Magnus Steinstø

Implementation of a Miniature Autonomous Directional Drilling Rig With Nonlinear Model Predictive Trajectory Control

June 2021

Implementation of a Miniature Autonomous Directional Drilling Rig With Nonlinear Model Predictive Trajectory Control

Magnus Steinstø

Industrial Cybernetics (MIIK)
Submission date: June 2021
Supervisor: Lars Struen Imsland

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

The petroleum industry has an interest in increased automation to improve safety, reduce costs, and improve efficiency. Fully autonomous drilling operations can improve safety by reducing the number of humans in line of dangerous equipment, while reducing cost and improving efficiency by addressing human operator shortcomings. Human drillers have limitations in term of data processing capabilities, reaction time, attention span, task execution precision, and long-term planning.

The miniature scale drilling rig competition Drillbotics hosted by DSATS aims to accelerate digitization by challenging student teams to find innovative solutions to industry problems. The challenge for the 2020/2021 competition was to design and implement a fully autonomous drilling rig capable of directional drilling to intersect one or multiple target points within a rock sample with up to 30-degree inclination and 15-degree azimuth adjustments from the starting point. The NTNU student team approached the challenge by creating a fixed bent sub Bottom Hole Assembly (BHA) with a directional steering concept of adjusting the orientation velocity of the BHA over time to control the future well path trajectory. Nonlinear Model Predictive Control (NMPC) was tested as fixed bend directional steering required long term planning to intersect target points. Position was estimated from hoisting movement and an orientation estimate of the BHA. Downhole IMU measurements were used for closed-loop orientation estimates. An attempt at changing the control software system software environment to Simulink was made, requiring a complete control system redesign to allow fast paced prototyping of controllers in a simulated environment.

Results suggest that NMPC trajectory control is a viable option for 3-dimensional steering with a fixed bend sub BHA and can replace more complex Rotary Steerable System (RSS) systems for certain applications in the petroleum or other industries requiring high precision directional drilling. A complete redesign of the control system in Simulink was successfully implemented.

Sammendrag

Petroleumsindustrien har interesse av økt automasjon for å forbedre sikkerhet, redusere kostnader og øke effektivitet. Full-autonome boreoperasjoner kan forbedre sikkerhet ved å redusere antall mennesker i nærheten av farlig utstyr, samtidig som at kostnader reduseres og effektiviteten øker ved å eliminere boreoperatørens menneskelige begrensninger fra prosessen. En boreoperatør har begrensninger i form av evne til å behandle data, reaksjonstid, fokus, presisjon og langsiktig planlegging.

Konkurransen for borerigger i miniatyrskala Drillbotics arrangert av DSATS har som målsetning å øke digitalisering i petroleumsindustrien ved å utfordre studenter til å finne innovative løsninger på industrirelevante utfordringer. Utfordringen for konkurransen i 2020/2021 var å utvikle en full-autonom borerigg for å treffe et eller flere målpunkter i en steinprøve med opptil 30 grader helning og 15 grader rotasjonsavvik fra startpunktaksen. Studentlaget fra NTNU forsøkte å løse dette med å lage nedhullsutstyr (BHA) med en fastsatt bøyningsvinkel. Styringskonseptet var å justere rotasjonshastigheten til orienteringen av nedhullsutstyret over tid for å kontrollere den fremtidige brønnbanen. Ikke-lineær Modell Prediktiv Kontroll (NMPC) ble brukt da styring med fastsatt bøy krever langsiktig planlegging for å treffe målpunktene nøyaktig. Posisjon ble estimert fra forflytninger av hevesystemet og orienteringsestimat fra sensormålinger i nedhullsutstyret. Akselerometer og magnetometer målinger fra nedhullssensoren ble brukt for orienteringsestimat i lukket sløyfe. Det ble forsøkt å endre softwareplattform til Simulink. Endringen krevde en full ombygging av hele kontrollsystemet med hensikt i å kunne teste kontrollere i simulering.

Resultatene antyder at NMPC kan brukes for presisjonsstyring av fremtidig brønnbane for nedhullsutstyr med fastsatt bøy. Dette kan muligens erstatte bruken av mer komplekse styrbare bits (RSS) systemer for enkelte bruksområder i petroleumsindustrien og andre sektorer med behov for høypresisjon retningsstyrt boring. En fullverdig kontrollsystemimplementasjon ble utviklet i Simulink.

Acknowledgements

Firstly, I would like to thank the student team of Gaute Hånsnar, Trygve Mikal Viga Skretting, and Benedicte Gjersdal for their hard work, dedication, insight, and knowledge required for making this multi-disciplinary project possible. I would also like to thank my supervisor Lars Struen Imsland for great support and guidance throughout the duration of the project. Project leader Alexey Pavlov and the petroleum supervisor support team has been of great help when discussing ideas and concepts in the biweekly meetings and on request, and I would like to thank them for their contribution.

I want to show appreciation to the supporting engineering team. Without your insight, guidance, and help with daily challenges the rig would never have been operational. I would like to thank Torkjell Breivik for helping with implementation of electrical components and advice regarding automation in general. Noralf Vedvik, Terje Bjerkan, and Håkon Myhren has been invaluable for helping with design, implementation, general advice, and machining for the mechanical system.

Thanks to DSATS and SPE for hosting and organizing the Drillbotics competition. This year's challenge has been interesting and forced the team to find creative solutions across disciplines. I am confident that everyone participating in Drillbotics has learned a lot and are better prepared for their future careers.

Huge thanks to our sponsors BRU21, Equinor, and Lyng Drilling for making this project possible with their financial funding and sponsorship. A special thanks to John-Morten Godhavn from Equinor for his long-term commitment to the project.

I would also like to thank Kim Anderssen from Nordic Automation for great technical support while implementation servo motors.

Contents

List of Figures

List of Tables

Abbreviations

AC	Alternating Current.
AI	Artificial Intelligence.
AoA	Angle-of-Arrival.
AoD	Angle-of-Departure.
API	Application Protocol Interface.
BFGS	Broyden-Fletcher-Goldfarb-Shanno.
BHA	Bottom Hole Assembly.
BLE	Bluetooth Low Energy.
CAN	Controller Area Network.
CIP	Common Industrial Protocol.
CPU	Central Processing Unit.
DAQ	Data Acquisition.
DC	Direct Current.
DLS	Dogleg Severity.
DSATS	Drilling Systems Automation Technical Section.
EDS	Electronic Data Sheet.
FIFO	Fist In First Out.
GB	Gigabyte.
GUI	Graphical User Interface.
HMI	Human Machine Interface.
HSE	Health and Safety Executive.
I2C	Inter-Integrated Circuit.
IDE	Integrated Development Environment.
IGP	Department of Geoscience and Petroleum.
IMU	Inertial Measurement Unit.
IP	Internet Protocol.
ITK	Department of Engineering Cybernetics.
KKT	Karush-Kuhn-Tucker.
KOP	Kickoff Point.
LED	Light Emitting Diode.
MAC	Media Access Control.
MD	Measured Depth.
MPC	Model Predictive Control.
MVP	Minimum Viable Product.
NIC	Network Interface Card.
NMPC	Nonlinear Model Predictive Control.
NOK	Norwegian Krone.
NPT	Non-Productive Time.
NTNU	Norwegian University of Science and Technology.
ODE	Ordinary Differential Equation.
OS	Operating System.
OSI	Open Systems Interconnection.
P2P	Peer-To-Peer.
PCB	Printed Circuit Board.
PCI	Peripheral Component Interconnect.
PDM	Pressure Displacement Motor.
PID	Proportional Integral Derivative.
PLC	Programmable Logic Controller.
QP	Quadratic Programming.
RAM	Random Access Memory.
ROP	Rate Of Penetration.
RPM	Revolutions Per Minute.
RSS	Rotary Steerable System.
RSSI	Received Signal Strength Indication.
SDK	Software Development Kit.
SPE	Society of Petroleum Engineers.
SPI	Serial Peripheral Interface.
SQP	Sequential Quadratic Programming.
SSD	Solid State Drive.
TCP	Transmission Control Protocol.
TVD	True Vertical Depth.
UA	Unified Architecture.
UART	Universal Asynchronous Receiver-Transmitter.
UDP	User Datagram Protocol.
USB	Universal Serial Bus.
USD	United States Dollar.
UUID	Universal Unique Identifier.
WOB	Weight On Bit.
XML	Extensible Markup Language.

1 Introduction

The petroleum industry has an interest in increasing the degree of automation to improve safety, reduce costs, and eliminate or reduce incidents. An oil and gas installation will have humans in line of potentially dangerous equipment that may cause serious harm to personnel. Fixed costs of petroleum installation are enormous, making downtime and Non-Productive Time (NPT) costly. Incidents may cause damage to personnel and the environment, damaged critical components, and major economic losses.

Drilling is executed by a human operator in the industry. The operator must constantly make informed decisions based on drilling theory, available data, and previous experience to drill the intended well path. Drillers need extensive training, with experienced drillers usually having a lower failure rate and higher efficiency. Drilling does currently have a low degree of autonomy. Some newer installations have software for executing automated drilling tests, incident detection, and drilling state alerts.

Autonomous drilling has the potential of removing human limitations from the drilling process in terms of data processing capabilities, reaction time, reasoning, and attention span. An autonomous system does not require rest or extensive training. The computer can process multiple incoming streams of data in real-time. Complex relations between parameters can be used for incident detection and drilling optimization. The system will always be on high alert and detect and react to incidents faster than any human.

Directional drilling is to drill a well path that has inclination relative to the gravity field. Most production wells are drilled with a high inclination to increase extraction rates from the reservoir. Two common approaches to drill a well path with inclination is to use a Rotary Steerable System (RSS) or bend sub system for the Bottom Hole Assembly (BHA). RSS is the most modern approach allowing the bit to be orientated in any direction relative to the BHA. The BHA will move in the direction of the bend when applied Weight On Bit (WOB). RSS systems are complex and expensive. Bent sub solutions used to be the standard approach to drilling deviated well paths. The BHA has either a fixed or adjustable bend angle. Bend sub BHAs are cheaper, less complex, and more robust but are more difficult and less intuitive to steer accurately.

The NTNU Drillbotics student team has spent the fall of 2020 and spring 2021 developing a fully autonomous directional drilling rig. The team competes in the yearly Drillbotics competition hosted by DSATS were universities build a fully autonomous miniature scale drilling rig. This year's challenge was to intersect one or multiple target points in a deviated well path with up to 30-degree inclination and 15-degree azimuth orientation deviation between the points. The NTNU team reasoned that an RSS system would be difficult to build mechanically robust at a miniature scale and therefore opted for creating a fixed bend BHA.

The steering concept relied on adjusting the orientation of the BHA over time to control future trajectory. By planning far ahead it was assumed to be possible to accurately steer the BHA in 3 dimensions to intersect all target points. The possibility of using Nonlinear Model Predictive Control (NMPC) for controlling future well path trajectory was therefore explored. NMPC was chosen as it relies on a system model and it was assumed that the future trajectory would be in the direction of the bend angle orientation of the BHA. The input in the form of BHA rotation could then be used for steering. The NMPC controller will try to predict future states in a simulation, optimize the simulation by adjusting inputs in the prediction horizon, before applying the first input of the optimized range to the system.

Accurate steering relies on an accurate position estimate. It was investigated how to accurately estimate position throughout the drilling operation inside the rock sample and make use of downhole sensor data to improve estimates in closed-loop.

New requirements for advanced controller and simulations capabilities for this year's competition has encouraged the team to change the software environment from LabVIEW to Simulink. A complete rebuild of the control system would then be required with the benefit of allowing controllers to be tested in simulation before being applied to the physical rig for fast paced prototyping.

The work in this thesis is based on research and experiments from the "Design Report NTNU - Drillbotics 2021 Phase I"[13] where the rig implementation was planned and a simulation environment prototype for the system model tested. The thesis written by the petroleum students on the team called "Design and Implementation of a Miniature Rig for Autonomous and Directional Drilling"[12] is frequently referenced.

The research could benefit the petroleum industry by allowing fixed bend BHAs to replace more complex Rotary Steerable System (RSS) systems for certain applications. Other industries requiring accurate high inclination directional drilling on a small-scale where it is difficult, expensive, or impractical to use a RSS systems could also benefit.

The thesis has 10 chapters excluding the introduction, conclusion, and future work. A brief description for each follow:

- **Chapter 2: Project Overview**: The Drillbotics competition, organization structure, available resources, workflow, and safety
- **Chapter 3: Mechanical Design**: Mechanical concept and brief description of key components
- **Chapter 4: Electrical System**: Overview of servo motors, sensors, and electrical components present on the rig
- **Chapter 5: MATLAB and Simulink**: Comparison to LabVIEW and introduction to software tools used in the project
- **Chapter 6: Downhole Sensor**: Description of new downhole sensor and its implementation
- **Chapter 7: Hardware Communication**: Communication and control structure for automation hardware implementation
- **Chapter 8: Low-Level Control System Implementation**: Controllers, incident detection, and features in the responsive low-level control system
- **Chapter 9: Position Estimation**: Position model and uncertainty reduction for the position estimate
- **Chapter 10: Directional Drilling MPC**: Implementation and design choice discussion for NMPC trajectory controller
- **Chapter 11: Autonomous Operation**: Description of state machine implementation

2 Project Overview

NTNU Drillbotics is a student project designing and implementing an autonomous miniature drilling rig to participate in the yearly Drillbotics competition. The project is organized by the Department of Geoscience and Petroleum (IGP) with support from Department of Engineering Cybernetics (ITK). The main purpose of the project is to research and accelerate digital innovation in the petroleum industry.

2.1 Drillbotics Competition

Drilling Systems Automation Technical Section (DSATS) has since 2015 hosted the Drillbotics competition challenging universities to build a fully autonomous drilling rig. DSATS is a subgroup of Society of Petroleum Engineers (SPE), a non-profit organization to collect and exchange technical knowledge in the petroleum industry for the public benefit[68]. SPE manages multiple publications and organizes knowledge sharing events. NTNU has participated every year since the 2015/2016 competition.

DSATS and the Drillbotics committee will early fall provide set of guidelines describing the objective, rules, guidelines, and evaluation criteria for next year's competition. Guidelines are changed every year and will typically have a major revision once a student team has mastered the competition objectives. All teams are restricted to a maximum budget of 10.000USD and a maximum of 5 student team members. The 2020/2021 guidelines can be found in subsection F.

2.1.1 Motivation

DSATS believes that challenging students to collaborate in a multi-disciplinary team is highly beneficial for future petroleum, mechanical, electrical, and control engineers and builds a solid foundation for post-graduate careers[10]. The petroleum industry seeks to improve safety, lower costs of operation, and improve efficiency.

An autonomous system can react and respond to incidents faster, process more data simultaneously, and execute drilling commands more precisely and reliably than any human operator. Fully automated drilling operations have not yet been implemented due to technical complexity, and DSATS encourage students to find creative solutions to industry problems.

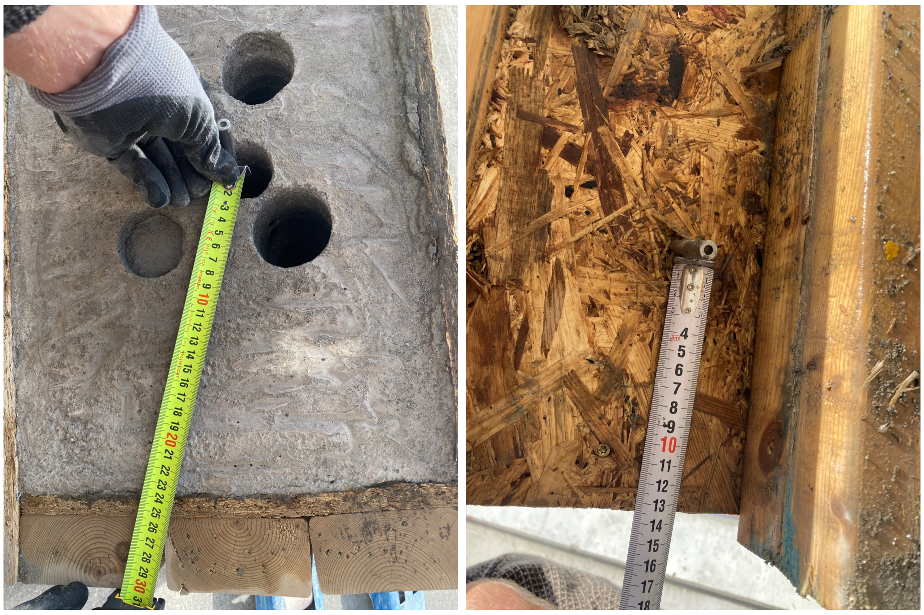
2.1.2 Competition Groups

Since the 2018/2019 competition there have been a group A and group B competition. Group A has traditionally been the physical rig competition. The first group B competition had last year's competition guidelines and was intended to attract universities that had never participated in the competition before. For the 2019/2020 competition group B was changed to a virtual rig simulation competition. Due to the covid-19 pandemic the Drillbotics 2019/2020 competition was cancelled, and the group labels inverted for the 2020/2021 competition to give all universities a fair chance to participate regardless of covid-19 restrictions. The NTNU student team was most motivated to participate in the group B physical rig 2020/2021 competition.

2.1.3 2020/2021 Competition

Because the 2019/2020 competition was cancelled, only minor revisions were done to 2020/2021 guidelines. The challenge was to drill a deviated well path intersection one or multiple target points with a maximum inclination and azimuth of 30 and 15 degrees, respectively in a 60cm tall rock sample. The first 4 inches had to be drilled vertically. Hard requirements for the rig included fully autonomous operation, closed-loop directional control based on downhole sensor data, and remote operation from an API.

On the day of the competition NTNU demonstrated a highly capable miniature drilling rig meeting all hard requirements of the competition. The operation was executed fully autonomously with a highly respectable drilling performance. Judges were so impressed they requested an extra run testing maximum Rate Of Penetration (ROP) and inclination of the rig. The resulting well path was drilled in 6 minutes with a 35-degree inclination, equivalent to a displacement of 17cm. NTNU was later announced as the winners of the competition.



(a) Top Measurement. (b) Bottom Meaurement

Figure 1: Competition run with displacement measurement.

2.2 Student Team

The 2020/2021 NTNU Drillbotics student team had 4 members with 1 cybernetics and 3 petroleum students. Each team member was assigned roles during the fall of 2020.

- **Gaute Hånsnar**: Petroleum engineering student and main responsible for BHA design and implementation. Gaute has also created the control system API for remote operation of the rig, and he has been the main responsible for economics.
- **Magnus Steinstø**: Industrial cybernetics student and student team leader. He has a bachelor's degree in computer science and has previously been a part of the UiS Drillbotics team for the 2018/2019 competition. Main responsible for the control system, system modelling, and BHA steering.
- **Trygve Mikal Viga Skretting**: Petroleum engineering student and main responsible for the custom bit design. Trygve has been the HSE responsible and he has been the team member that has done the most practical work on the rig.
- **Benedicte Gjersdal**: Petroleum engineering student and main responsible for the hydraulic system. She has designed the hydraulic swivel and components for the azimuth control system while actively researching reports and papers.

Team members must be agile in a small team and may have to take on tasks outside their area of expertise, requiring close collaboration and creative problem solving.

(a) Gaute Hånsnar (b) Magnus Steinstø (c) Trygve Mikal Viga Skretting (d) Benedicte Gjersdal

Figure 2: Student team members.

2.3 Project Supervisors

Professors from IGP and ITK has followed the project closely and helped in problem solving, planning, knowledge transfer, and thesis guidance among other tasks. Supervisors have been available at biweekly meetings and on request for consultation. The supervisors have followed the project for multiple years and have in depth knowledge of petroleum and cybernetics, and how these areas of expertise can be combined. With no student continuation from last year, supervisors have been an invaluable resource.

2.4 Available Resources

Students have almost exclusive access to a miniature scale drilling rig carried over from previous student teams. The student team is allowed to make permanent modifications to the rig to best fit the requirements for the Drillbotics competition and their research.

The student team has a dedicated office space located close to the miniature scale drilling rig at IGP. Automation and mechanical engineers have been available for consultation on short notice with offices located next to the student team office. A small electronics workshop with a 3D-printer is next to the office.

IGP has an in-house workshop with skilled personnel for machining mechanical components. The workshop has full-time employees that have created most of the custom mechanical components used in the project. An in-house workshop has allowed fast paced design alteration and minimal downtime while fixing broken parts.

2.5 Sponsors and Contributions

Most of the financial funding comes from the BRU21 project. BRU21 is a research and innovation program with the ambition of accelerating digitization and automation in the petroleum industry[9]. The program unites academic and industrial experts within the fields of petroleum, cybernetics, robotics, computer engineering, and data science, among others. The project is centred around PhD and Postdoc research projects.

Lyng Driling is a Schlumberger company located close to Trondheim. The company has helped the student team design and produce custom drilling bits. The team has been on a company visit to their facilities to learn more about how their bits are designed and produced.

Equinor has provided financial support to the project. The student team has had a concept presentation and a drilling demonstration to their drilling automation division.

2.6 Kanban

Kanban is a structure for organizing work. Kanban is commonly used in the software industry to maintain an overview of what a team is currently doing and limit any person to one or very few concurrent ongoing tasks. Current state is tracked with a Kanban board where tasks are arranged into columns depending on their execution state. The team has used 7 different states.

- **To Be Done**: Column for all tasks that should be done at some point. This could be tasks that needs to be done everything from 1 week to multiple months into the future. A task is move to "Prioritized" if it is to be completed within the next week.
- **Prioritized**: Tasks that are not currently in progress but should be completed by the end of the week. Tasks that are in progress and assigned to specific people and moved to "In Progress".
- **In Progress**: Tasks currently being worked on. Any person should be assigned to an absolute minimum number of concurrent tasks. Completed tasks are moved to "Done".
- **Done**: Tasks and any subtasks have been successfully completed. "Done" tasks are archived after the end of week meeting.
- **Paused**: If a higher priority task occurs while a low priority task is "In Progress", the task is moved to "Paused" to be resumed later.
- **Blocked**: An in progress task has been temporarily aborted because of something outside the control of the student team. This could be restrictions, issues, or dependencies on other tasks.
- **Parked**: A task in any of the previous states have temporarily been removed from the list of tasks to be done. If a task becomes irrelevant it is discarded from the "Parked" column.

Each task description should be self-descriptive and with well-defined completion criteria. Tasks are not assigned to any person before they are in progress. A rule of thumb is that a task should be limited such that it does not last more than 1-2 days. Tasks are moved from "To Be Done" to "Prioritized" during the Monday meeting. Tasks are added to the "To Be Done" column once any of the team member have recognized something that needs to be done.

2.7 Work Philosophy

The team has had a work philosophy centred around making Minimum Viable Product (MVP) for each task. Iterative improvements are then made before they are tested in combination with other MVPs. Low effort solutions are often sufficient and should be tested before more complex solutions are attempted. Testing a component or software as early as possible often uncovers challenges and implications that are difficult to discover in the planning phase. Simple prototypes like 3D prints or minor software alterations can be tested to help isolating the issue before a new MVP is created.

2.8 Budget

Team members are equally responsible for maintaining an up-to-date budget with their expenses in a shared spreadsheet. Budget is tracked in both NOK and USD to comply with both the internal budget restriction from the university and the competition guidelines. The student team has 75.000NOK at their disposal with funding up to 100.000NOK if strictly necessary. A final expense overview is shown in Table 1[12].

Table 1: Budget Overview.

Component	**Price per item (USD)**	**Quantity**	**Total Cost (USD)**
Drill Chuck	43.42	1	43.42
Hollow shaft gearbox	626	1	626
Shipping gearbox	135	1	135
Servomotor + Driver	602	1	602
Universal joint	27.4	4	109.6
Radial ball bearing	30.92	4	123.68
Aluminium drill pipes	2.57	110	282.7
Aluminium rod	1.1	120	132
Shipping drill pipe and rod	182	1	182
Generic drill bit 1	23	3	69
Generic drill bit 2	20	5	100
Sensor card and USB cable	41	5	205
Inspection camera	33	2	66
Azimuth gearbox	377.5	2	755
Lenze EtherNet I/P module	430	1	430
Torque sensor	156	3	468
Hard drive and cables	166.57	1	166.57
Drill pipe connector	39.25	4	157
Mixing nozzle glue	30	1	30
Pilot hole bit	31.67	3	95
Silicone material	304	1	304
Titanium and steel rod	9.65	75	724
Top Drive motor	1730	1	1730
Epoxy glue and sensor card	192	1	192
Total Cost (USD)			**$ 7 730** **NOK 63 850**
Balance (NOK)			**NOK 11 150**

2.9 Covid-19 Precautions

The global pandemic forced everyone involved in the project to take extra precautions to avoid contamination and potential serious illness. People were encouraged to stay home and get tested if they had any symptoms of the virus and maintain at least 1 meter distance to each other or wear a face mask if this was not practically possible. Hand hygiene was of upmost importance and disinfectant dispensers were located at most entrances to the building where sinks were not available. Personal protective gear was used to reduce shared contact surfaces. The team was assigned two additional offices to be able to maintain sufficient distance.

Throughout the duration of the project there have been three longer periods, totalling up to around 4 weeks, were access to lab and offices was either restricted or strongly discouraged. Home office limited the progress on some tasks requiring on sight access.

None the personnel directly involved with the project have been infected with covid-19 during the duration of the project. Some team members were quarantined because of contact with people who had tested positive for covid-19.

2.10 HSE

The miniature drilling rig has the potential of causing serious harm to personnel if safety procedures are not followed. The first step was to identify potential risk elements and discuss what can be done to eliminate or reduce the risks. Analysis and procedures are further described by the petroleum engineering students of the team[12]. A shortlist of identified risks are:

- **Hazardous Energy**: Electrical components on the rig has the potential of causing electrical shock to personnel. The rig requires water as a drilling fluid to flush out cuttings and bind dust. All electrical components near the drilling fluid has some water resistance and all cables have shielding from water. Untrained personnel should under no circumstances make electrical alteration in the automation cabinet on the rig.
- **Mechanical Hazards**: Mechanical failures may cause components with high kinetic energy to come lose during operation. A protective glass is lowered while drilling to protect the personnel. Some components like the bit will be exposed while drilling with a high rotational velocity. The hoisting system can exert large forces capable of causing serious harm to misplaced extremities.
- **Chemical Hazards**: Personnel is exposed to potentially harmful chemicals like epoxy, grease, and concrete. Concrete has fine dust that can interfere with the soft tissues in the breathing system. Concrete dust is present on the floor after a drilling operation and will be circulated up into the air if not sufficiently cleaned.
- **Ergonomic Hazards**: Moving heavy equipment must be done with caution to avoid injuries. Moving the rock sample in place requires some manual adjustments. A trolley jack is used to move the rock sample for longer distances.

2.10.1 Safety Measures

Procedures and preventive actions for some potential safety hazards are listed. A more thorough description is made by the petroleum students of the team[12].

- **During Rig Construction**: Assembling the rig requires personnel to be in close proximity to the moving parts of the rig. All movements are stopped when somebody has extremities in line of potential movement. Lose hair and clothing are not permitted near the rig. The BHA may have sharp edges and protective gloves should be used to avoid cuts. Before commencing the drilling operation, all motors should be tested at a low rotational velocity to verify that all components are mounted correctly.
- **During Rig Operation**: There should always be two people present at the rig during operating. At least one of the personnel should be able to stop the operation at short notice, both through software and using the emergency stop button. The protective glass should be lowered unless access to components is required. Parameters like top drive torque, Weight On Bit (WOB), and azimuth torque must always be monitored to ensure the rig is within safe operating range.
- **Limiting Operational Forces**: Motors are capable of exerting forces exceeding the what the mechanical system can sustain. Rods can twist or wind-up during operation, drill pipes will buckle under excessive load, and the azimuth system can disassemble the BHA. Closed-loop control is used to maintain forces within a safe operating range and the control system has features to react to incidents.
- **Fire Safety**: Electrical system and some of the chemicals used may cause fire. Fire hoses and fire extinguishers are placed near the rig to minimize potential damage. No unqualified personnel are allowed to make modifications to the electrical system.
- **Ergonomics**: Lifting heavy items manually should be avoided as much as possible. Jack trolleys and a forklift are available. If a heavy item must be moved manually for practical reasons, team members should assist to reduce the load on an individual and all lifts should be done with a straight back.
- **Stop Work Authority**: If any personnel at any point feels is uncomfortable or detects a potential risk that should be addressed before continuing, all work should be stopped immediately. Safety requires trust and any request to halt progress should be respected and followed at any time.

- **Emergency Stop Button**: The software control system may stop working as intended during operation. An emergency stop button is installed within arms reached of the rig operator. The threshold for pushing the button should be low if something unexpected occurs or if the software control system is not responding.

Safety procedures and measures have no value if they are not known or respected. Any new personnel operating the rig should study the safety evaluation on the rig before attempting to operate the rig.

2.11 GitHub

GitHub is a source control tool that simplifies the organization of code and collaboration when there is multiple software developers[16]. GitHub is especially useful when multiple people make alterations to different parts of the code as changes from multiple sources can be merged automatically. Code is organized into repositories and further divided into branches. Branches holds code at different stages of development within the repository.

A common organization of branches is:

- **master**: The root branch all code is based on. Master is generally the code in production and is expected to be stable and all features functional. For the control system this was a well-tested version of the control system.
- **develop**: Merge branch for features that are tested independently. Code is expected to be stable enough to build new features based on the current code. For the control system, develop branch was used during drilling tests when experimenting with some new features.
- **feature-[name of feature]**: Features currently in development based on code from develop. Feature branches are created by a developer creating a new feature and the code is not expected to be stable. Feature branches should be removed once a new feature is tested and pushed to develop. When developing the control system, the feature branch was used for regular progress backups and during drilling operations to test new features that are not finalized.

This project had one main developer. GitHub was still useful for transferring code between multiple computers, code backups, code rollback, and documenting changes.

3 Mechanical Design

The miniature drilling is the main asset of the project and is carried over from previous years. The rig is used almost exclusively by NTNU Drillbotics and permanent modification can be made to best fit competition and research objectives.

3.1 Change of Concept

The bit propulsion system has been changed compared to previous years. The two recent teams have attempted to design a downhole Pressure Displacement Motor (PDM) motor. Calculations performed by the team suggested that such a motor would not provide sufficient torque without an impractically long Bottom Hole Assembly (BHA) at this scale[12]. Attempts from previous years have shown that creating a reliable PDM motor is difficult[18].

Inspired by the 2018/2019 contribution from University of Oklahoma, a new drilling concept with a rod transferring torque from a top drive motor to the bit with a rod was attempted. The rod is rotated inside of the hydraulic system, the drill pipe, and the BHA, and connect to a drive shaft in the BHA rotating the bit.

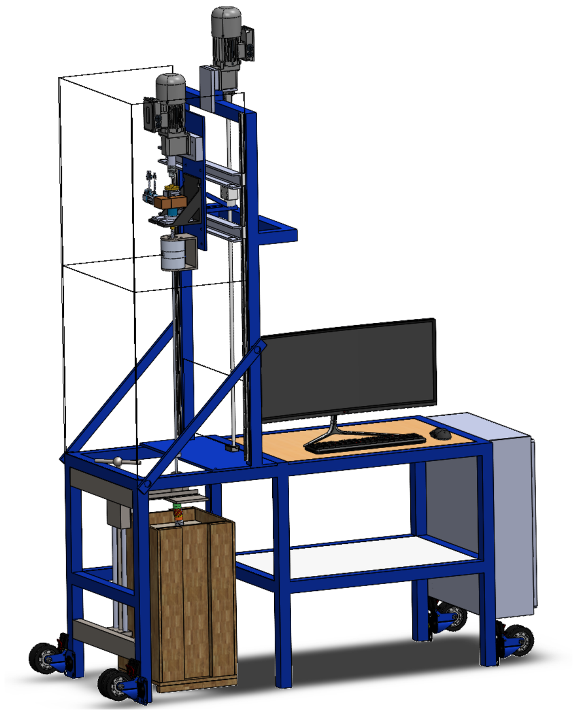


Figure 3: Solidworks overview of the rig.

The steering concept relies on a fixed bend BHA that will build a deviated well path in the direction of the bend relative to its current orientation when Weight On Bit (WOB) is applied. Orientating the BHA can therefore be used to control the direction of the well path. The drill pipe is connected to the BHA and can be used to change its orientation. The top drive was previously used to orientate the drill pipe, and the concept therefore required a new dedicated azimuth control system. With sufficient long-term planning the BHA trajectory can be steered in 3 dimensions.

3.2 Rig Frame

All rig components are directly or indirectly connected to a large metal structural frame. The frame is a robust platform and static reference for the rest of the mechanical components. The hoisting system, automation cabinet, operator desk, protective glass, and a movable protective cover is directly attached to the frame.

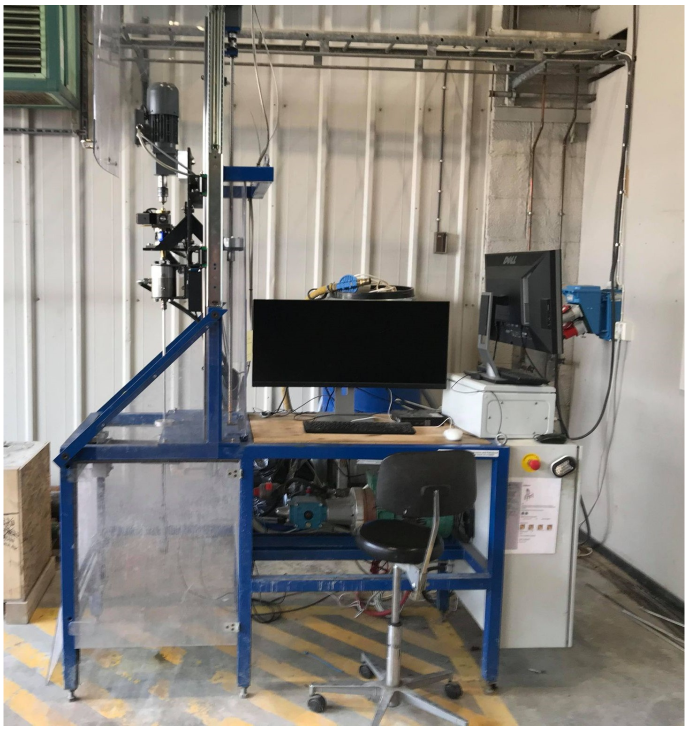


Figure 4: Rig frame with all mechanical components attached.

3.3 BHA

The part of the drilling system holding the bit is called the Bottom Hole Assembly (BHA). The BHA is connected to the drill pipe and has a propulsion rod in the center providing torque to rotate the bit. The BHA has 5 major components: upper stabilizer, sensor housing, bend angle, drive shaft, and bit sub.

The top part of the BHA is the upper stabilizer acting as a point of contact to the well path. This part has a slightly larger diameter than the rest of the BHA with surface grooves for added stability. The drill pipe is attached to a friction mount in the top of the stabilizer.

Below the upper stabilizer is the sensor sub with a compartment for the downhole sensor. The component protects the sensor from mechanical stress and water. The downhole sensor cable is passed out the top of the housing and through a hole in the upper stabilizer. Sensor implementation is described in section 6. Set screws in top and bottom holds threads in place while rotating.

The bend angle has a connection from the drilling rod to the drive shaft providing torque to the bit. The rod is attached to a flexible joint with set screws. The other end of the joint is attached to the drive shaft going through the lower stabilizer and into the bit sub. Two bend angles were tested with approximately 5- and 7-degrees bends.

Between the bend angle and the bit sub is a component with the same shape and diameter as the upper stabilizer to act as a point of contact to the well path. The drive shaft is passed through the stabilizer and into the bit sub. There is a low friction pad between the lower stabilizer and the bit sub to hold the bit sub in place while drilling.

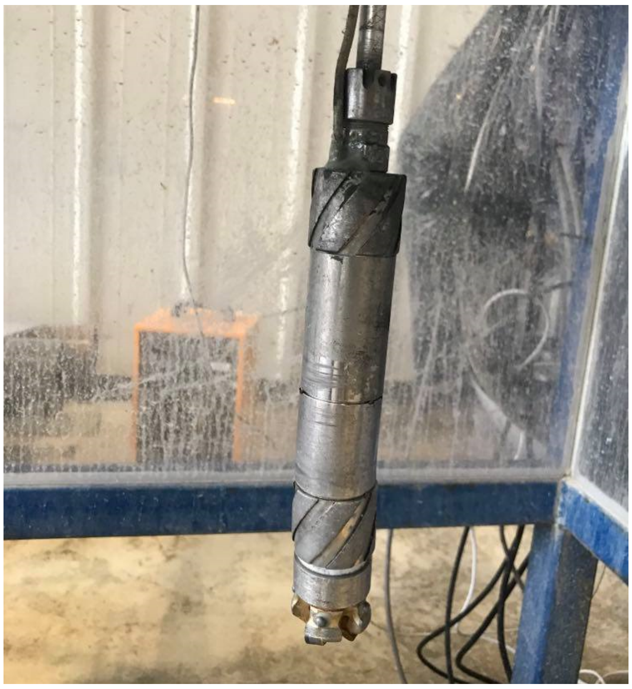


Figure 5: The BHA connected to the drill pipe.

3.4 Riser

To reduce movements and vibrations in the BHA before the lower stabilizer is inside the well path, a riser with a slightly larger diameter than the BHA is placed around the BHA to limit horizontal movement. The riser can be moved up and down using a threaded ball screw with a manual lever.

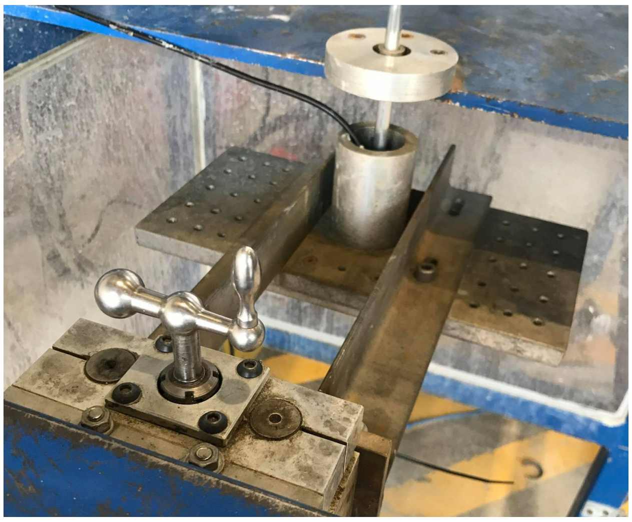


Figure 6: Riser with manual lever with the BHA inserted.

3.5 Hydraulic Swivel

Drilling fluids are used to flush out cuttings, cool the bit, lubricate components, and binds dust among other functions. Tap water is used as the drilling fluid for practical reasons. The swivel allows fluid to be directed to the bit through the drill pipe while allowing unidirectional rotation of the drill pipe. Load transfer from applied WOB is propagated to the swivel, transferring the load to the hoisting system frame.

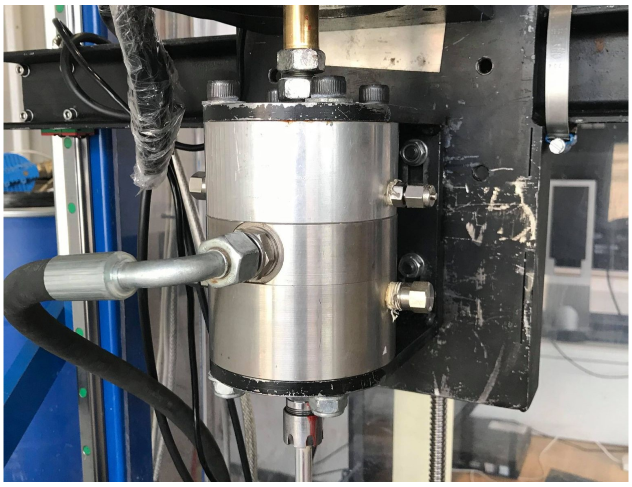


Figure 7: Hydraulic swivel with drill pipe at the bottom and T-shaft at the top.

The component is a modified version of the hydraulic swivel designed for the 2018/2019 competition. The original swivel was designed to only allow unidirectional rotation. Rotating in the opposite of the intended direction would unscrew and dismantle the swivel connections. Modifications was also required to make the swivel hollow, allowing the rod from the top drive to pass through.

Unidirectional rotation was achieved with a combination of 3 locking nuts on the shaft in the top of the swivel connecting to the T-shaft. The locking nut combination ensures that the threads of the shaft will not loosen if there is an applied pressure to the nut. Inlets inside the swivel allows water to be forced into the drill pipe and out of the bit nozzles.

3.6 Azimuth Control System

The steering concept required an accurate BHA orientation system that could be regulated with closed-loop control. Initially, the plan was to use a hollow shaft servo motor. Hollow shaft servo motors with appropriate dimensions, torque range, and software compatibility proved difficult to find. Some robotic joint motors met our requirements, but quotes for such motors indicated that over half the budget would have to be spent on the azimuth control system. Cheaper alternatives were therefore investigated.

An alternative to a hollow shaft servo motor was to use a hollow shaft gearbox with a conventional servo or stepper motor. A gearbox system would reduce the orientation accuracy of the drill pipe and makes torque estimates from a servo motor less accurate. This was an acceptable trade-off as the differences in accuracy were in the ratio of a few arc-min, and the main purpose of measuring torque from the azimuth system was to enforce an upper torque limit through closed-loop control while orientating the BHA inside the rock.

3.6.1 Hollow Shaft Gearbox

A hollow shaft gearbox of the model GSH60 by Gigager was acquired to precisely orientate the pipe using a servo motor. The gearbox has a 30:1 gear ratio, permissible rotation speed of 90 RPM, and permissible torque of 30 Nm. Repeatability of and positioning accuracy are ≤ 10 and ≤ 50 arc-sec, respectively. The gearbox must be protected from water and dust as it has a IP40 rating[14]. The hollow shaft of the gearbox has a diameter of 18mm, allowing the T-shaft to be passed through.

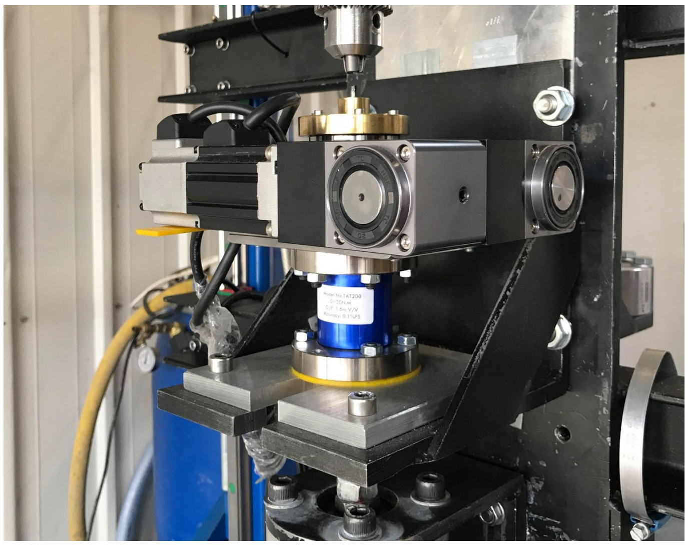
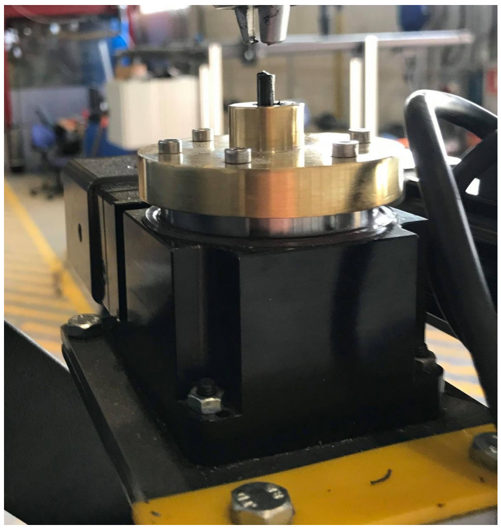
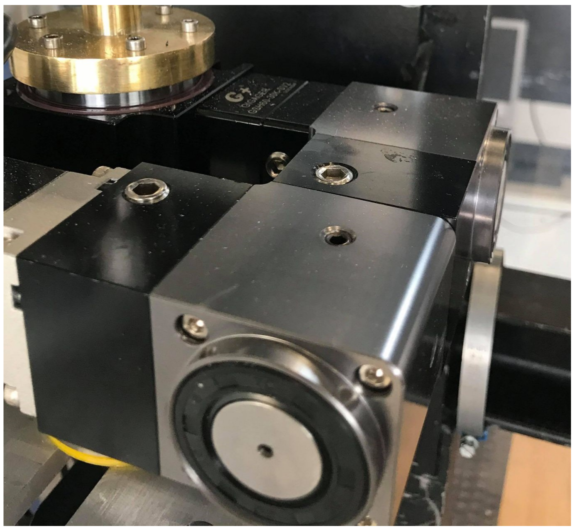


Figure 8: Azimuth control system with servo motor, gearbox system, torque sensor, and T-shaft.



(a) Hollow shaft gearbox.



(b) Two Gigager GSZ042-03K-SV connected in series.

Figure 9: Gearbox system for azimuth control.

3.6.2 Higher Gear Ratio

During initial testing it was discovered that the minimal rotation speed of the motor was too fast for accurate velocity control of the BHA orientation. This is described further in subsection 4.3. To be able to rotate the drill pipe with a slower speed, two additional gearboxes of type GSZ042-03K-SV from Gigager were added between the hollow shaft gearbox and the servo motor. This model was chosen as it was compatible with the hollow shaft gearbox and there was no single gearbox with sufficient gear ratio from the manufacturer. Each gearbox has a 3:1 gear ratio resulting in a total gear ratio of 270:1. Permissible input speed of the gearboxes are 2500 RPM, and permissible torque is 12 Nm. Each gearbox has a backlash of less than 0.5 arc-min[15].

Increasing the number of gearboxes increases the backlash and should be avoided if possible.

3.6.3 Torque Sensor

When further increasing the gear ratio with multiple gearboxes it was assumed that the torque estimate from the motor would not be accurate enough to be used in closed-loop control. An external torque sensor was therefore added. The sensor measures the difference between a mechanical reference and the torque provided by the motor and gearbox system.

Finding a torque sensor with dimensions closely matching the mounting points of the gearbox, with water resistance, a hollow shaft of at least 18mm, and a sensible torque proved difficult. Initially, the plan was to use a low-profile sensor to minimize the length of the T-shaft and mount the sensor directly to the hydraulic swivel. No off-the-shelf components meeting all hard requirements was found and custom options were orders of magnitude more expensive.

A more common type of static torque sensor is a cylinder instead of a plate. An alternative design was proposed with the torque sensor connected to a mechanical reference with the gearbox mounted on top of the sensor with an adapter plate. To be compatible with already manufactured components a shelf was used as the mechanical reference.

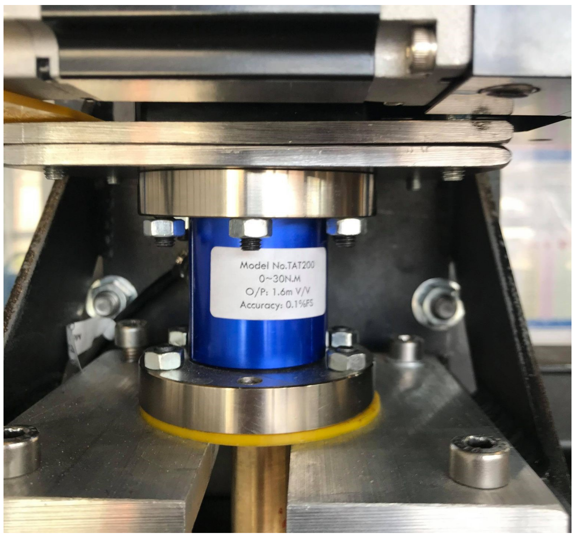


Figure 10: Torque sensor with a mechanical reference and the gearbox system in either end.

3.6.4 T-Shaft

A T-shaft transfers torque from the hollow shaft gearbox to inner rotating part of the swivel connected to the drill pipe. The T-shaft is mounted to the top of the hollow shaft gearbox and goes through the hollow shaft of the gearbox and torque sensor to connect with the rotating inner part of the swivel. Locking nuts allows unidirectional movement without unscrewing the threads. The T-shaft is made of brass to reduce rod friction. The inner diameter of the T-shaft closely matches the diameter of the rod to improve water sealing.

3.7 Hoisting System

WOB is applied to the bit by moving the hoisting mounting plate slide vertically. The slide has the top drive, swivel, and azimuth control system is directly attached. The drill pipe is attached to the drill pipe applying WOB. The mounting plate is connected to a ball screw vertical movement system. The screw axle is connected to the hoisting motor with the ball screw to the mounting plate. There are rails connected to the rig frame and the mounting plate to stabilize the slide and ensure that movement from the ball screw will only move the plate in the vertical direction. The hoisting system is unchanged from previous years.

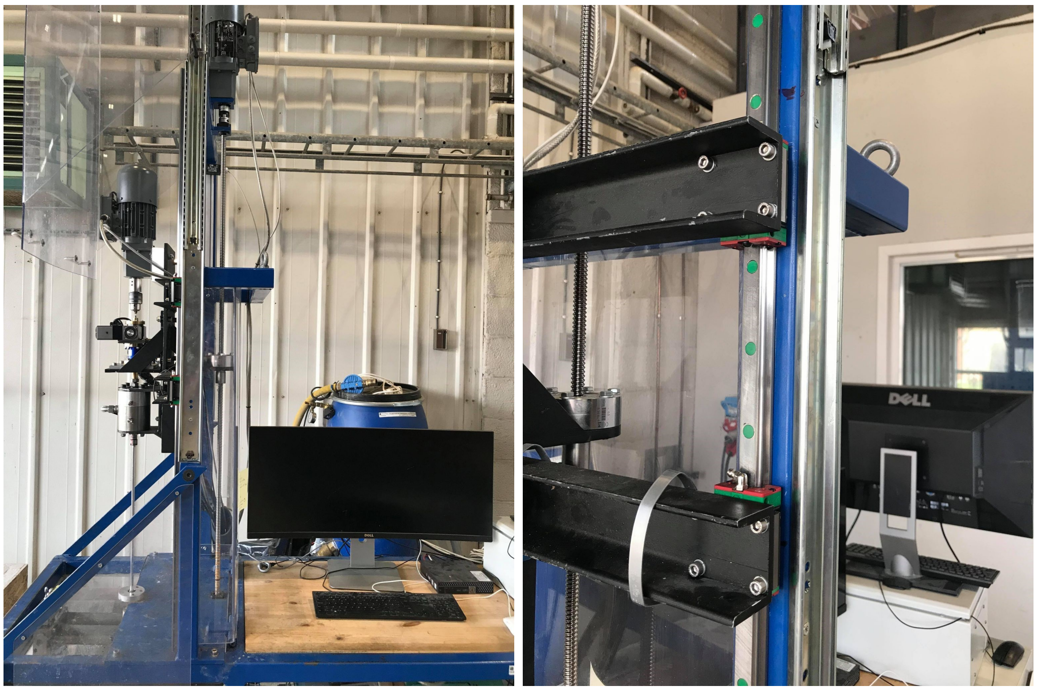

(a) Overview. (b) Rails for movement.

Figure 11: Hoisting system with motor, ball screw, and the rails holding and moving the slide in the vertical direction.

3.8 Bit Propulsion System

Torque is transferred from the top drive to the bit through a rod. The rod is attached to a drill chuck and goes through the azimuth control system, hydraulic swivel, drill pipe, and into the drive shaft in the BHA moving the bit.

3.8.1 Drill Chuck

A chuck typically used for mounting bits in power tools is used to connect the rod to the top drive. The drill chuck allows rods to be changed with minimal effort while having enough friction to transfer the torque.

3.8.2 Drill Pipe

The drill pipe is used as a casing for the rod and is rotated to steer the BHA inside the rock. Applied force is transferred from the swivel to the BHA through the drill pipe. The drill pipe is made of a material that allows bending without plastic deformation. Material options and length is explicitly stated in the Drillbotics competition guidelines. A friction connection intended to be used in the mounting of bits in a CNC-machine connects the BHA to the drill pipe and the drill pipe to the swivel.

3.8.3 Drill Rod

The drill rod transfers torque from the top drive to the bit. The rod must be flexible and strong to transfer the torque from the top drive while the drill pipe is bent from the drilling deviation. Titanium rods were used as they are very strong and flexible.

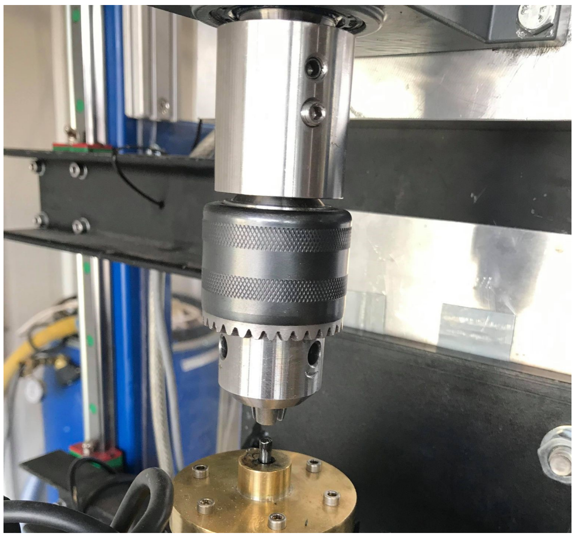


Figure 12: Drill chuck used to transfer torque from top drive to rod.

3.8.4 Bit

The bit is mounted with a threaded connection in the bit sub. Multiple bits have been tested and a custom bit has been designed in collaboration with Lyng drilling. Bits affect drilling performance, and the ideal bit for the steering concept has a predictable movement characteristics and minimal vibrations.

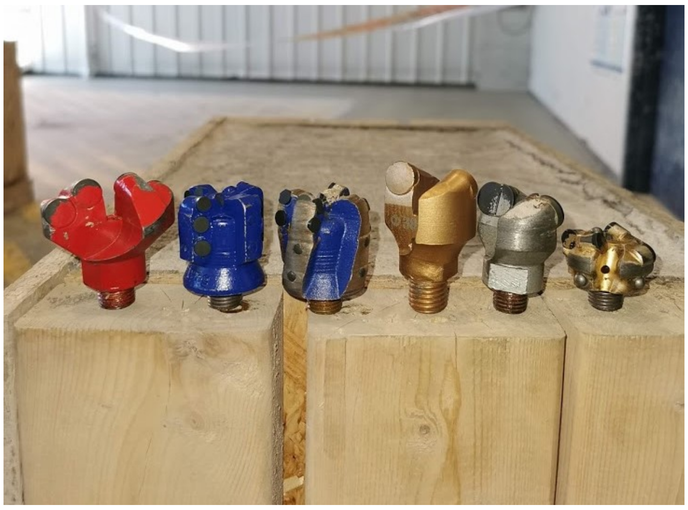


Figure 13: Different bit designs tested on the rig. The blue bits are designed by the student team and produced by Lyng Drilling.

4 Electrical System

The rig has a sophisticated electrical system for controlling the rig while taking measurements during operation. Most components are in an automation cabinet to protect from water and dust.

The rig has 3 servo motor inputs. The motors are used to move the hoisting system, rotate the bit, and orientating the BHA. All motors are used in closed-loop control and support one or multiple standardized automation protocols to be compatible with the software control system. In addition to servo motor measurements there are two sensors used for closed-loop control.

A load cell measures applied WOB to control the hoisting system, and a torque sensor is used to limit the maximum torque when orientating the BHA inside the rock.

4.1 Hoisting Motor

The hoisting motor is unchanged from previous years. The motor is of the type Lenze GST03-2M VBR 063C42, with a driver of the type Lenze E84AVTCE7512SX0. The driver has a maximum output of 0.75kW, maximum motor RPM of 3400, and a maximum torque of 45Nm[62]. A gearbox limits the maximum output rotational velocity to approximately 380RPM. A communication module of the type Lenze E84AYCEO enables EtherNet/IP external control. The motor controls the velocity of the hoisting system movement and applied WOB in closed-loop control with measurements from the load cell.

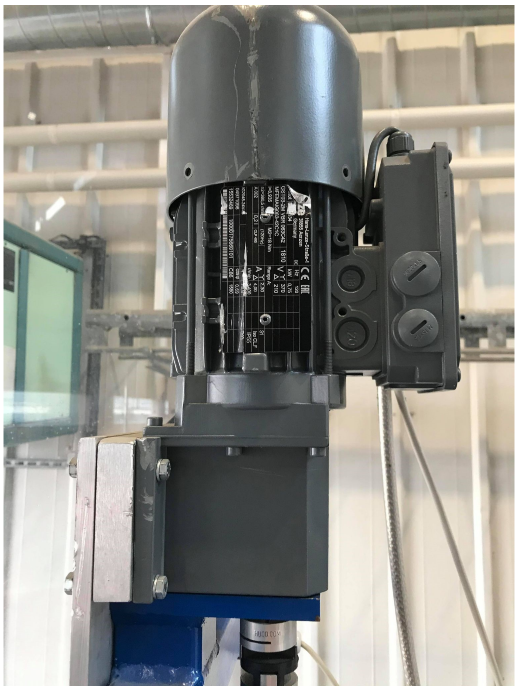


Figure 14: Hoisting motor connected to the ball screw.

A safety feature is upper and lower limit switches for hoisting movement. Switches are triggered if the hoisting system slide moves out of bounds, and the motor will not accept any further commands until the error is reset. A brake resistor is installed to dissipate heat when decelerating with the high inertia load of the hoisting system.

4.1.1 Lenze Engineer

Lenze provides software for configuring and controlling their drives. The software includes programs to update, configure, tune, and manually control Lenze devices. By selecting drive, motor, gearbox, and communication module, relevant parameters are configured automatically for the specific hardware configuration. The interface between the computer and driver is a diagnose adapter that connects to the computer using an USB port. Measured motor states and parameters can be monitored in real-time using the adapter, even if the driver is operated through a communication protocol.

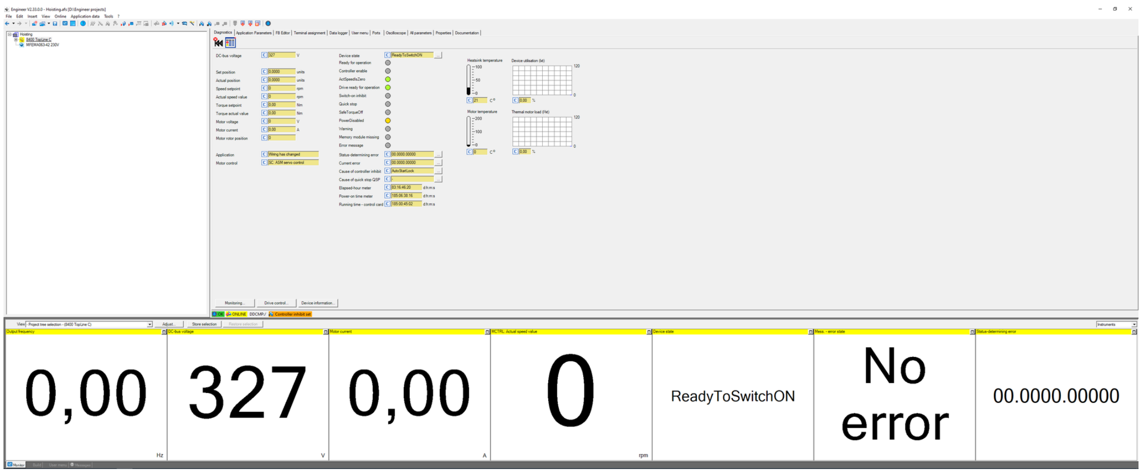


Figure 15: Lenze Engineer software overview.

The driver is programmable through a block programming interface called FB editor. EtherNet/IP communication and input logic can be configured in the editor. Communication inputs and outputs are fully customizable. External electrical signals to the driver can be used in the logic, such as the limit switches installed on the hoisting system.

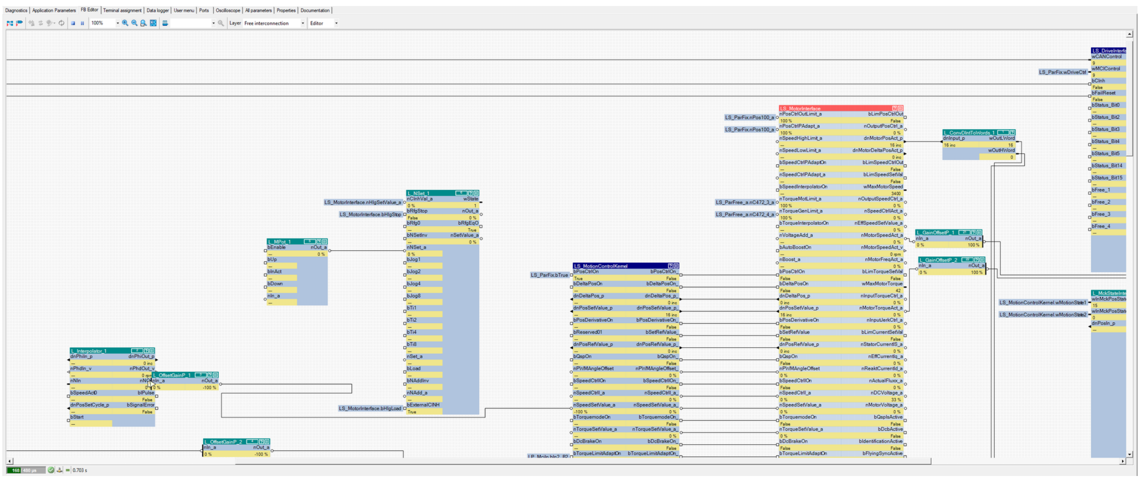


Figure 16: Block programming interface for the drive.

Motor tuning tools are available the Lenze Engineer software.

4.2 Top Drive Motor

The top drive motor has been replaced this year due to CANopen automation protocol limitations[67]. An equivalent motor and driver to the hoisting system was re-purposed from a previous research project. Communication and control of the hoisting motor had proved reliable, and the software integration was expected to be less time consuming than acquiring a different motor and drive, as much of the programming could be re-used from the hoisting system. The drive did not have a communication module and another Lenze E84AYCEO was purchased.

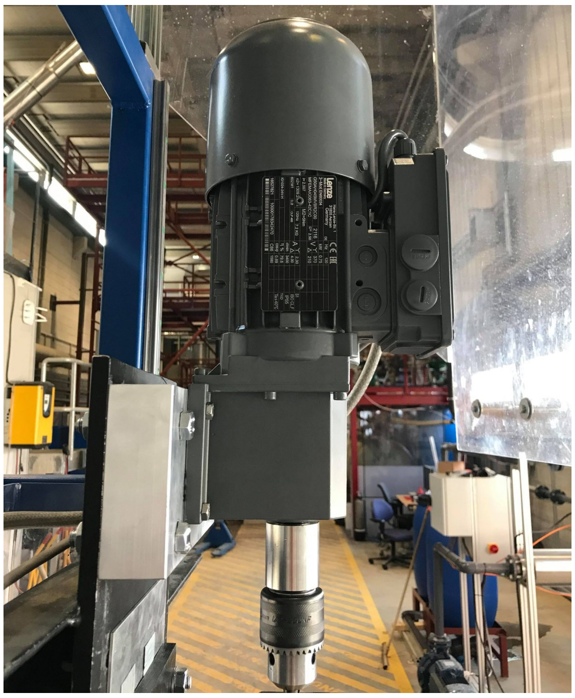


Figure 17: Top drive motor with drill chuck.

The motor is used to provide torque to the bit through a rod. The motor was not the ideal choice for top drive as it was optimized for high torque and high inertia loads. The maximum 380RPM output rotational velocity was expected to be sufficient but proved to be a limiting factor while drilling.

It was attempted to remove the gearbox from the motor to allow the rotation to operate at the 3400 motor RPM. When opening the gearbox, it was discovered that the output shaft from the motor was not suited for mounting the drill chuck without permanent damage. Gearboxes were not sold separately, and time limitations required a new top drive to be a drop-in replacement. A similar motor with a lower ratio gearbox of type Lenze G50AH045MVBR2C00 was ordered with a maximum output velocity of 1300RPM and 5Nm of torque.

There is no brake resistor as the inertia load is minimal. Lenze Engineer configuration of the hoisting and top drive is equal except for limit switches, brake resistor, gearbox parameters, and encoder resolution.

4.3 Azimuth Control Motor

Drill pipe orientation is controlled by a servo motor connected to a hollow shaft gearbox system. The motor and driver were purchased this year for the new mechanical concept. Components are made by MOONS' with a SM0402AE4-KCD-NNV motor and a M2DV-1D82IP driver. Motor specifications are a maximum output power of 0.1kW, maximum RPM of 3000, maximum rated torque of 0.32Nm, and a peak torque of 0.93Nm[65][66]. The specific components were chosen as they were compatible with the hollow shaft gearbox and could be ordered from and customized by the same manufacturer selling the hollow shaft gearbox. The drive supports both EtherNet/IP and Modbus TCP communication.

During initial testing it was discovered that the minimum rotation velocity when connecting the servo motor directly to the hollow shaft gearbox exceeded the rotation speed required to accurately control the rotation of the drill pipe. The lower limit was found to be around 6RPM yielding a minimum 8 deg/min output rotation velocity. Noises while operating suggested that the motor was struggling. Additional gearboxes were purchased to increase the torque and allow finer control.

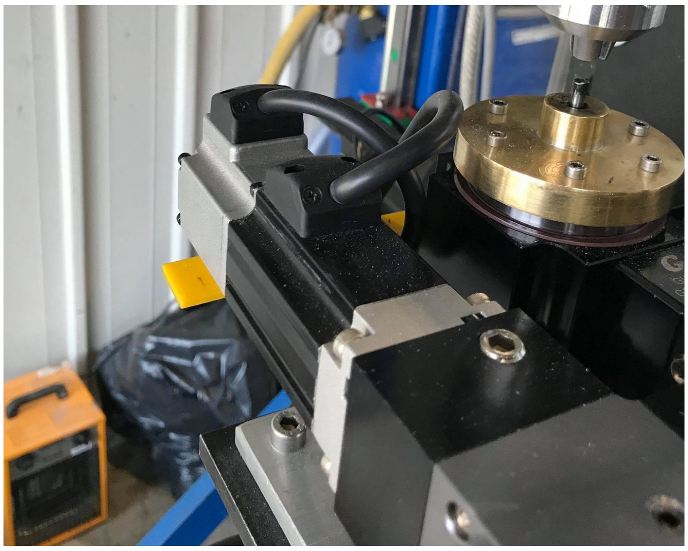
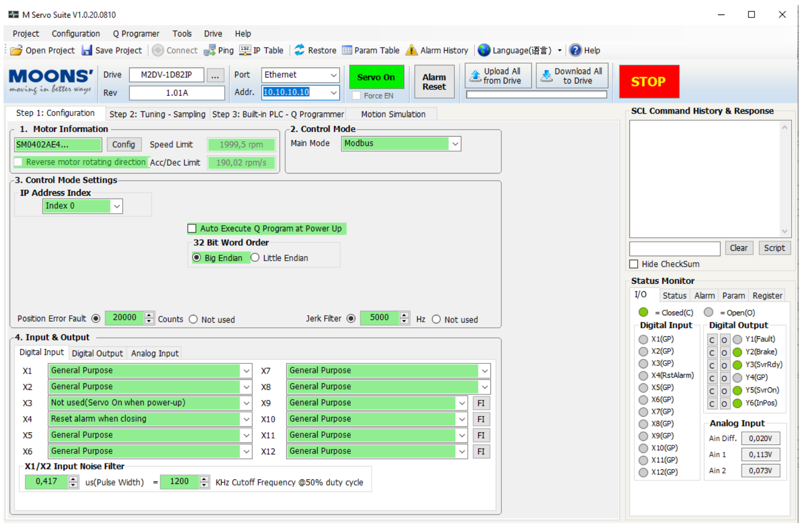


Figure 18: Azimuth motor connected to the gearbox system.

A more inexpensive stepper motor or hybrid stepper motor could have been used in the final design. A servo motor was chosen to have a torque estimate from the motor. The increased gear ratio required an external torque sensor for closed-loop control as measurements would be unreliable.

4.3.1 M Servo Suite

Drive configuring is done using the M Servo Suite software by MOONS'. By selecting the driver model and motor, parameters are automatically configured to match the hardware setup. The drive is configured through the same Ethernet port used for sending and receiving data using the communication protocol. The drive has dual Ethernet ports but only supports interaction from one Ethernet port at a time.



Drive: M2DV-1D82IP
Rev: 1.01A
Port: Ethernet
Addr.: 10.10.10.10
Force EN

1. Motor Information
SM0402AE4...
Speed Limit: 1999,5 rpm
Reverse motor rotating direction
Acc/Dec Limit: 190,02 rpm/s

2. Control Mode
Main Mode: Modbus

3. Control Mode Settings
IP Address Index: Index 0
Auto Execute Q Program at Power Up
32 Bit Word Order: [✓ Big Endian] Little Endian
Position Error Fault [✓] 20000 Counts / Not used
Jerk Filter [✓] 5000 Hz / Not used

4. Input & Output
Digital Input | Digital Output | Analog Input

X1	General Purpose	X7	General Purpose
X2	General Purpose	X8	General Purpose
X3	Not used(Servo On when power-up)	X9	General Purpose
X4	Reset alarm when closing	X10	General Purpose
X5	General Purpose	X11	General Purpose
X6	General Purpose	X12	General Purpose

X1/X2 Input Noise Filter
0,417 us(Pulse Width) = 1200 KHz Cutoff Frequency @50% duty cycle

SCL Command History & Response
Hide CheckSum

Status Monitor
I/O | Status | Alarm | Param | Register
● = Closed(C) ○ = Open(O)

Digital Input	Digital Output
○ X1(GP)	C O ○ Y1(Fault)
○ X2(GP)	C O ● Y2(Brake)
○ X3(GP)	C O ● Y3(SvrRdy)
○ X4(RstAlarm)	C O ○ Y4(GP)
○ X5(GP)	C O ● Y5(SvrOn)
○ X6(GP)	C O ● Y6(InPos)
○ X7(GP)	
○ X8(GP)	
○ X9(GP)	
○ X10(GP)	
○ X11(GP)	
○ X12(GP)	

Analog Input
Ain Diff. 0,020V
Ain 1 0,113V
Ain 2 0,073V

Figure 19: Interface of M Servo Suite.

Limitations can be set on velocity, acceleration, and position of the motor. Motor tuning features are available. M Servo Suite is less sophisticated than Lenze Engineer in terms of capabilities and customizing options.

4.4 Load Cell

A load sensor is placed between the hoisting ball screw and the hoisting slide to estimate applied WOB. The load cell is the TC4-AMP model made by APE Transducer. A built-in amplifier has options for voltage output in the range ±5V and ±10V, and current output of 4-20mA[2]. The sensor can measure force in the range ±2.5kN.

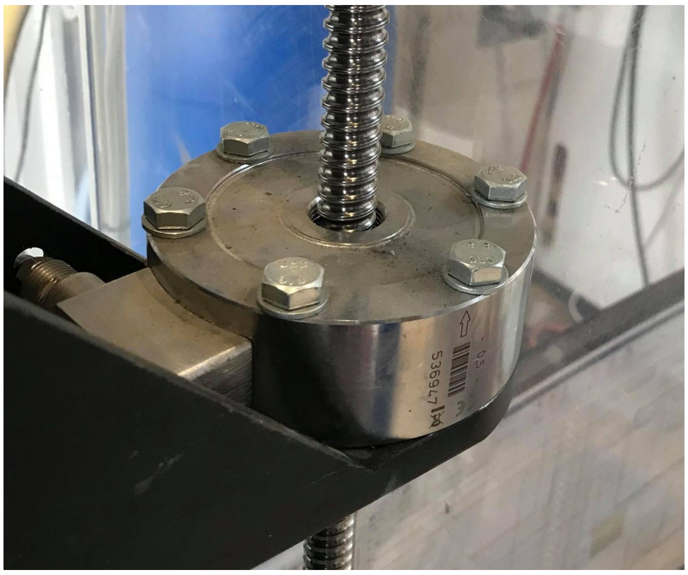


Figure 20: Load cell for estimating WOB.

Voltage output was chosen as it was simpler to implement with the DAQ. Current output would have been more robust for longer cable runs as there would be no voltage drop in the cable. Highly accurate measurements are not a strict requirement, and the cable length is relatively short.

WOB is estimated from the output voltage of the sensor, gravity, and the mass offset of components attached to the hoisting system slide.

$$WOB = V \cdot \frac{F_2 - F_1}{g(V_2 - V_1)} - m_{offset} \tag{1}$$

4.5 Gateway

EtherNet/IP is a widespread automation protocol, however not all control system software environments support the standard. Modbus TCP is an older protocol that has been the standard in many industries for a long time with great software support. Previous teams have used a gateway to translate a EtherNet/IP subnet to a Modbus TCP subnet for communication with the hoisting driver[67]. The gateway is of the type Hilscher NT 100-RE-EN and can be configured to control multiple devices simultaneously. The maximum poll rate of the gateway is 1000Hz[19].

4.5.1 SYSCON.net

Configuration of the gateway is done using the SYSCON.net software with a drag-and-drop block programming interface. The EDS file to a component can be imported and added as a node into the subnet configuration. The gateway block has an input and output subnet where nodes can be connected. The subnets are configured in the gateway block for the gateway where IP addresses and address mapping between the two protocols are set. If multiple devices are used, memory registers are mapped with an offset equal to the maximum number of memory addresses of the

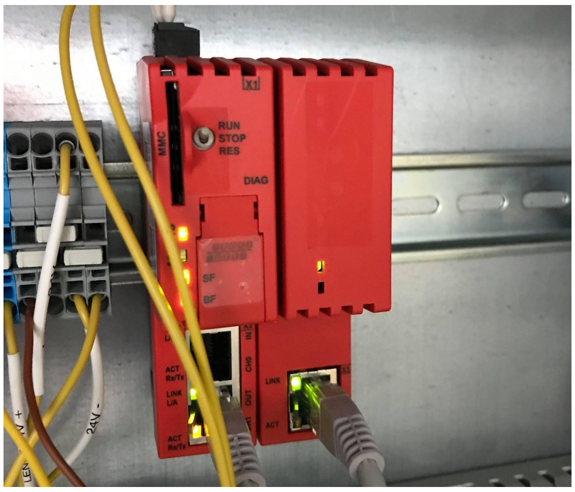


Figure 21: Automation protocol gateway.

subsequent component. Communication configuration changes or software updates in the drives will require the gateway to be reprogrammed to re-establish communication.

4.6 Network Switch

The control system computers have a single Ethernet port with no expansion slots. A network switch is used to allow multiple devices to be accessible without a direct connection. The switch accepts and redirects data to the intended recipient based on the MAC address of each device. NICs of the control system computers must be configured to a subnet that can access all devices in the network. The switch is unmanaged with few configuration options. A network switch will add some latency and overhead to communication between devices.

4.6.1 IP Addresses

All devices must be configured to a local IP address to be accessible on a local network. Drives typically comes with a set of default IP addresses that can be configured by a hardware toggle or through software. The Lenze drives can be easily configured to any local IP address, while the MOONS' drive has a limited set of possible addresses. While setting up the addresses for the network, it was still unknown how to change the default address of the MOONS' drive, it was later discovered that the index from the predefined table could be assigned in the M Servo Suite software. All IP addresses were therefore configured in the subnet of the default MOONS' address.

Table 2: Device IP Addresses.

Device	IP Address	Connected To
High Level Control Computer	10.10.10.4	Network Switch
Control System Computer	10.10.10.5	Network Switch
Gateway Modbus Subnet	10.10.10.8	Network Switch
Gateway Ethernet/IP Subnet	10.10.10.9	Top Drive Motor Driver
Azimuth Control Motor Driver	10.10.10.10	Network Switch
Top Drive Motor Driver	10.10.10.20	Gateway
Hoisting Motor Driver	10.10.10.30	Top Drive Motor Driver

EtherNet/IP supports daisy chaining of devices. Because the network switch did not have sufficient ports for all devices to be connected directly, the hoisting motor driver was connected to the top drive driver. This has the disadvantage of not allowing the top drive and hoisting driver to be pinged from the control system computers to check connectivity.

4.7 Torque Sensor

For reasons described in subsection 3.6, an external azimuth sensor was added to the system for closed-loop control. The sensor measures the static torque between a mechanical reference shelf and the static part of the azimuth control gearbox system. This will estimate the torque forces exerted on the drill pipe when rotating. The sensor is of the model TAT200 made by HT Sensor Technology and measures torque in the range ±30Nm and has an IP67 rating[70].

Calculating torque from the voltage reading is done by assuming a linear sensor response and extrapolating the ±10V output range to the maximum torque value.

$$\tau = -3 \cdot V_{Torque,Sensor} \tag{2}$$

The sign of the output is inverted for the torque value sign to corresponds to azimuth rotation direction. The sensor must be centred and aligned as close as possible to the T-shaft to avoid measurement irregularities when rotating.

4.7.1 Amplifier

There is no built-in amplifier in the torque sensor. The cylinder is a transducer with changing resistance based on the torque difference between the two sides. An amplifier power source is used to create a voltage output that can be measured by the DAQ. Voltage differences of a few mV caused by change in resistance in the transducer is amplified into the ±10V range. The amplifier is of the type SIC-A5 made by HT Sensor Technology and uses a 24V DC power source.

The amplifier is placed underneath the tabletop in a protective case as the wire was not long enough to reach the automation cabinet and there were concerns about electromagnetic interference affecting the measurements. Resistors in the amplifier is used to calibrate the zero offset and output gain.

4.8 DAQ

Some sensors on the rig outputs an analog voltage or current signal. Analog signals are sampled by a DAQ and converted into digital measurement values that can be utilized by the software control system. The DAQ is of the model USB-6212 made by National Instruments. Maximum resolution is 16-bit with a maximum sample rate of 400 kS/s[59]. Voltage signals can be read with a common ground or as differential inputs. The DAQ is connected to the low-level control system computer with an USB 2.0 interface.

4.9 Low-Level Control System Computer

The low-level control system software runs on a small desktop computer. A desktop computer is more convenient, easier to develop for, and more widely available compared to dedicated PLCs. The computer is of the model Dell Optiplex 7060 with a six core Intel Core i7 8700T processor and 32GB of RAM.

The computer is managed by the university and will restrict most files from being accessed between users. It is convenient that multiple team members can log in to the control system computer without sharing passwords. An additional internal SSD was therefore added to allow control system documents to be saved and accessed from all users on the computer.

There is a significant reduction in performance once the computer has been running with a high system load for a longer duration. Modern desktop CPUs adjusts their clock speed based on the temperature. The tiny case with limited airflow is not meant for high load continuous operation,

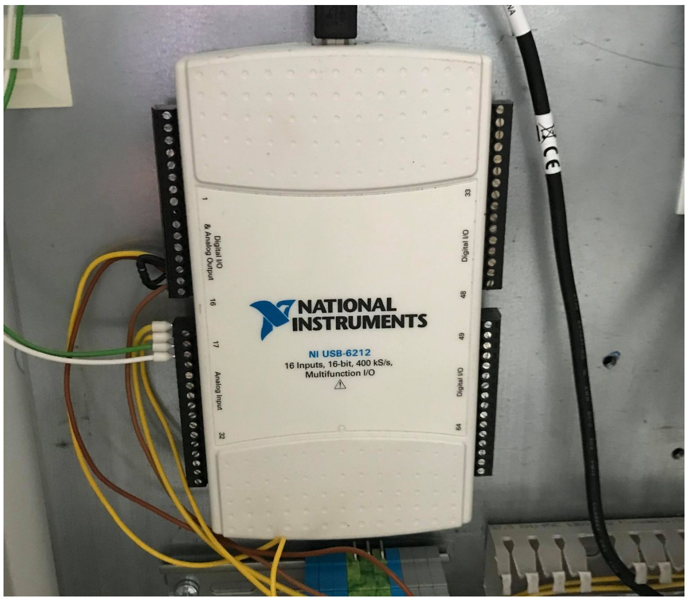

NATIONAL INSTRUMENTS
NI USB-6212
16 Inputs, 16-bit, 400 kS/s, Multifunction I/O
Digital I/O & Analog Output
Analog Input
Digital I/O

Figure 22: DAQ with connected inputs from load cell, torque sensor, and pressure transmitter.

and the temperature will increase and clock speed decrease until the computer reaches a temperature equilibrium. The temperature of the computer must therefore be considered when profiling the performance of the control system software.

4.10 High-Level Control System Computer

High computational cost required an additional on-hand computer to be used, as budget constraints did not allow a new more powerful computer to be purchased. The control system was separated into low-level and high-level systems. The control systems share data over UDP network streams. The computer is of type Dell Optiplex 7080 with a six core Intel Core i5 10500T and 16GB of RAM.

4.11 External Bluetooth Card

A Class 1 Bluetooth adapter from Laird was acquired to give Bluetooth communication the best possible chance of maintaining a stable connection to the downhole sensor while inside the rock. Class 1 adapters have the most powerful sender and receiver that the Bluetooth standard permits with up to 100mW maximum power and a typical range of up to 100m[75]. The adapter is connected to the high-level control computer with an USB extender cable to be placed as close as possible to the rock.

4.12 Power Distribution

Power to the rig is provided by a 1-phase 230V input. The automation cabinet has support for 3-phase 400V as this has previously been a requirement. The load cell and torque sensor amplifier are powered by a 24V DC power supply, converted from 230V AC. The DAQ and downhole sensor is powered by the 5V DC USB cable from the control system computers.

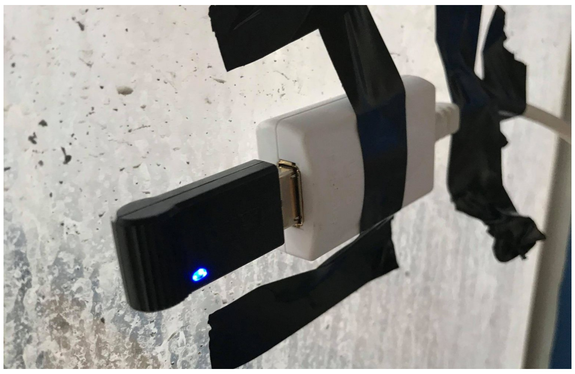


Figure 23: Bluetooth sender and receiver connected to an USB extender.

4.12.1 Emergency Stop

There is an emergency stop button installed on the automation cabinet. The switch breaks the power circuit to the top drive, hoisting motor, and azimuth motor.

Figure 24: Emergency stop button on the automation cabinet.

5 MATLAB and Simulink

The possibility of changing software environment from LabVIEW to Simulink was investigated during the planning phase of the project[13]. The change was motivated by new Drillbotics competition requirements with increased focus on simulations and high-level control. Completing this year's competition objectives would be difficult without having the possibility for fast prototyping, tuning, and debugging of advanced controllers in a simulation environment before testing them on the physical rig.

5.1 LabVIEW

LabVIEW is a system engineering software for measurements and control systems[58]. LabVIEW uses a graphical programming interface to communicate with hardware components and processing measurement data. The software has great hardware support, especially for hardware components made by National Instruments.

Recent teams have used LabVIEW to control the rig. Main motivation for using LabVIEW was great hardware support[67]. Competitions requirements from previous years could be solved with a basic low-level control system controlling WOB and a state machine implementation. Hardware integration was the main challenge for the control system and LabVIEW did generally have better hardware support than Simulink.

LabVIEW supports some advanced modelled controllers like MPC but has very limited simulation capabilities and documentation compared to Simulink. Converting controllers tested in Simulink to LabVIEW would be time consuming and prone to bugs.

5.1.1 Hybrid Solution

The advantages of Simulink are mainly for high-level control. A possibility was to create a low-level control system in LabVIEW and stream data to Simulink for high-level control. More code could then be reused from previous while having advanced simulation and controller capabilities of Simulink.

This was the backup plan if it was not possible to implement low-level control in Simulink. Using two different software environments adds complexity to the system and streaming data introduces more potential points of failure. Many of the hardware components had to be replaced for the new concept, making the time-save for reusing code less significant.

5.2 MATLAB Overview

MATLAB is a computing environment developed by MathWorks. The programming language was designed for numerical computing, matrix manipulation, plotting, and algorithm development[35]. MATLAB can also be used for control systems, machine learning, robotics, and measurements. The language is well documented with active troubleshooting forums.

MATLAB functionality can be expanded using toolboxes. Commonly used toolboxes include expansions for control system, optimization, signal processing, and machine learning. Cybernetics students at NTNU are familiar with MATLAB as the recommended programming language for multiple courses during their studies. MATLAB 2020a was used as it was the latest stable version at the start of the project.

5.3 Simulink Overview

Simulink is an addition to MATLAB designed for modelling and simulating systems in a block diagram interface[47]. Like MATLAB, toolboxes are available for extended functionality. MATLAB Function blocks allow MATLAB code to be used as a part of Simulink simulations and variables can be accessed directly from the MATLAB workspace. Code generation allows a selection of blocks and MATLAB functions to automatically generate efficient C, C++, or HDL code from a model.

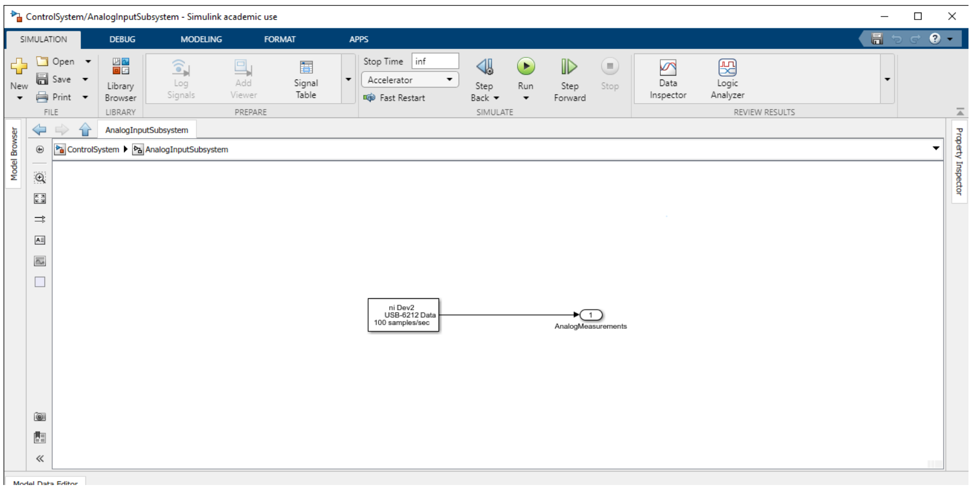


Figure 25: Overview of Simulink model GUI.

5.4 Simulation Configurations

Simulink can simulate both continuous and discrete systems. Simulink will automatically suggest a solver based on identified system dynamics. Solver selection can be overridden by the user. Other configuration options can be used to improve performance, debugging, and set guidelines for how blocks interact with each other.

5.4.1 Continuous Solver Setup

System dynamics simulations requires a continuous time solver. The solver has a variable step size as this is generally more efficient and accurate than using a fixed step solver. The choice of solver is discussed in subsection 9.4.

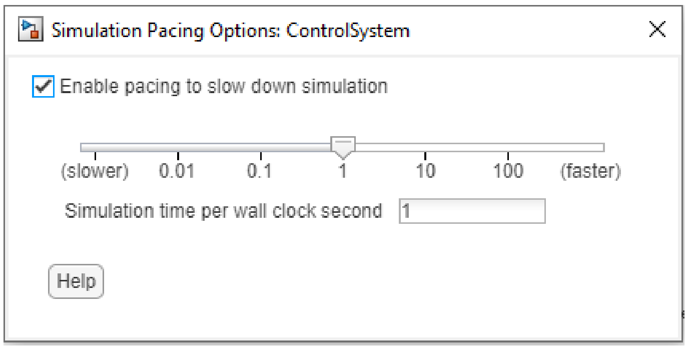
5.4.2 Control System Solver Setup

Control system measurements are sampled at discrete time-intervals. Because the control system is not trying to do any state simulations, the solver can be set to discrete with no continuous states.

5.4.3 Simulation Pacing

A Simulink model will by default execute as fast as possible. Any external hardware may behave differently depending on the model execution time. Execution time can be slowed down to not run faster than real time using simulation pacing.

The Analog Input block for the DAQ will slow down simulation time as described in subsection 7.1.



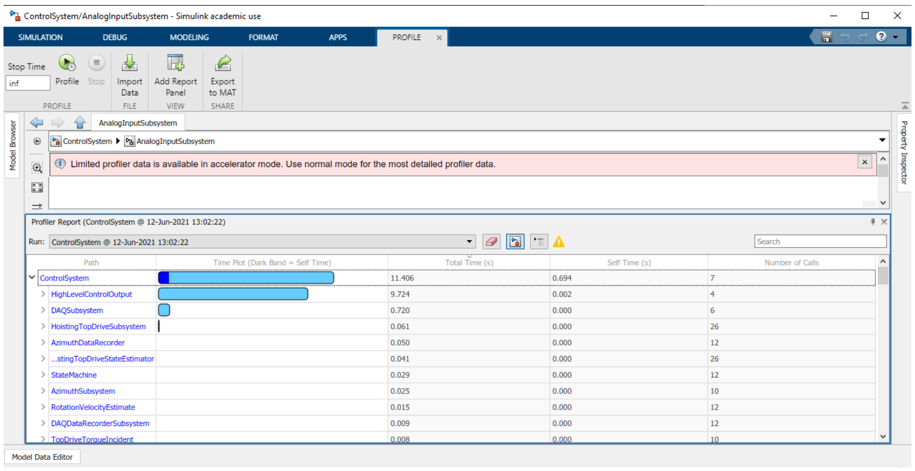
[✓] Enable pacing to slow down simulation

Simulation time per wall clock second: 1

Figure 26: Simulation pacing configuration options.

5.5 Performance Profiling

Simulink has debug features to measure block execution time. Execution time of blocks is used to identify blocks that should be optimized or have a reduced sample rate. The control systems have limited computational resources, and resources must be prioritized for the best overall control performance without overloading the system.



Stop Time: inf

Limited profiler data is available in accelerator mode. Use normal mode for the most detailed profiler data.

Profiler Report (ControlSystem @ 12-Jun-2021 13:02:22)

Run: ControlSystem @ 12-Jun-2021 13:02:22

Path	Time Plot (Dark Band = Self Time)	Total Time (s)	Self Time (s)	Number of Calls
ControlSystem		11.406	0.694	7
HighLevelControlOutput		9.724	0.002	4
DAQSubsystem		0.720	0.000	6
HoistingTopDriveSubsystem		0.061	0.000	26
AzimuthDataRecorder		0.050	0.000	12
...stingTopDriveStateEstimator		0.041	0.000	26
StateMachine		0.029	0.000	12
AzimuthSubsystem		0.025	0.000	10
RotationVelocityEstimate		0.015	0.000	12
DAQDataRecorderSubsystem		0.009	0.000	12
TopDriveTorqueIncident		0.008	0.000	10

Figure 27: Simulation Profiler interface.

Profiles of the system have typically been executed for 60 seconds to reduce the effects of differences in initialization times between runs while being short enough for fast paced testing. Thermal throttling of the control system computers must be considered as explained in subsection 4.9.

It has been observed that some features like the GUI can affect the execution time of unrelated blocks in the model. Blocks limiting the model execution time will be shown in the profiler to use more than their minimum required execution time.

5.6 Code Generation

Some MATLAB functions supports code generation allowing MATLAB Function blocks and Simulink blocks to be compiled an executed as efficient C or C++ code. Functions will otherwise be executed by the MATLAB interpreter. The model must be set to Accelerated execution time to use compiled code[25].

The default MATLAB compiler is LCC. Some MATLAB functions like Modbus caused issues while compiling, and the compiler is changed to the MathWorks recommended MinGW-w64 compiler.

5.7 Concurrent Execution

Single core performance is only a fraction of the potential performance of a modern desktop CPU. Utilizing multiple CPU cores have the potential of increasing the maximum throughput of the control system, allowing a higher computational load for each time-step, and add scalability to the control system. A multicore control system requires special design considerations to ensure consistent states when passing data between threads using mutexes and semaphores[31]. Simulink requires simulations that should run concurrently to be organized into independently executed blocks called Atomic Subsystems.

5.7.1 Atomic Subsystems

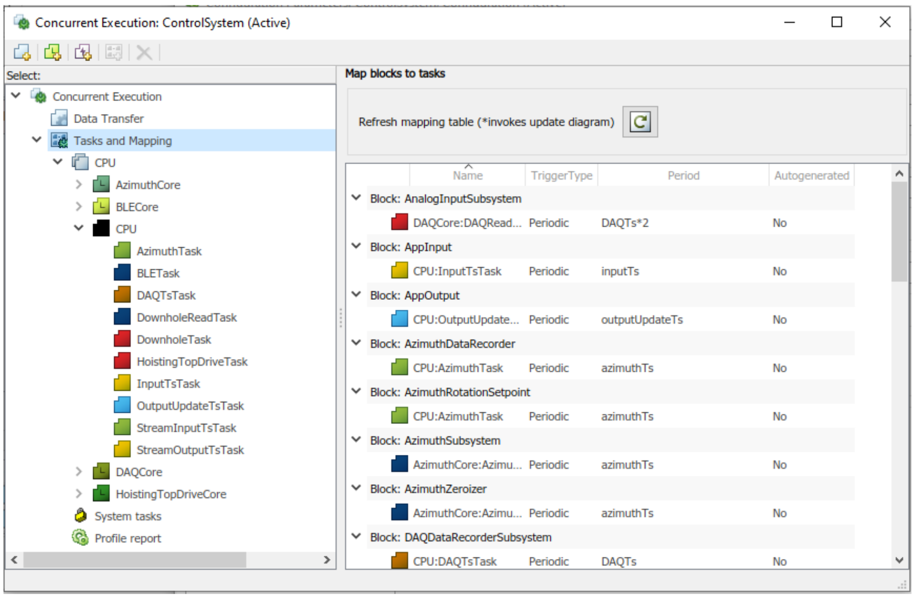
Atomic Subsystems are independently executed subsystems in Simulink. A subsystem is a grouping of multiple blocks with only the inputs and output exposed to the rest of the model. Subsystems are useful for feature abstraction and maintaining an easy overview for large simulations[52]. Atomic subsystems are used in discrete time systems with a fixed sampling rate.

Concurrent execution requires all inputs and outputs of a subsystems to have a fixed and equal sampling rate. Atomic subsystems are more complex to implement but allows utilization of multiple CPU cores.

5.7.2 Configuration

Concurrent execution can be enabled in the solver settings of the model. Simulink requires all blocks to be inside an atomic subsystem before a model can be run concurrently.

Enabling concurrent execution gives access to a task configuration tool. Tasks are assigned periodic or aperiodic triggers. Having multiple tasks will make use of multiple CPU cores on a multicore target device. Period tasks are assigned a sample time that must correspond to the sample time of the atomic subsystem assigned to it. All periodic triggers within a task must be a multiple of the smallest assigned sample time.


Concurrent Execution: ControlSystem (Active)

Select:

- Concurrent Execution
  - Data Transfer
  - Tasks and Mapping
    - CPU
      - AzimuthCore
      - BLECore
      - CPU
        - AzimuthTask
        - BLETask
        - DAQTsTask
        - DownholeReadTask
        - DownholeTask
        - HoistingTopDriveTask
        - InputTsTask
        - OutputUpdateTsTask
        - StreamInputTsTask
        - StreamOutputTsTask
      - DAQCore
      - HoistingTopDriveCore
    - System tasks
    - Profile report

Map blocks to tasks

Refresh mapping table (*invokes update diagram)

Name	TriggerType	Period	Autogenerated
Block: AnalogInputSubsystem			
DAQCore:DAQRead...	Periodic	DAQTs*2	No
Block: AppInput			
CPU:InputTsTask	Periodic	inputTs	No
Block: AppOutput			
CPU:OutputUpdate...	Periodic	outputUpdateTs	No
Block: AzimuthDataRecorder			
CPU:AzimuthTask	Periodic	azimuthTs	No
Block: AzimuthRotationSetpoint			
CPU:AzimuthTask	Periodic	azimuthTs	No
Block: AzimuthSubsystem			
AzimuthCore:Azimu...	Periodic	azimuthTs	No
Block: AzimuthZeroizer			
AzimuthCore:Azimu...	Periodic	azimuthTs	No
Block: DAQDataRecorderSubsystem			
CPU:DAQTsTask	Periodic	DAQTs	No

Figure 28: Simulink Concurrent Execution configuration tool.

Assigned triggers are indicated in the Simulink model with a small icon above the subsystem. A subsystem may be assigned to multiple triggers if a subsystem is directly connected to other atomic subsystems assigned to different tasks.

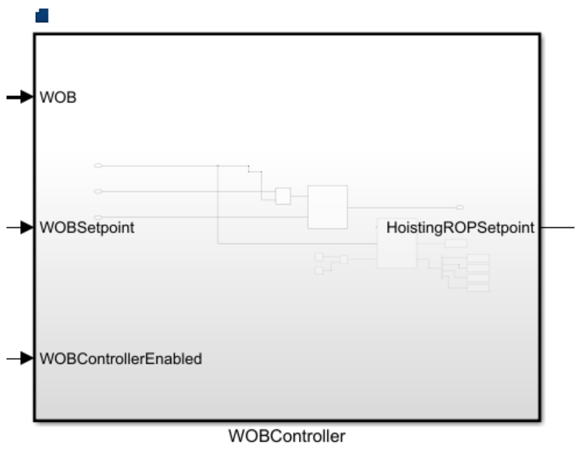


Figure 29: Atomic Subsystem assigned to a periodic task.

5.7.3 Data Transfer

Atomic subsystems are set to different sampling rates to make best use of available computational resources. Transferring data between blocks with different sampling frequencies requires a Rate Transition block as inputs and outputs of atomic subsystems in Simulink must have the same sample rate. Sample rates of the system is set such that they are a multiple of each other to ensure deterministic transfers of data[46].

If the sample rate of the transition block is set to output more often than the input, the same input will be outputted multiple times to the subsequent subsystem with a higher sampling rate. Down conversion will buffer higher frequency inputs and output the latest value on request from the lower sample rate block. Operation mode is indicated in the upper left corner of the Rate Transition block.

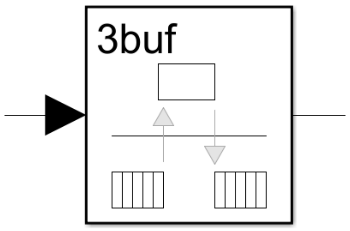


Figure 30: Rate Transition block buffering inputs to downsample a higher frequency input signal.

Data transmissions between task may require a unit step delay for synchronization. Tasks should be assigned such that unit delays are avoided for latency sensitive functionality.

5.8 Simulink Real-Time Support

It is possible to use Simulink for hard real-time applications. There are two toolboxes with real-time functionality: Simulink Desktop Real-Time and Simulink Real-Time.

A real-time system is more predictable as the worst-case execution time is known, ensuring a consistent time-steps and that computational deadlines for incident reaction are always met. Both toolboxes have features for analysing time-steps and using threads that will not be interrupted by the general purpose OS.

5.8.1 Simulink Desktop Real-Time

Simulink Desktop Real-Time is designed to run real-time systems on Windows or macOS computers. An independent real-time kernel is used to prevent interrupts from the general purpose OS. The toolbox has support for some selected hardware components.

The toolbox is not used in the system because of hardware support. To use the toolbox the DAQ would have to be replaced with a supported device[30]. Most of the real-time compatible DAQs uses the variation of the PCI interface. The current low-level control system computer does not have room for internal expansion cards and would likely have to be replaced.

Desktop Real-Time does not support Modbus TCP communication natively. There is real-time support for TCP communication and it would be possible to manually create an implementation of the low-complexity Modbus TCP standard for real-time communication. No attempt at manually implementing the standard was made as it was assumed too time consuming. Using a code generated MATLAB Function block for Modbus communication is a possibility. Tests with MATLAB Function blocks with Modbus TCP communication were inconclusive as the block did not compile with the standard compiler configuration and no further attempts were made.

Bluetooth Low Energy (BLE) has no native real-time support. It would be possible to run an instance for BLE in a different programming environment and pass data to Simulink using a real-time capable communication interface like CAN. A more practical alternative would be to use serial communication over a wired connection to the downhole sensor. Serial communication is supported by Simulink Desktop Real-Time[49].

The current GUI interface is not compatible with Simulink Desktop Real-Time and would have to be altered. Examples of real-time compatible GUI implementations are available[27].

Simulink Desktop Real-Time integration could be attempted if the DAQ was replaced. Budget and time constraints meant that no further testing was done, and further investigation left for future work. It should also be considered if a soft real-time system is sufficient based on further analysis of required sampling rates and response times.

5.8.2 Simulink Real-Time

Simulink have a toolbox for non-desktop real-time systems. The implementation relies on target computers and real-time DAQs from speedgoat. Simulink I/O blocks with native support for Modbus TCP and EtherNet/IP are available[50].

Using this system would have been the optimal choice in terms of hardware integration and hard real-time support. The toolbox would require the control system computer, DAQ, and an I/O module for serial communication would have to be acquired from speedgoat. A quote was not requested as it was expected to cost more than the entire budget.

5.9 Operating System

Windows 10 is a general purpose OS used by both control system computers. Using Windows for a real-time system is not ideal as unpredictable OS interrupts may be triggered at any time, potentially affecting time-steps and latency critical functionality. Some frequently accessed programs like Lenze Engineer, M Servo Suite and SYSCON.net only supports Windows. Having the same computer as both target and host is practical when developing software.

Thread management is handled by the OS. Operating systems have different policies for prioritizing threads and gives different level of access to programs that is not a part of the OS. Compared to hard real-time operating systems, Windows does not have support to run a thread that cannot be interrupted by any other tasks including the OS. A thread assigned to the maximum available priority level in Windows may therefore be affected by OS threads. Interrupts cause unpredictable execution times for tasks.

Real-time optimized Linux was considered to allow the control system to run with minimal and shorter OS interrupts at a higher thread priority. Restrictions on university managed computers, added complexity of the system, software compatibility concerns, and uncertainty if subsequent teams would be sufficiently familiar with Linux meant that this was not prioritized.

5.9.1 Real-Time Kernel

Simulink Desktop Real-Time has support for a dedicated real-time kernel. By running threads outside the Windows kernel, they will not be interrupted by the OS. Hard real-time performance would then be possible while running the simulation from Windows.

The Real-Time kernel requires the entire model to be compatible with Simulink Desktop Real-Time[32]. Changes to hardware and software are then required.

5.10 GUI

MATLAB has an App Designer tool to create GUIs. GUIs are created in an intuitive drag-and-drop interface in the Design View, while logic with MATLAB syntax is implemented in Code View.

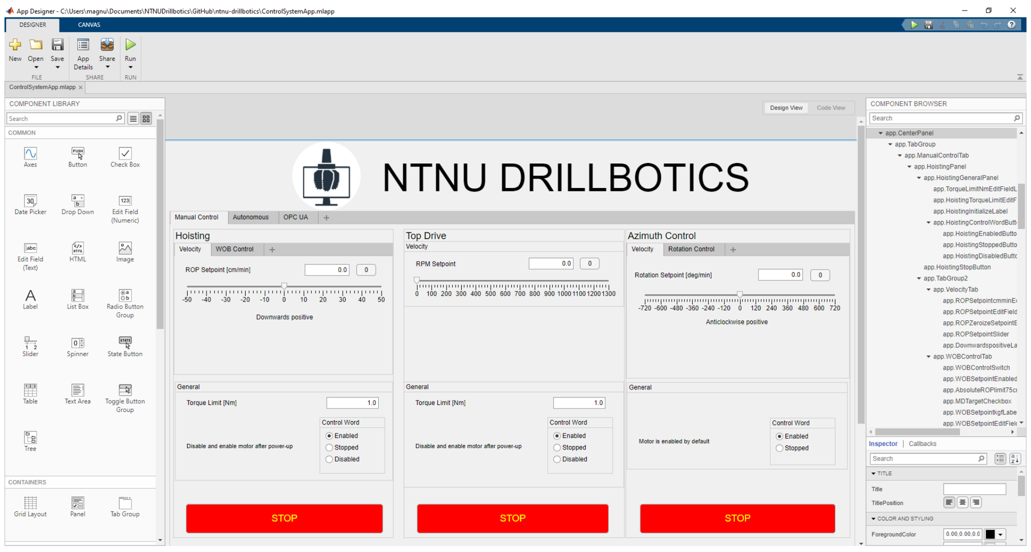


Figure 31: MATLAB App Designer Overview.

Components added in the Design View can be connected to handlers triggering function calls when the state of a component is changed. Logic in the Code View can be used to interface with a Simulink model.

5.10.1 Simulink Interface

An App Designer GUI can interface with a Simulink model using MATLAB commands. Constants in Simulink can be edited, and Display block values read while the simulation is running. A constant is changed with the `set_param` command, and displays can be read with the `get_param` command.

```
% Write to constant in the subsystem AppInput called InputConstant
set_param([bdroot,'/AppInput/InputConstant'],'Value', num2str(inputValue));
...
% Read from display in subsystem AppOutput called OutputDisplay
rto = get_param([bdroot,'/AppOutput/OutputDisplay'],'RuntimeObject');
```

`bdroot` returns the top level model of the current Simulink system[22]. Inputs are only accepted if they are converted to a string before being set to the constant. Value is the parameter to be edited in the constant block. A `RuntimeObject` is required to access the display values while a simulation is running[20].

Runtime objects are computationally expensive. Comparisons using the Simulink Profiler suggested that it was less computationally expensive to use a single display block with multiple values and a single runtime object rather than multiple display blocks with their own runtime object.

All input constants are in the same Atomic Subsystem for simpler overview of input logic. The input block subsystem is polled very frequently to reduce user input latency.

5.10.2 Input Data Update

Some App Designer GUI components can take inputs from the user. These components have callback functions that are triggered when a component event occurs. Components used in the GUI for user input values are buttons, edit fields, switches, and sliders. Input values from the components can be passed to the event functions.

The Simulink model is controlled from the App Designer GUI by altering constant blocks in the Simulink model connected to the rest of the blocks in the model. Constant blocks will only accept string values of a numeric input.

```
% DummyConstant is the constant bock and AppInput the subsystem
set_param([bdroot,'/AppInput/DummyConstant'],'Value', num2str(dummyValue));
```

Constant blocks will retain their values after the Simulink model is stopped. This must be carefully considered when developing new features to avoid unexpected behaviour when the model is restarted.

5.10.3 Output Data Update

The GUI polls data from output Display blocks when the `PostOutput` event is triggered. The app is initialized in the `InitFcn` callback of the model.

```
% DummyDesignerApp is the name for the app file without file extension
hApp = DummyDesignerApp;
```

The trigger is connected to a public GUI function in the model `StartFcn` event.

```
% Initialize post output trigger for output display
% AppOutput is a subsystem containing the output display
set(0,'ShowHiddenHandles','on');
blk = 'DummyDesignerApp/AppOutput/OutputDisplay';
event = 'PostOutputs';
listener = @(app, event) updateGUI(hApp);
h = add_exec_event_listener(blk, event, listener);
```

Event based polling ensures minimal latency from new model value output and GUI update for a particular poll rate.

5.11 Plotting

Plotting commands are equal to plot commands used in MATLAB scripts. Plots are displayed in the `UIAxes` component with support for 2D and 3D plots. Plot type is determined by the first plot command. Initialization commands are called in the GUI initialized callback. Plots are defined as local objects that can be edited later.

```
% 2D plot initialization by linking the GUI element to object
% b specifies the color of the plot and handle visibility disables legend
app.plotObjectLine = plot(app.UIAxesObject, nan, nan, 'b',
    'HandleVisibility', 'off');
% 3D plot initialization
app.plot3ObjectLine = plot3(app.UIAxesObject, nan, nan, nan, 'b',
    'HandleVisibility', 'off');
```

Plotting in App Designer is highly resource demanding and plot count and update frequency must be kept to a minimum. Using the Simulink Profiler it was discovered that resizing the plot, editing axes, and recreating the plot object had a significant computational cost if done at every GUI update cycle. Calling these functions once or during initialization reduced the plotting execution time multiple orders of magnitude.

Once the plot is initialized it can be updated by changing the axis data in the plot object. Data in the plot object is retained from last iteration and does not need to be updated if the data is equal as for the last update.

```
% Update the data in plot object
app.plotObjectLine.XData = dummyXData;
app.plotObjectLine.YData = dummyYData;
% If 3D plot
app.plotObjectLine.ZData = dummyZData;
```

Finding more efficient methods of updating plots is left for future work.

5.12 App Designer MATLAB Compiler

One possible solution to reduce computational cost of the GUI is to use the App Designer MATLAB Compiler[36]. The compiler creates a standalone application relying on compiled rather than interpreted code execution. There is also a Simulink compiler to create executable programs from models[48]. Time restrictions limited testing and research into how the app and Simulink could communicate if one or both are compiled.

5.13 Simulink Stateflow

To simplify the implementation of state space machines, Simulink has a graphical programming tool called Stateflow. Stateflow allows decision logic to be programmed as a flowchart of a finite number of states. Parameters and logic are specified with MATLAB syntax. Stateflow is further described in subsection 11.1.

6 Downhole Sensor

Downhole sensor measurements used for closed-loop control have been a requirement since the 2018/2019 Drillbotics competition. Most teams have attempted to solve this by placing a small microcontroller and an IMU sensor inside the BHA.

6.1 Previous Years

The 2018/2019 NTNU Drillbotics team designed a custom sensor package with a microcontroller and IMU sensor[67]. The required bit diameter was then 1.25 inches, limiting off-the-shelf options. The custom design allowed for a tiny PCB with an ICM-20948 IMU sensor from TDK and EMF32 Gecko (EFM32G210F128-QFN32) microcontroller from Silicon Labs.

6.1.1 Software Setup

This year's team intended to use the custom sensor package designed by the 2018/2019 team. The thesis specified that Simplicity Studio 4 was used to program the microcontroller. The microcontroller requires a specific combination of core tools version, SDK, compiler, and sensor driver for code to be compiled. Software versions were not documented in the thesis and the latest version of the tools did not compile code for the device.

The primary issue was a pin configuration tool for communication between the microcontroller and sensor. The latest version of the tool failed to open the pin configuration file. An older version of the tool allowed the file to be opened but failed to autogenerate support files as intended. The tool was deprecated after the release of Simplicity Studio 5. The engineer who assisted in the development of the sensor package was no longer working at the department. Multiple combinations of core tools, SDK, compiler, and driver versions were tested without resolving the issue.

With enough effort it would probably be possible to use the old sensor package. After multiple days of trying to compile basic programs for the device, other options were considered. The 2020/2021 Drillbotics competition has increased the bit size to 1.5 inches, making some off-the-shelf components viable.

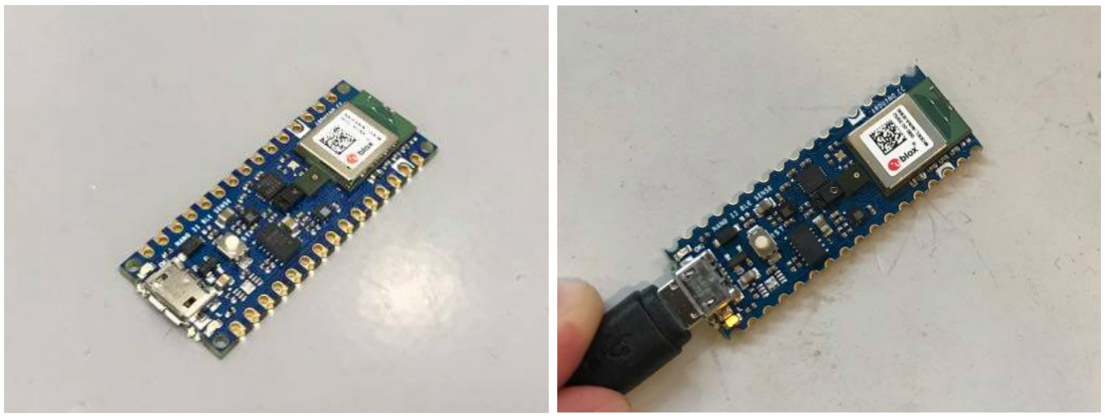
6.2 Arduino Nano 33 BLE Sense

Limited technical personnel and PCB design competence within the student team required a new sensor package to be made from off-the-shelf components. A new downhole sensor package would have to fit inside the BHA, preferably have an IMU sampling rate of approximately 100Hz, and an interface to transmit data to surface.

Having the sensor and microcontroller on the same PCB was preferred. Tiny wires between a separate microcontroller and sensor PCB would be difficult to solder and exposed to electromagnetic noise.

IMU sensors often connected to a microcontroller in a master-slave configuration using a Inter-Integrated Circuit (I2C) or Serial Peripheral Interface (SPI) interface. I2C and SPI communication is not robust enough for the sensor to be placed downhole and the microcontroller on surface.

A widely available, inexpensive, and easy to work with microcontroller is an Arduino. Their smallest microcontroller is the Arduino Nano 33 BLE Sense. Conveniently, this model has a built-in IMU sensor. The Arduino has a width of 18mm and a length of 45mm. The width exceeds the maximum of 16mm that is possible to fit inside the sensor housing of the BHA. It was observed that the PCB had soldering points that potentially could be removed to reduce the width of the PCB.



(a) Unmodified. (b) Modified.

Figure 32: Unmodified and modified Arduino Nano 33 BLE Sense.

6.3 PCB Modification

An Arduino was acquired to test if the microcontroller and IMU sensor would be functional if the soldering points were trimmed. The schematics drawings showed no connectors except for soldering points being located within 14mm from the centre of the PCB. The electronics lab at NTNU was consulted and they recommended using a circular saw to remove the sides and use sandpaper for finer alterations.

Lab personnel was able to reduce the width of the Arduino to 15.5mm with the microcontroller and IMU sensor still being functional. The width of the saw blade made it difficult to remove more of the PCB without risking damage to the personnel cutting the tiny Arduino. A successful modification indicated that the Arduino was a viable option for a downhole sensor package.

Using a circular saw was not optimal as it required lab personnel to have fingers very close to the saw blade to make a precise cut. A tool in the local workshop designed to remove straight sections of metal and plastic plates was tested. A sharp blade is moved by a manual lever applying pressure to a small cross section. The edge cutter proved more safe, accessible, and precise than the circular saw.

6.3.1 Sensors and Components

Arduino microcontrollers are made from off-the-shelf sensors and components. The microcontroller is the model nRF52840 by Nordic Semiconductors[4].

Table 3: Arduino Nano 33 BLE Sense Sensors.

Name	**Type**	**Sample Rate [Max]**	**Unit**	**Range**
LSM9DS1	Accelerometer	119Hz[74]	g	$\pm2/\pm4/\pm8/\pm16$
LSM9DS1	Magnetometer	119Hz[74]	gauss	$\pm4/\pm8/\pm12/\pm16$
LSM9DS1	Gyroscope	119Hz[74]	dps	$\pm245/\pm500/\pm2000$
MP34DT05	Microphone	62.5kHz[1]	-	0/1
APDS9960	Gesture, Light, Proximity	-	-	-
LPS22HB	Barometric Pressure	75Hz[73]	hPa	260-1260
HTS221	Temperature	12.5Hz[72]	°C	-40-120
HTS221	Humidity	12.5Hz[72]	rH	0-100

6.4 Communication

The Arduino has two built-in interfaces that could be used to transfer data to surface: USB serial and Bluetooth Low Energy (BLE). Concerns about wireless transfer reliability meant that the initial plan was to use serial communication.

USB is a common interface found on both microcontrollers and computers. USB can transfer data up to 10 meter under ideal conditions. USB 2.0 has 4 conductors, two of which transfer data in half-duplex mode. More robust serial standards exist but are difficult to find in off-the-shelf components.

The Arduino supports wireless connectivity with BLE 5.0. BLE is intended for low powered devices used in home automation, short range wireless connectivity, and remote control. Bluetooth is not directly compatible with BLE, however most senders and receivers supporting Bluetooth 4.0 or later will have support for BLE. Compared to other low powered wireless communication standards, BLE has a relatively high transfer rate with a theoretical maximum of 2Mbit/s. Range has a theoretical maximum of around 100 meters under ideal conditions and signals are sent in the 2.4 GHz radio frequency band like traditional Bluetooth[76].

6.4.1 BLE Communication Structure

Traditional Bluetooth and serial communication have an asynchronous communication structure were data is sent to one or multiple receivers directly. BLE uses a client-server architecture. The server is called a peripheral and a client is called a central in BLE terminology.

Information to be passed between devices is structured into services that are subdivided into characteristics. A characteristic is updates by the peripheral server, and a central client requesting data from the characteristic will receive the data from the latest characteristic update. There is no pairing as with traditional Bluetooth. Requesting and receiving data is called a transaction. Each transaction takes a few milliseconds to execute[7]. A central client can subscribe to changes in data in the peripheral server to receive the new data without requesting if the peripheral server supports notify.

6.4.2 Limited BLE Range

It was observed during drilling that BLE communication signal strength would decrease throughout the drilling operation when the BHA was inside the rock. Comparisons between signal strength at similar distances with and without the BHA suggested that the metal enclosing severely affected the signal strength. Loss of signal strength resulted in a slower sampling rate and sometimes a disconnect while drilling. No singular sender placement position managed to maintain connection with full transfer both at the top and bottom while inside the rock. The sender has the largest permissible power for the Bluetooth and BLE standard. The problem could potentially be solved by having multiple antennas placed around the rock.

6.5 BHA Integration

The sensor package must be protected from mechanical stress, water, and sand during operation. Ideal sensor position is as close to the bit as possible as explained in subsection 9.11. A sensor housing compartment in the BHA holds the epoxy sealed sensor package while drilling. A wired USB connection to the sensor package was tested first as it was uncertain if BLE would be reliable and a cabled connection would allow both interfaces to be tested.

6.5.1 USB Cable Trimming

The Arduino uses a micro USB connector. The connector holds the cable in place and lines up the connectors of the cable with the PCB pins. USB cable connectors are generally bulky to protect the wires from stress while connecting and disconnecting. The female connector on the Arduino is of the maximum height that will fit inside the sensor housing. No widely available off-the-shelf low-profile USB cables with a micro USB connector was found. Soldering wires directly into the connector would require high precision soldering and be difficult to execute consistently.

Multiple different cables between 3 to 5 meters were ordered to find out if they could be modified to fit inside the sensor compartment. A bulky USB cable can be trimmed in the connector to about the same height as the female connector. USB cables differs in how the 4 wires are protected inside the connector. Datasheets generally do not specify how the connector is constructed as it is not intended to be modified.

A knife was used to remove the outer rubber or plastic shielding of the wires. Some models have wires molded into hard rubber. These are difficult to trim as there is no indication how much can be removed until the wires are exposed by the knife and then likely damaged. Some have a small plastic enclosing the width of the male connector. These allow the connector to be trimmed down with minimal risk of damaging the conductors, making it the preferred choice.

From the ordered cables the RS PRO generic cable was inexpensive, shielded, and had the wires in the connector enclosed in a protective plastic case. The Aluminium enforced cable from Nedis ordered from Elfa Direct had the same properties but were twice as expensive.

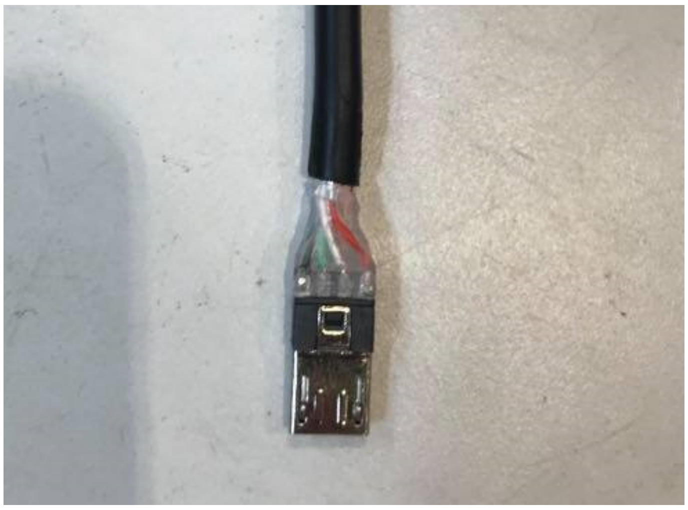


Figure 33: Trimmed RS PRO USB cable.

6.6 Cabled Implementation

The Arduino is inserted into the sensor compartment of the sensor housing in the BHA. A hole was drilled in the upper stabilizer to allow the USB cable to pass. The next challenge was to protect the sensor package from water and short circuiting in the metal BHA. Previous teams have used a protective layer of epoxy[67].

A total of 5 attempts has been made at implementing the sensor package with sufficient water seal. All attempts are described in Appendix B. A consistent procedure for water proofing was eventually found.

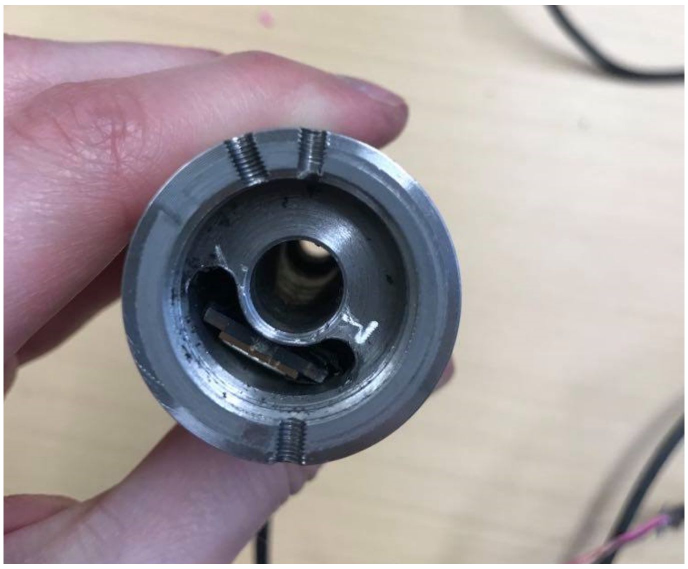


Figure 34: Inserted sensor package without epoxy.

6.6.1 Insertion and Sealing Procedure

The most consistent implementation procedure used a combination of a Plasti Dip coating and epoxy. The Arduino with the cable was covered with multiple layers of Plasti Dip with a 30-minute wait between each layer. The cable was then cut close the largest USB connector for the cable to be threaded through the upper stabilizer hole.

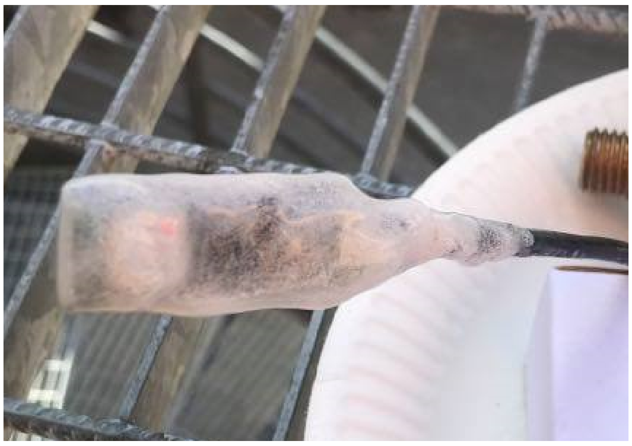


Figure 35: Arduino and USB cable with multiple layers of Plati Dip.

The upper stabilizer and sensor housing must be connected before the sensor package can be inserted. The upper stabilizer is filled with as much epoxy as possible before the sensor housing is threaded in, and the set screw inserted. The cable is then threaded through the upper stabilizer hole. Stiffening the end with tape made this process simpler.

Once the cable is through the hole in the upper stabilizer, the cable is pulled such that there are only a few centimetres left before the sensor package is inserted. The sensor compartment is then filled approximately half full of epoxy.

The sensor package is before insertion painted with a liquid layer of epoxy on top of the Plasti Dip. A flat head screwdriver is used to push the Arduino into the sensor chamber while carefully pulling the cable out at the same rate in the other end. The sensor package should not be inserted longer than necessary to avoid breaking the USB connectors in the micro USB connector. 1 mm clearance to the lower metal edge of the sensor chamber has proven sufficient.

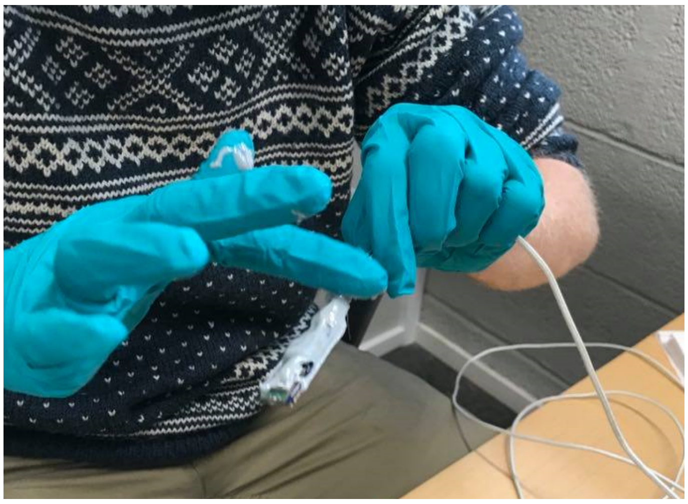


Figure 36: Arduino epoxy application.

A heat gun can be used during the process to reduce the viscosity of the epoxy and stop it from hardening. The upper part of the cable is sealed with epoxy, before letting it harden overnight. The USB connector is resoldered to the large connector that was removed before threading the cable through the upper stabilizer hole. Epoxy in the threads can be removed with a rotating steel brush.

6.7 Power

The Arduino can be powered with 3.3V soldering pads, a 5V micro USB connector, or 4.5-21V using the VIN input[4]. VIN uses a soldering connector removed in the modification process and soldering pads for 3.3V are located underneath the PCB. USB connector power was therefore the preferred option.

Cable to surface can be avoided when using BLE by having a battery pack inside the BHA and connecting the ground and power conductors of the micro USB to the battery.

6.7.1 Minimum Voltage

Battery voltages are determined by their chemical components and are typically not available with a 5V voltage. Size restrictions makes it impractical to use voltage dividers inside the sensor housing. Batteries will have a decreasing voltage as they deplete. Finding the voltage operating range was required to select a battery pack.

The power source was first set to the nominal operating voltage of 5V with maximum current while the Arduino was sending IMU data over BLE. Voltage was gradually reduced until there was no communication or LED lights on the Arduino.

Communication stopped and LEDs turned off at 4.0V. The experiment was repeated with the voltage gradually being increased from below 4.0V to check capacitor charge affected the results. As soon as the power source was increased from 3.9V to 4.0V, the microcontroller LEDs lighted up and BLE communication resumed.

6.7.2 Battery Options

A battery bank would have been a practical option, but no off-the-shelf models small enough to fit inside the BHA was found. Another issue is that most battery banks have a minimum current draw controller to prevent depletion while not in use. The current draw of the Arduino is not sufficient to activate most battery banks, and other options were therefore explored.

Rechargeable batteries were researched next. A single battery with voltage in the range 4.5-5V would have been the best option. The only widely available batteries that could potentially fit inside the BHA was found to be around 1.5V and there would only be enough space for a single battery of this type.

Non-rechargeable batteries have a higher power density than rechargeable batteries. No small widely available single battery with a 4.5-5V voltage was found. Button sized batteries are widely available and generally have a voltage around 1.5V. 3 batteries in series gives a voltage of 4.5V. Surplus button batteries of type LR54/189 was available in the workshop and small enough to fit inside the BHA.

6.7.3 Battery Proof of Concept

3 button cell batteries were connected in series and used to power the Arduino. Batteries were stacked on top of each other. Electrical tape was applied around the edges of the battery on the top and bottom of the battery to avoid short circuits. A small piece of copper tape in the centre on the top and bottom of the batteries completed the circuit. The voltage reading from the battery pack was measured by a voltmeter to approximately 4.5V. The sensor package powered by the batteries was able to transfer IMU data over BLE.

6.8 Wireless Implementation

Making a wireless sensor package was not prioritized due to time restrictions, communication reliability with wireless communication, and minimal issues when using a wired connection. Unsolved problems for the implementation were how to create a waterproof connector between the Arduino and battery package, and how to hold the batteries in place while allowing the battery cartridge to be removed.

6.9 Programming

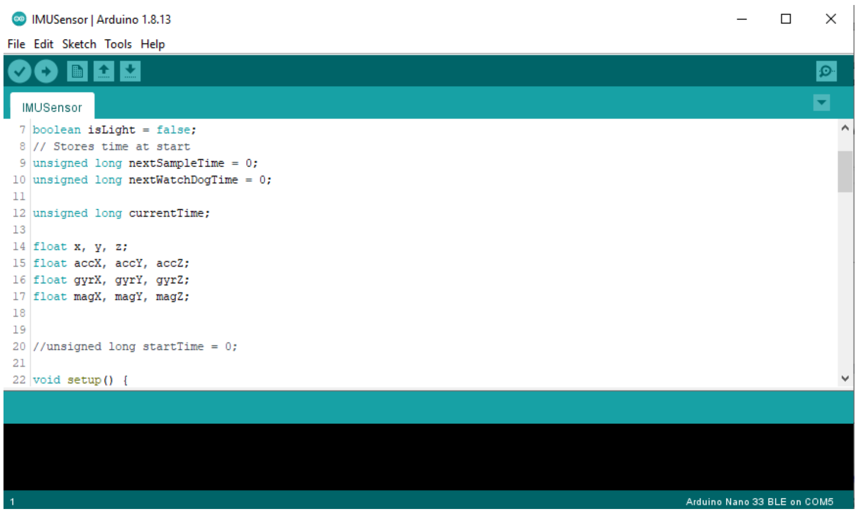
The Arduino is a general-purpose microcontroller. To use it as a IMU sensor package it had to be programmed to sample and transfer data to surface.

6.9.1 Arduino IDE

The microcontroller is programmed in a code editor purposely designed for Arduino products, intended to make it easy to get started. A new project is organised into two functions: setup and loop.

```
void setup() {
    // put your setup code here, to run once:
}
void loop() {
    // put your main code here, to run repeatedly:
}
```

The IDE has a text editor, compiler, debugger, and support for packages. The programming language is based on C and is well documented with an active community.


IMUSensor | Arduino 1.8.13
File Edit Sketch Tools Help
IMUSensor
1
Arduino Nano 33 BLE on COM5

```
boolean isLight = false;
// Stores time at start
unsigned long nextSampleTime = 0;
unsigned long nextWatchDogTime = 0;

unsigned long currentTime;

float x, y, z;
float accX, accY, accZ;
float gyrX, gyrY, gyrZ;
float magX, magY, magZ;


//unsigned long startTime = 0;

void setup() {
```

Figure 37: Arduino IDE interface.

6.9.2 Libraries

The IMU sensor and BLE uses drivers installed in the Arduino IDE library manager.

- **Arduino_LSM9DS1.h**: Communication with the IMU sensor. Functionality for configuring the sensor and sampling data[3].
- **ArduinoBLE.h**: Library for using the BLE module on the Arduino. Functionality for both sending and receiving data[7].

6.9.3 Data Sampling

Sample commands are sent from the microcontroller to the sensor using function calls to the Arduino LSM9DS1 library. There is dedicated function call for sampling the accelerometer, magnetometer, and gyroscope in all 3-axis, respectively.

```
// Import library
#include <Arduino_LSM9DS1.h>
// Define the sample variables
float accX, accY, accZ;
float gyrX, gyrY, gyrZ;
float magX, magY, magZ;

void setup() {
    // Wait until IMU is initialized
    if (!IMU.begin()) {
        while (1);
    }
    // Read data into variables once
    IMU.readAcceleration(accX, accY, accZ);
    IMU.readGyroscope(gyrX, gyrY, gyrZ);
    IMU.readMagneticField(magX, magY, magZ);
}
```

Sample commands can be placed in the loop section of the code to sample more than once.

6.9.4 Serial Data Transfer

Data can be passed using a wired serial connection. Serial does not require any libraries when using Arduino IDE.

```
void setup() {
    // Start serial with baud rate as parameter
    Serial.begin(9600);
    // Ensure serial port is ready.
    while (!Serial);
    // Send arbitrary message over serial
    Serial.println("Hello world");
    ...
    // If the receiver expects a 4 byte float value instead
    float value = 1.2345;
    Serial.write((byte*)&value, 4);
}
```

The serial connection for the Arduino Nano 33 BLE Sense is virtual and the baud rate can be set arbitrarily[4].

6.9.5 BLE Data Transfer

The peripheral data server can be started and updated from the ArduinoBLE library.

```
// Import library
#include <ArduinoBLE.h>
// Define names and ids used to detect the device
const char* nameOfPeripheral = "DownholeIMU";
const char* uuidOfService = "00001101-0000-1000-8000-00805f9b34fb";
const char* uuidOfTxChar = "00001143-0000-1000-8000-00805f9b34fb";
// Define the BLE Service
BLEService imuService(uuidOfService);
// Define the BLE Characteristics
int numberOfBytesToTransfer = 2;
BLECharacteristic txChar(uuidOfTxChar, BLERead | BLENotify |
↪   BLEWriteWithoutResponse,numberOfBytesToTransfer,true);

void setup() {
    // Wait until BLE is initialized
    if (!BLE.begin()){
        while (1);
    }
    // Check if device is connected
    BLEDevice central = BLE.central();
    if (central.connected()){
        // Send arbitrary value
        int value = 123;
        txChar.writeValue(value,numberOfBytesToTransfer);
    }
}
```

Service UUIDs are taken from default examples in the library documentation[7]. Sending data from surface to downhole would require a rx characteristic.

6.9.6 Sample Loop

IMU data is intended to be used for real-time orientation estimation described in section 9. The discrete-time observers require a consistent sample interval. There are limitations to how fast each sensor can sample and transfer data to surface, and how fast the control system can process the incoming data.

Placing the code for sampling and sending data in the loop section would allow the fastest possible IMU transfer rate. Samples and transmissions will have slight deviations in execution time resulting in an inconsistent sampling rate. A busy wait loop frequently checking an accurate timer can be used to maintain a consistent sampling frequency. The sample execution code is not reached until the next sample time is surpassed or equal to the internal clock. A variable stores the next sampling time with the period added for each sample.

```
// Initialize timers and period
unsigned long nextSampleTime = 0;
int PERIOD = 10;

void setup() {
    nextSampleTime = millis() + PERIOD;
}

void loop() {
    if(nextSampleTime - millis() <= 0){
        nextSampleTime += PERIOD;
        // Read from sensors
        ...
    }
}
```

Arduinos have a more accurate timing function called `micros()`. This function is not used as it would cause a long overflow after about 70 minutes[8]. Logic to handle overflows could have be implemented but would add further complexity to the code. The `millis()` value is checked as fast as possible and the sampling period is set as an integer value of milliseconds, meaning the two functions will be equally accurate at maintaining loop timings.

There is no synchronization between the control system and microcontroller clock. Some samples rate irregularities may therefore occur over time. The Arduino was able to consistently sample and transfer data at a rate of 100Hz.

The surface transfer rate can be reduced by using a downhole filter. A simple filter implementation is described in Appendix D.

6.10 Magnetometer Calibration

An ideal magnetometer will measure magnetic field strength without any distortions and offsets. Samples orientated in all directions for the same position would then create a perfect sphere with measurements centred at the origin. Metal objects, electromagnetic interference, and manufacturing defects affects magnetometers sensor measurements and requires calibration to provide useful orientation data[34].

6.10.1 Hard Iron Effects

Hard iron effects are stationary magnetic noise sources. Common sources of hard iron effects are metallic objects and currents in the PCB surrounding the magnetometer sensor. Hard iron effects move the origin of the ideal sphere away from the zero center. The metallic BHA enclosing the sensor and the stationary metal frame of the rig are potential sources of hard iron effects. Biases are introduced to counter out the expected displacement relative to the zero centre.

6.10.2 Soft Iron Effects

Objects affecting the magnetic field around the sensor is called soft iron effect sources. Soft iron effects distort the shape of the ideal sphere into an ellipsoid. Some sources of soft iron effects on the rig are motors and the rotating rod. Soft iron effect calibration stretches the shape of the ellipsoid into a sphere and rotates it to align with the global frame.

6.10.3 Calibration Data

A magnetometer can be calibrated for hard and soft iron effects. Magnetometer data is sampled while the sensor is being orientated in every possible orientation. Calibration conditions should be as close as possible to the operating conditions in terms of position and distortion effects. The sensor can be orientated manually by hand while trying to sample data uniformly in all orientations.

6.10.4 MATLAB Calibration Implementation

MATLAB has a dedicated function to correct for soft and hard iron effects called `magcal`. The function takes an array of uncalibrated magnetometer measurements in all 3 axes as input and outputs the correctional coefficients A, b, and the residual error of the calibration. The correction technique compares the quadratic deviation from measurements x, soft iron effect bias b, positive definite matrix R, and the magnetic field strength β.

$$(x-b)R(x-b)^T = \beta^2 \tag{3}$$

R is chosen by the `magcal` function automatically based on the supported algorithm with the smallest residual error defined by Equation 4. The supported options are identity matrix, diagonal matrix, and symmetric matrix[33].

$$E = \frac{1}{2\beta^2}\sqrt{\frac{\sum ||(x-b)A||^2 - \beta^2}{N}} \tag{4}$$

A is a square matrix of order 3. `magcal` uses a variety of solvers to minimize E by adjusting A. Calibrated magnetometer measurements are calculated by subtracting the biases and correcting distortions with the A matrix.

$$m = (x-b)A \tag{5}$$

6.11 Additional Features

The Arduino has a microphone that could be used for high frequency drilling phenomena identification. The microcontroller is computationally capable of running simple AI models and potentially Fast Fourier transforms on real-time data. Time limitations meant that these features were not tested.

6.12 Potential Improvements

There are some drawbacks of the Arduino Nano 33 BLE Sense and its implementation.

- **Size**: The Arduino is large relative to the BHA. Having a smaller sensor package would allow the BHA to be shorter and the sensor to be placed closer to the bit.
- **Wireless BHA**: A wired connection may cause issues with cable wind-up while using the azimuth control system and makes waterproofing more difficult. Testing indicates that a fully wireless BHA is possible with the current Arduino using multiple antennas.
- **Sample rate**: The maximum IMU sample rate while transferring is limited to slightly over 100Hz. Higher sampling frequency could be used to improve orientation estimates and analysis of high frequency trends.
- **Bluetooth**: The non BLE version of Bluetooth has higher transmission rate, more powerful antennas, and better software support. Bluetooth or BLE 5.1+ could be used for position estimates described in subsection 9.15.
- **Sensor quality**: Sensors used in the microcontroller are intended for consumer electronics. More accurate sensors do exist with less gyroscopic drift, smaller unit to unit differences between the sensors, and higher measurement accuracy.
- **Off-center sensor placement**: The IMU is not located in the centre of the BHA. This has implications further described in subsection 9.11.

A new custom sensor package is a possibility, but future teams should be certain they have sufficient skill, time, and capabilities to do it before committing[67]. New Arduino Nanos such as Arduino Nano RP2040 CONNECT with improved feature sets should also be considered[5][6].

7 Hardware Communication

The main concern when changing software environments to Simulink was hardware communication. LabVIEW is known for having great hardware support while Simulink only supports a few communication standards and hardware components. Replacing proven components on the rig with components known to be compatible with Simulink was not an option because of budget constraints. Most of the existing critical components therefore had to be compatible with Simulink for the change to be sensible.

7.1 DAQ

A concern when changing from LabVIEW to Simulink was DAQ hardware support. The DAQ is made by National Instruments that makes the LabVIEW software. Integrating a device on a competing software platform was expected to be more difficult. Simulink lists the USB-6212 as a supported device in the Data Acquisition Toolbox[40].

Simulink has an Analog Input block used to sample data from the DAQ. Asynchronous and synchronous samplings are supported. USB-6212 is not real-time compatible and is intended for buffered reads with multiple measurements being taken and outputted as a synchronized sample array. Asynchronous mode drastically reduces the maximum sample rate while allowing values to be outputted frequently, reducing latency for closed-loop control. Sample data is placed in a FIFO buffer before being read by the control system. Sample time is specified in the block configuration.

ni Dev2
USB-6212 Data
100 samples/sec

Figure 38: Simulink Analog Input Block.

Block size must be defined for the Analog Input block specifying the number of samples that the DAQ should sample before outputting the measurements to the model. Minimum block size is 2 for the Analog Input block. Sampling frequency must therefore either be set to double the desired sampling frequency or set to the desired sampling frequency and samples delayed by a time-step.

Tests increasing the sampling frequency until a DAQ buffer overflow occurred with only the Analog Input block in a model indicated that single channels can be sampled at up to 200Hz with a block size of 2. Any additional channels reduced the sampling rate approximately linearly. Sampling and outputting from load cell and azimuth torque sensor require the sample rate to be set not much higher than 100Hz with a block size of 2.

7.1.1 Sample Selector

Both load cell and azimuth torque sensor data are used in closed-loop control. Sensor data cannot be sampled at different rates using the Analog Input block. Sample rate is set to 100Hz, and a latency is introduced at every time-step to make use of the full sampling frequency. A sample selector can be implemented in Simulink inside an atomic subsystem sampled at a 100Hz sampling rate. The analog read block is sampled at half the desired sampling rate to correspond to the block size of 2. The matrix with column and row count 2 is split by Selector blocks. The split matrix is inputted into a MATLAB Function block as first and second input. The sample selector will output the first sample at the first iteration and the second sample at the second iteration.

```
function y = SampleSelector(first,second)
    % Choose every other sample from two inputs
    persistent firstInput;
    if isempty(firstInput)
       firstInput = true;
    end
    if firstInput
       y = first;
       firstInput = false;
    else
        y = second;
        firstInput = true;
    end
end
```

7.1.2 Reduced Sampling Rate

An alternative to introducing delay is to have a 50Hz sampling frequency and outputting the second value from the selector and discard the first. Drilling causes high frequency vibrations and reducing the sampling frequency would further reduce the Nyquist frequency for data analysis. Previous teams have successfully drilled using a 50Hz sampling for WOB with minimal latency in LabVIEW[67]. A higher sampling was favoured to detect and react to more of the occasional spike values that may cause significant damage to the mechanical system.

7.1.3 Output Rate Warning

When initializing the Simulink model with an Analog Input block, a warning is displayed indicating that sampling frequencies more than 20 might not be achievable with the specified block size.

The warning does not change based on how many channels that are being read simultaneously, suggesting that the warning assumes the worst-case scenario of all channels being read. During initialization, the buffer may overflow and have some seconds of reduced responsiveness, however buffer overflows during operating while sampling at 100Hz has not been observed as long as the model is not computationally overloaded by other blocks or background programs.

7.1.4 Simulation Pacing

Simulink models will by default run as fast as possible. The Analog Input block will automatically reduce the execution time to match the sample time without the use of model simulation pacing. The Analog Input block may therefore appear to use more system resources in Simulink Profiler as it holds simulation time between time-steps.

7.2 Automation Protocols

Automation protocols are standardized communication interfaces used by multiple vendors. Standards allow communication with components without a device specific driver or API. Product and vendor specific drivers and APIs typically have very limited software support and documentation. Standards simplifies industrial automation implementation, and vendors benefit from not having to design and implement their own structure. Some common standards are Modbus, EtherCAT, PROFINET, and EtherNet/IP. Standards can be proprietary or made as a collaboration between multiple companies and organizations.

7.3 Modbus Communication

Modbus is an automation standard developed by Modicon in 1979[80]. Despite its age it is still widely used as it is royalty free and easy to implement. The standard has less capabilities than more modern protocols, but its widespread use means great software support.

MATLAB has support for Modbus TCP communication through the Instrument Control Toolbox. There is no dedicated Simulink block for Modbus TCP communication, however there is support in Simulink for running MATLAB code in the MATLAB Function block[37].

7.3.1 Modbus TCP Request/Response Structure

Modbus TCP is a master-slave communication structure. Multiple master devices can be used in the same network, and a master device can read from and write to any other master device. Client devices can only read from other masters. Modbus TCP uses IP-networks operating in level 7 of the OSI model[80].

There are 4 types of data objects. Coils and discrete inputs are a single bit in size. Coils can be both written to and read from while discrete inputs are read-only. Input registers and holding registers are both 16 bits in size. Input registers are read-only while holding registers are read-write. An overview inspired by [80] is shown in Table 4.

Table 4: Modbus TCP Object Type Address Range.

Object Type	**Access**	**Size**	**Address Space**
Coil	Read-write	1 bit	00001-09999
Discrete Input	Read-only	1 bit	10001-19999
Input Register	Read-only	16 bit	30001-39999
Holding Register	Read-write	16 bit	40001-49999

Reading data in Modbus is requesting a range of memory addresses from a master. Each memory address corresponds to either a single bit or 16 bits depending on the data object type. The relation between memory address and value can be found in the register value overview in the technical documentation for the master device. Data type is specified in the register list. A value larger than 16 bits will span multiple consecutive memory addresses.

Commands are identified by the master with a function code. Function codes for reading and writing data is shown in Table 5.

Table 5: Modbus TCP Read/Write Function Codes.

Function Name	**Function Code**
Read Coils	1
Read Discrete Inputs	2
Read Multiple Holding Registers	3
Read Input Registers	4
Write Single Coil	5
Read Single Holding Register	6
Write Multiple Coils	15
Write Multiple Holding Registers	16

Modbus TCP commands and responses are sent as frames. The size of the frames depends on the function code as different commands has different number of parameters. Table 6 is inspired by [80].

Sending frames requires the IP address of the server to be known. Modbus TCP has a default port of 502.

Table 6: Modbus TCP Frame.

Name	Length (bytes)	Function
Transaction Identifier	2	For synchronization between messages of server and client
Protocol Identifier	2	0 for Modbus TCP
Length Field	2	Number of remaining bytes in this frame
Unit Identifier	1	Slave address (255 if not used)
Function Code	1	Function codes as in other variants
Data Bytes	n	Data as response or commands

7.3.2 Modbus Explorer

Modbus Explorer is a MATLAB tool for testing Modbus communication. The user interface operates as a Modbus client capable of testing read and write communication with a Modbus server. The configuration has options for IP address, port, timeout, byte order, bit order, and server id. Server values can be read and plotted continuously and write commands can be sent from a form. Datatype, address, and register type is specified for all reads and writes. Read values can be assigned a name for identification purposes. There is an option for exporting a MATLAB script for the current connection with read and write values in a template containing connection, configuration, reads, and successful writes.

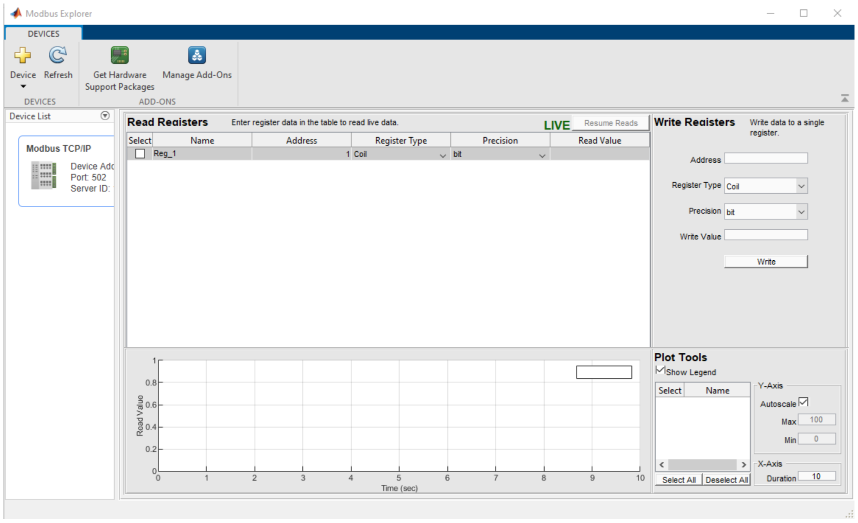


Figure 39: Modbus Explorer interface.

7.3.3 MATLAB Implementation

MATLAB has support for Modbus TCP and RTU communication using the Instrument Control Toolbox. MATLAB has functions for establishing connection to the server, reading, writing, and obtaining device information.

A connection is established to a server by creating a Modbus object. The required parameters for connecting to a TCP server is the transport protocol and device address. Server port and additional connection options can also be specified, such as timeout, bit order, or byte order. Options can be applied to the object after it has been created.

```
m = modbus('tcpip', ipAddress);
% Some optional configurations of the connection object
m.Timeout = 3;
m.WordOrder = 'little-endian';
```

Once the Modbus object has been created it can be used to read and write from the server. Reading from the server requires a minimum of 3 parameters: Modbus object, data object type, and starting address.

```
    % Read the first input register
    data = read(m, 'inputregs', 1);
```

A range of multiple values can be read using the same command with a different set of parameters. The count parameter specifies how many subsequent registers to read. Specifying precision in the command reads the returned values as a specific data type. Default data type is `uint16`. The command requires server id to be specified. Server id is 1 by default.

```
    % Reads 3 uint32 values starting from the first register
    data = read(m, 'inputregs', 1, 3, serverId, 'uint32');
```

Returned data type is an array when multiple values are read.

Writing is executed by a MATLAB write command. Required parameters are the communication object, data object type, starting address, and the values to be written.

```
    % Write 'value' to the first holding register
    write(m,'holdingregs',1,value);
```

It is possible to write to multiple subsequent values using the same command with additional parameters. Precision can also be specified.

```
    % Write 2 uint32 values starting at the first holding register
   write(m,'holdingregs',1,[value1 value2], serverId, 'uint32');
```

7.3.4 Simulink Implementation

There is currently no dedicated Modbus TCP communication block in Simulink without using speedgoat hardware described in subsubsection 5.8.2. Dedicated blocks for TCP communication in the Instrument Control Toolbox could potentially be used to control the drives. Using the dedicated block would require the Modbus TCP communication structure to be implemented manually. This was assumed to be time consuming, even for the simple Modbus TCP standard. If maximal performance or real-time support is a requirement this may have be a viable option and could be investigated in future work.

Modbus communication is Simulink is handled by a MATLAB Function block running MATLAB code. MATLAB Function blocks does not allow objects as input values. The MATLAB Function block will call its internal function at every time-step. The Modbus object is created as a persistent variable inside the function block to reduce computational cost. Persistent variables are maintained between function calls. The object is created at the first function call time-step.

```
function [output1,output2] = ModbusTCP(input1,input2)
    % Must be specified to avoid compile error in Code Generation
    output1 = 0.0;
    output2 = 0.0;
    persistent m;
    if isempty(m)
        m = modbus('tcpip', ipAddress);
    end
    % Read 2 uint16 values from input registers starting from first register
    data = read(m, 'inputregs', 1, 2);
    % Write 2 uint16 values to holding registers starting from first register
    write(m,'holdingregs',1,[input1 input2]);
    % Extract read values to be outputted
    output1 = data(1);
    output2 = data(2);
end
```

Code generation requires the output variables to be specified before they are overwritten to specify its data type.

7.4 EtherNet/IP

Ethernet/IP is an industrial network protocol created in the mid-1990s[79]. It is commonly used in factories and the process industry for automation purposes. The standard is optimized for real-time control and built to allow automation over standard IP networks. Measurements and drive commands are sent as UDP datagrams and TCP packets[84][83]. EtherNet/IP operates in the level 5, 6, and 7 of the OSI model, meaning non-specialised switches and routers operating on layer 2 and 3 can be utilized.

A device using EtherNet/IP has an EDS file defining what the device is, what parameters it has, current configuration, and which values that can be read from the device. Registers are not accessed directly as with Modbus TCP. Common Industrial Protocol (CIP) messages are passed to and from the devices in the network with instructions on data to accessed and altered[78].

7.4.1 Gateway Translation

Devices for translating between automation protocols exists. These devices create two subnet with two different protocols and translates commands and data between them. Gateways can be used to improve software support or connect legacy equipment to a new system. Feature sets of standards differ, and some functionality cannot be translated. The gateway described in subsection 4.5 translates EtherNet/IP to Modbus TCP. EtherNet/IP organises data in objects while Modbus TCP organises data in registers that can be written to and read from.

Translating between standards in a gateway is a suboptimal solution as it introduces more points of failure, makes hardware integration more complex, is more difficult to troubleshoot, increases latency, and reduces the feature set of standards. Gateways should therefore be avoided if possible.

7.5 Hoisting and Top Drive Motor

Hoisting and top drive motor control and measurement relies on the EtherNet/IP automation protocol. Simulink does currently not support EtherNet/IP natively without specialized hardware from speedgoat[71]. Previous teams have used a programmable gateway described in subsection 4.5 to translate commands and data between EtherNet/IP and Modbus TCP with better software support. The gateway was then used to control a single device.

Based on the technical documentation of the gateway it was assumed that the device could be programmed to control multiple devices. The backup plan was to acquire another gateway if multiple devices was not supported, or other technical issues was encountered. Purchasing another gateway was undesirable as each unit costs 800USD and add further complexity to the system.

7.5.1 SYSCON.net Configuration

The gateway has a configuration software called SYSCON.net described in subsubsection 4.5.1. Devices can be added to the EtherNet/IP subnet by dragging and dropping another block into the block programming interface. Because the new top drive had an equal driver and communication module to the hoisting system, the existing block was copied and connected to the EtherNet/IP subnet. The configuration is shown in Figure 40.

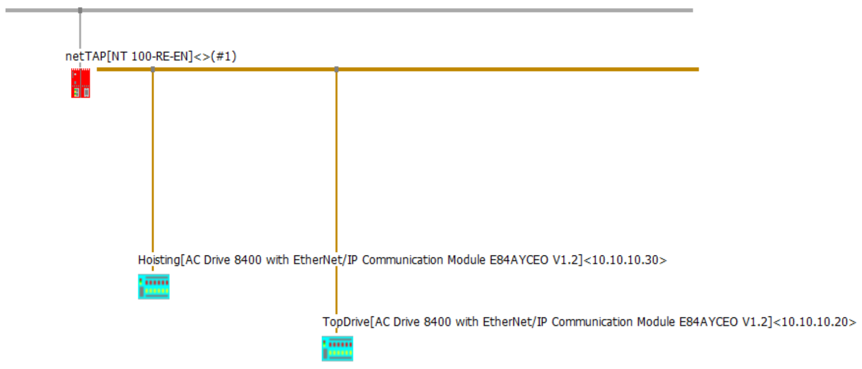

netTAP[NT 100-RE-EN]<>(#1)
Hoisting[AC Drive 8400 with EtherNet/IP Communication Module E84AYCEO V1.2]<10.10.10.30>
TopDrive[AC Drive 8400 with EtherNet/IP Communication Module E84AYCEO V1.2]<10.10.10.20>

Figure 40: SYSCON.net device configuration interface.

Modbus TCP requires every read value to be assigned to a register. The read values are mapped from EtherNet/IP to Modbus TCP as unsigned 16-bit integers. When multiple devices are connected, the range of registers for a device starts after the total number of read and write values of any device added previously to the subnet.

Default configuration will automatically map all readable values from EtherNet/IP to Modbus TCP. If an input or output should not be mapped, they can be removed in the mapping configuration. Most outputs are not mapped in Lenze Engineer and currently unused. Values are still mapped for ease of future configuration in Lenze Engineer. Reducing the number of mapped values will not reduce the register displacement between the two devices.

7.5.2 Inputs and Outputs

Register indexes of the Modbus TCP subnet is shown in Table 7 and write values in Table 8. All registers are holding registers.

Table 7: Hoisting and Top Drive Modbus Read Addresses.

Value	Address	Data type
Hoisting RPM	2	uint16
Hoisting Torque	3	uint16
Hoisting Position	4-5	uint32
Top Drive RPM	18	uint16
Top Drive Torque	19	uint16

7.6 Azimuth Control Motor

The azimuth control motor supports both Modbus TCP and EtherNet/IP. MATLAB has native support for Modbus TCP and this protocol was tested first to allow direct communication without the use of a gateway. Documentation of the MOONS' drive had examples on how to send commands to the drive and the overview of register addresses[64]. The drive was configured for

Table 8: Hoisting and Top Drive Modbus Write Addresses.

Value	Address	Data type
Hoisting Control Word	2	uint16
Hoisting RPM Setpoint	3	uint16
Hoisting Torque Limit	4	uint16
Top Drive Control Word	18	uint16
Top Drive RPM Setpoint	19	uint16
Top Drive Torque Limit	20	uint16

Modbus communication mode in the M Servo Suite software. Modbus Explorer was used to test communication and create a MATLAB script for communication.

The motor is controlled by setting the velocity setpoint for the drive and sending an opcode to enable and disable movement.

Table 9: Azimuth Modbus Commands.

Command	Hexadecimal	Decimal
Start Jogging	0x96	150
Stop Jogging	0xD8	216

7.6.1 Inputs and Outputs

Read registers are listed in Table 10 and write registers in Table 11. Full list of registers is available in the driver manual[64].

Table 10: Azimuth Modbus Read Addresses.

Value	Address	Data type
Azimuth Position	7-8	int32
Azimuth RPM	11	int16

Table 11: Azimuth Modbus Write Addresses.

Value	Address	Data type
Azimuth RPM Setpoint	49	int16
Azimuth Command Opcode	125	int16

An attempt at controlling the motor with EtherNet/IP was made motivated by reduced computational cost. The attempt is described in Appendix C

7.7 Modbus Communication Optimization

It was observed from Simulink Profiler test that hardware communication was very computationally expensive. Efficient Modbus communication was crucial to achieve sufficient sampling and input frequency for closed-loop control. It was discovered that the computational cost scaled approximately linearly with the number of Modbus commands being executed.

7.7.1 Shared Modbus Connection Object

It was assumed that most of the execution time was spent holding the execution thread while waiting for a response from the Modbus master. A shared Modbus connection object with non-exclusive properties could potentially allow waiting to be done in parallel for reads and writes.

Sharing an object between multiple MATLAB Function blocks proved difficult. Simulink does not allow objects to be passed as inputs or outputs between blocks and workspace and global MATLAB objects cannot be passed to a function as a parameter. Other options were therefore considered.

7.7.2 Write-Read Modbus Command

MATLAB has a Modbus command for both writing and reading called `writeRead`[56]. The command is limited to holding registers only. Because both the gateway and azimuth control motor communication read from input registers there is no benefit in using this command.

7.7.3 Command Aggregation

The amount of data being passed with each command had no detectable difference in execution time. Values that the control system reads may not be in subsequent memory registers. Modbus supports reading and writing multiple registers as a range with the start and number of bytes specified.

Based on these observations, read commands were aggregated into a single command reading the full range from the first desired value to the last. Data from the registers between the desired values are discarded by the MATLAB Function block in the control system.

For consecutive writes it is not possible to skip values in the range. Not connecting inputs values to logic in Lenze Engineer means they can be set arbitrarily, and a single write command be used to control both drivers. Mapping could also have been removed in SYCON.net, but this would make it more complex to add new input values in Lenze Engineer later.

7.7.4 Write On Change

It is possible to have persistent values in a MATLAB Function block. Drivers will retain the last sent command values until they are powered down. If none of the values are changed from last time-step it is not strictly necessary to send a new command.

This feature was implemented and tested before being removed. While using a closed-loop controller the commands will change very frequently limiting the potential gains of the feature. It would be possible to have a threshold before a new command is sent for some values, but this would make the system less predictable and be prone to bugs.

Real-time systems are less about the total throughput and more about predictability for every time-step. Writing at every time-step is the worst case and there should always be enough computational resources to handle this scenario. Furthermore, the feature is prone to bugs and unpredictable behaviour if a communication failure occurs. The feature can be used as a last resort if additional computational throughput is needed.

7.7.5 Write Command Reduction

Drive states must be sampled multiple times a second to detect torque spikes, estimate rotational velocity, and detect incidents. Writes can be executed less frequent as the motors will not be able to respond to a new setpoint or commands immediately. Reads and writes have approximately the same computational cost. This feature is implemented by having a persistent state keeping track of how many writes have been executed since last write.

```
function [output1,output2] = ModbusTCP(input1,input2,writeInterval)
    % Must be specified to avoid compile error in Code Generation
    output1 = 0.0;
    output2 = 0.0;
    persistent m;
     % Define write counter
    persistent writeCounter;
    if isempty(m) || ismepty(writeCounter)
        m = modbus('tcpip', ipAddress);
        % Dummy value to write on first loop
        writeCounter = Inf;
    end
    % Read data
    data = read(m, 'inputregs', 1, 2);
    % Write at certain interval
    if writeCounter >= writeInterval
        write(m,'holdingregs',1,[input1 input2]);
        writeCounter = 1;
    else
        % Increase the counter
        writeCounter = writeCounter + 1;
    end
    % Output data
    output1 = data(1);
    output2 = data(2);
end
```

The write interval defines how many reads that are executed for every write. The sample rate of the drive must be set such that it could be maintained with both reads and writes for every time-step to ensure consistent sample times.

7.7.6 Concurrent Execution

Concurrent execution allows multiple tasks to be executed in parallel utilizing multiple CPU cores. Parallel hardware communication reduces the impact on minimal time-step when having multiple hardware components and increases the computational budget. Synchronizing and passing data between threads introduces overhead and profiling is required to determine if concurrent execution is beneficial.

7.8 Optimized Hoisting and Top Drive Communication

It is possible to use a single read command to receive values from both the hoisting and top drive with command aggregation. Some additional read values are retrieved by the function call and disregarded by the control system.

```
% Values between the two registers will be all 0.
% uint64 and uint32 reads are used for spacing in-between
% The hosting and top drive registers
data = read(m, 'inputregs', 1, [3, 1, 2, 1, 4], serverId, {'uint16',
    'uint32', 'uint64', 'uint32', 'uint16'});
% Extract desired hoisting data
hoistingRPM = data(2);
hoistingTorque = data(3);
hoistingPos = data(4);
% Extract desired top drive data
topDriveRPM = data(10);
topDriveTorque = data(11);
```

A single write command can be used to both the hoisting and top drive motor as the values between the two ranges are not connected to logic in Lenze Engineer and can be set to an arbitrary value.

```
% Write input values to the drives
% 0 values are used for spacing and can be set to an arbitrary uint16 value
write(m,'holdingregs',2,[hoistingControlWord hoistingRPMSetpoint
    hoistingTorqueLimit 0 0 0 0 0 0 0 0 0 0 0 0 0 topDriveControlWord
    topDriveRPMSetpoint topDriveTorqueLimit],serverId,'uint16');
```

Write command reduction is used and write commands sent less frequent than read commands. The atomic subsystem communication block is mapped to its own task to be ran in parallel using concurrent execution.

7.9 Optimized Azimuth Control Communication

Position and RPM from the drive is used for closed-loop control. Position and RPM are not in subsequent registers. Both measurements can be retrieved with a single read command by disregarding any values between the two ranges.

```
% Position is input register 7-8 and RPM is 10
data = read(m, 'inputregs', 7, [2, 1], serverId, {'int32', 'int16'});
% Extract desired data
azimuthPos = data(1);
azimuthRPM = data(3);
```

RPM setpoint and opcode registers are used to control the drive from the control system. RPM setpoint specifies the target rotational velocity and opcode the operation mode. Write registers for the input commands are spread out and writing to both would require many other registers to be overwritten. M Servo Suite does not allow registers to be unmapped as with Lenze Engineer. Write on change can be used for the opcode as it is rarely altered.

```
% Check if opcode input has changed since last time-step
if lastAzimuthCommandOpcode ~= azimuthCommandOpcode
    % Opcode changed, send a write command
    write(m,'holdingregs',125,azimuthCommandOpcode,serverId,'int16');
    lastAzimuthCommandOpcode = azimuthCommandOpcode;
end
```

RPM is used in closed-loop control to regulate azimuth torque and is updated more frequently.

```
% Send RPM setpoint to drive
write(m,'holdingregs',49,azimuthRPMSetpoint,serverId,'int16');
```

Write command reduction and concurrent execution is used to reduce computational cost equal to the hoisting and top drive implementation.

7.10 Downhole Communication

The downhole sensor described in section 6 has two options for transferring data to surface; serial and BLE. BLE was the preferred option to avoid USB cable runs along the pipe and allow better water sealing of components in the BHA.

The downhole sensor supports Bluetooth Low Energy (BLE). BLE is intended for low powered devices and is less capable than the traditional Bluetooth standard in terms of sender power and maximum transmission rate. Before testing there were concerns that Bluetooth would either not have sufficient range when enclosed inside the metal BHA or not have sufficient transmission rate to transfer the required data in real-time to surface, requiring the serial interface to be used.

7.10.1 BLE Communication

Simulink does not currently have BLE communication support. MATLAB has BLE communication support and a MATLAB Function block can be executed in a Simulink model.

To get a list of available BLE devices the `blelist` command can be executed. The list will show name, address, and RSSI of available devices. UUID or name is used to create a BLE peripheral device object with the `ble` command.

```
% Connect using device name
deviceName = "DownholeIMU";
b = ble(deviceName);
...
% Alternatively, connect using address
address = "B9D8EFA56700";
b = ble(address);
```

Characteristics of the peripheral device of interest can be specified for the connected device using the `characteristic` command. Characteristics are a subdivision of a service. A specific characteristic is accessed either by UUID or name of the service and characteristic.

```
    % Specify characteristic using names
    serviceName = "DownholeService";
    characteristicName = "IMUCharacteristic";
    c = characteristic(b,serviceName,characteristicName);
    ...
    % Alternatively, specify characteristic using UUID
    % UUID values are determined by peripheral device
    serviceUUID = "1101";
    characteristicUUID = "1143";
    c = characteristic(b,serviceUUID,characteristicUUID);
```

Latest peripheral device data from the characteristic can be accessed with the `read` command.

```
    data = read(c);
```

If the peripheral device support notify, values can be received for each new update without sending a request from the central device with the `subscribe` command.

```
    subscribe(c);
```

Subscriptions can bet set to trigger a MATLAB function when new data is received.

```
    c.DataAvailableFcn = @BLEDataAvailableFcn;
```

There is currently no Code Generation support for BLE and all BLE commands must be executed by the MATLAB interpreter.

7.10.2 Serial Communication

Serial communication has a dedicated Simulink block in the Instrument Control Toolbox. When placing a Serial Receive block in Simulink, a Serial Configuration block is automatically added. The Serial Configuration block specifies the connection parameters such as baud rate, data bits, parity, and stop bits.

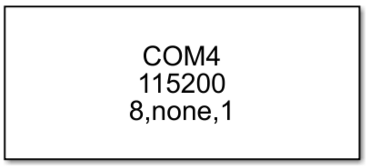


Figure 41: Serial Configuration Block.

Serial Receive requires incoming data to be formatted. For increased robustness on transmission errors, header and terminator can be added to indicate start and end of a transmission. Data type, number of values, and sampling time must be specified in the block. Data is read into a serial FIFO buffer. Slight deviations in timing between the control system and downhole sensor requires the serial sampling rate to be higher than the intended sampling rate to avoid buffer overflows. A Rate Transition block downsamples the oversampled serial values. A Serial Receive block will output its last received value if no new serial data has been received.

Data
COM4
Status

Figure 42: Serial Receive Block.

Headers and trailers must be sent from the Arduino. Header is set to be of type S and trailer of type CR/LF.

```
// Header 'S'
Serial.write(83);
// Write payload data
...
// Terminator "Carriage Return" '\r' and "Linefeed" '\n'
Serial.write(13);
Serial.write(10);
```

7.11 Downhole Communication Optimization

BLE was the preferred choice for downhole to surface communication to have a wireless BHA. BLE communication has a high computational cost to the control system computer and a limitation to maximum data payload size that can be passed with each transfer. Different approaches were tested to circumvent these limitations.

Data from the downhole IMU is primarily used to estimate orientation. Changes in orientation are slow with minuscule change each second. BLE downhole communication is computationally expensive and the high-level control computer is not able to maintain a transfer rate of more than 50Hz under optimal conditions. Maximum transfer rate decreases rapidly if the signal is obstructed by the rock sample.

7.11.1 Buffered Data Transfer

Profiler data suggested that there was minimal difference in computational cost for small and large BLE data payloads. Sending data in larger buffered chunks instead of at each time-step can reduce the computational cost on the control system computer at the expense of increased latency. Orientation estimation is not latency sensitive, and the trade-off is therefore acceptable. Data can be delayed by the product of the time-step and buffer length in the Simulink model to be used for orientation estimation. MATLAB BLE communication is limited to a maximum 256 bytes data size for each transfer.

7.11.2 Gyroscope Data Exclusion

A gyroscope measures the rate of change in orientation. Gyroscopes is used to improve orientation estimates for fast moving objects when the accelerometer is affected by system acceleration. The BHA will change orientation very slowly while drilling. Changes in orientation are minuscule relative to the ±2000dps measurement range. Measurements are affected by vibrations with frequent extreme noise values.

Limited usefulness of values and an upper data transfer payload limit led to the gyroscope measurements being excluded from BLE data transfers.

7.11.3 Downhole Data Encoding

IMU measurement read functions in the Arduino returns float value of 4 bytes. Accelerometer and magnetometer have a measurement range of ±4g with two decimals and ±400uT with zero decimals, respectively. Both values can be encoded as a 16-bit signed integer whiteout data loss. Accelerometer measurements are multiplied with a gain factor of 100 before the conversion.

16-bit signed int values are transferred to surface before being converted back to float numbers with the inverse gain factor.

7.11.4 Downhole Data Processing

The Arduino Nano 33 BLE Sense has a powerful microcontroller. When using measurements for the purpose of orientation estimation it is possible to process the raw data downhole before transferring it to surface.

Some possible filter implementations include median and Kalman. Fast Fourier transform could potentially be used to reduce the required data transfer rate with minimal data loss. It is expected that transfer rates to surface could be reduced multiple orders of magnitude with a downhole filter while making use of the maximal sampling frequency. Orientation estimations could potentially be computed downhole to further reduce sample rate. Not transferring data to surface has the drawback of not allowing post analysis of raw data.

A tested extreme exclusion median filter is found in Appendix D.

7.12 API

The Drillbotics competition guidelines requires an Application Protocol Interface (API) for remote rig operation. No industry adopted API for drilling rigs was found during the research phase.

An API inspired by the UiS Drillbotics 2018/2019 team was developed[17]. The API was implemented by a petroleum student on the team and is further described in their thesis[12]. OPC UA is commonly used in the oil and gas industry and was chosen as the API framework.

7.12.1 OPC UA

OPC is an industrial interoperability standard maintained by the OPC Foundation[11]. Unified Architecture (UA) is the latest version from 2008. Older versions, now referred to as OPC classic, had multiple versions for specific purposes like data access, alarms and events, and historical data access. OPC UA unifies functionality from OPC classic while adding features such as platform independence, security, and expandable feature set.

7.12.2 Architecture

MATLAB has support for basic OPC UA functionality with the OPC Toolbox such as reading and writing to a server[43]. A typical API implementation would have the node to receive requests and that holds the current state as the server. OPC UA server functionality is currently not supported.

Python has packages for OPC UA server functionality. MATLAB functions can be called from Python using the MATLAB Engine API. A MATLAB function can control and read data from the rig with the same interface as the GUI.

The API uses an information model defining available nodes on the server. A node contains methods and events defined in hierarchical structure in a XML file.

7.12.3 Request Structure

Every request node contains a method child node. The child node is an executable function. The executable function will fetch any parameters passed to the Python server and call the corresponding MATLAB function through the MATLAB Engine API with its function parameters. Nodes are accessed by their id or browse-name. An example of a request is to set the ROP or WOB setpoint.

7.12.4 Events

A client can request to be notified when a rig state has been changed. Event based messaging has less overhead than a client frequently polling the state. The server sets a trigger that calls will call a MATLAB function when the event occurs. A typical event trigger could be for the client to be notified when a certain MD is reached or if a drilling incident has occurred.

7.12.5 Rig Data

Data can be read from the rig on request by the client. A read request command is sent from the OPC UA client to the Python server. The Python server forwards the request to the MATLAB instance. Data is retrieved with the same access point as the GUI before being passed through the MATLAB Engine API to the Python server and forwarded to the OPC UA client.

7.13 Control System Computers Communication

High computational cost required the control system to be ran on two computers as a distributed system. Both computers are connected to the local network switch with Ethernet. It was briefly considered to use the OPC UA API for communication between nodes. High frequency reading and writing with the API would introduce significant overhead to both control systems computers. Peer-To-Peer (P2P) communication has less overhead and Simulink has support for TCP and UDP communication.

Transmission Control Protocol (TCP) is used for reliable communication between nodes[83]. The receiver will send a confirmation back to the sender for each packet it has successfully received. The protocol requires an open connection between the sender and receiver before any data can be passed. TCP is used in applications were it is crucial to know if a package has been received as intended at the expense of network and computational overhead.

User Datagram Protocol (UDP) are used in cases where there is no strict requirement to confirm that data has been received[84]. This is commonly used in streams of data between two nodes. Datagrams can be sent without an open connection and there is no read confirmation from the receiver. UDP has less network overhead as datagrams have a smaller header and footer compared to packages, and the total number of packages or datagrams being passed is smaller as there is no read confirmation.

UDP was selected for the communication to reduce overhead. No open connections allow the control systems to be initialized and stopped independently without causing a timeout or requiring a reestablishment of the connection.

7.13.1 Simulink Implementation

Communication is handled by the UDP Send and UDP Receive blocks from the Instrument Control Toolbox. Values to be sent over the network is multiplexed and passed through a Byte Packing block. Data type for inputs and outputs must be explicitly specified and byte alignment can be adjusted. Most signals in Simulink will be of datatype double.

UDP Send block requires the remote address and port of the receiver. An available port is automatically suggested by the block and the remote address is set to the receivers IP address. Local address is the IP address on the local NIC the datagram should be sent from. UDP packet size specifies the maximum number of bytes to be sent[54]. Byte order is set to big endian by default. There is an option to enable blocking mode. Blocking mode will limit simulation time from running faster than real time.

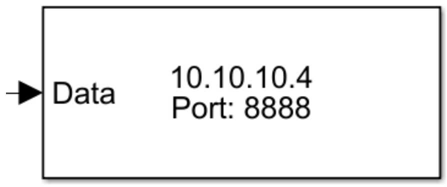


Figure 43: UDP Send block.

UDP Receive requires the local IP address and port number to correspond to the remote address and port in the UDP Send block. Remote address and port correspond to the local address and port set in the UDP Send block in the other node. The block is configured to always output the latest available data. UDP datagrams may arrive out of order. Data size specifies the size of the matrix to be received. Source data is set to the same datatype as the Byte Package output in the sender. Byte order is set to big endian. Blocking mode is disabled. Sample time is set equal to the sender.

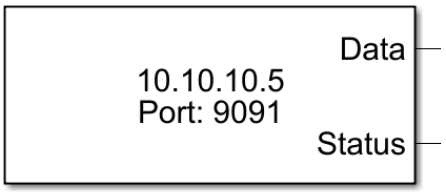


Figure 44: UDP Receive block.

If sender and receiver blocks are present in a single model, block priority of the sender must be set higher than the receiver to ensure that the operations are executed in the intended order.

7.13.2 IP Addresses and Ports

Both computers have a single NIC with IP addresses specified in Table 2. Assigned ports are default port suggestions and an increment of the suggested ports. IP addresses are in the same subnet as the default address of the MOONS' drive.

Table 12: Control System Computers UDP Port Assignment.

Role	**Address**	**Port**
High-Level Receiver	10.10.10.4	8888
High-Level Sender	10.10.10.4	8889
Low-Level Receiver	10.10.10.5	9091
Low-Level Sender	10.10.10.5	9090

8 Low-Level Control System Implementation

Latency sensitive controllers with a short prediction horizon are implemented in the low-level control system. Low-level controllers in the system includes a WOB and an azimuth torque controller. Latency sensitive incident detection must also be implemented in the low-level control system. The operator can control the rig through a Human Machine Interface (HMI).

8.1 PID

A simple and versatile monovariable controller is the Proportional Integral Derivative (PID) controller. The controller provides an input to the system based on the deviation, deviation over time, and rate of change of deviation from the reference. The difference between the measured state and the reference is the error. The controller is tuned using constants for proportion (P), integral (I), and derivative (D) gain. Increasing a constant will increase the input to the system from each of the specific component.

$$u(t) = K_P \cdot e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \tag{6}$$

The error $e(t) = r(t) - y(t)$ is dependent on the reference r and the system output y. K_P, K_I, and K_D are tunable constants.

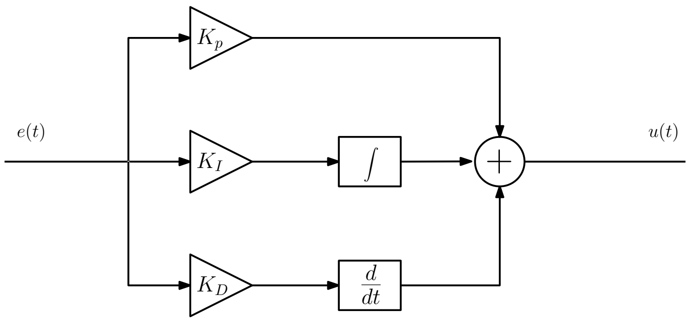


Figure 45: PID diagram illustration.

A standard PID controller has no upper or lower limit for the input to the system. Physical systems generally have some limitation to the magnitude of the input that can or should be provided. Software implementations of the PID controllers often have an option for saturating the input to the system.

8.1.1 Simulink Implementation

Simulink has a dedicated block for PID regulation. The block supports both continuous and discrete time systems. Measurements are discrete and uses a discrete implementation found in Equation 7.

$$u[k] = (K_P + K_I \cdot T_s \frac{1}{z-1} + K_D \frac{N}{1 + N \cdot T_s \frac{1}{z-1}}) \cdot e[k] \tag{7}$$

With N being a large filter constant for the derivative estimate.

8.1.2 Anti-Windup

Output saturation sets an upper and lower limit for maximal output from the block. By default, there is no anti-windup for the saturation. If the saturation limit is exceeded for longer periods of time, the controller integration would continue accumulating and will struggle to not overshoot the reference when approaching the reference value.

Anti-windup techniques are used to limit the integral accumulation for PID controllers. One common method available in the PID Simulink block is back-calculation. Back-calculation has an internal tracking loop that will discharge the integrator output when the controller hits the saturation limit, preventing accumulation and slow response when approaching the reference after a period of output saturation[21]. This is implemented by feeding back the difference between the saturated and unsaturated control signal to the integrator. This feature can be tuned using a back-calculation coefficient specifying the gain in the wind-up loop.

8.1.3 Tuning

Simulink has closed loop auto-tuning tool for finding the optimal PID coefficients. The block is placed after the controller output and will based on the measurements and slight sinusoidal alterations to the output try to estimate the frequency response of the system and provide PID coefficients based on a target bandwidth and phase margin[24]. The PID controller remains unchanged during the tuning and stability will not be affected.

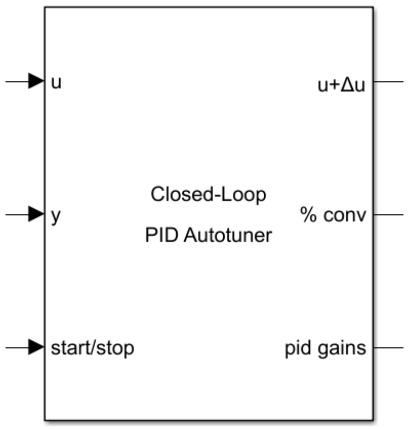


Figure 46: PID Autotuner block.

The block takes input from the PID output, plant output, and a start/stop signal, and outputs an altered PID input, convergence state, and the new suggested controller gains. Target bandwidth, target phase margin, and sinusoidal amplitude must be specified as block parameters. Target bandwidth is the 0-dB gain crossover frequency of the tuned open-loop response.

8.2 WOB Controller

While drilling it is ideal to maintain a constant force on the bit, as it minimizes mechanical stress and maximizes ROP while drilling. Bit rotation with applied WOB causes major vibrations requiring responsive closed-loop regulation. The WOB controller regulates the velocity of the hoisting system based on the measurements from the load cell. The output from the controller is a ROP setpoint that is sent to the hoisting motor.

The controller assumes that moving the hoisting system downwards will increase WOB and moving upwards will decrease the WOB. The drill pipe can be modelled as a stiff spring that will remain in tension if the bit is in contact with the rock. Drilling vibrations can be modelled as process noise.

Output saturation with back-calculation is enabled to avoid high ROP setpoints when the bit is not in contact with the rock.

8.2.1 Simulink Implementation

The controller is implemented in its own subsystem with a Simulink PID block. The error input is the difference between the WOB setpoint and the measured WOB estimate. There is a switch in the GUI for toggling the WOB controller. The toggle signal is connected to an external reset input on the PID controller to avoid unpredictable behaviour when enabling the controller.

8.2.2 Tuning

It was attempted to use the Closed-Loop PID Autotuner block to automatically obtain tuning PID parameters. WOB measurement, PID output, and a manual start/stop witch was inputted into the block, and the ROP output with sinusoidal alterations was inputted as ROP setpoints.

Bit rotation was started and the autotuner enabled when the bit was very close to tagging the rock. The WOB controller was then enabled with a 10kg setpoint. Once the WOB had converged to the setpoint value and convergence rate output indicated close to 100%, the autotuner block outputted suggested parameters for the PID controller.

Suggestions from the autotuner were very inconsistent between attempts. Testing some of the suggested parameters yielded recommendations that were very aggressive while some highly underdamped. Target bandwidth was tested for target convergence time in the range from 0.2 to 10 second target. Target bandwidth parameter has the unit of rad/sec, and target convergence time was multiplied with 2π.

Drilling causes major vibrations making frequency response analysis difficult. The controller was tuned manually by observing system response. Slow responsiveness was compensated for by increasing the P gain value and stationary deviations in WOB by increasing the I gain. Overshoots while converging was compensated for by reducing the I gain, and oscillations by decreasing P.

Table 13: WOB PID Parameters.

Description	**Notation**	**Value**
Proportional	P	0.354
Integral	I	0.25
Derivative	D	0

Autotuning was not tested with the bits from Lyng Drilling due to time constraints. Less vibrations may give better tuning results.

8.3 Azimuth Torque Controller

The azimuth motor rotates the BHA and forces the bit into the well path wall to change azimuth while drilling. A torque is exerted from the azimuth control system that must be kept within an upper and lower limit to avoid mechanical damage. The controller should attempt to maintain the rotational velocity setpoint value until the torque limits are exceeded. If the torque limit is exceeded the controller should maintain a rotational velocity as close to the setpoint as possible. Setpoint have the unit of deg/min for drill pipe rotation.

A PID controller can be used for this purpose. Depending on the rotation direction, the torque setpoint is set to the torque limit in the direction of rotation. The output from the PID controller is normalized to not exceed an absolute value of 1. The output from the controller is then multiplied with the absolute value of the setpoint. Backcalculation anti-windup is used to avoid slow response and overshoot when approaching the torque limit.

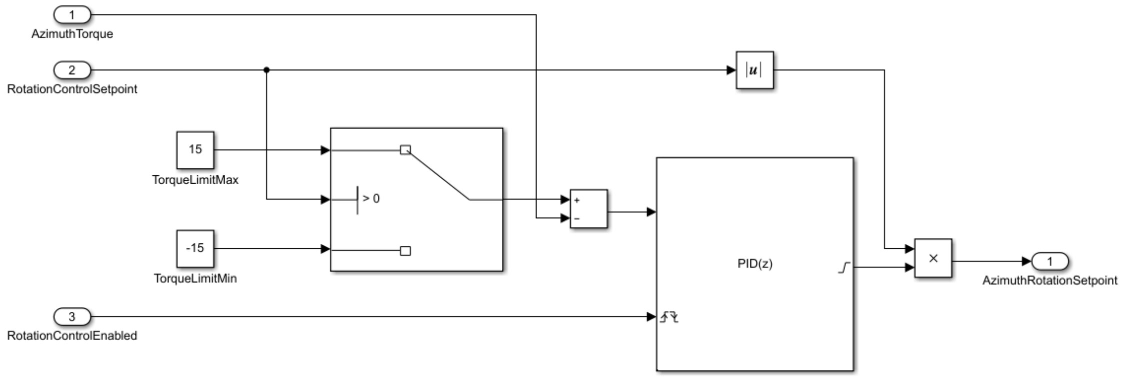


Figure 47: Azimuth torque controller implementation.

During normal operation, the controller output will be saturated and the PID will try to maintain the closest possible velocity value to the setpoint.

8.3.1 Tuning

The controller was tuned manually to a slightly underdamped behaviour. Changes in azimuth torque is slow with less process noise than the WOB controller. Tuning was done by setting the torque limit value within the range of what is normally observed during rotation inside the rock. The controller was enabled until the torque converged to the limit and the response observed. Underdamped performance was achieved by setting P lower than what a critically dampened controller would have and having a relatively small I parameter to avoid major overshoots.

Table 14: Azimuth Control PID Parameters.

Description	Notation	Value
Proportional	P	0.4
Integral	I	0.2
Derivative	D	0

8.4 Kalman State Estimation

A Kalman filter can be used as a low-pass filter for estimating system states. Linear Kalman filter blocks are simpler to implement in Simulink for this purpose than making an observer from multiple blocks or a MATLAB Function block.

8.4.1 State-Space Representation

Linear system models can be written in the state-space representation describing system dynamics ODEs using matrices. States and inputs are given as vectors.

$$\dot{x} = Ax + Bu \tag{8a}$$

$$y = Cx + Du \tag{8b}$$

In Equation 8, x is the state vector, u is the input vector, A is the state matrix, B is the input matrix, C is the output matrix, and D is the feedthrough matrix[82]. The state-space model can be discretized by approximating the derivative of the state vector.

$$\dot{x}(t) \approx \frac{x(t) - x(t - \Delta t)}{\Delta t} \tag{9}$$

Using the approximation in Equation 9 and rearranging Equation 8 yields a discretized state-space representation. A_d and B_d may be different from A and B.

$$x_{k+1} = A_d x_k + B_d u_k \tag{10a}$$

$$y_k = C_d x_k + D_d u_k \tag{10b}$$

8.4.2 Kalman Filter Theory

A Kalman filter is a state estimator for a stochastic system. Kalman filters provides the optimal state estimate based on the covariances of one or multiple measurements. Kalman filters enables sensor fusion and estimates can be combined and weighted based on their covariances. The standard implementation of the Kalman filters assumes that all measurements have a Gaussian distribution around a mean value. Width of the distribution depends on the expected deviation for each estimate called the covariance.

Kalman filters rely on a system model. The filter will first provide an estimate of the future output based on the current state and the current input called the prior estimate. The principles of the Kalman filter are illustrated using a state-space model for convenience.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{11}$$

The covariance of an estimate may change over time. The Kalman filter uses a predicted prior estimate of the covariance to weight the current estimate towards other measurements and estimates. The estimate depends on the previous covariance and the process noise covariance Q.

$$P_k^- = AP_{k-1}A^T + Q \tag{12}$$

Each iteration of the algorithm updates the estimate. A value used to minimize the variance of the optimal estimate is the Kalman gain. Kalman gain depends on the measurement covariance R. A larger measurement noise covariance result in a smaller Kalman gain.

$$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R} \tag{13}$$

A new state estimate is calculated from the Kalman gain, prior estimate, and system output.

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-) \tag{14}$$

The new error covariance is calculated based on the Kalman gain and prior estimate.

$$P_k = (I - K_k C)P_k^- \tag{15}$$

The combination of the previous equations results in the optimal state estimate for a state-space system.

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k)) \tag{16}$$

8.4.3 Simulink Implementation

Linear Kalman filter has a dedicated block in Simulink. A linear state-space model is specified within the block and represent the system model. System matrices are defined within the block with inputs for measurement y and input u, and an output with the filtered estimate.

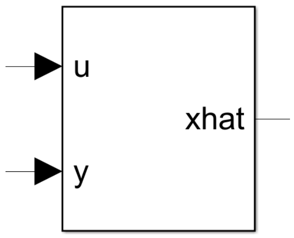


Figure 48: Linear Kalman filter block.

8.4.4 Tuning

Kalman filters are tuned based on a covariance estimate of each state. A covariance estimate can be found with statistical analysis of sampled data. The sampled data should be as equal to the operating conditions as possible. Covariance can be adjusted during runtime in Simulink and can be modelled or changed depending on operating mode and conditions.

8.5 State Observer

Linear Kalman filters can be used as a low-pass filter observers by setting the state-space matrices A and C to one and adjusting measurement and state covariances. Linear Kalman filters has a dedicated Simulink block that makes it simple to implement and can be tuned based on covariance of the input data.

$$x[k+1] = x[k] \tag{17a}$$

$$y[k] = x[k] \tag{17b}$$

The observer is tuned using the expected covariance of the input data. When using the Kalman filter as an observer, the measurement noise scalar will adjust how much each new measurement is expected to deviate from the mean value. Setting this value higher will give a slower response to measurement changes while being more noise resistant.

8.6 Hoisting and Azimuth Velocity Estimation

It is useful for the operator to know the current Rate Of Penetration (ROP) and rotational velocity while drilling to monitor progress and quantify drilling performance. The estimate should be responsive to sudden changes in movement while outputting a consistent value with no major fluctuations while at a constant velocity.

A state observer will give a low-pass estimate that will be more representative to velocity over time. Stationary deviations between RPM and rotational velocity estimates from change in position was observed on both the hoisting and azimuth motor. This is assumed to be caused by limited RPM output resolution. Change in position was therefore used to estimate velocity. Time-step velocity can be estimated from difference in current and last position gained by the inverse of the time-step.

Parameters for hoisting velocity are specified in Table 15 and azimuth velocity in Table 16.

Table 15: Hoisting Velocity Observer Parameters.

Description	**Notation**	**Value**
Process Noise Covarinace	Q	0.01
Measurement Noise Covarinace (ROP setpoint $\neq 0$)	R	50
Measurement Noise Covarinace (ROP setpoint $= 0$)	R	1
Process and Measurement Noise cross-covariance	N	0

Table 16: Azimuth Rotation Velocity Observer Parameters.

Description	**Notation**	**Value**
Process Noise Covarinace	Q	0.01
Measurement Noise Covarinace (ROP setpoint $\neq 0$)	R	5
Measurement Noise Covarinace (ROP setpoint $= 0$)	R	1
Process and Measurement Noise cross-covariance	N	0

Estimates are tuned conservatively to reduce fluctuations when using the WOB or azimuth torque controller and intended to give an indication of drilling performance over a short time frame to the operator. Estimates will take some time to converge after major changes in setpoint. Responsiveness when disabling rotation is improved by changing the measurement noise covariance if the velocity setpoints are set to 0.

8.7 System State Estimations

Estimates of fluctuating states such as torques and WOB are useful for the operator while drilling. Plotting values to provide a short-term perspective is highly resource demanding and multiple less important plots are distracting to the operator. Kalman state observers can be tuned to provide a responsive short-term window of system states. Estimates are displayed as a numeric value in the GUI.

State estimate observer parameters are shown in Table 17.

Table 17: System State Observers Parameters.

Description	**Notation**	**Value**
Top Drive RPM Process Noise Covariance	Q	0.1
Top Drive RPM Measurement Noise Covariance	R	1
Top Drive Torque Process Noise Covariance	Q	0.1
Top Drive Torque Measurement Noise Covariance	R	1
Hoisting Torque Process Noise Covariance	Q	0.05
Hoisting Torque Measurement Noise Covariance	R	5
Azimuth Torque Process Noise Covariance	Q	0.01
Azimuth Torque Measurement Noise Covariance	R	10
WOB Process Noise Covariance	Q	0.01
WOB Measurement Noise Covariance	R	10

Process and Measurement Noise cross-covariance were all set to 0. Minimal effort was put into making very accurate estimations. Raw measurements were plotted simultaneously with the estimates and observed to be responsive to significant changes in state and with no major fluctuations for relatively constant values.

8.8 Hoisting and Top Drive Hardware Communication Implementation

The communication structure from subsection 7.8 is implemented in a MATLAB function block. The commands to the drives are sent as integer values corresponding to the range of maximum and minimum of motor output. Values are converted to and from intuitive units in the Modbus communication subsystem.

8.8.1 Hoisting Inputs

ROP in cm/min is passed as a setpoint to the hoisting motor. The setpoint is passed through gain block to convert ROP to RPM, and output RPM to motor RPM using a gear ratio gain. Motor RPM is converted to a 16-bit unsigned integer output that the hoisting inverter can accept. The 16-bit conversion is taken from LabVIEW code previously used to control the rig[67].

$$u_{uint16}[k] = 4.81867 \cdot u_{Motor,RPM}[k] + 16383.5 \tag{18}$$

Hoisting torque limit is converted to a 16 bit output by Equation 19.

$$u_{uint16}[k] = \frac{32768}{2 \cdot 4.68} \cdot u_{TorqueLimit}[k] + 16384 \tag{19}$$

Hoisting inputs is floored to closest integer and clamped to the range between 0 and 65535. Hoisting control word is set by a GUI function.

8.8.2 Top Drive Inputs

RPM is an intuitive unit for the user to control the top drive. The input is multiplied with the top drive gear ratio before the same conversion as for the hoisting from Equation 18 is applied and subsequently floored to closest integer value before being clamped to 0 to 65535 range.

Torque limit is affected by the difference in top drive and hoisting motor gear ratios. Top drive torque is calculated from a gear ratio relation compensation of Equation 19.

$$u_{uint16}[k] = \frac{8.935}{2.579} \cdot (\frac{32768}{2 \cdot 4.68} \cdot u_{TorqueLimit}[k]) + 16384 \tag{20}$$

Control word passing is implemented equally to the hoisting motor.

8.8.3 Hoisting Outputs

Motor measurements are outputted as 16-bit unsigned integer values. Conversions are reused from previous LabVIEW code.

$$y_{Motor,RPM} = \frac{y_{uint16} - 16384}{4.81867} \cdot 100 \tag{21}$$

The motor RPM output is converted to ROP by multiplying with the inverse gearbox ratio and reverse ROP to RPM gain. A misconfigured scaling factor in Lenze Engineer requires a gain factor of 100. Scaling was not altered to remain compatible with LabVIEW code.

Hoisting position is represented by a 32-bit unsigned integer. Position is reset to zero when the driver is powered cycled. Moving the hoisting system upwards will cause an integer underflow and output a value in the upper range for a 32-bit unsigned integer. Power cycling the hoisting motor to reset the position is impractical. Simple logic was implemented to check if the position has underflowed.

```
function signedPosition = fcn(position)
    % Maximum value of uint32 is 4.294.967.295 approx 4.2e9
    if position > 2e9
       % Underflow has occurred
       signedPosition = position - double(intmax('uint32'));
    else
        % No underflow
        signedPosition = position;
    end
end
```

Positions are represented by a 32-bit unsigned integer corresponding to approximately 34 meters of hoisting movement. Cut-off logic is set at around 17 meters of either upwards or downwards movement. The hoisting system can only move less than 2 meters.

Hoisting position output is divided by the relation between motor encoder ticks and movement in position found in LabVIEW scripts calibrated by previous teams[67].

$$y_{Position,mm} = \frac{y_{Position,bits}}{117113} \tag{22}$$

Millimetres are thereafter converted to centimetres and outputted to the rest of the Simulink model.

Hoisting torque conversion from bits is the inverse calculation of Equation 19 with the same misconfigured scaling factor in Lenze Engineer.

$$y_{Torque,Nm} = \frac{y_{Torque,bits} \cdot 4.68 \cdot 2}{32768} \cdot 100 \tag{23}$$

8.8.4 Top Drive Outputs

Top drive RPM is calculated similarly to the hoisting found in Equation 21 multiplied by the inverse gear ratio of the motor.

Position from the top drive is disregarded by the control system. The value is received by Simulink and could be used in the future.

Torque is converted in the same way as with hoisting using Equation 23 with an inverse gear ration relation gain.

$$y_{Torque,Nm} = \frac{2.579}{8.935} \cdot \left(\frac{y_{Torque,bits} \cdot 4.68 \cdot 2}{32768} \cdot 100\right) \tag{24}$$

8.9 Azimuth Hardware Communication

The azimuth system has the same structure and as the hoisting and top drive communication with setpoints and outputs converted to and from intuitive units.

8.9.1 Inputs

The azimuth control drive takes a signed 16-bit integer values as input. Rotation velocity is inputted to the block in degrees per minute of drill pipe rotation. Degrees per minute is converted to RPM of the azimuth system and multiplied with the azimuth gearbox ratio. The conversion from motor RPM to input is shown in Equation 25.

$$u_{Input} = -4 \cdot u_{Motor,RPM} \tag{25}$$

Motor is positive clockwise. Coordinate frames are positive anticlockwise and the input is therefore inverted. The input value is rounded and clamped to the maximum acceptable input to the drive of ± 2160[64].

Operation mode of the drive is set using an opcode passed from GUI inputs.

8.9.2 Outputs

Azimuth motor measurements RPM is outputted in the same unit as the input. Values must be converted to RPM using the inverse calculation of Equation 25.

$$y_{Motor,RPM} = -\frac{1}{4} \cdot y_{Output} \tag{26}$$

The motor RPM is multiplied with the inverse of the gear ratio and converted to degrees per minute.

Position is measured with a rotational encoder in the motor. The datasheet states that there are 10.000 ticks for each full revolution[64], with the inverse being used to calculate the number of revolutions from the position output. Revolutions are converted to orientation in degrees gained with the inverse of the azimuth gearbox ratio.

8.9.3 Velocity Control Limitations

Inputs to the motor can only be adjusted to integer precision. Gearbox output rotation precision can therefore only be adjusted in steps of around 0.5 deg/min. Minimum setpoint for the motor to move is around 1.0 deg/min assumed to be a minimum acceptable motor RPM estimated to be around 8 RPM. Precision is more than sufficient for orientation control but may cause irregularities between setpoints and measured velocity.

8.10 DAQ Implementation

Reading data from the DAQ is descried in subsection 7.1. An Analog Input block is placed in an atomic subsystem sampled at the desired sampling rate with a block size of 2. The atomic subsystem with the Analog Input is sampled at half the desired rate matching the output rate of the Analog Input block. The output is sent to a rate converter and passed to an atomic subsystem sampled at the desired sampling rate. The sample block is split and both measurements delayed by a unit step to when they were sampled. Outputs are converted to intuitive outputs using Equation 1 and Equation 2.

8.11 Incident Detection and Reaction

An autonomous system must detect and react to drilling incidents. Incidents can be detected by monitoring system state measurements. Appropriate action can be taken by the control system automatically without operator intervention.

8.11.1 Stuck Bit

Stuck bit occurs when the bit is not rotating while the top drive motor rotates and winds up the drilling rod. The incident can be detected by a spike in top drive torque. The control system must react quickly to prevent twist-off, plastic deformation, and fatigue in the drilling rod.

Normally the top drive torque will have a consistent low torque in the range between 0.1 and 0.5Nm while drilling. If the bit gets stuck there will be a sudden spike in torque stabilizing at the top drive torque limitation, generally set at 1.0Nm. The bit is assumed to be stuck if one of the last 3 measurements in a rolling window exceeded 0.5Nm.

If stuck bit occurs, the WOB controller sets the WOB setpoint to -5kg-f. ROP velocity will reverse and lift the bit. Lifting the bit from the rock will usually be enough for the bit to start rotating again and drilling can be continued as shown in **??**.

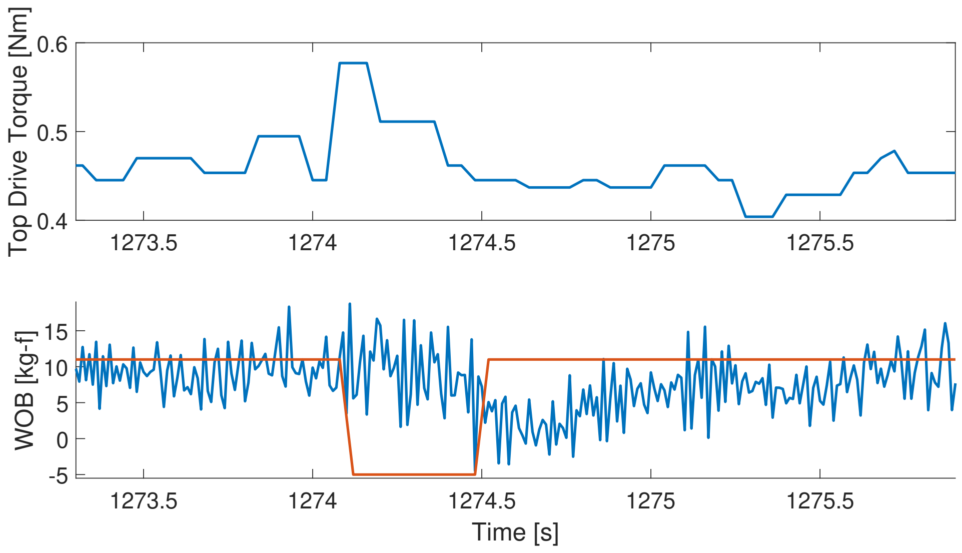


Figure 49: Stuck bit incident and reaction. WOB is lowered and the drilling operation can continue.

8.11.2 Twist-Off/Rod Wind-Up

Observations while drilling suggests that the bit rarely starts rotating again after being stuck for longer than 4 seconds. Depending on the drilling rod material the drilling rod will either twist-off or be deformed inside the drill pipe. If the stuck bit top drive torque threshold is exceeded for 3 seconds the drilling rod has likely deformed and the operation must be aborted. A sudden drop in top drive torque after a longer period of stuck-bit suggests a twist-off has occurred.

8.11.3 MD and Orientation Target Reached

It is possible to set a target to stop hoisting movement and azimuth rotation when a certain MD or orientation is reached. The incident remains active until a new target is set or the WOB or rotation controller has been disabled.

8.12 Concurrent Execution

All blocks in the low-level control systems are implemented as atomic subsystems to allow concurrent execution and utilization of multiple CPU cores. Dedicated tasks are created for the most computationally demanding operations such as motor and DAQ communication. Transferring data between threads may cause a unit delay from synchronization in Simulink. Tasks are configured to avoid unit delays between latency critical systems such as closed-loop controllers and measurements.

8.13 Update Frequency

For the control system to be responsive and react to drilling incidents, the DAQ and motors are sampled and new setpoints sent multiple times a second. Rates are mostly limited by computational resources and the computational budget must be prioritized between the components. Computational cost for sample rate configurations is tested using the Simulink Profiler.

8.13.1 Sample Rates

Computational cost of hardware communication limits the maximum sample rate from the drives. The DAQ is mainly limited by the USB interface and that the device is not intended for continuous real-time operation.

Top drive motor measurements are used to detect torque spikes to prevent stuck bit and the hoisting motor must react quickly to incidents and give a responsive ROP estimate to the operator. The samples rates of the hoisting and top drive cannot be set independently as they are both connected to the gateway and share the same Modbus interface. The position from the azimuth motor is sampled frequently to provide a responsive rotation velocity estimate of the azimuth system. The GUI must be updated frequently enough to inform the user about drilling incidents with a perception of being responsive. Table 18 is a list of implemented sampling frequencies.

Table 18: Low-Level Control Sample Rates.

Component	Sample Rate[Hz]	Period[s]
DAQ	100*	0.01*
Hoisting Motor	25	0.04
Top Drive Motor	25	0.04
Azimuth Motor	12.5	0.08
GUI Input Poll	25	0.04

DAQ is sampled with a 100Hz frequency delayed by a single time-step described in subsubsection 7.1.1. Azimuth motor sample rate is set smaller than the hoisting and top drive motor as the states changes are slower while communication has approximately the same computational cost for each sample. GUI polling has a small computational cost and is set high to decreases latency for user inputs.

8.13.2 Update Rates

How often a command is sent to a motor is set differently from the sample rate to reduce computational cost. Updating motor setpoints more frequently has diminishing returns as the motor is unable to converge infinitely fast to the setpoint value. Write are set to be executed at a certain rate relative to the reads defined as the write interval. Hoisting and top drive commands is sent as a single Modbus command and will have the same update frequency.

The hoisting motor setpoints are frequently altered by the WOB controller and must have a high update rate. Top drive RPM is rarely changed during operation but is sent with the same

Table 19: Low-Level Control Update Rates.

Component	Update Rate[Hz]	Period[s]	Write Interval
Hoisting Motor	12.5	0.08	2
Top Drive Motor	12.5	0.08	2
Azimuth Motor	≈4.17	0.24	3
GUI Output	2.5	0.40	-

aggregated write command to the gateway. Observations during drilling indicates that azimuth torque will change slowly, and the write rate can therefore be set lower relative to the hoisting and top drive. GUI frequency is set to the minimum for what is perceived as responsive for the user when monitoring drilling parameters.

8.13.3 Improvements

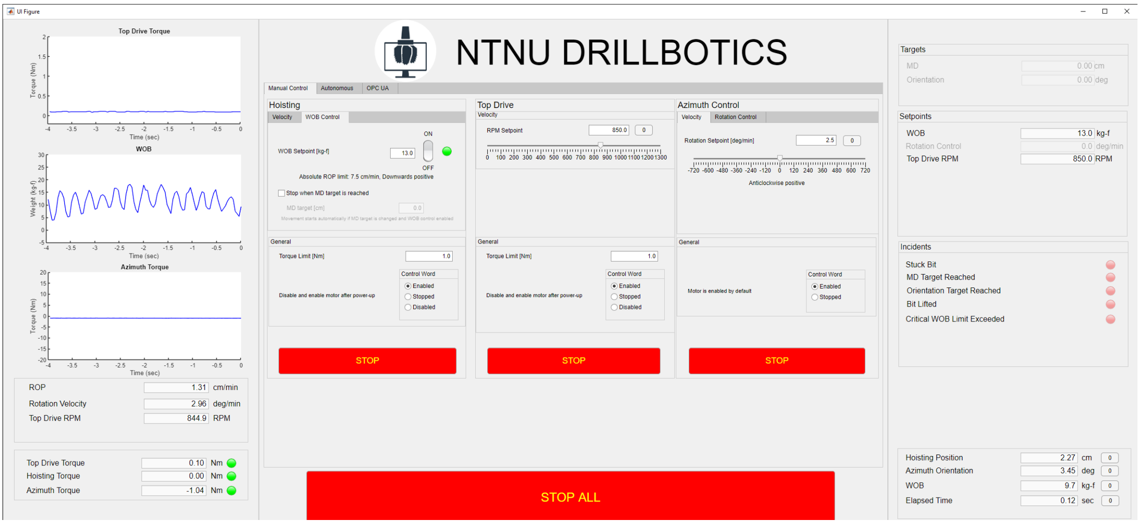
Sample rates are apart from the DAQ limited by computational cost. The DAQ would have to be replaced to achieve higher sampling rates in real-time operation. A reduction in computational cost for Modbus communication or a more capable computer would allow the sample and update rate of the hoisting, top drive, and azimuth motor to be increased. The gateway has a maximum sample rate of 1000Hz and is unlikely to be the limiting factor.

No upper limit to hoisting, top drive or azimuth communication rates have been found. Communication is tested for a single read or write command up to 200Hz in a Simulink model with no other blocks present. Communicating with the azimuth motor over EtherNet/IP would allow all motor sample and update rates to be increased as discussed in subsection 7.7.

Analytically determining minimum required sampling and update rates are left for future work due to time restrictions.

8.14 GUI

The control system HMI is a GUI created in MATLAB App Designer. The GUI is designed to be intuitive to use without in-depth knowledge of the drilling system.



ROP	1.31	cm/min
Rotation Velocity	2.96	deg/min
Top Drive RPM	844.9	RPM

Top Drive Torque	0.10	Nm	●
Hoisting Torque	0.00	Nm	●
Azimuth Torque	-1.04	Nm	●

Hoisting
Velocity | WOB Control
WOB Setpoint [kg-f] 13.0 ON [✓ OFF] ●
Absolute ROP limit: 7.5 cm/min, Downwards positive
☐ Stop when MD target is reached
MD target [cm] 0.0
Movement starts automatically if MD target is changed and WOB control enabled

General
Torque Limit [Nm] 1.0
Disable and enable motor after power-up
Control Word: [✓ Enabled] Stopped Disabled

Top Drive
Velocity
RPM Setpoint 850.0 0

General
Torque Limit [Nm] 1.0
Disable and enable motor after power-up
Control Word: [✓ Enabled] Stopped Disabled

Azimuth Control
Velocity | Rotation Control
Rotation Setpoint [deg/min] 2.5 0
Anticlockwise positive

General
Motor is enabled by default
Control Word: [✓ Enabled] Stopped

Targets

MD	0.00	cm
Orientation	0.00	deg

Setpoints

WOB	13.0	kg-f
Rotation Control	0.0	deg/min
Top Drive RPM	850.0	RPM

Incidents

Stuck Bit	●
MD Target Reached	●
Orientation Target Reached	●
Bit Lifted	●
Critical WOB Limit Exceeded	●

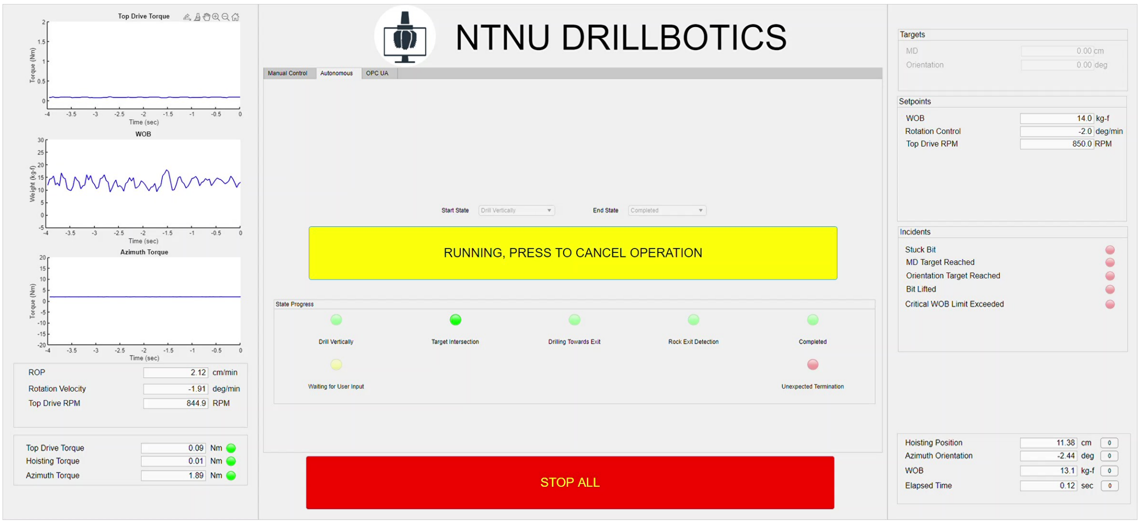
Hoisting Position	2.27	cm	0
Azimuth Orientation	3.45	deg	0
WOB	9.7	kg-f	0
Elapsed Time	0.12	sec	0

Figure 50: Low-level control GUI in manual operation mode.

Data that must be closely monitored over time has a 4 second plot of high frequency data displayed to the user. Plots are computationally expensive and must be kept to a minimum. Less important drilling states are displayed as numerical displays. Plots are placed on the left to be visible in the peripheral vision while looking at the drilling operation. Less important states values are located on the right. Incidents are displayed as lamps to indicate their binary states. The centre of the GUI changes based on operating mode. An emergency stop button is always visible and will immediately stop all motors if pressed.

The rig is manually controlled by inputting setpoints and drive commands with input fields, sliders, toggles, and radio buttons. Different motor control modes are separated in tabs to avoid operator confusion and inconsistent input states. Setpoint inputs are range limited to reduce the risk of operator error.

Autonomous mode can be configured and initialized from the autonomous tab. State progress is indicated by lamps, and the start button will change based on current state.



Start State: Drill Vertically
End State: Completed

ROP	2.12	cm/min
Rotation Velocity	-1.91	deg/min
Top Drive RPM	844.9	RPM

Top Drive Torque	0.09	Nm
Hoisting Torque	0.01	Nm
Azimuth Torque	1.89	Nm

Targets

MD	0.00	cm
Orientation	0.00	deg

Setpoints

WOB	14.0	kg-f
Rotation Control	-2.0	deg/min
Top Drive RPM	850.0	RPM

Hoisting Position	11.38	cm	0
Azimuth Orientation	-2.44	deg	0
WOB	13.1	kg-f	0
Elapsed Time	0.12	sec	0

Figure 51: Low-level control GUI in autonomous operation mode.

8.15 State Machine

Autonomous operation is controlled by a state machine described in subsection 11.1. The state machined is in the low-level control system to be controllable by the GUI, have minimal response time to incidents, and allowing the operation to be stopped even if a high-level control communication error has occurred.

8.16 Data Logging

Post analysis of data requires measurements, setpoints, incidents, and estimates to be logged during the drilling operation. Data is written to file by a dedicated Simulink block as a MAT file. MAT files that can be opened as an array in MATLAB workspace. The array will contain the simulation time and multiplexed values for each time-step as a column. Data files are stored in a dedicated folder. If the model is restarted, any previous data files will be overwritten.

Separate logging files are used for the DAQ, hoisting and top drive, and azimuth system as they may have different sample rates and each logging block has minimal impact on the execution time according to the Simulink profiler.

9 Position Estimation

To intersect the target points the position controller must know the current position of the BHA inside the rock. Measuring absolute position inside the rock with a high degree of accuracy is difficult. Position is estimated from the assumption that the BHA will move in the direction of its current orientation at the rate of the hoisting movement.

9.1 Reference Frame

The centre of the top of the rock sample is defined as the zero point of the system. The z-axis is defined positive downwards, the x-axis positive in the opposite direction of the door for inserting the rock, and y-axis in the direction away from the operator desk. The rock sample is aligned perpendicular to the rig frame.

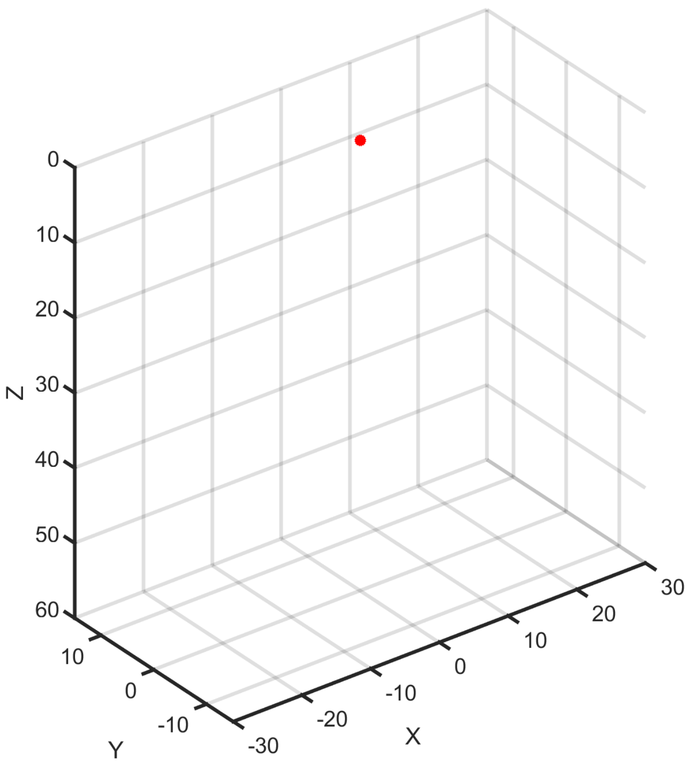


Figure 52: Inertial coordinate frame as seen from the drilling operator desk. Zero point is indicated in red. All units are in cm.

Orientation is represented using Euler angles. Euler angles are more intuitive to use than quaternions with the drawback of having a singularity point. The singularity occurs if $\cos(\theta)$ is zero. Because inertial pitch of 90 degrees will not occur during drilling due to mechanical constraints, the singularity is not a concern.

9.2 Rotation Matrices

Finding relative orientation between frames is often useful. Rotation matrix properties are described in Equation 27.

$$(R_a^b)^{-1} = (R_a^b)^T = R_b^a \tag{27a}$$

$$R_b^c R_a^b = R_a^c \tag{27b}$$

$$det(R_a^b) = 1 \tag{27c}$$

The inverse of a rotation matrix is equal to its transpose. The inverse reverses the frame translation origin and destination. Rotation matrices can be sequenced to transform rotations with an intermediate frame. The sequence order of the rotation matrices is arbitrary. Rotation matrices rotates a vector from one frame to another without affecting the magnitude of the vector.

Euler angle rotation matrices are shown in Equation 28.

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{28a}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{28b}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{28c}$$

Euler angle rotation matrices can be sequenced to rotate from one Euler frame reference to another as shown in Equation 29.

$$R_b^a = R_z(\psi)R_y(\theta)R_x(\phi) \tag{29}$$

The subscript of the rotation matrix is the original frame, and the superscript is the destination frame.

9.2.1 Angular Velocity Transformation

Rotation matrices can be used to translate angular velocities between frames.

$$\omega_{ab}^a = \omega(\dot{\psi}) + R_z(\psi)\omega_y(\dot{\theta}) + R_z(\phi)R_y(\theta)\omega_x(\dot{\phi}) \tag{30}$$

With $\omega(\dot{\psi})$, $\omega_y(\dot{\theta})$, and $\omega_x(\dot{\phi})$ being shorthand notations for angular velocity vectors in Equation 31.

$$\omega(\dot{\psi}) = \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}, \omega_y(\dot{\theta}) = \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix}, \omega_x(\dot{\phi}) = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} \tag{31}$$

9.3 BHA Frame

It is more intuitive to make a system model based on the frame of the BHA. The frame of the BHA can be translated to the inertial frame with rotation matrices and positional displacements to obtain differential equations describing system behaviour.

The zero coordinate of the BHA is placed at an extension of the straight BHA at the length of the lowest edge of the bit. The zero point is defined at this point as it can easily be defined when by tagging the rock.

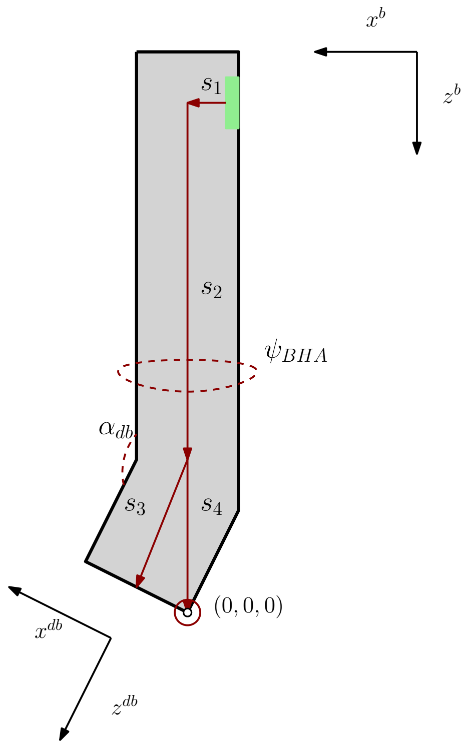


Figure 53: BHA reference frame. Zero point is set at the lowest point of the angled bit when tagging the rock.

Placing the zero point at the bit would increase dependency on an accurate bend angle orientation estimate as the bit changes position with BHA rotation, increasing uncertainty. Position of the bit can be found from the positional displacement relative to the zero point and a rotation matrix with the orientation estimate. Notation is defined in Figure 53. The orientation translation from the sensor to the bit is shown in Equation 32.

$$R^b_{db} = \begin{bmatrix} \cos(\alpha_{db}) \cdot \cos(\psi_{BHA}) & -\sin(\psi_{BHA}) & \sin(\alpha_{db}) \cdot \cos(\psi_{BHA}) \\ \cos(\alpha_{db}) \cdot \sin(\psi_{BHA}) & \cos(\psi_{BHA}) & \sin(\alpha_{db}) \cdot \sin(\psi_{BHA}) \\ -\sin(\alpha_{db}) & 0 & \cos(\alpha_{db}) \end{bmatrix} \tag{32}$$

The positional displacement from the sensor to the bit is shown in Equation 33.

$$\Delta p = \begin{bmatrix} s_1 + s_3 \cdot \sin(\alpha_{db}) \cdot \cos(\psi_{BHA}) \\ s_3 \cdot \sin(\alpha_{db}) \cdot \sin(\psi_{BHA}) \\ s_2 + s_3 \cdot \cos(\alpha_{db}) \end{bmatrix} \tag{33}$$

α_{db} is 7 degrees with the most recent BHA bend. Length of s_3 will depend on the bit. Current best estimates of s_1, s_2, and s_3 are 1cm, 8cm, and 3cm. s_4 will depend on the cutter configuration of the bit. Accurate lengths can be extracted from mechanical design drawings.

9.4 Digital Twin Simulation

It is practical to be able to test controllers and position estimation without access to the rig. A digital twin simulation allows fast paced prototyping of controllers and estimators. A key feature is that controllers can be tested and debugged in simulation before being applied to the physical rig. The digital twin simulation is based on a physics model of expected movement behaviour given the mechanical properties of the rig. The digital twin is implemented in Simulink.

9.4.1 Model

The model assumes that the BHA will move in the direction of its current orientation at rate of hoisting movement in Measured Depth (MD). Rate of orientation change is based on the bend angle of the BHA and defined as DLS. Modelled movements of the BHA are relative to its own frame. BHA frame orientation can be translated to the inertial frame with rotation matrices. Euler angles are used to describe the frame orientation in roll, pitch, and yaw.

$$\dot{\phi}_{BHA} = 0 \tag{34a}$$

$$\dot{\theta}_{BHA} = DLS \cdot \dot{d} \tag{34b}$$

$$\dot{\psi}_{BHA} = \dot{\psi}_{I} \tag{34c}$$

Change in hoisting position and azimuth orientation is denoted $\dot{d}$ and $\dot{\psi}_I$, respectively.

All movements in Equation 34 are relative to the zero point of the BHA. Angular velocity can be written on vector form to simplify further calculations.

$$\omega_{BHA} = \begin{bmatrix} \dot{\phi}_{BHA} \\ \dot{\theta}_{BHA} \\ \dot{\psi}_{BHA} \end{bmatrix} \tag{35}$$

Movements in the frame of the BHA can be translated to the BHA frame using rotation matrices. The rotation matrix can be simplified as the system model assumes no movement in roll relative to the BHA. Euler angle orientation is relative to the inertial frame.

$$\omega_I = \frac{1}{cos(\theta)} \begin{bmatrix} \cos(\theta) & \sin(\phi)\cdot\sin(\theta) & \cos(\phi)\cdot\sin(\theta) \\ 0 & \cos(\phi)\cdot\cos(\theta) & -\sin(\phi)\cdot\cos(\theta) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \omega_{BHA} \tag{36}$$

Integration of the angular velocity during the drilling operation yields the orientation. The BHA is assumed to move in the direction it is orientated at the rate of change in MD equal to velocity of the hoisting system.

$$\dot{p_I} = R^I_{BHA} \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{d} \end{bmatrix} + \dot{R}^I_{BHA} \cdot \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix} \approx R^I_{BHA} \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{d} \end{bmatrix} \tag{37}$$

The rate of change of the orientation matrix is minuscule and estimated to be zero.

9.4.2 Modelled Build Rate

Dogleg Severity (DLS) is a measurement in the petroleum industry defining how much angle a bent sub will build over a certain distance. DLS is defined in this system as an expected ratio of angle build rate to movement in MD in degrees. Inertial orientation will accumulate as the BHA moves in MD in the direction of the bend sub orientation.

$$DLS_{nom} = \frac{\alpha_{br}}{360} \tag{38}$$

Experimental results showed a strong correlation between rotational velocity of the azimuth system and decrease in build rate. Inspired by the formulas for normal distribution, a model was made for DLS given the current rotational velocity.

$$DLS = DLS_{nom} \cdot \exp\left(-\frac{\dot{\psi}_I^2}{q}\right) \tag{39}$$

With q being a tunable constant for build profile and DLS_{nom} the build rate without rotation. Distribution for some values of q is shown in Figure 54.

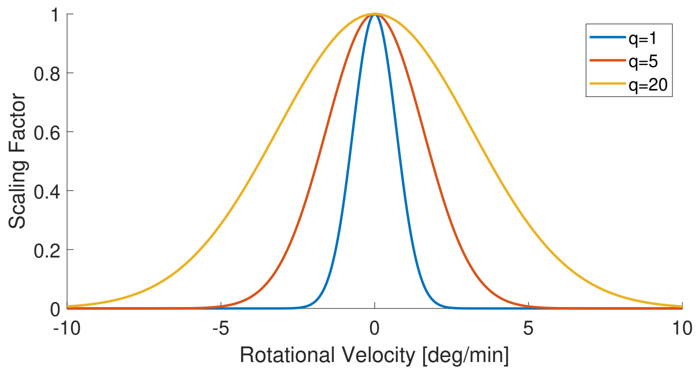


Figure 54: Scaling factors for different values of q with azimuth rotation.

9.4.3 Simulink Implementation

The model is implemented as a block diagram with MATLAB function blocks for more advanced calculations. Simulink will by default use the variable step `ode23t` solver for continuous time estimations. Variable step solvers are generally more efficient and accurate than fixed step solvers for continuous systems.

9.5 Model

Sensor data is sampled at discrete time intervals, requiring a discrete time model. The model is based on the continuous time model from subsubsection 9.4.1 to estimate the position in the inertial frame.

$$\hat{\phi}_{BHA}[k+1] = \hat{\phi}_{BHA}[k] = 0 \tag{40a}$$

$$\hat{\theta}_{BHA}[k+1] = \Delta t \cdot DLS \cdot \dot{d}[k] + \hat{\theta}_{BHA}[k] \tag{40b}$$

$$\hat{\psi}_{BHA}[k+1] = \Delta t \cdot \dot{\psi}_I[k] + \hat{\psi}_{BHA}[k] \tag{40c}$$

DLS relies on the same model as the digital twin found in Equation 39 with a sampled $\dot{\psi}_I[k]$. Equation 40 can be written as a vector to simplify further calculations.

$$\hat{\Theta}_{BHA}[k+1] = \begin{bmatrix} \hat{\phi}_{BHA}[k+1] \\ \hat{\theta}_{BHA}[k+1] \\ \hat{\psi}_{BHA}[k+1] \end{bmatrix} \tag{41}$$

Rate of change in the frame of the BHA must be converted to rate of change in the inertial frame. Rotation matrix rate of change is estimated to be zero.

$$\hat{\Theta}_I[k+1] = \frac{1}{cos(\hat{\theta}[k])} \begin{bmatrix} \cos(\hat{\theta}[k]) & \sin(\hat{\phi}[k]) \cdot \sin(\hat{\theta}[k]) & \cos(\hat{\phi}[k]) \cdot \sin(\hat{\theta}[k]) \\ 0 & \cos(\hat{\phi}[k]) \cdot \cos(\hat{\theta}[k]) & -\sin(\hat{\phi}[k]) \cdot \cos(\hat{\theta}[k]) \\ 0 & \sin(\hat{\phi}[k]) & \cos(\hat{\phi}[k]) \end{bmatrix} \hat{\Theta}_{BHA}[k+1] + \hat{\Theta}_I[k] \tag{42}$$

Orientation estimates are in the inertial frame. Integrating the change in orientation yields the current orientation estimate.

$$p[k+1] = \Delta t \cdot \hat{R}^I_{BHA}[k] \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{d}[k] \end{bmatrix} + p[k] \tag{43}$$

Rotation matrix is equal to Equation 29 with inertial frame orientation estimates at the current time-step.

The model relies on multiple estimates and there is significant process noise. Position estimation covariance will increase rapidly in open-loop. Measurements from the downhole IMU sensor can be used for closed-loop estimations of inertial BHA orientation.

9.6 Extended Kalman Filter

Nonlinear system models are used to improve orientation and position estimates. Linear Kalman filters requires a linear state-space model. Some models will not have an accurate linear estimation for the entire range of states and inputs, but will have a close linear approximation around an operating point, as shown in Equation 44.

$$f(x) \approx f(\hat{x}) + f'(x)(x - \hat{x}) \tag{44}$$

Linear Kalman filters requires noise values to have a Gaussian distribution to be an optimal estimator. Passing noise values through a non-linear function will in general not give a Gaussian output distribution. Kalman filters can assume that the noise will be approximately linearly distributed called additive noise as in Equation 45 or modelled as a part of the nonlinear function as shown in Equation 46 called nonadditive noise[28].

$$x_k = f(x_{k-1}, u_k) + w_k \tag{45a}$$

$$y_k = g(x_k) + v_k \tag{45b}$$

$$x_k = f(x_{k-1}, u_k, w_k) \tag{46a}$$

$$y_k = g(x_k, v_k) \tag{46b}$$

With x being the state, u the input, y the output, with w and v the noise values for states and outputs. Combinations of additive and nonadditive state and output equations can be used interchangeably.

The linearization can be used directly in the state-space modelled Kalman filter by using the Jacobian of the state and input.

$$F = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1},u_k} \tag{47a}$$

$$G = \frac{\partial g}{\partial x}|_{\hat{x}_k} \tag{47b}$$

If either the state or output function is linear, the Jacobians can be used interchangeably with state-space matrices. Discrete nonlinear approximations are shown in Equation 48.

$$\Delta x_k \approx F \Delta x_{k-1} + w_k \tag{48a}$$

$$\Delta y_k \approx G \Delta x_k + v_k \tag{48b}$$

9.6.1 Simulink Implementation

The nonlinear model is specified as a MATLAB function. The function takes the current state and input as parameters and outputs the next state estimate. Analytical Jacobian is an optional MATLAB function to speed up estimation. The Jacobian will be estimated numerically if no function is provided. Process noise type can be selected, with additive as the default option[28].

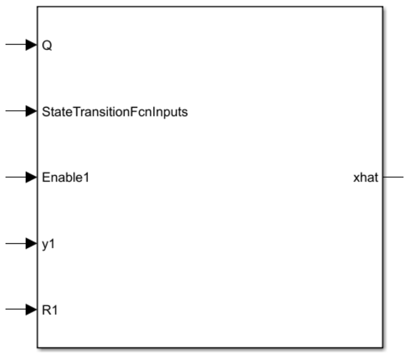


Figure 55: Extended Kalman filter block.

A covariance matrix is used to weight the model estimate accuracy. A small covariance implies an estimate of high accuracy. Covariance can be specified as time-varying and changed during runtime from an external input to the block.

Like the linear Kalman Filter block, the Extended Kalman filter supports multirate inputs and output of the covariance estimate.

9.7 Orientation Estimate

It is difficult to accurately measure position of the BHA while drilling. Process noise and an inaccurate position model requires closed-loop measurements to maintain a good approximation of orientation throughout the drilling operation. Deviations in the estimated orientation and BHA orientation will cause an accumulating error in the position estimate.

Orientation can be estimated from downhole accelerometer and magnetometer measurements. Gravity is aligned with the inertial z-axis and the acceleration vector can be used to estimate inertial roll and pitch from trigonometric relations.

$$\hat{\phi}[k] = \arctan \frac{a_y[k]}{a_z[k]} \tag{49a}$$

$$\hat{\theta}[k] = -\arctan \frac{a_x[k]}{a_z[k]} \tag{49b}$$

Yaw can be estimated from magnetometer measurements. The magnetometer measurement vector will point towards magnetic north with a magnetic field inclination. The x and y components of the magnetometer measurement are used to estimate the magnetic north heading.

$$\hat{\psi}_{north}[k] = \arctan - \frac{m_y[k]}{m_x[k]} \tag{50}$$

Quadrant alignment requires the `atan2` function to be used instead of `arctan` in code for Equation 49 and Equation 50.

It is not practical to align the rig with the direction of magnetic north. An offset between magnetic north and the inertial x-axis is added as a bias.

$$\hat{\psi}[k] = \hat{\psi}_{north}[k] - \psi_{offset} \tag{51}$$

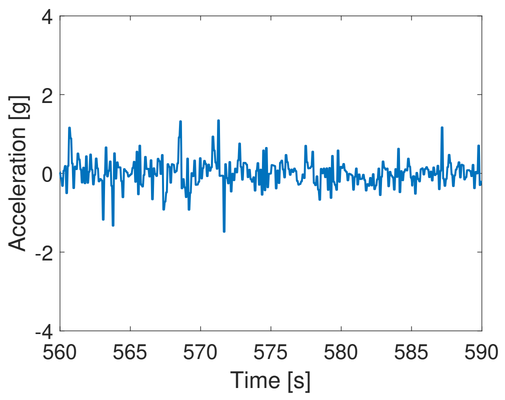
9.8 Downhole Accelerometer Data

The accelerometer will mainly measure the gravitational field and vibrations while drilling. Bit selection has a profound effect on drilling vibration magnitude. A tested bit from Alibaba had extreme noise values larger than ±3g from the mean, exceeding the measurement range of ±4g in some orientations. The bit from Lyng Drilling exceeding ±1.5g from the mean.

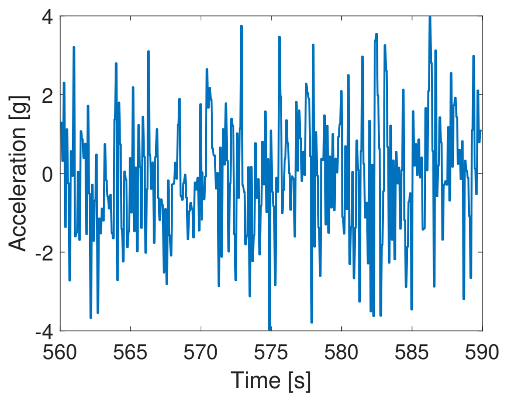
Looking at the data in Figure 56 it is possible to see a trend in the mean. Data does also appear to have a Gaussian distribution.

9.8.1 State Estimate Deviations While Drilling

It was observed that pitch and roll estimates would converge to different values depending on if the bit contacted the rock. Deviations could be as large as 4 degrees for a well-tuned low-pass filtered acceleration measurements. This is assumed to be caused by vibration patterns while drilling because of the angled bit orientation. This phenomenon was identified at the later stages of the project and finding a modelled relation for continuous orientation estimate is left for future work.



(a) Lyng bit.



(b) Alibaba bit.

Figure 56: Acceleration data from drilling tests with Lyng and Alibaba bits sampled at 10Hz.

9.8.2 Orientation Surveys

Mean deviation in acceleration requires bit rotation to be stopped to take accurate orientation measurements for roll and pitch. Stopping bit rotation and taking a measurement with less process noise is called a survey. Between surveys the orientation estimate relies on the position model. A survey interval of 5cm MD movement was chosen as a trade-off between time spent drilling and positional accuracy.

9.8.3 Kalman State Observer

Accelerometer data can be low-pass filtered with a Kalman state observer. The state observer is tuned for estimating orientation while bit rotation is disabled.

Table 20: Roll and Pitch Observer Parameters.

Description	**Notation**	**Value**
Process Noise Covarinace	Q	1e-3
Measurement Noise Covarinace	R	5
Process and Measurement Noise cross-covariance	N	0

9.8.4 Gravity Field Model

Expected gravitational vector measurements can be modelled from the current orientation estimate. The model assumes the gravitational field will point in the inertial z-axis direction with a strength of 1g. The gravitational vector is rotated with the inverse rotational matrix relative to Equation 37.

$$\hat{a}[k] = (\hat{R}^{I}_{BHA}[k])^{T} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{52}$$

The modelled Kalman filter was implemented as an attempt to further improve orientation estimates and make measurements more noise resistant while being responsive. An extended Kalman filter can model both the orientation and magnitude of the gravitational field and compare it to accelerometer measurements from the downhole IMU.

As shown in Figure 57, the model requires the subsequent orientation estimate to be used in the estimation of the current orientation in a cascade of two Kalman filters. The setup proved difficult

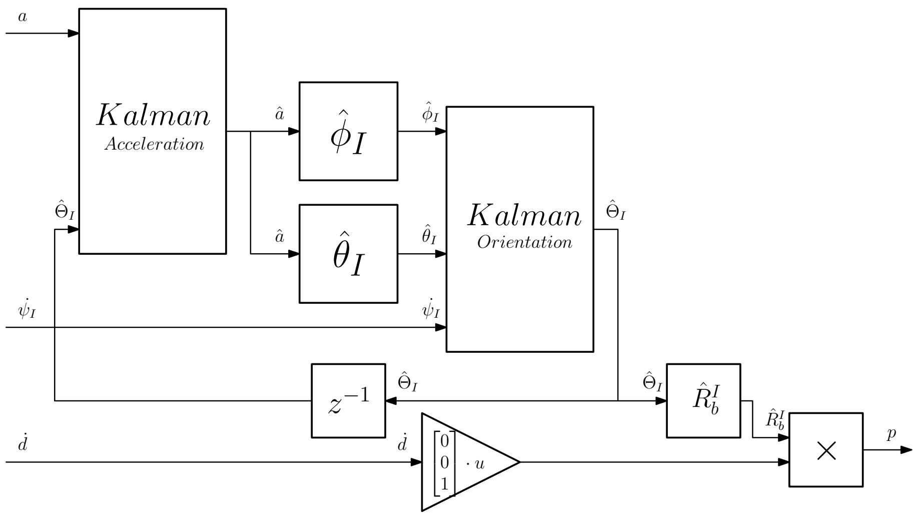


Figure 57: Gravity field model block diagram.

to tune. Reducing the covariance of the of the measurements gave a highly fluctuating estimate. Reducing the model covariance sometimes resulted in a run-away integration error for pitch and roll without the measurements being able to correct for it. An easier to tune Kalman state observer implementation was used instead.

9.9 Downhole Magnetometer Data

The magnetometer will measure the electromagnetic field of the earth and any soft and hard iron disturbances. The electromagnetic field of the earth can be estimated to be constant in the inertial frame for a small area. Earths electromagnetic field differs based on location and is estimated to have a 74°inclination with a 52-gauss magnitude in Trondheim[63]. Orientation and magnitude of the electromagnetic field is less reliable than that of the gravitational field. Magnetometer measurements are less affected by the drilling process noise compared to the accelerometer data.

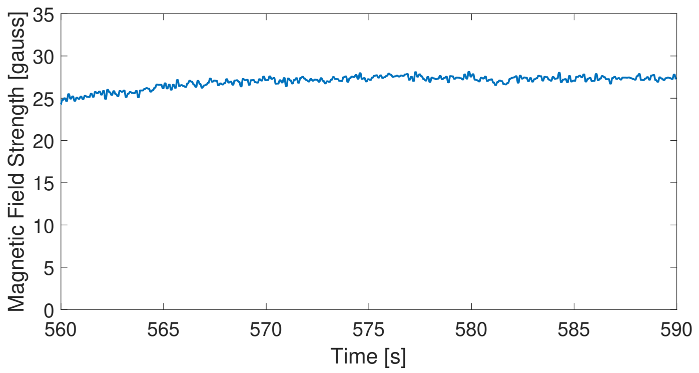


Figure 58: Magnetic field data snapshot sampled at 10Hz.

An ideal magnetic field for orientation would be orientated perpendicular to the gravity field. The high inclination magnetic field in Trondheim is not ideal for estimating yaw as Equation 51 uses the x and y components of the magnetic field vector. Magnetic field measurements are usually in the range ±30 gauss while drilling.

9.9.1 Kalman State Observer

Magnetometer measurements are low-pass filtered with a Kalman state observer. Magnetometer measurements are less affected by process noise and can be sampled continuously.

Table 21: Yaw Observer Parameters.

Description	Notation	Value
Process Noise Covarinace	Q	1e-2
Measurement Noise Covarinace	R	4e2
Process and Measurement Noise cross-covariance	N	0

9.9.2 Electromagnetic Field Model

A model for the magnetic field takes the product of a magnetic field inclination rotation matrix, the inverse rotation matrix estimate, and a vector with the magnitude of the magnetic field.

$$\hat{m}[k] = R_y(\theta_{mag}) \cdot ((\hat{R}^I_{BHA}[k])^T \cdot \begin{bmatrix} m_{magnitude} \\ 0 \\ 0 \end{bmatrix}) \tag{53}$$

Modelled Kalman filter tests suggested that the model was not accurate enough to further improve the yaw estimate from the unmodeled filter. The same issues as with the gravity model with cascading filter tuning and run-away integration error was present.

9.9.3 Nonuniform Magnetic Field

Drilling test with no azimuth rotation showed deviations in azimuth of up to 15 degrees from the expected value. The riser would alter the yaw estimate by multiple degrees and the estimate would drift over time. Multiple calibrations with and without all motors enabled was tested with no major changes in result. Experiments at the same position and orientation gave similar consistent drift results. Observations suggest that the magnetic field is nonuniform and affected by large metallic objects such as the rig frame.

9.9.4 Electromagnetic Field Mapping

As the magnetic field appears to nonuniform it would be possible to map and model the magnetic field. Such a model would have to rely on an orientation and position estimate to be effective. An accurate model could potentially be used to estimate current position based on estimated orientation.

9.9.5 Artificial Magnetic Field

A powerful magnet could be used to create an artificial magnetic field. The field could be orientated in any direction with a greater magnitude than earth's magnetic field up to the downhole sensor measurement range of ± 400 gauss[74]. A more elaborate setup could use shielding from external magnetic fields and strategically placed magnets creating a predictable or uniform magnetic field.

A small electromagnet was briefly tested inspired by previous teams[67]. Experiments placing the downhole sensor at different positions relative to the magnet suggested that the magnetic field strength was approximately equal to earth's magnetic field at 20cm. A more powerful magnet would be beneficial to have a stronger magnetic field than the earth at any position inside the rock.

9.9.6 Modelled Yaw

The most accurate yaw estimates while drilling was achieved by using the movement model. The model assumes that the BHA rotates as much as the azimuth controller in the frame of the BHA. Pitch and roll estimates are relatively accurate and testing suggest small measurable deviations in expected azimuth. The solution is suboptimal as it estimates yaw in open-loop. Further work is required to reliably have an accurate yaw estimate.

9.10 Downhole Gyroscope Data

Major vibrations, slow changes in orientation, and significant measurement noise makes gyroscope readings difficult to utilize to improve orientation estimates. Measurements are therefore disregarded by orientation estimators to reduce complexity. Rate of change would have to be integrated to estimate orientation.

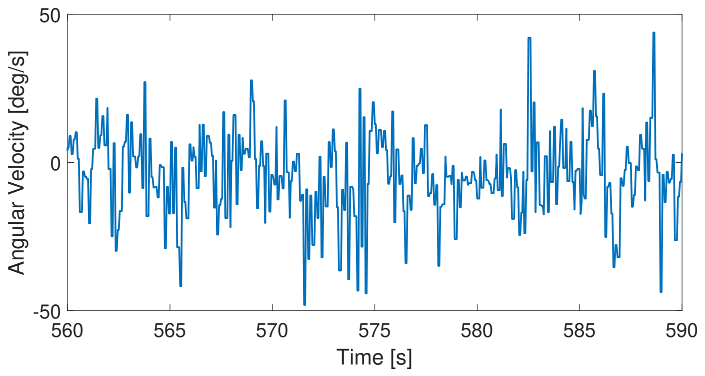


Figure 59: Angular velocity data sampled at 10Hz.

9.11 Sensor Displacement Translation

It is not possible to place the current downhole sensor close to the bit or at the centre of the BHA. An accurate estimate of the current BHA orientation requires alignments and corrections from the raw sensor measurements.

9.11.1 Axes Misalignment

Measurement axes of downhole sensor is not aligned with the reference or BHA frame when the sensor is inside the BHA. Frames are rotated to make further calculations more intuitive.

$$x_{ref} = -z_{sensor} \tag{54a}$$

$$y_{ref} = y_{sensor} \tag{54b}$$

$$z_{ref} = -x_{sensor} \tag{54c}$$

9.11.2 Ideal Sensor Orientation

When the sensor must be in the outer edges of the BHA the ideal positioning is to have the sensor in the exact opposite orientation of the BHA bend. Exact opposite orientation does not require any yaw displacement compensation and has the largest possible curvature radius when measuring roll and pitch. The larger required movement distance relative to change in orientation makes the measurement more accurate and resistant to noise.

9.11.3 Orientation Displacement

The BHA is made up of threaded components. It is difficult to create threads that will align in a certain position. Sensor orientation deviations from the ideal orientation will affect all Euler angle estimations. For yaw the orientation displacement angle is added as an offset value similar to axis alignment compensation in Equation 51.

Roll and pitch are compensated for by rotating the acceleration measurements with the rotation matrix in Equation 28a with the yaw displacement value.

9.12 Position Model Verification

Open-loop position estimation was tested while drilling to verify accuracy and tune the model. Accurately measuring position inside the rock was difficult. Deviations in exit hole position was used instead and compared to the model estimate to tune the build rate estimate.

Like in the competition requirements, a vertical well path was drilled for the first 4 inches. Drilling was executed autonomously to reduce variations between each run. Experiments for estimating the nominal build rate was simplest to verify by drilling through the rock sample with no azimuth rotation and measure exit hole displacement as shown in Figure 60.

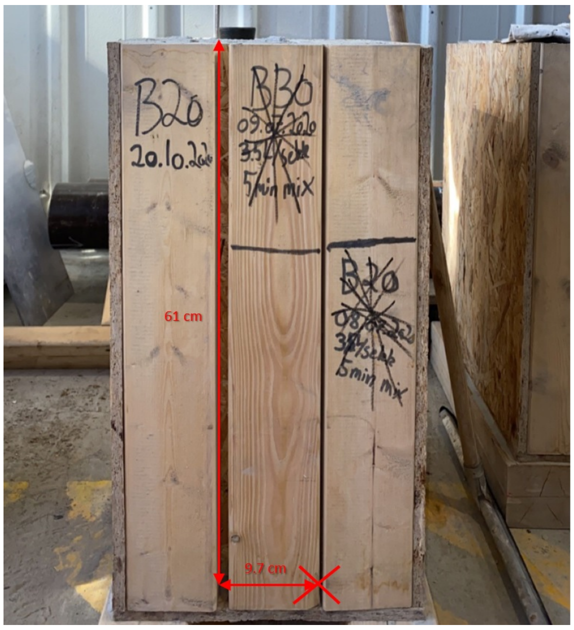
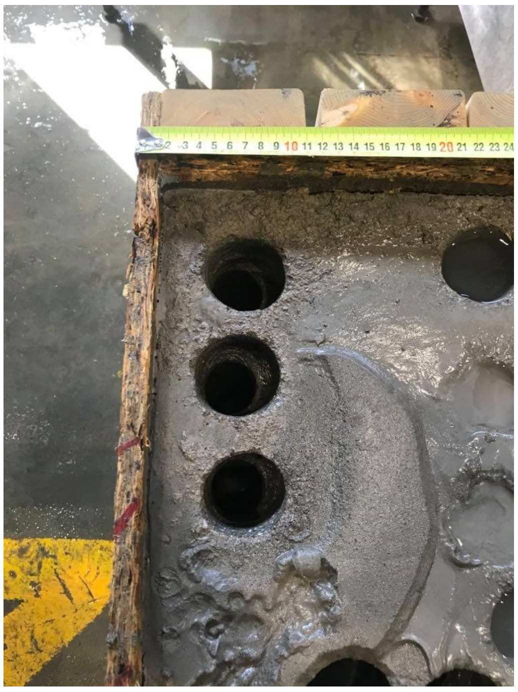


Figure 60: Exit hole displacement measurement without azimuth orientation[12].

The build rate relative to azimuth rotational velocity model was simplest to test by starting close to the edge of the rock sample and measuring the displacement position of the side exit hole. BHA orientation in the exit hole could also be observed but was difficult to measure accurately. A tuning experiment is shown in Figure 61.


(a) Entry holes.

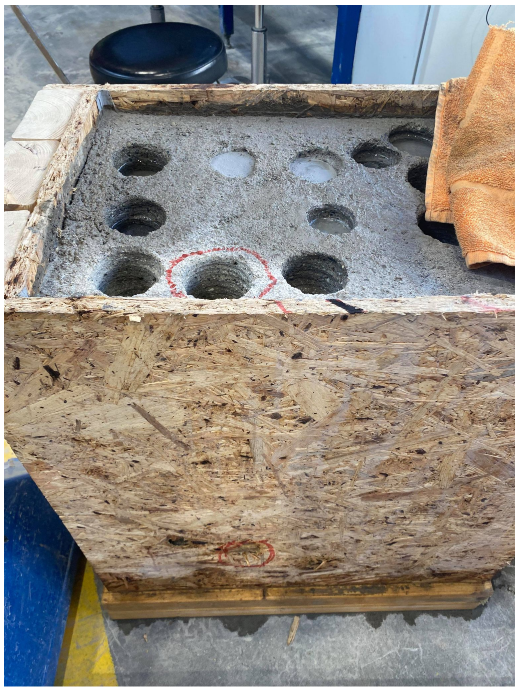

(b) Side exit holes.

Figure 61: Side exit holes for 1, 3, and 10 deg/min azimuth rotation with a 4 inch vertical section.

9.12.1 Movement Integration Error

The ROP estimate intended for the operator was first used to determine the change in position for every time-step in the position estimation. After 60cm of drilling the estimate of position in the z-axis could be up to 10% smaller than expected. This is assumed to be caused by slow response to rapid changes by the observer, resulting in a too low estimate.

Time-step velocity estimates comparing motor position was used instead to reduce integration error. The velocity estimate will be highly fluctuating with a mean equal to the rotational velocity. Because orientation changes very slowly and the orientation estimates can be tuned for minimal fluctuations while moving, the model will give an accurate estimation with minimal position estimation error.

9.12.2 Orientation Verification and Tuning

After obtaining a rough estimate of build rate from positional displacements, the orientation model was tested in closed-loop with regular surveys. Between surveys the position estimation relied on model predictions. Before a downhole survey was taken, the orientation estimate was written down and compared to the converged orientation estimate from IMU measurements. Nominal build rate was set such that it minimized mean deviation between model estimation and orientation measurement estimations.

9.12.3 Rotational Build Rate Model Tuning

With a well-tuned nominal build rate obtained it was attempted to find a modelled relation between azimuth velocity and angle build rate. The relation between azimuth rotational velocity and decrease in build rate was verified about 2 weeks before the competition and was based on observations from drilling experiments with fixed azimuth rotation velocities more than 5 deg/min.

A controlled experiment had drilled out the side of the rock sample with different fixed rotational velocities. From a small sample size, 10 deg/min rotation resulted in a straight well path while 1 deg/min had minimal change. 3 deg/min gave a significant reduction in build rate, but there was still a noticeable change in pitch. The q parameter was tuned to best fit the observed test results.

A larger data set and more experiments is expected to improve the model estimates and should be prioritized in future work.

9.12.4 Obtained Tuning Parameters

Tuning parameters used by the position model is shown in Table 22. DLS_{nom} is calculated from α_{br} with Equation 38.

Table 22: Position Model Tuning Parameters.

Description	Notation	Value
Nominal Build Rate Ratio	α_{br}	2
Nominal DLS	DLS_{nom}	0.0056
Build Rate Distribution Tuning Parameter	q	5

9.12.5 Testing Results

Closed-loop positional estimations of deviated drilling suggests an accuracy of ±1.5cm in the x and y axis deviation without azimuth rotation. z-axis position deviations rarely deviated more than 0.5cm from theoretical TVD value displacement. Measuring exit holes with a high degree of accuracy is difficult and zero-point displacement from the bit depends on the exit orientation of the BHA. Measurement tape on the outside of the rock was used to check exit hole displacement.

Azimuth rotation made the estimates less accurate with typical deviations from a limited sample size in the range ±2.5cm for the x and y axis. z-axis position was within the margin of error equal to no vertical rotation. Results indicate that the movement and build rate model has potential for improvement.

9.12.6 Reference Orientation Inconsistency

Before drilling is started, orientation estimates from the downhole sensor is calibrated. The BHA is orientated as close as possible to the reference frame and the measurement is set as an offset by the operator from the GUI. Consistently setting the reference at exactly the right orientation is crucial as this cannot be corrected for by measurements and will significantly affect the position estimate.

The riser is slightly misaligned to the vertical axis, making it difficult to set consistent reference frame orientations. To improve consistency, marks are set for the riser position and a spirit level was used to improve consistency. The process does still have some variance and a poor calibration with 1 degree offset may cause the position estimate to be off by multiple centimetres. Aligning the riser with the vertical axis and further improving the calibration process should be prioritized for future work.

9.13 Azimuth and Inclination Calculation

Azimuth and inclination are more common than Euler angles to describe orientation of the BHA in the petroleum industry. Azimuth is the orientation of the BHA relative to some reference direction, typically magnetic north. Azimuth is equal to the inertial yaw.

$$\alpha = \psi \tag{55}$$

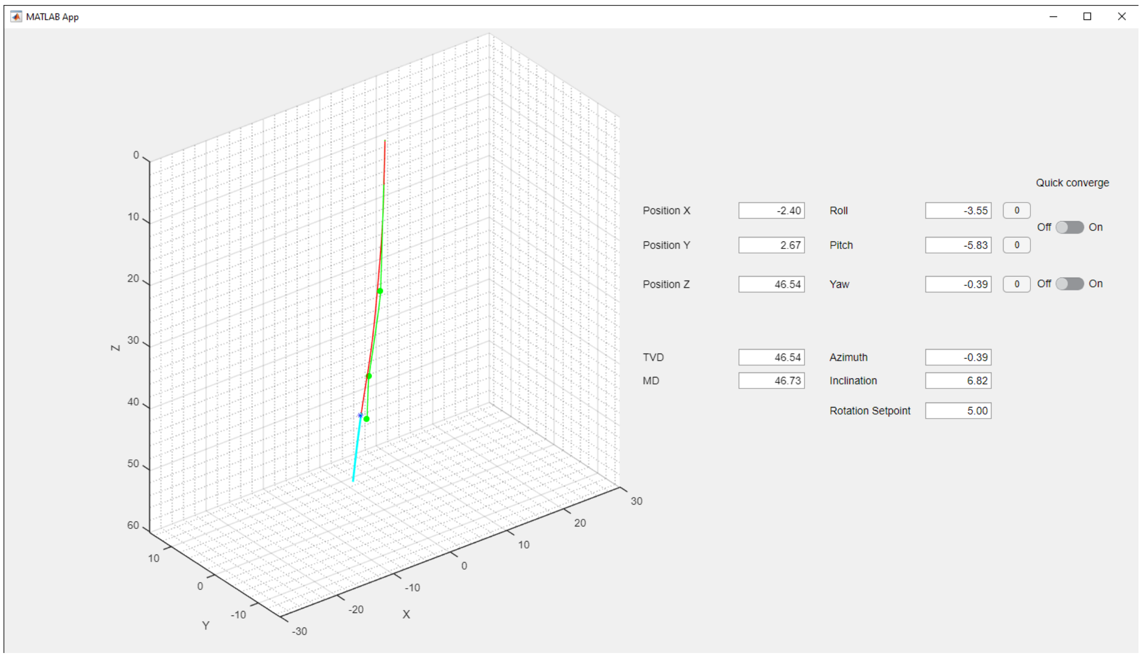
Inclination is the relative angle of the BHA to the gravity vector. Inclination can be calculated from Euler angles using trigonometric identities by aligning the angle deviation from the gravity vector in a 2D plane and calculate the relative angle.

$$\psi_{Inclination} = \arccos(\cos\phi \cdot \cos\theta) \tag{56}$$

Bit inclination will depend on current bend angle orientation.

9.14 GUI

Current position and orientation are displayed to the operator in a GUI. The GUI has a 3D plot of current and past position estimates, and a prediction horizon described further in section 10. The plot can be rotated to display the well path from different view positions while drilling.



Position X	-2.40	Roll	-3.55	0
Position Y	2.67	Pitch	-5.83	0
Position Z	46.54	Yaw	-0.39	0
TVD	46.54	Azimuth	-0.39	
MD	46.73	Inclination	6.82	
		Rotation Setpoint	5.00	

Figure 62: GUI for the high-level control system.

9.15 Direct Position Estimation

The position estimation relies on other states estimates over time. Estimation accuracy will decrease over time because of modelling, measurement and integration error. A potential solution would be to use features of the Bluetooth 5.1 protocol called Angle-of-Arrival (AoA) and Angle-of-Departure (AoD). The technology is intended for indoor positioning using antenna arrays measurements with known positional displacements to estimate the position of a Bluetooth device[69].

A possible implementation of a system could be to have a Bluetooth sender in the BHA and an array of antennas positioned near the rock sample. Angle measurements from multiple receivers with known positions could be used to create a higher accuracy position estimate using sensor fusion. Each measurement can under ideal conditions have an accuracy of a couple of degrees[61]. Received Signal Strength Indication (RSSI) could give an indication of distance to the BHA to further improve the position estimate. Accumulating uncertainty could be reduced or eliminated with direct position estimation.

Arduino Nano BLE Sense has Bluetooth 5.0 without support for angle measurement of incoming and outgoing signals. RSSI measurements has major fluctuations between each sample, suggesting it is not consistent enough for high accuracy position estimation without AoA and AoD. The recent introduction of Bluetooth 5.1 limits availability of devices, however it is reasonable to assume that the standard will have more widespread adaptation in the immediate future. The Arduino supports peripheral devices with communication over UART, SPI, and I2C. Arduino compatible Bluetooth receivers and transmitters for older standards are available and it may be possible to add direct position estimation without replacing the downhole microcontroller.

9.16 Petroleum Industry Scalability

Position is estimated from magnetometer measurements for a full-scale drilling rig to within a few meters of precision. This is generally sufficient for drilling operations. Modelled movement models could be used to further improve the current position estimate for high precision drilling. Orientation can be estimated from accelerometer and magnetometer data.

The position estimation system could be used for other industry applications requiring small scale deviated drilling with high precision such as intersection water reservoirs, mining, geothermal energy, construction work, and small-scale subsea installations.

10 Directional Drilling MPC

To intersect target points, the azimuth system is rotated over time to steer the BHA in 3 dimensions. The steering concept assumes that the trajectory will deviate in the direction of the BHA bend. Movements must be planned far ahead as angle build rate and azimuth rotation velocity is limited.

Model Predictive Control (MPC) is a control algorithm relying on a system model to simulate and optimize system inputs for a prediction horizon before applying the first optimized input to the system. MPC allows hard limits to be set for inputs and can be used for multivariable control. A drawback of MPC is high computational cost as simulations and optimizations must be executed at each time-step. NMPC is an MPC controller for nonlinear models.

NMPC was chosen for positional steering as long term planning was a requirement and no good linear approximation of movement dynamics was found. Changes in positional state is slow and the controller can have a time-step of multiple seconds, making long term predictions achievable. Simulink has support for NMPC controllers with a dedicated block using the Model Predictive Controller Toolbox.

An introduction and comparison between different MPC controllers in the Model Predictive Control Toolbox for an alternative system model is found in Appendix A.

10.1 MPC Model

Position estimation relies on a system model described in subsection 9.4. Simulink NMPC has support for continuous time models. 6 states are used by the controller: Euler angles and Cartesian coordinates in the inertial frame.

State equations are found by matrix multiplication of the position estimation model from Equation 36 and Equation 37.

$$\dot{\phi} = \frac{\cos\phi \cdot \sin\theta \cdot \dot{\psi}_I + DLS_{nom} \cdot \exp(-\frac{\dot{\psi}_I^2}{q}) \cdot \sin\phi \cdot v_{nom}}{\cos\theta} \tag{57a}$$

$$\dot{\theta} = DLS_{nom} \cdot \exp(-\frac{\dot{\psi}_I^2}{q}) \cdot \cos\phi \cdot v_{nom} - \sin\phi \cdot \dot{\psi}_I \tag{57b}$$

$$\dot{\psi} = \frac{\cos\phi \cdot \dot{\psi}_I + DLS_{nom} \cdot \exp\,(-\frac{\dot{\psi}_I^2}{q}) \cdot \sin\phi \cdot v_{nom}}{\cos\theta} \tag{57c}$$

$$\dot{x} = v_{nom} \cdot (\sin\phi \cdot \sin\psi + \cos\phi \cdot \cos\psi \cdot \sin\theta) \tag{57d}$$

$$\dot{y} = -v_{nom} \cdot (\cos\psi \cdot \sin\phi - \cos\phi \cdot \sin\psi \cdot \sin\theta) \tag{57e}$$

$$\dot{z} = v_{nom} \cdot \cos\phi \cdot \cos\theta \tag{57f}$$

v_{nom} is a constant median estimate of ROP per second while drilling.

10.2 Prediction and Control Horizon

The number of time-steps to be simulated by the MPC is called the prediction horizon, and the number of time-steps with optimized input values is called the control horizon. Increasing the prediction and control horizon increases the number of optimization variables, increasing the computational cost.

The prediction time horizon is dependent on the size of the time-step. A larger time-step will increase the time into the future that can be predicted at the expense of how often controller inputs are updated. Systems have different control requirements and must consider the trade-off between prediction time horizon and frequent adjustments of inputs.

It is common that the time-step of unequal size in an MPC prediction horizon. The first time-steps are set smaller to allow finer adjustments of inputs. The rational is that predictions further into the horizon will be less accurate and computational cost can be reduced while having high frequency control adjustments on a short-term time horizon.

Optimization complexity will not scale linearly with the length of the prediction or control horizon, limiting the maximal potential prediction horizon.

10.2.1 Directional Drilling MPC Considerations

Directional drilling requires long term planning. Long term planning comes at the expense of reduced input control adjustment frequency. Movement in position is slow and potential controller performance gains from high frequency adjustments of inputs are minimal. Time-step is set to multiple seconds to increase the prediction time horizon while being able to optimize for every time-step.

Computational cost makes planning the entire well path with an acceptable input update frequency difficult. The controller must therefore be designed to follow a well path reference plan rather than planning the well path. Following the reference requires constant azimuth control velocity adjustments. Usefulness of having a longer prediction horizon than control horizon is therefore minimal, and both are set equal.

A variable time-step horizon would be beneficial as long-term predictions will be less accurate and short-term control frequency could be increased.

10.3 Simulink Implementation

Simulink has a dedicated block for nonlinear MPC control[42]. The block requires an object defined in MATLAB workspace for configuration of model, outputs, weights, and options[41]. The components are a part of the Model Predictive Control Toolbox[38].

10.3.1 MATLAB MPC Object Configuration

The nonlienar MPC object can be edited by MATLAB commands. It is convenient to have all commands defining the object in a MATLAB script. The object is generated with the `nlmpc` command taking the number of states, outputs, and inputs are parameters.

```
% Number of states
nx = 6;
% Number of outputs
ny = 3;
% Number of inputs
nu = 1;
% Create the object
nlobj = nlmpc(nx,ny,nu);
```

Time-step, prediction horizon, and control horizon can be configured. Default values are 1, 10, and 2 respectively.

```
    nlobj.Ts = MPCTs;
    nlobj.PredictionHorizon = predictionHorizonLength;
    nlobj.ControlHorizon = controlHorizonLength;
```

Controller model and outputs are defined with functions with x and u as parameters. The nonlinear MPC object takes continuous time models as the default. Additional parameters may also be passed to the function as MATLAB functions cannot access MATLAB workspace variables directly. Jacobians for states and inputs may also be added to reduce computational cost and accelerate the optimization. Jacobians will be numerically estimated if no analytical function is provided. Some solvers will also accept a Hessian model function. The number of input parameters must be explicitly defined.

```
    % Define number of input parameters
    nlobj.Model.NumberOfParameters = 2;
    % Use external function as state model
    nlobj.Model.StateFcn = @(x,u,param1,param2)
     ↪  ContinuousStateFcn(x,u,param1,param2);
    nlobj.Jacobian.StateFcn = @(x,u,,param1,param2)
     ↪  ContinuousStateJacobianFcn(x,u,param1,param2);
    % Use local function as output model
    % Output the last 3 states
    % Dummy local function to show structure
    nlobj.Model.OutputFcn = @(x,u,param1,param2) [x(4);x(5);x(6)];
    % Dummy local function to show structure
    nlobj.Jacobian.OutputFcn = @(x,u,param1,param2) [
        0 0 0 1 0 0;
        0 0 0 0 1 0;
        0 0 0 0 0 1];
```

State and Jacobian MATLAB function must follow a specific structure to be accepted as a model function by the NMPC object.

```
function xDot = ContinuousStateFcn(x,u,param1,param2)
    % Dummy state function to show model structure
    xDot = [param1*u;param2*x(2);param2*u;param1*x(6);param1*x(1);param2];
end
```

```
function [A,Bmv] = ContinuousStateJacobianFcn(x,u,param1,param2)
    % Dummy Jacobian matrices to show output structure
     A = [param1*u, param2*u, 0, 0, 0, 0;
        -param2*u, 0, 0, 0, 0, 0;
        x(1), x(2)*u, 0, 0, 0, 0;
        param2*x(4), 0, 0, 0;
        param1*x(5), 0, x(2), 0, 0, 0;
        param2*u, -x(4), 0, 0, 0, 0];
    Bmv = [param1*x(1); x(3); x(6); 0; 0; 0];
end
```

Default solver options can be overridden in the nonlinear MPC object.

```
    % Set the solver algorithm to "sqp" and allow parallel execution and limit
    ↪  maximum iterations for each optimization
    % More solver options are found in the documentation
    nlobj.Optimization.SolverOptions.Algorithm = "sqp";
    nlobj.Optimization.SolverOptions.UseParallel = true;
    nlobj.Optimization.SolverOptions.MaxIterations = 200;
```

Outputs are called Measured Variables (MV) in MATLAB. Limits can be set to inputs for maximum and minimum values and rate of change. Hardness of input constraint enforcement can also be specified.

```
    % Set dummy limits for single input system
    nlobj.MV = struct('Min',-1, ...
        'Max',1, ...
        'RateMin',{-0.1}, ...
        'RateMax',{0.1});
```

Weights for states and inputs and their rate of change can be assigned.

```
    % Specify dummy weights
    nlobj.Weights.OutputVariables = [0.5 0.5 0.5];
    nlobj.Weights.ManipulatedVariables = 1;
    nlobj.Weights.ManipulatedVariablesRate = 1;
```

An NMPC object must be validated before it can be applied to a Simulink block. The validation function runs automated test to checks the structure of the object. Initial state and input must be defined for the function to run.

```
    % Validate nonlinear MPC controller
    x0 = [0;0;0;0;0;0];
    u0 = 0;
    validateFcns(nlobj,x0,u0,[],{param1,param2});
```

If the test is successful, the object can be linked to a nonlinear MPC Simulink block. The Simulink block will only accept parameter inputs from a bus signal. The bus input block is configured from the MATLAB script. Configuration requires the Simulink model to be open.

```
    % Add bus selector for parameters
    mdl = 'NameOfSimulinkModel';
    createParameterBus(nlobj,[mdl '/Subsystem Name/Nonlinear MPC
    ↪  Controller'],'dummyBusObject',{param1,param2});
```

10.3.2 Simulink Block Configuration

The Simulink block takes current state estimate, reference, and last input as input connections and will output the system input as default. Model parameters requires an additional parameter input connection to the block with the bus connection configured from the MATLAB script. Predicted state outputs are enabled as an output to be displayed in the GUI to the operator.

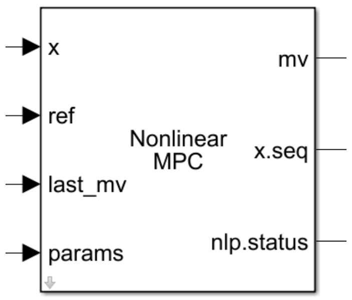


Figure 63: Nonlinear MPC block.

An optimization state output is used for debugging the solver. The state will change if the solver was not able to find a solution satisfying the optimization tolerances for the maximum number of iterations.

The reference is compared to the output variables of the model. Variable time-step is not mentioned in the nonlinear MPC controller Simulink documentation. This could be implemented by writing a custom cost function for the controller.

10.4 Time-Step Management

The NMPC controller must be able produce a new input to the system within the assigned model time-step. Optimization with the solver algorithms in subsection 10.5 will have different execution times depending on initial conditions and current state. The nonlinear MPC block will by default hold simulation time until a new input is available. This will either be the solver finding a solution satisfying the optimization tolerances or if the iteration limit is reached[42]. The position control system is a soft real-time system that should consistently provide inputs to the azimuth control system for optimal steering. Major slowdowns may also affect serial communication with the downhole sensor, potentially causing a serial buffer overflow or timeout. The serial connection cannot be re-established without restarting the simulation.

Iteration limits can be set in the solver options when defining the nonlinear MPC object in MATLAB[41]. Reducing the iteration limit will decrease the chance of an optimal solution to be found while the worst-case execution time will be reduced. The status output of the nonlinear MPC block indicates the termination criteria that was met by the optimization. The iteration limit must be set conservatively such that a worst-case optimization will be executed within the MPC time-step. A test for worst-case optimization time is to run a hardware-in-the-loop test without applying controller inputs to the system and set initial orientation in the opposite direction of the reference.

Controller action if no solution satisfying optimization tolerances was found can be configured. The block will by default reuse the input from the previous time-step finding a solution within the tolerances. If suboptimal solutions are enabled, the controller will output the current best optimization result at the last iteration[39]. The default option is selected as the input frequency is relatively high relative to movement, and inputs are not expected to change a lot between iterations. The default iteration limit is 400.

Reducing the prediction horizon and increasing the MPC time-step will increase the chance of finding an optimal solution without slowing down the simulation. Parameters in Table 23 were set conservatively to avoid model slowdowns.

Table 23: Nonlinear MPC Parameters.

Description	Notation	Value
MPC Time-Step	-	15
Prediction Horizon Length	-	30
Control Horizon Length	-	30
Nominal ROP	ROP_{nom}	3
Nominal Movement Velocity	v_{nom}	0.05

Nominal ROP will affect the length of the prediction horizon, and time-step must be tuned based on this parameter. Increasing the nominal ROP increases the distance horizon of the controller, allowing a smaller MPC time-step to be set.

10.5 Optimization Solver

The nonlinear MPC will by default use the `fmincon` optimization algorithm with an interior-point solver. `fmincon` is the recommended MATLAB optimization function for smooth nonlinear functions with either bound, linear, cone, or general smooth constraints in the MATLAB optimization decision table for problems with multiple variables[44].

A general inequality constrained nonlinear optimization problem is defined in Equation 58.

$$\min_x f(x) \quad s.t \quad c(x) \leq 0 \tag{58}$$

10.5.1 Hessian Approximation

The default recommended Hessian approximation in MATLAB is BFGS. BFGS estimates the Hessian from a quasi-Newton approximation of the Lagrangian function[57][77]. Some optimization algorithms in MATLAB requires the Hessian estimate shown in Equation 59 to be used and does not allow an analytical Hessian to be specified.

$$H_{k+1} = H_k + \frac{q_k q_k^T}{q_k^T s_k} - \frac{H_k s_k s_k^T H_k^T}{s_k^T H_k s_k} \tag{59}$$

Where s_k and q_k is defined in Equation 60a and Equation 60b.

$$s_k = x_{k+1} - x_k \tag{60a}$$

$$q_k = (\nabla f(x_{k+1}) + \sum_{i=1}^{m} \lambda_i \cdot \nabla g_i(x_{k+1})) - (\nabla f(x_k) + \sum_{i=1}^{m} \nabla g_i(x_k)) \tag{60b}$$

The estimated Hessian will be used by `fmincon` if no analytical Hessian is provided or if the selected solver algorithm does not support it.

10.5.2 Interior-Point Algorithm

Interior-point is the default solver for the MATLAB `fmincon`. Interior-point will solve a sequence of approximate minimization of equality constrained problems[26].

$$\min_{x,s} f_\mu(x,s) = \min_{x,s} f(x) - \mu \sum_{i=1} \ln(s_i), \quad s.t. \quad s \geq 0, \quad h(x) = 0, \quad g(x) + s = 0 \tag{61}$$

With s_i being slack variables and the logarithmic component called a Barrier function. μ will converge to 0 as the optimization converges towards a solution.

When solving the approximate the optimizer will try to make a direct Newton step to solve the KKT conditions found in Equation 62.

$$\nabla_x \mathcal{L}(x, \lambda) = 0 \tag{62a}$$

$$\lambda_{g,i} g_i(x) = 0, \quad \forall i \tag{62b}$$

If the algorithm is unable to make a direct Newton step due to infeasibility, it will take a conjugate gradient step instead. Conjugate step gradient will do a quadratic approximation of the approximated optimization function. A trust region-based algorithm with linear constraints is used to approximately solve the KKT conditions.

$$\nabla_x \mathcal{L} = \nabla_x f(x) + \sum_{i=1} \lambda_i \nabla g_i(x) + \sum_{j=1} y_j \nabla h_j(x) = 0 \tag{63}$$

With a positive λ the approximate solution is found by minimizing a least-squares function.

$$\begin{aligned} \min_{\Delta x, \Delta s} \; & \nabla f^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2_{xx} \mathcal{L} \Delta x + \mu e^T S^{-1} \Delta s + \frac{1}{2} \Delta s^T S^{-1} \Lambda \Delta s \\ s.t. \quad & g(x) + J_g \Delta x \Delta s = 0, \quad h(x) + J_h \Delta x = 0 \end{aligned} \tag{64}$$

With Δx and Δs the step for state and slack variables, e is a vector of ones with the same size as g, Λ and S the diagonal of the λ and s values, and J_g and J_h the Jacobians of the constraint functions.

Steps are taken until the optimization tolerance is met or until the optimization iteration maximum is reached.

States at an iteration may be infeasible or violate constraints. Loosening constraints for the first iterations is usually beneficial to allow faster convergence with larger steps sizes. To force the optimization algorithm towards a feasible solution, a merit function is used to penalize constraints violations based on a function ν that may be dependent on the iteration number.

$$\min_{x,s} f_\mu(x, s) + \nu \big\| h(x), g(x) + s \big\| \tag{65}$$

The MATLAB implementation has additional optimizations and robustness features not described in this thesis. More details about this are found in the algorithm documentation[26]. Interior-point allows an analytical Hessian function to be specified. A Hessian estimate will otherwise be used with BFGS from Equation 59 as the default option.

10.5.3 SQP Algorithm

Sequential Quadratic Programming (SQP) is iterative method for constrained smooth nonlinear optimization supported by the `fmincon` function[81]. The MATLAB SQP implementation makes an approximation of the Hessian of the Lagrange function at each iteration with a quasi-Newton update method[26]. The estimate is used to create a Quadratic Programming (QP) subproblem used to find the search direction for a line search. SQP has strict feasibility with respect to bounds.

A general Lagrangian with constraints $g(x)$ is shown in Equation 66.

$$\mathcal{L}(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot g_i(x) \tag{66}$$

The default Hessian estimate function is BFGS described in Equation 59. The quadratic subproblem based on the Hessian estimate is shown in Equation 67.

$$\begin{gathered} \min_{d} \frac{1}{2} d^T H_k d + \nabla f(x_k)^T d \\ s.t \quad \nabla g_i(x_k)^T d + g_i(x_k) = 0, i = 1, ..., m_e, \\ \nabla g_i(x_k)^T d + g_i(x_k) \leq 0, i = m_e + 1, ..., m, \end{gathered} \tag{67}$$

QP subproblems can be solved efficiently with a QP optimization algorithm. MATLAB will by default use an interior-point-convex algorithm for a QP problem[45].

The solution of the QP problem is used to determine the new iterate of the optimization algorithm. The initial conditions must be feasible.

$$x_{k+1} = x_k + \alpha_k d_k \tag{68}$$

With α_k the step length. Step length is determined by a line search algorithm. The algorithm will iterate until optimization tolerances are met or iteration limit reached.

The SQP MATLAB implementation has more complex features for added robustness, finding a feasible starting point if none is provided, and improved search efficiency further described in more detail in the MATLAB documentation[26]. A constrained optimization problem can often by solved faster than an unconstrained problem with SQP as the optimizer can make more informed decisions of search direction and step length from the constraints.

10.5.4 Solver Selection and Configuration

Initial testing used the default interior-point algorithm. The controller was successful at following the reference path an made sensible control decisions. If the position deviation between the reference and prediction horizon was too small for the controller to converge to the reference, the default iteration limit would often be reached. Depending on the suboptimal solution solver configuration option the controller would either output the last inputs satisfying the iteration limit or a suboptimal solution with an infeasible prediction horizon as shown in Figure 64.

Researching the constrained nonlinear optimization documentation from MATLAB[26][23] and solver issues guide[55], SQP was suggested as an algorithm that would satisfy bounds at all iterations while maintaining a high efficiency. Choosing SQP as the solver resolved the issue with infeasible prediction horizon states. Tests with different prediction horizon lengths also suggested that SQP was more efficient as it allowed a longer prediction horizon without exceeding the iteration limit. The interior-point algorithm has the option of using an analytical Hessian. An ideal performance comparison would require the Hessian to be specified for the interior-point algorithm and was not prioritized.

10.6 Straight Drilling

The BHA has a fixed bend angle. The well path may require certain sections to be drilled with a lower angle build rate than the nominal DLS value.

The competition guidelines require the first 4 inches to be drilled vertically. This is accomplished by rotating the azimuth system with a high rotational velocity. The resulting well path will have

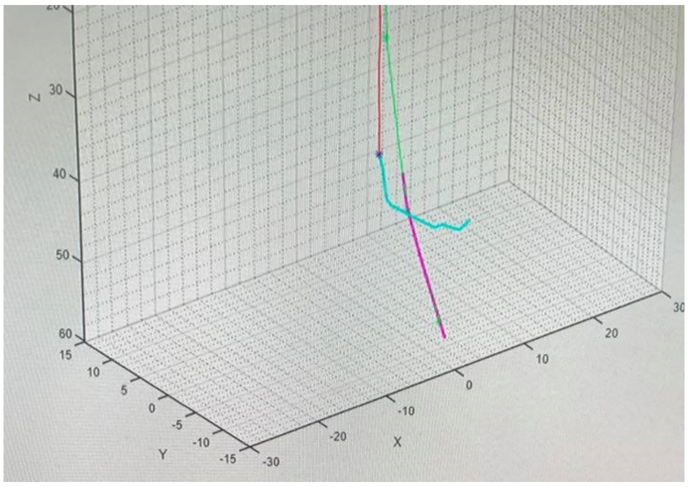


Figure 64: Picture of infeasible prediction horizon from the interior-point algorithm.

a larger diameter than the bit itself. WOB must be set conservatively to limit ROP, allowing multiple full rotations of the azimuth system to avoid poor hole quality with large grooves. An autonomously drilled vertical section is shown in Figure 65.

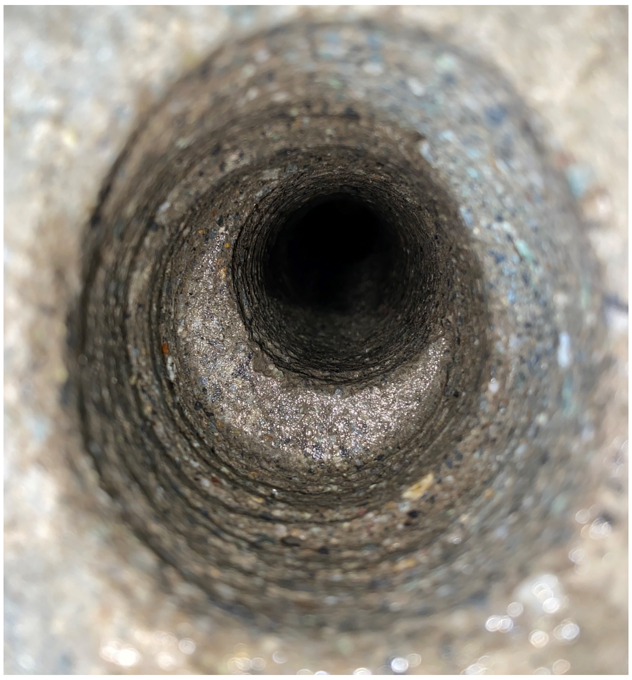


Figure 65: Autonomously drilled vertical section.

Fast azimuth rotating with high WOB while drilling often resulted mechanical failure. The friction mounts for the pipe would often slip for rotational velocities more than 20 deg/min while inside the rock. Lifting WOB allowed full rotations to be made as with the vertical section by spinning. Azimuth rotation more than 100 degrees over the duration of the drilling operation with applied WOB would often result in a stuck BHA.

Spinning while drilling would require different operation modes to be toggled while drilling as `fmincon` requires continuous constraint functions. Binary states or mixed-integer optimization for inputs would require a different solver and probably require a genetic algorithm to be used. Drilling results indicating fast azimuth rotation was not possible was obtained late in the project and major changes to the model was not possible due to time restrictions.

10.6.1 Rotational Velocity Build Rate Decrease

It was observed from drilling experiments that even small rotational velocities while drilling would significantly decrease the angle build rate in the direction of the BHA bend. This assumed to be caused by the BHA taking the path of less resistance, and instead of being forced in the direction of the BHA bend will slip on the well path wall and therefore build less angle. The BHA was expected to go straighter while rotating as the direction of build is constantly changing, however the reduction in build rate was significantly larger than what an unmodeled DLS would suggest.

The observation was used as a part of the steering concept to drill vertically without spinning. A modelled relation between build rate and azimuth rotation velocity is described in Equation 39.

10.7 Steering Concept Alternatives

Alternative steering concepts was considered based on test results and general observations.

10.7.1 Orientation Based Inputs

Rotational velocity inputs require a representative nominal ROP estimate to be accurate. Changing the steering concept to not rotate with applied WOB and rather make orientation corrections at certain MD or time intervals would make the steering less dependent on ROP. The concept would also allow more aggressive changes in azimuth.

The concept would require the WOB to be lifted frequently, limiting ROP for the drilling operation. It would be possible to implement a required orientation change threshold for WOB to be lifted, increasing the ROP. The concept would require the movement model to be altered and was therefore not tested.

10.7.2 Spinning Operational Mode

Because drilling results showed that the system did either need to rotate slowly to steer or rotate fast to go straight with spinning it was considered to use two operational modes that could be toggled by the MPC controller.

`fmincon` requires the constraints and inputs to be a continuous smooth function to be efficient. There is no support for binary or mixed-integer states with `fmincon`. The controller would then allow values in the entire range ± 360 deg/min. Velocity inputs exceeding 20 deg/min had during testing been known to cause issues with the connection between the BHA and drill pipe, and values around ± 180 deg/min will give very poor hole quality as the rotation is not fast enough to drill a straight uniform well path. Weights with `fmincon` can only be set as linear values without using a custom cost function.

10.7.3 Mixed-Integer Inputs

Limited setpoint resolution of the azimuth control system means that inputs are set at certain steps described in subsection 8.9. Continuous inputs are rounded to the closest acceptable value by the azimuth motor block in the low-level control system. Deviations between controller outputs and system inputs will therefore occur. If outputs of the controller were limited to certain values for each time-step the controller input would match the motor rotation values inputted to the system. It would then be possible to have a spinning mode by setting some of the accepted inputs to a value that will have a very fast rotational velocity without the use of a binary operational mode.

Discrete input would require a mixed-integer solver such as a genetic algorithm. The number of optimized states scales with the multiple of the size of option values at each time-step. The `fmincon` solver does not support mixed-integer optimization. Genetic algorithms solvers are available in

MATLAB[29]. Genetic algorithms are ideal for highly nonlinear problems without analytical Jacobians or discontinuous functions. Using a genetic algorithm for smooth functions with analytic Jacobians will not allow efficient analytical Jacobian and Hessian functions to be used, decreasing efficiency. Prediction and control horizon would then be very limited due to branching factor, and the number of potential inputs would have to be kept at an absolute minimum, limiting inputs resolution. Increasing the prediction and control horizon would drastically decrease the chance of good controller inputs.

10.8 Well Path Reference

The prediction horizon of the MPC is limited by computational cost. The prediction horizon of the controller would have to span all target points to make good long term control decisions. This could have been achieved with a MPC controller with a very large time-step and with a custom cost and reference function.

An approach to have both long-term planning capabilities and a high controller output frequency is to use a reference path. The controller will try to follow the reference path and compare current deviation from it at each time-step. A reference path is also beneficial in terms of solvers as the optimizer can make use of the default cost function. The reference can be generated before the drilling operation is started.

10.9 Bezier Curve

After suggestions from supervisors, Bezier curves was tested as they are commonly used in the industry and relatively simple to implement with a low computational cost. Bezier curves are a line segments with the curvature being adjusted by a one or two parameters for second and third order curves, respectively. A well-path with multiple target points will have one Bezier line segment between each of the subsequent target points.

Bezier curves of second order is mostly used for well path planning with free ends, and third order for set-end trajectories[60]. Free-end Bezier curves are used as they have unspecified azimuth and inclination at the end points. The end azimuth and inclination of a Bezier curve will be the initial orientation of the subsequent Bezier curve.

Each segment has a starting point S and an end point E. A unit tangent vector t_s is drawn from the starting point.

$$t_s = \begin{bmatrix} \cos\theta_s \\ \sin\theta_s \cdot \cos\phi_s \\ \sin\phi_s \end{bmatrix} \tag{69}$$

With θ_s and ϕ_s being the starting point inclination and azimuth, respectively. A tunable scalar d_s is used to adjust the curvature of the line segment. The scalar determines the position of the attractor point C_s. The attractor point is used to define the curvature of the line segment. The straight-line segment between the starting point and attractor point, and attractor point and the end point is normalized. The curve is created by moving at a step size along the two normalized vectors and drawing a line between the two points of the current step of the two vectors. The line between the points on the attractor point lines is also normalized and moved along with the same step size as the lines to and from the attractor point defining a point on the Bezier curve. The concept is illustrated in Figure 66.

The attractor point is placed from the tangent in Equation 69, the starting point and the tunable parameter d_s.

$$C_s = S + d_s \cdot t_s \tag{70}$$

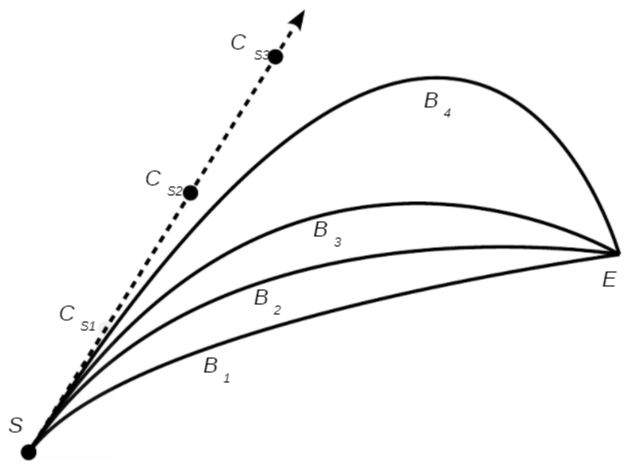


Figure 66: Bezier curve attractor point concept[60].

Movement along the normalized lines with the current normalized step u will describe the position along the curve. Step size is chosen based on resolution requirements, with a smaller step size resulting in more points along the Bezier curve. Equation 71 describes a second order Bezier curve.

$$B(u) = (1-u)^2 S + 2(1-u)u \cdot C_s + u^2 E \tag{71}$$

A well path with multiple target points will have multiple free-end Bezier curves in succession. Target points are starting points for one segment and the end point for subsequent line segment. The first target point will have the Kickoff Point (KOP) as the starting point. The well path should be continuous, and the initial azimuth and inclination is set from the inclination at the end point of the previous segment. Start inclination is set to zero as the BHA will point in the direction of the gravity vector initially. Starting azimuth is set to zero, however the Bezier algorithm will automatically find the optimal azimuth at the second iteration. Inclination and azimuth must be calculated in the end points of a segment for the well path to be continuous.

Equation 72 calculates the derivative at a Bezier point for a second order free-end trajectory.

$$\dot{B}(u) = -2(1-u)S + 2(1-2u)C_s + 2uE \tag{72}$$

The unit tangent used for further calculations is shown in Equation 71.

$$t = \frac{\dot{B}}{\sqrt{\dot{B} \cdot \dot{B}}} \tag{73}$$

Inclination and azimuth is calculated from trigonometric identities using the Bezier unit tangent vector in Equation 73.

$$\theta = \arccos t_z \tag{74a}$$

$$\phi = atan2(t_x, t_y) \tag{74b}$$

Second order Bezier curves are tuned with a single parameter for each segment. An optimization algorithm with a cost function can find the optimal parameters for the well path reference.

10.9.1 Cost Function

By penalizing undesired characteristics of the well path, a cost function can be used in combination with an optimization algorithm to find an optimal well path for a set of target points. An optimal well path has a constant DLS with the nominal build rate. DLS can be calculated at a given point based on the curvature of the path k.

The curvature calculation requires the double derivative of the Bezier curve. Double derivative of B for free-end well paths is shown in Equation 72.

$$\ddot{B}(u) = 2S - 4C_s + 2E \tag{75}$$

Curvature k can be calculated with the double derivative.

$$k = \frac{1}{\dot{B} \cdot \dot{B}} \ddot{B} - \frac{\dot{B} \cdot \ddot{B}}{(\dot{B} \cdot \dot{B})^2} \dot{B} \tag{76}$$

k is then used to calculate DLS.

$$DLS = \|k\| \cdot \frac{180}{\pi} \tag{77}$$

Deviations from the target DLS is penalized by the algorithm with the square of the deviation. DLS changes between Bezier curves are heavily penalized to create the smoothest curve possible.

$$\begin{aligned} \min \sum_{n=1}^{m-1} ((q_1(DLS(n) - DLS_{target}))^2 + (q_2(DLS(n+1) - DLS(n))^2) \\ +(q_1(DLS(m) - DLS_{target}))^2 \end{aligned} \tag{78}$$

Penalization for change in azimuth and curvature between time-steps was tested. Tuning and appropriate weighting of factors proved difficult for a wide range of potential paths.

10.9.2 Optimization

MATLAB Optimization Toolbox has a decision table for selecting the recommended solver based on problem type[44]. The objective function is smooth nonlinear. d_s must be positive and linear constraints are therefore required. The optimization decision matrix recommends `fmincon` for the problem type. `fmincon` uses an interior-point algorithm by default.

`fmincon` can be called with an objective function, initial conditions, and matrices describing the linear constraints.

```
[~,targetCount] = size(targetPoints);
% Define optimization variables
x = optimvar('x',1,targetCount);
% Bezier steps for the [0 1] range. Smaller value, more reference points
stepSize = 0.05;
% Arbitrary target points
p = zeros(3,targetCount);
for i = 1:targetCount
    p(:,i) = [targetPoints(3,i);targetPoints(2,i);targetPoints(1,i)];
end
```

```
% Objective function with free optimization variable
obj = @(x) ObjectiveFcn(x,p,stepSize,DLSTarget);
```

The problem is not convex and may have multiple local minima. Different initial conditions are therefore tested, and the cost functions outputs compared to increase the chance of finding a good well path. Initial conditions are set as a range of values for each of the target point parameters. The number of combinations will increase with the number of target points by the factor of the number of values in the search range for each value.

Table 24: Bezier Well Path Tuning Parameters.

Description	**Notation**	**Value**
Nominal DLS (deg/m)	DLS_{nom}	56
Step Size	-	0.05
Starting Point Min	-	0.2
Starting Point Max	-	0.75
Starting Point Delta	-	0.2
DLS Target Deviation	q_1	5
DLS Step Change	q_2	4

The optimized parameter with the smallest cost is then set as the reference path.

10.9.3 Limitations

Each segment of a Bezier curve is limited to curve in 2 dimensions. Some target point layouts may therefore have suboptimal results with DLS deviating from the target DLS and abrupt changes in azimuth and inclination in the transaction between the line segments. The cost function can be tuned to penalize abrupt changes but there are limitations to how good well paths with large change in azimuth that can be created from this method.

Finding a good general tuning of the cost function for different well paths is difficult. The cost function must be tuned very conservatively to work with a large range of inclinations and azimuths, resulting in some well paths being almost straight-line segments to avoid extreme inclination and azimuth values for other well paths. The phenomena can be observed in Figure 67.

Bezier curves must intersect all points. In some cases, the smallest total deviation from all target points would be achieved if some of multiple points are not intersected, for instance to build sufficient inclination to intersect other points later on.

There are large potential gains by selecting an alternative well path planning approach and this should be prioritized for future work.

10.10 Well Path Reference Alternatives

Other alternatives to second order Bezier for well path reference generation were considered.

10.10.1 Third Order Bezier Curve

Equal concept to second order Bezier with 2 attractor points for each line segment allowing curvature in 3 dimensions[60]. Curvature in 3 dimensions would allow the well path to have characteristics closely matching the expected build rate of the system with adjustments in azimuth. More flexibility is expected to make the optimization simpler to tune for a larger range of target point configurations.

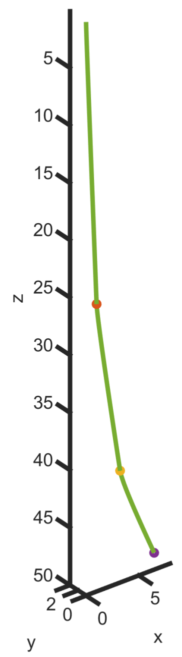


Figure 67: Optimized Bezier well path. Sections have low curvature to avoid infeasible well paths for other target point configurations.

Double the number of attractor points require double the number of optimization variables, increasing computational cost. Third order Bezier was not tested due to time restrictions.

Third order Bezier would require all targets to be intersected by the well path. This might not always be ideal as the mean deviation of all points might be smaller by intentionally not intersecting some target points perfectly.

10.10.2 Splines

Splines is a mathematical function creating a continuous function between two or multiple points. A simple spline implementation is defined as a set of equations of polynomials of a certain order with the constraint that the derivative and double derivative must be equal before and after a point that the spline will follow. The spline curve will have an optimized polynomial function satisfying the continuous constraint between each of the points, and the line segment between each point is a usually different polynomial function.

Bezier curves was chosen over splines as it was easier to limit a Bezier curve to a fixed DLS and papers with petroleum industry values calculated were available[60]. Bezier curvature is limited by the optimization function. If a spline implementation with the possibility of increased control of curvature was found splines may be a viable option.

Splines may not always intersect all points. This can be beneficial in some cases, but splines will tend to take the shortest path possible. If points are provided to form a curvature, the spline estimate is likely to have a much smaller radius of curvature and not intersect most of the points.

10.10.3 Model Based Optimization

It would be possible to use the model of the MPC controller to create the well path reference with a custom cost function. The cost function could be set to penalize the positional deviation of the closest target point intersection to the planned well path. The prediction horizon would have to span all potential target points from start to finish to make good predictions. The nominal velocity input could be used to tune the length of the prediction horizon at the trade-off of input resolution. The model could also provide a reference for orientation to further assist the controller and allow a shorter prediction horizon or less advanced controller to be used.

Time limitations meant this idea was not further explored. The optimization is expected to be more computationally expensive but is expected to be feasible within the 30-minute permitted time window in the Drillbotics guidelines on either local hardware or using a cloud service.

A highly efficient well path planner spanning all target points in the closed-loop MPC controller with an acceptable input frequency would make a pre-planned path unnecessary.

10.10.4 Dynamic Well Path Planning

If any of the well path references generation methods has a sufficiently small computational cost, the well path could be regenerated while drilling based on the current position estimate and remaining target points. The BHA steering controller would then have a more relevant reference if there were deviation from the initial reference plan and improve long term control decisions.

10.11 Reference Horizon Generator

The reference to the nonlinear MPC is a point along the Bezier well path for every time-step in the prediction horizon. The reference path cannot be generated at an infinite resolution requiring the reference points from the controller to be an interpolation between the Bezier points. Bezier points are calculated with a small step size and a linear approximation has minimal well path length deviations to the continuous curve length.

The closest point to the current position is found with a binary search algorithm. A binary search will half the length of the search space at each iteration. Three points are evaluated at every iteration; first, middle, and end of the search space array. The range between the two points with the smallest distance to the current position estimate are used to define the new smaller search space until the length of the search space until the middle point is equal to either the first or second point. The length from the current position to a reference point is evaluated with the norm of the deviation.

The search algorithm was chosen as it was simple to implement and can quickly search a very large range of values with $O(\log n)$. The algorithm is not guaranteed to return the exact closest point. The reference path will always move in positive z direction for any feasible BHA orientation. Finding the exact closest point is not a strict requirement if the references generator can consistently give a point that is close to the current position and create a relevant reference.

After the closest Bezier point is found, the previous and subsequent reference index are found if they are available. A binary search is used to find the closest point on each of the maximum of two line segments to the current position from the norm value. The search terminates once an accuracy threshold is met. The closest point on each of the line segments are compared and the closest point position and previous index passed to the reference window generator.

A reference horizon is generated from linear segments between Bezier well path reference points. The nominal drilling velocity is used to estimate the expected movement for each time-step. The remaining movement distance is tracked for each point while keeping track of the current reference index. The algorithm will first check if it is possible to move to the next reference point directly by checking if the norm of the deviation is less than the remaining movement distance. If this is possible the current point index is incremented, and the norm subtracted from the remaining distance for the time-step. If a full movement step is not possible the vector between the current and subsequent point is normalized and multiplied with the remaining distance. Distance moved from last point is saved by the algorithm and used for the next reference point. The point along the vector is added to a reference horizon before algorithm is repeated for the next reference point

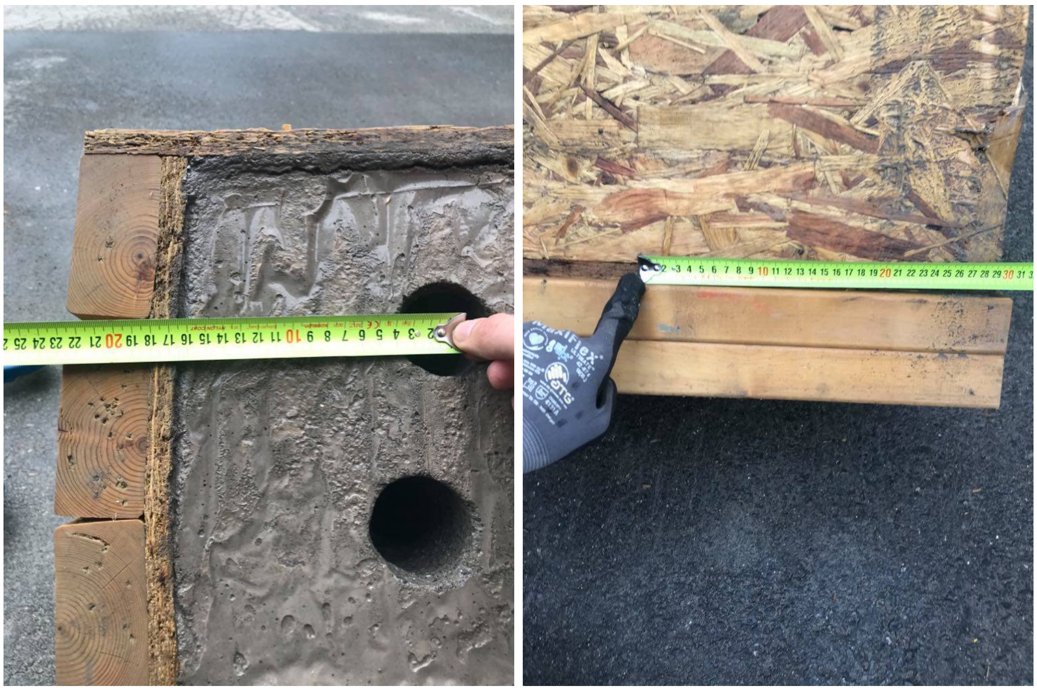

(a) Top measurement. (b) Tilted rock bottom measurement.

Figure 68: Measurement of x-axis deviation for closed-loop position control. Deviation from estimated exit position was less than 2cm after compensating for the offset of the bit.

until the reference horizon is of equal length to the prediction horizon.

10.12 Open-Loop Test Results

An inconveniently timed sensor card failure required the first physical rig tests to be ran without the downhole sensor card correcting for deviations orientation. The test was essentially a hardware-in-the-loop test of the system to verify that the controller would send inputs to the rig as intended and that the inputs were sensible based on the current position deviations to the reference.

The test was a successful considering that the controller made sensible decisions to follow the reference path and the general azimuth and inclination trend of the pre-planned well path. Positional displacement estimations had deviations of up to 2-3cm for x and y position, suggesting that closed-loop orientation is a requirement for an accurate position estimate.

10.13 Closed-Loop Test Results

Testing the controller with downhole sensor data was expected to give a better position estimates, improving target intersection accuracy in the rock. Position estimates were improved relative to the open-loop test and the resulting well path would more accurately follow the reference path than with the open-loop tests.

It was observed that the controller would often not build sufficient inclination to intersect target points at the later stages of the well path. This is expected to be caused by insufficient long-term planning capabilities, partly caused by the length of the prediction horizon and partly by a suboptimal well path reference. Increasing the BHA bend angle would allow larger corrections to be made on a shorter time horizon. With a few exceptions the controller would make sensible control decisions to reduce the deviation from the reference path.

The steering performance is best for deviated well paths with either a slightly smaller or larger

DLS to the nominal value. Insufficient tuning of the straight drilling model is assumed to be the main cause. Closed-loop positional control was only tested for a few drilling operations and is expected to be more accurate and consistent with more tuning of the controller and position estimate, comparable to simulation results.

10.13.1 Y-axis Orientation Bug

The final version of the controller appears to have a bug when the target points are in the y-axis direction. The controller will output the maximum orientation velocity in the opposite direction of what is sensible considering the reference path. The behaviour does not occur in digital twin simulations, suggesting that the cause of the bug is related to an inverted orientation value or incorrect rotational matrix. A possible cause could be the azimuth compensation system for BHA orientation displacement or a reversed roll estimate.

10.13.2 MPC Controller Parameters

Weighs and parameters for the controller with best simulation performance is shown in Table 25.

Table 25: MPC Tuning Parameters.

Description	Notation	Value
MPC Time-Step	Ts_{MPC}	15
X-axis Position Deviation Weight	q_x	2
Y-axis Position Deviation Weight	q_y	2
Z-axis Position Deviation Weight	q_z	2
Azimuth Rotation Manipulated Variable Weight	q_{MV}	1
Azimuth Rotation Manipulated Variable Rate Weight	q_{MVrate}	1
Azimuth Rotation Maximum Velocity (deg/min)	-	15
Azimuth Rotation Minimum Velocity (deg/min)	-	-15
Azimuth Rotation Velocity Rate Maximum (deg/min)	-	10
Azimuth Rotation Velocity Rate Minimum (deg/min)	-	-10

10.14 Petroleum Industry Scalability

Bent sub directional drilling systems are used in the petroleum industry but have been replaced with more Rotary Steerable System (RSS) capable of adjusting the bit orientation in any direction for high accuracy directional drilling. Planning well paths in 3 dimensions with a directional sub is difficult for a human driller as it requires long term planning and is less intuitive than operating an RSS. RSS systems are highly complex, expensive, and are difficult to make mechanically robust, especially at a small scale.

NMPC directional drilling could either be implemented as an assistance to a human operator with real-time drilling path prediction and input suggestions, or as a fully autonomous system following a reference path autonomously. The controller would allow fixed bent systems to be used for high accuracy direction drilling were RSS is currently the industry standard.

Sending data to and from the BHA can either be done with mud pulsing or using wired pipes. Mud pulsing encodes data into the mud being sent back to surface by adjusting the pressure. Mud pulsing has limited data rate capabilities. If filtered values are calculated downhole the required transmission rate could be reduced multiple order of magnitude. It would also be possible to have the computer with the NMPC controller downhole to reduce the required data rate to surface for fully autonomous operation.

Directional drilling MPC with a bent sub could also be used in other industries such as geothermal energy, mining, construction work, water reservoir intersection, and space exploration.

11 Autonomous Operation

During the Drillbotics competition drilling demonstration, the rig is required to operate fully autonomously without human operator intervention. The system must detect and respond to incidents in real-time and make closed-loop control drilling decisions. Different phases of the drilling operations use different setpoints, algorithms, and drilling parameters. Phases can be defined as states of operation to make logic implementation more intuitive with a state machine.

11.1 State Machine

A state machine is a behaviour model with a finite number of states. Transactions between states depends on the current state and one or multiple logical conditions. State space machines abstracts modes of operation into separated instances. Separated instances are more intuitive to understand and makes logic easier to implement, maintain, and debug. Basic state machines implementation will only have a single active state at a time. A single state may have multiple transaction inputs from other states and should behave similarly independent of the origin state.

State space machines can be visualized using a flowchart. Transactions between states are represented as arrows and states a simple figure with the name of state.

11.1.1 Autonomous Operation States

There are a total of 7 states in the system. Each state represents a specific mode of operation. The flowchart of the states is shown in [**fig:state˙machine**].

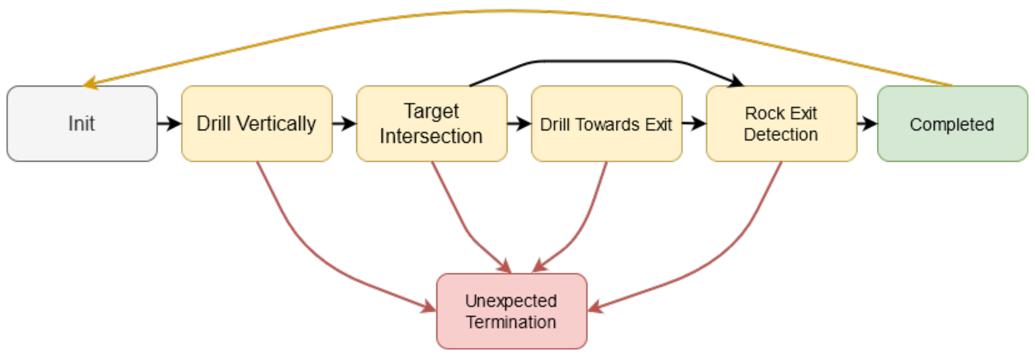


Figure 69: State machine flowchart.

Each state is described in further details in subsection 11.3. A short description of each state follows.

- **Init**: Waiting for user input to start the autonomous operation. Disables all movement.
- **Drill Vertically**: Drills a 4 inch vertical well path from the tag position.
- **Target Intersection**: Drills a deviated well path with the azimuth being controlled by the high-level control system. State transaction occurs when the z position of the last target is passed.
- **Drill Towards Exit**: All targets are intersected and drilling until the edge of the rock is within close proximity.
- **Rock Exit Detection**: Edge of rock is estimated to be within close proximity. Activate detection to check if bit has exited the rock.

- **Completed**: Drilling operation has successfully been completed. Disable all movements and wait for operator input.
- **Unexpected Termination**: Some critical incident caused the operation to end prematurely. Wait for operator input.

11.2 Simulink Implementation

Simulink Stateflow is a toolbox for graphical programming of state machines[51]. Each state is represented as a block and transactions between states represented by lines with transition logic conditions. A state can have an internal state machine system subsystem. Initial state entry points are indicated with a blue dot.

Graphical programming of state machines allows simple overview of state transitions for advanced logic. Logic programming errors and therefore less likely when the complexity of the state machine increases.

Each state may have logic executed when the state is initialized, during state operation, and when the state is exited.

11.3 States Overview

11.3.1 Init

Autonomous operation is disabled when the control system is initialized. The operator can enable autonomous operation from the control system GUI. All controller outputs are disabled and motor setpoints are set to zero. The init state can also be accessed from the completed state if the operator resets after completion.

11.3.2 Drill Vertically

Competition guidelines require the first 4 inches of well path to be drilled vertically. Vertical drilling is achieved by rotating the azimuth system quickly at a setpoint of 540 deg/min. To achieve good hole quality with a close to smooth well path, WOB is set relatively low to limit the ROP, allowing the bit to be rotated multiple rotations for the vertical section to improve hole quality.

To avoid wind-up of the sensor card cable, rotation direction is reversed if the current orientation is larger than 360 degrees or smaller than -360 degrees. The azimuth control system has a slow acceleration and deceleration. A pause of 20 second without applied WOB is used to allow the bit to accelerate in the opposite direction before continuing drilling. The substate machine is shown in Figure 70.

There is more mechanical stress and vibrations while tagging the rock. The system will therefore drill vertically for two centimetres without azimuth rotation. The state transitions to the target intersection state occurs when 4 inches or 10.16 cm has been moved in MD.

11.3.3 Target Intersection

The state starts by homing towards the first target point. Homing is done by alternating between two states setting rotation in opposite direction depending on the orientation relative to the first target point. A substate transition occurs when a threshold for position and maximum orientation velocity is met. A downhole orientation survey is then taken. Surveys will first stop all azimuth rotation and lift WOB for a duration for cuttings to be cleared in front of the bit. Bit rotation is then stopped for a longer duration for the orientation estimate to converge. Bit rotation and the

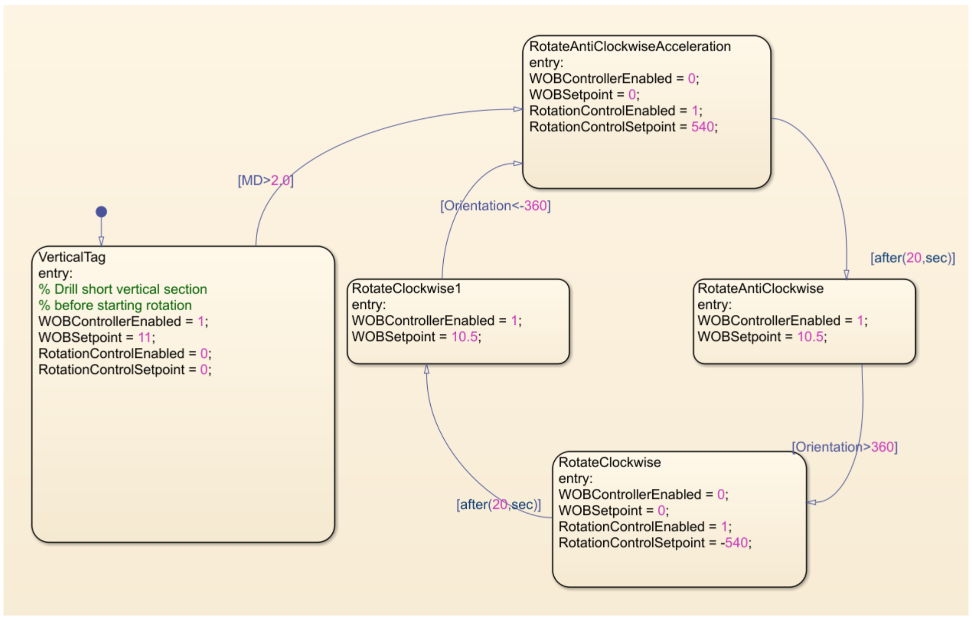

RotateAntiClockwiseAcceleration
entry:
WOBControllerEnabled = 0;
WOBSetpoint = 0;
RotationControlEnabled = 1;
RotationControlSetpoint = 540;
[MD>2,0]
[Orientation<-360]
[after(20,sec)]
VerticalTag
entry:
% Drill short vertical section
% before starting rotation
WOBControllerEnabled = 1;
WOBSetpoint = 11;
RotationControlEnabled = 0;
RotationControlSetpoint = 0;
RotateClockwise1
entry:
WOBControllerEnabled = 1;
WOBSetpoint = 10.5;
RotateAntiClockwise
entry:
WOBControllerEnabled = 1;
WOBSetpoint = 10.5;
[Orientation>360]
[after(20,sec)]
RotateClockwise
entry:
WOBControllerEnabled = 0;
WOBSetpoint = 0;
RotationControlEnabled = 1;
RotationControlSetpoint = -540;

Figure 70: Substate machine for vertical drilling.

WOB controller is enabled after the survey has been taken before drilling towards the next survey point.

Between samples the azimuth rotation system will take inputs from the MPC controller. The inputs are used to follow the reference path and intersect target points. Orientation surveys are taken for each 5 cm of movement in MD. The exception is when the last target point is less than a survey interval away. This is mainly done to avoid the rock exit detection to trigger prematurely if the last target point is surpassed right after a survey. The substate machine is shown in Figure 71.

The state is ended if either the last target point is passed in TVD to transition to the drilling towards exit mode, or to rock exit detection if x, y, and z position estimate is close to exiting the edges of the rock.

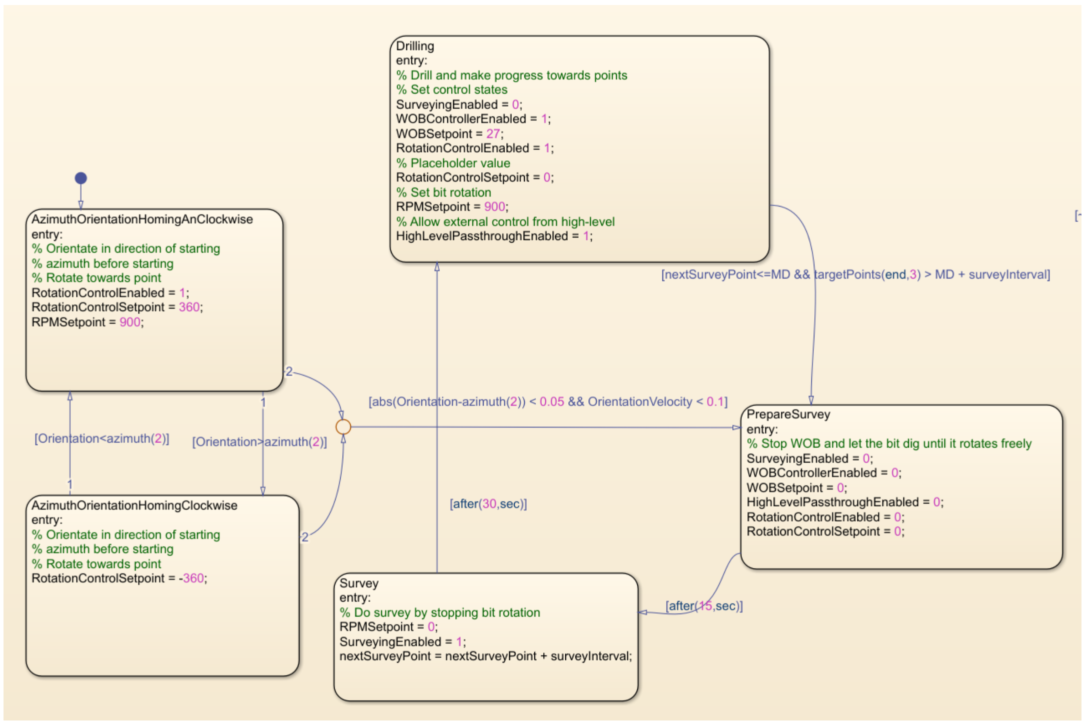
11.3.4 Drilling Towards Exit

If the last target point is passed and the TVD is not close to the edge of the rock, the drilling towards exit mode is activated. The WOB setpoint can be set higher to increase ROP as there is no azimuth rotation steering required. The state will transition to the rock exit detection state when the bit is close to exiting the rock.

11.3.5 Rock Exit Detection

When the bit is approaching the edge of the rock the rock exit detection is activated. WOB is lowered and ROP monitored to detect rock exit. It was observed when drilling that there would be a small increase in ROP followed by a decrease and constant low ROP as the bit hit the concrete block container made from wood.

Low ROP detection was less prone to false positives. A logical input of the average ROP value compared to a threshold value is inputted as a boolean signal to the state machine. When the rock exit boolean ROP detector is triggered, the state machine will transition to the completed state. The thresholds was set with an average ROP of less than 0.3cm/min for a window of 15 seconds.



```
Drilling
entry:
% Drill and make progress towards points
% Set control states
SurveyingEnabled = 0;
WOBControllerEnabled = 1;
WOBSetpoint = 27;
RotationControlEnabled = 1;
% Placeholder value
RotationControlSetpoint = 0;
% Set bit rotation
RPMSetpoint = 900;
% Allow external control from high-level
HighLevelPassthroughEnabled = 1;
```

```
AzimuthOrientationHomingAnClockwise
entry:
% Orientate in direction of starting
% azimuth before starting
% Rotate towards point
RotationControlEnabled = 1;
RotationControlSetpoint = 360;
RPMSetpoint = 900;
```

```
AzimuthOrientationHomingClockwise
entry:
% Orientate in direction of starting
% azimuth before starting
% Rotate towards point
RotationControlSetpoint = -360;
```

```
PrepareSurvey
entry:
% Stop WOB and let the bit dig until it rotates freely
SurveyingEnabled = 0;
WOBControllerEnabled = 0;
WOBSetpoint = 0;
HighLevelPassthroughEnabled = 0;
RotationControlEnabled = 0;
RotationControlSetpoint = 0;
```

```
Survey
entry:
% Do survey by stopping bit rotation
RPMSetpoint = 0;
SurveyingEnabled = 1;
nextSurveyPoint = nextSurveyPoint + surveyInterval;
```

[nextSurveyPoint<=MD && targetPoints(end,3) > MD + surveyInterval]
[abs(Orientation-azimuth(2)) < 0.05 && OrientationVelocity < 0.1]
[Orientation<azimuth(2)]
[Orientation>azimuth(2)]
[after(30,sec)]
[after(15,sec)]
1
2
1
2

Figure 71: Substate machine for target intersection.

11.3.6 Completed

Stops all motor movements and waits for operator input. The operator can reset autonomous operation back to the init state by pressing the button used to start the operation. Changing text on the start button will indicate that the operation has been completed.

11.3.7 Unexpected Termination

The state machine has a logical input that can be triggered if a critical incident is detected, and the rig operation must be stopped immediately. The state can be reached from any state and will stop all motor rotation and indicate to the operator that the operation has terminated.

11.4 Starting and End States

The autonomous system can be started from any state. This is useful for restarting after an aborted operation, only executing some of the autonomous procedures, and debugging the state machine. The end state is checked for each state transition. Each state is assigned a numerical value depending on the intended execution order. If the subsequent state value is greater than the end state, the state machine will transition to the completed state.

12 Conclusion and Future Work

A fully autonomous drilling rig with NMPC trajectory steering was successfully implemented. The control system has been completely redesigned to use the MATLAB software environment instead of LabVIEW with benefits such as fast paced simulation prototyping and advanced controller support. A model for BHA trajectory and current position estimation is implemented and tuned. A downhole IMU sensor has been implemented and tested. Measurements are used in closed-loop to estimate orientation of the BHA.

Results indicate that a fixed bent BHA is a viable option for directional drilling trajectory steering in 3 dimensions if combined with a MPC controller. The research can be used to replace RSS based systems for certain applications in the petroleum industry with benefits such as reduced cost and mechanical robustness. High inclination small scale directional drilling systems were RSS systems are difficult to implement have the greatest benefit from the findings. The position model can be used to improve high accuracy small scale drilling systems.

A shortlist of recommended future work described in this thesis follows:

- **Position Controller Tuning**: The trajectory controller has shown promising results for directional steering. More tests and debugging is expected to further improve target intersection capabilities. The model for expected build rate is based on very few test experiments due to time limitations and would benefit from additional tuning. Controller performance is also expected to improve with a more accurate and reliable position estimates and well path planning.
- **Position Estimation**: The position estimation has shown promising results but could benefit from additional tuning. Reference frame alignment consistency has been an issue, invalidating the results of some drilling operations and caused poor target intersection performance. Methods of estimating the position directly or using an artificial magnetic field should be investigated to reduce accumulating position error.
- **Well Path Planning**: A second order Bezier well path reference curve is suboptimal as all curves are restricted to a 2 dimensional plane. Increasing the order of the Bezier curve or using a model based well path generator would improve long-term planning capabilities, allowing more consistent target intersection for multiple subsequent points.
- **Hardware Communication Optimization**: The current interfacing over Modbus TCP have a very high computational cost when using the integrated MATLAB function. A manual Modbus TCP implementation using Simulink TCP communication blocks is expected to decrease computational load. Controlling the azimuth motor from EtherNet/IP interface through the gateway is expected drastically reduce the Modbus communication computational load.
- **Downhole Sensor Improvements**: The current downhole sensor requires a long BHA with a large displacement between the bit and sensor. The sensor is not aligned with the centre of the BHA and has rotational displacement relative to the bend of the BHA. The process of inserting the microcontroller into the BHA and using Plasti Dip and epoxy for water sealing is unreliable. A wireless BHA may also be attempted.
- **GUI Optimization**: The current GUI implementation is very resource demanding. Plots in particular limit the maximum overall sampling frequencies of the system. Finding a more efficient GUI implementation could significantly reduce computational cost. A compiled GUI from the MATLAB Solver in App Designer with a different interface between the GUI and Simulink model should be investigated.
- **Real-Time System**: With the exception of the DAQ, it is assumed to be possible to implement the control system as a Simulink Desktop Real-Time system. Replacing the DAQ with a Simulink Desktop Real-Time compatible model will probably require the control system computer to be replaced as there is no room for expansion cards in the current model and real-time compatible models will require a PCI based interface. A real-time system will allow more reliable worst-case execution times for low-level control.

- **State Covariance Estimation**: There is currently no continuous covariance estimate for orientation and position. Kalman filters and analysis of drilling data could be used to estimate the covariance of states in real-time.
- **Downhole Microphone**: There is a microphone connected to the downhole microcontroller. Microphone measurements can detect high frequency vibration trends while drilling. The microphone was not tested due to time restrictions.

Bibliography

[1] victorromeo (GitHub username). *PDM library.* URL: https://github.com/victorromeo/ArduinoPDM.

[2] AEP. *TC4 AMP.* URL: https://www.aep.it/wp-content/uploads/2020/02/TC4AMP.pdf.

[3] Arduino. *Arduino LSM9DS3 library.* URL: https://www.arduino.cc/en/Reference/ArduinoLSM9DS1.

[4] Arduino. *Arduino Nano 33 BLE Sense.* URL: https://store.arduino.cc/arduino-nano-33-ble-sense.

[5] Arduino. *Arduino Nano Family.* URL: https://store.arduino.cc/new-home/nano-family.

[6] Arduino. *Arduino Nano RP2040 CONNECT.* URL: https://store.arduino.cc/nano-rp2040-connect.

[7] Arduino. *ArduinoBLE Library.* URL: https://www.arduino.cc/en/Reference/ArduinoBLE.

[8] Arduino. *micros().* URL: https://www.arduino.cc/reference/en/language/functions/time/micros/.

[9] BRU21. *BRU21: Research and Innovation Program in Digital and Automation Solutions for the Oil and Gas Industry.* URL: https://www.ntnu.edu/bru21.

[10] Drillbotics. *International Competition.* URL: https://drillbotics.com/international-competition/.

[11] OPC Foundation. *Unified Architecture.* URL: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[12] T.M.V Skretting G. Hånsnar and B. Gjersdal. *Design and Implementation of a Miniature Rig for Autonomous and Directional Drilling.* Master's Thesis. NTNU Department of Engineering Cybernetics, 2021.

[13] T.M.V Skretting G. Hånsnar M. Steinstø and B. Gjersdal. *Design Report NTNU - Drillbotics 2021 Phase I.* Specialization Project. Department of Geoscience and Petroleum, 2020.

[14] Gigager. *GSH Hypoid Gear Rotary Table.* URL: https://5.imimg.com/data5/VT/DC/CF/SELLER-2308089/hypoid-gear-rotary-table.pdf.

[15] Gigager. *Right Angle Reducer.* URL: https://www.gigager.net/uploads/201919612/Right-Angle-Gearboxes.pdf?rnd=479.

[16] Inc GitHub. *The tools you need to build what you want.* URL: https://github.com/features.

[17] C. B. Guggedal and M. Steinstø. *Control System Architecture and API Integration.* Bachelors's Thesis. UiS Faculty of Science and Technology, 2019.

[18] M.U. Azam H.E. Helle and J.M. Montoza. *Design and Implementation of an Autonomous Miniature Drilling Rig for Directional Drilling.* Thesis. Department of Geoscience and Petroleum, 2019.

[19] hilscher. *Gateway EtherNet/IP Scanner to Open Modbus/TCP.* URL: https://www.hilscher.com/products/product-groups/gateways/for-the-control-cabinet-ip20/fieldbus-gateways/nt-100-re-eneimomb/.

[20] The MathWorks Inc. *Access Block Data During Simulation.* URL: https://se.mathworks.com/help/simulink/ug/accessing-block-data-during-simulation.html.

[21] The MathWorks Inc. *Anti-Windup Control Using a PID Controller.* URL: https://se.mathworks.com/help/simulink/slref/anti-windup-control-using-a-pid-controller.html.

[22] The MathWorks Inc. *bdroot.* URL: https://se.mathworks.com/help/simulink/slref/bdroot.html.

[23] The MathWorks Inc. *Choosing the Algorithm.* URL: https://se.mathworks.com/help/optim/ug/choosing-the-algorithm.html#brppuoz.

[24] The MathWorks Inc. *Closed-Loop PID Autotuner.* URL: https://se.mathworks.com/help/slcontrol/ug/closedlooppidautotuner.html.

[25] The MathWorks Inc. *Code Generation from MATLAB Code.* URL: https://se.mathworks.com/help/robotics/ug/code-generation-from-matlab-code.html.

[26] The MathWorks Inc. *Constrained Nonlinear Optimization Algorithms.* URL: https://se.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html.

[27] The MathWorks Inc. *Developing an App Designer App for a Simulink Model using CAN.* URL: https://se.mathworks.com/help/vnt/ug/developing-an-app-designer-app-for-a-simulink-model-using-can.html.

[28] The MathWorks Inc. *Estimate State of Nonlinear System with Multiple, Multirate Sensors.* URL: https://se.mathworks.com/help/control/ug/multirate-nonlinear-state-estimation-in-simulink.html.

[29] The MathWorks Inc. *ga.* URL: https://se.mathworks.com/help/gads/ga.html.

[30] The MathWorks Inc. *Hardware Support from Simulink Desktop Real-Time.* URL: https://se.mathworks.com/hardware-support/simulink-desktop-real-time.html.

[31] The MathWorks Inc. *Implement Data Parallelism in Simulink.* URL: https://se.mathworks.com/help/simulink/ug/implement-data-parallelism-in-simulink.html.

[32] The MathWorks Inc. *Install Real-Time Kernel.* URL: https://se.mathworks.com/help/sldrt/ug/real-time-windows-target-kernel.html.

[33] The MathWorks Inc. *magcal.* URL: https://se.mathworks.com/help/nav/ref/magcal.html.

[34] The MathWorks Inc. *Magnetometer Calibration.* URL: https://se.mathworks.com/help/nav/ug/magnetometer-calibration.html.

[35] The MathWorks Inc. *MATLAB.* URL: https://se.mathworks.com/products/matlab.html.

[36] The MathWorks Inc. *MATLAB Compiler.* URL: https://se.mathworks.com/products/compiler.html.

[37] The MathWorks Inc. *MATLAB Function.* URL: https://se.mathworks.com/help/simulink/slref/matlabfunction.html.

[38] The MathWorks Inc. *Model Predictive Control Toolbox.* URL: https://se.mathworks.com/products/model-predictive-control.html.

[39] The MathWorks Inc. *MPC Controller.* URL: https://se.mathworks.com/help/mpc/ref/mpccontroller.html.

[40] The MathWorks Inc. *NI-DAQmx Support from Data Acquisition Toolbox.* URL: https://se.mathworks.com/hardware-support/nidaqmx.html.

[41] The MathWorks Inc. *nlmpc.* URL: https://se.mathworks.com/help/mpc/ref/nlmpc.html.

[42] The MathWorks Inc. *Nonlinear MPC Controller.* URL: https://se.mathworks.com/help/mpc/ref/nonlinearmpccontroller.html.

[43] The MathWorks Inc. *OPC Toolbox.* URL: https://se.mathworks.com/products/opc.html.

[44] The MathWorks Inc. *Optimization Decision Table.* URL: https://se.mathworks.com/help/optim/ug/optimization-decision-table.html.

[45] The MathWorks Inc. *quadprog.* URL: https://se.mathworks.com/help/optim/ug/quadprog.html.

[46] The MathWorks Inc. *Rate Transition.* URL: https://se.mathworks.com/help/simulink/slref/ratetransition.html.

[47] The MathWorks Inc. *Simulink.* URL: https://se.mathworks.com/products/simulink.html.

[48] The MathWorks Inc. *Simulink Compiler.* URL: https://se.mathworks.com/products/simulink-compiler.html#enabling-digital.

[49] The MathWorks Inc. *Simulink Desktop Real-Time.* URL: https://se.mathworks.com/products/simulink-desktop-real-time.html#monitor-signals.

[50] The MathWorks Inc. *Simulink Real-Time.* URL: https://se.mathworks.com/products/simulink-real-time.html.

[51] The MathWorks Inc. *Stateflow.* URL: https://se.mathworks.com/products/stateflow.html.

[52] The MathWorks Inc. *Subsystem, Atomic Subsystem, CodeReuse Subsystem.* URL: https://se.mathworks.com/help/simulink/slref/subsystem.html.

[53] The MathWorks Inc. *Symbolic Math Toolbox.* URL: https://se.mathworks.com/products/symbolic.html.

[54] The MathWorks Inc. *UDP Send.* URL: https://se.mathworks.com/help/dsp/ref/udpsend.html.

[55] The MathWorks Inc. *When the Solver Fails.* URL: https://se.mathworks.com/help/optim/ug/when-the-solver-fails.html.

[56] The MathWorks Inc. *writeRead.* URL: https://se.mathworks.com/help/instrument/writeread.html.

[57] The MathWorks Inc. *Writing Scalar Objective Functions.* URL: https://se.mathworks.com/help/optim/ug/writing-scalar-objective-functions.html#btt3sy0.

[58] National Instruments. *What Is LabVIEW?* URL: https://www.ni.com/en-no/shop/labview.html.

[59] National Intsruments. *USB-6212.* URL: https://www.ni.com/en-no/support/model.usb-6212.html.

[60] J. H. B. Sampaio Jr. "Designing 3D Directional Well Trajectories Using Bezier Curves". In: *ResearchGate* (2016).

[61] Silicon Laboratories. *Bluetooth Angle Estimation for Real-Time Locationing.* URL: https://www.silabs.com/whitepapers/bluetooth-angle-estimation-for-real-time-locationing.

[62] Lenze. *Inverter Drives 8400 TopLine C.* URL: https://download.lenze.com/TD/E84AVTCx__8400%20TopLine%20C__v11-1__EN.pdf.

[63] Magnetic-Declination.com. *What is magnetic declination in Trondheim, Norway?* URL: https://www.magnetic-declination.com/Norway/Trondheim/1877858.html.

[64] MOONS'. *M2 Series AC Servo User Manual.* URL: https://www.moonsindustries.com/medias/M2AC-Hardware-Manual-EN20190731-L.pdf?context=bWFzdGVyfHJvb3R8NjM3MjE3M3xhcHBsaWNhdGlvbi9wZG
attachment=true.

[65] MOONS'. *M2DV-1D82IP.* URL: https://www.moonsindustries.com/p/m2dv-series-servo-drives/m2dv-1d82ip-000004696351001660.

[66] MOONS'. *SM0402AE4-KCD-NNV.* URL: https://www.moonsindustries.com/p/m2dv-series-ac-servo-motors/sm0402ae4-kcd-nnv-000004611170000217.

[67] M.B. Nåmdal. *Design and Implementation of Instrumentation and Control System for an Autonomous Miniature Drilling Rig for Directional Drilling.* Master's Thesis. NTNU Department of Engineering Cybernetics, 2019.

[68] Society of Petroleum Engineers. *About Us.* URL: https://www.spe.org/en/about/.

[69] Nordic Semiconductor. *Bluetooth 5.1 Puts Bluetooth In Its Place.* URL: https://blog.nordicsemi.com/getconnected/bluetooth-5.1-puts-bluetooth-in-its-place.

[70] HT Sensor. *Static torque senso with flange TAT200.* URL: http://www.htc-sensor.com/products/187.html.

[71] speedgoat. *EtherNet/IP for Simulink.* URL: https://www.speedgoat.com/products-services/communications-protocols/ethernet-ip.

[72] STMicroelectronics. *HTS221.* URL: https://content.arduino.cc/assets/Nano_BLE_Sense_HTS221.pdf.

[73] STMicroelectronics. *LPS22HB.* URL: https://content.arduino.cc/assets/Nano_BLE_Sense_lps22hb.pdf.

[74] STMicroelectronics. *LSM9DS1.* URL: https://content.arduino.cc/assets/Nano_BLE_Sense_lsm9ds1.pdf.

[75] Wikipedia. *Bluetooth.* URL: https://en.wikipedia.org/wiki/Bluetooth.

[76] Wikipedia. *Bluetooth Low Energy.* URL: https://en.wikipedia.org/wiki/Bluetooth_Low_Energy.

[77] Wikipedia. *Broyden–Fletcher–Goldfarb–Shanno algorithm.* URL: https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm.

[78] Wikipedia. *Common Industrial Protocol.* URL: https://en.wikipedia.org/wiki/Common_Industrial_Protocol.

[79] Wikipedia. *Ethernet/IP.* URL: https://en.wikipedia.org/wiki/EtherNet/IP.

[80] Wikipedia. *Modbus.* URL: https://en.wikipedia.org/wiki/Modbus.

[81] Wikipedia. *Sequential quadratic programming.* URL: https://en.wikipedia.org/wiki/Sequential_quadratic_programming.

[82] Wikipedia. *State-space representation.* URL: https://en.wikipedia.org/wiki/State-space_representation.

[83] Wikipedia. *Transmission Control Protocol.* URL: https://en.wikipedia.org/wiki/Transmission_Control_Protocol.

[84] Wikipedia. *User Datagram Protocol.* URL: https://en.wikipedia.org/wiki/User_Datagram_Protocol.

Appendix

A Moving Particle MPC

To get started with the Model Predictive Control Toolbox in Simulink, a simplified system model was used to reduce complexity. The desired outcome was to have a functional implementation of a linear, adaptive, and nonlinear MPC and get familiar with the toolbox.

Model

The ideal model for testing purposes should have somewhat equal properties and dynamics to the trajectory control system. The model chosen simulates a particle moving in two dimensions. The input to the model is the orientation the inertial frame relative to the inertial x-axis. The movement is dependent on a constant velocity decomposed in the two dimensions.

$$\dot{x}_{pos} = v \cdot \cos(\theta) \tag{79a}$$

$$\dot{y}_{pos} = v \cdot \sin(\theta) \tag{79b}$$

The model is nonlinear but can be approximated using a linear model around $\theta = 0$. The states of the system are the positions in x and y with the input θ.

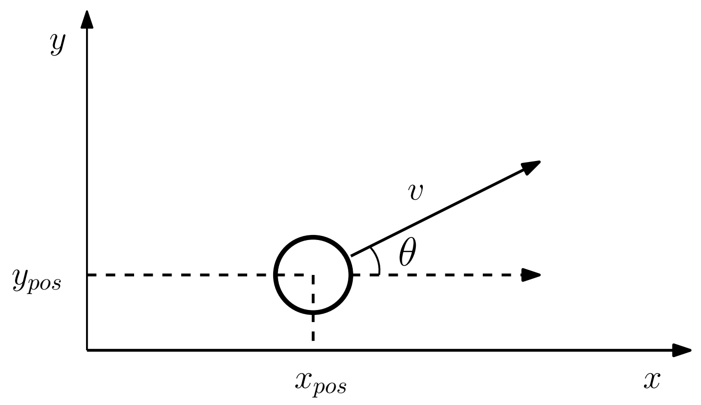


Figure 72: Illustration of states an inputs of the moving particle in two dimensions. The particle is infinitely small and has no inertia.

Plant Simulation

The plant is implemented in Simulink using a block-based programming interface. The plant takes θ and v as an input and outputs the current position in x and y. The model is implemented in continuous time. The input θ should not be allowed to change too fast to mimic a physical system. A rate limiter sets and upper and lower limit for the rate of change. The rate of change limit was set so that the dynamics was somewhat like how a car would be able to turn. For testing purposes, each controller is connected to their own plant simulation in closed-loop with the same reference.

Sinusoid Reference

One of the references used for the controllers was a sinusoidal function based on the current time. This reference was chosen as it would allow easy debugging in the early phase of implementation and using time instead of closest point to functions gives directly comparable results. The controller can be set to only optimize for a track a single axis at a time, making the controller easier to debug.

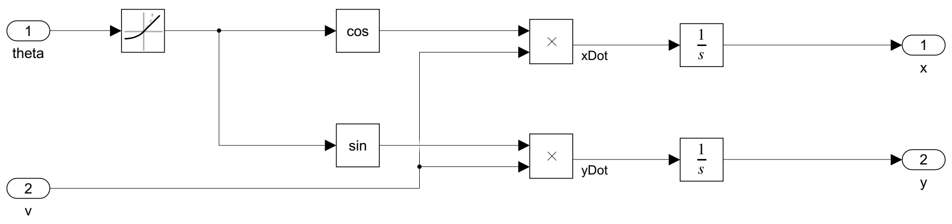


Figure 73: Simulink diagram of the continuous time plant.

All experiments with this reference only penalized y-axis deviation by setting the x-axis weight to zero.

$$y_{ref}(t) = 0.05 \cdot v \cdot \sin(0.04 \cdot t) \tag{80}$$

The constants where determined based on the maximum rate of change such that the slew rate would not limit the motion tracking capabilities of the controller. Slew rate limitations can be detected when there is a constant curvature arc that does not follow the reference with a sinusoid reference. The reference is an array of future desired y-coordinates for the prediction window. The coordinates will then depend on the velocity and the time-step.

Circle Reference

A circular reference was used to test tracking in two dimensions. A circle with $r = 10$ was chosen as it was simple to implement and was nonlinear.

$$x_{ref}(t) = r \cdot \sin(\frac{v \cdot t}{r}) \tag{81a}$$

$$y_{ref}(t) = r - r \cdot \cos(\frac{v \cdot t}{r}) \tag{81b}$$

The offset in radius was set such that system could start at the zero coordinate with θ in the zero position at the bottom of the circle. The radius was chosen such that the slew rate would not be an issue, even if the controller temporarily had a significant deviation from the reference.

Testing Parameters

Parameters used for the experiments unless explicitly noted are listed in Table 26.

The x-coordinate deviation depends on the type of reference. The sinusoid reference has deviation weight of zero.

Inertial Frame MPC

Controlling the particle in the inertial frame was attempted with a linear, adaptive, and nonlinear MPC. Controlling the system from the inertial frame does not require any translations of the reference or the control inputs, simplifying the implementation and reducing complexity.

Table 26: Parameters Used in Experiments.

Variable	Symbol	Value	Unit
MPC Time-Step	-	0.2	s
Velocity	v	1	-
Orientation, Initial	θ_0	0	rad
X-Coordinate, Initial	x_0	0	-
Y-Coordinate, Initial	y_0	0	-
Orientation Rate, Maximum	$\dot{\theta}_{max}$	0.3	rad/s
Orientation Rate, Minimum	$\dot{\theta}_{min}$	-0.3	rad/s
X-Coordinate Deviation Weight	q_x	0/1*	-
Y-Coordinate Deviation Weight	q_y	1	-
Prediction Horizon	n	10	-
Control Horizon	m	3	-

Linear MPC

A linear MPC can be implemented in Simulink using the MPC block. The block uses a static discrete system model in state-space form. The system model described in Equation 79 is nonlinear and cannot be described using a state-space form. The system must therefore be linearized. Because the model is static the linearization point cannot be changed based on position.

With the operating point chosen at $\theta = 0$ and choosing a nominal velocity v_{nom} and using the approximation of the derivative described in Equation 9, the linearized and discretized model becomes

$$x_{pos}[k+1] = x_{pos}[k] + (-v_{nom} \cdot \sin(\theta_0) \cdot \theta[k] + v[k] \cdot (\cos(\theta[k]) + \sin(\theta[k]) \cdot \theta[k])) \cdot \Delta t \tag{82a}$$

$$y_{pos}[k+1] = y_{pos}[k] + (v_{nom} \cdot \cos(\theta_0) \cdot \theta[k] + v[k] \cdot (\sin(\theta[k]) - \sin(\theta[k]) \cdot \theta[k])) \cdot \Delta t \tag{82b}$$

The linear MPC block has an option for adding a measured disturbance. The measured disturbances are included in the new state-space model as their own inputs. Measured disturbances can be calculated outside the controller block.

$$u = \begin{bmatrix} \theta \\ x_{pos,md} \\ y_{pos,md} \end{bmatrix} \tag{83}$$

Where $x_{pos,md}$ and $y_{pos,md}$ is defined as.

$$x_{pos,md}[k] = v[k] \cdot (\cos(\theta[k]) + \sin(\theta[k]) \cdot \theta[k]) \tag{84a}$$

$$y_{pos,md}[k] = v[k] \cdot (\sin(\theta[k]) - \sin(\theta[k]) \cdot \theta[k]) \tag{84b}$$

Using the new input vector, it is possible to create a state-space model that can be used by the MPC block.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -v_{nom} \cdot \sin(\theta_0) \cdot \Delta t & \Delta t & 0 \\ v_{nom} \cdot \cos(\theta_0) \cdot \Delta t & 0 & \Delta t \end{bmatrix} \tag{85a}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{85b}$$

Simulink Implementation

The linear MPC controller is implemented using the Simulink block described in **??**. In addition, a MATLAB function block is used to calculate the measured disturbances of the model. The memory blocks are used to avoid short circuits for values used both as inputs and outputs to the MPC block.

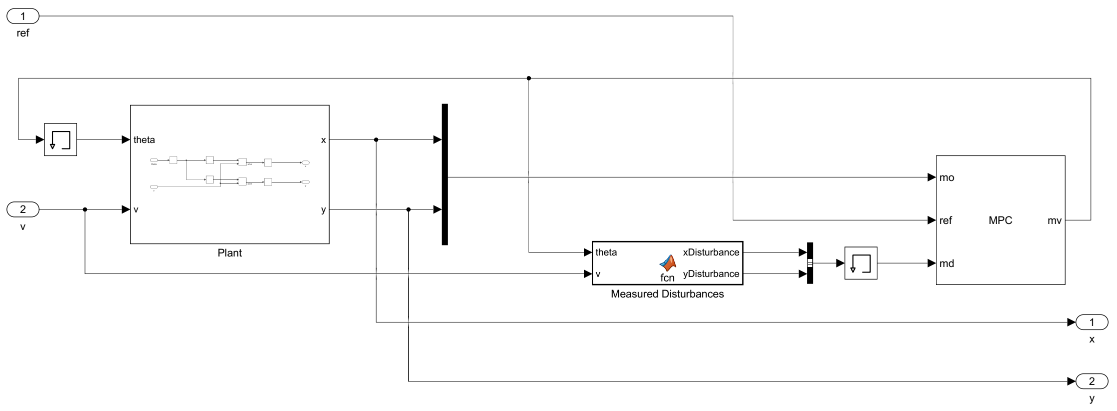


Figure 74: Simulink diagram of linear MPC.

Sinusoid Reference Tracking

The controller can follow the trend of the reference but trails behind and does not reach the same amplitude as the reference. The main reason for the poor tracking is an inaccurate model in the controller. With the initial conditions of $\theta = 0$, the controller assumes that the movement rate in x is not affected by the input and the movement rate in y is equal to the value in θ_0. Furthermore, the measured disturbance is assumed constant for the prediction horizon, while it is a function of the input. This is a limitation of the state-space model.

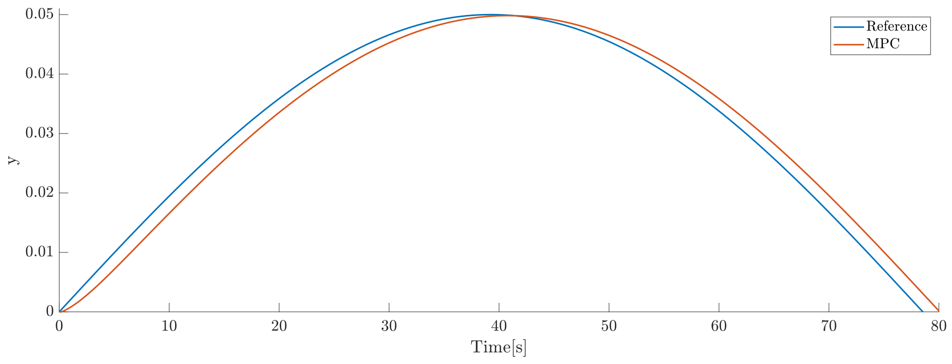


Figure 75: Sinusoid reference tracking using a linear MPC controller.

Circle Reference Tracking

The controllers track the trend of the reference initially for a small θ until about 45 degrees before diverging from the reference. The controller does not know how to manipulate its x position and has an inaccurate model of how the input manipulates the y position. Once the reference is outside the prediction horizon, the particle enters a spiral. The controller model is flawed and will not be able to track this reference.

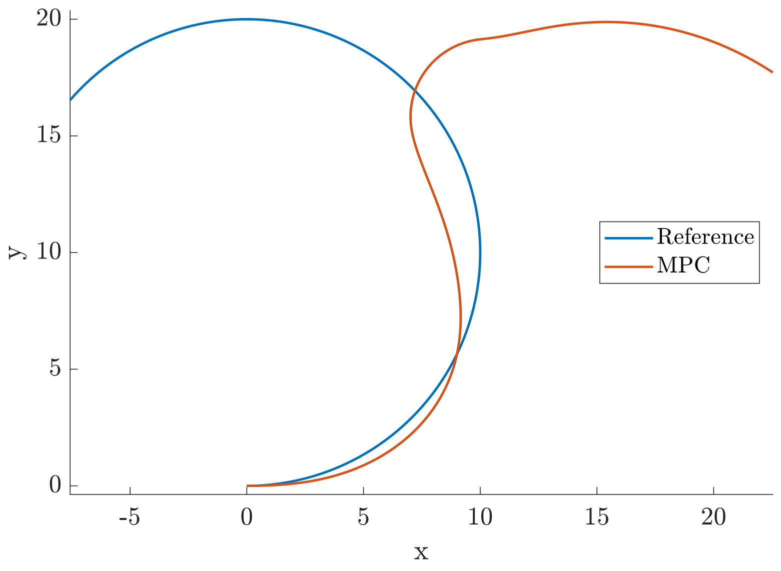


Figure 76: Circle reference tracking using a linear MPC controller for a duration of 40 seconds.

Adaptive MPC

The linear MPC controller was able to follow the trend of the reference when θ was close to the linearization point θ_0 for the circle reference, however the controller failed to track the reference for the circle reference. Looking at the B matrix in Equation 85, the model will produce a very poor approximation when θ is not close to zero and especially when the θ is larger than 90 degrees. Continuously updating the system model with a new linearization point enables the model to be more accurate for a larger range of values.

Model

The model is equal to the model in Equation 82 with the exception that there is no static point of linearization.

$$x_{pos}[k+1] = x_{pos}[k] + (-v_{nom} \cdot \sin(\theta[k]) \cdot \theta[k] + v[k] \cdot (\cos(\theta[k]) + \sin(\theta[k]) \cdot \theta[k])) \cdot \Delta t \tag{86a}$$

$$y_{pos}[k+1] = y_{pos}[k] + (v_{nom} \cdot \cos(\theta[k]) \cdot \theta[k] + v[k] \cdot (\sin(\theta[k]) - \sin(\theta[k]) \cdot \theta[k])) \cdot \Delta t \tag{86b}$$

The inputs u equal to the linear MPC described in Equation 83 and the disturbances $x_{pos,md}$ and $y_{pos,md}$ are equal to Equation 84 since the adaptive model will not accept values not related to any of the states or inputs.

The adaptive state-space model is equal to Equation 85 with the exception of the linearization point θ_0 being replaced with $\theta[k]$.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -v_{nom} \cdot \sin(\theta[k]) \cdot \Delta t & \Delta t & 0 \\ v_{nom} \cdot \cos(\theta[k]) \cdot \Delta t & 0 & \Delta t \end{bmatrix} \tag{87a}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{87b}$$

Simulink Implementation

The main difference from the linear controller is the addition of another MATLAB function block. The function block creates a new model for every time-step as described in Equation 87. The model still relies on measured disturbances that is assumed to be constant for the prediction horizon. The adaptive MPC block does only accept a bus input signal. The model outputs from the MATLAB function are converted to a bus signal using a Bus Creator.

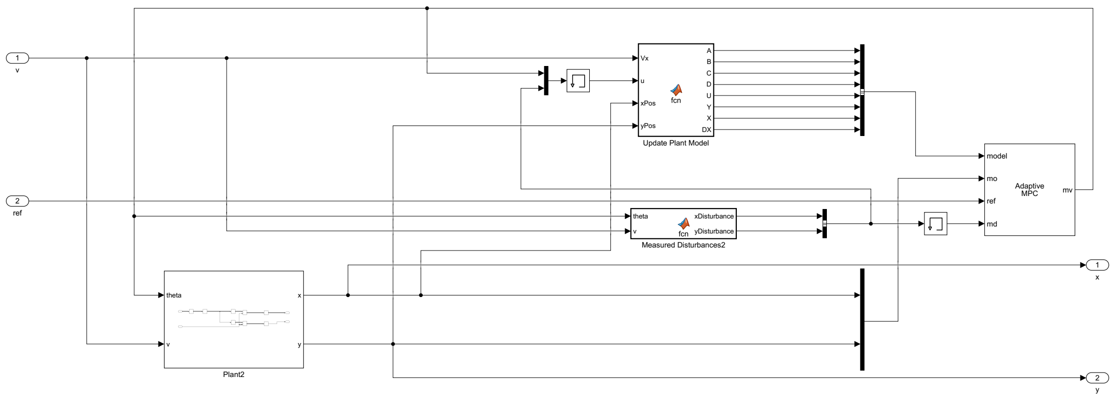


Figure 77: Simulink diagram of adaptive MPC.

Sinusoid Reference

Because the model can be updated during run-time the model is able to track the more closely than the linear model. The controllers still trail behind the reference, but to a lesser extent than the non-adaptive model. The measured disturbances are affected by the inputs without the controller being able to account for this. A time-based reference makes it difficult for the controller to converge when trailing behind if the exact reference. This results in a slightly smaller amplitude than the reference.

Circle Reference

The adaptive model is not able to follow the circular reference. The controller can follow the trend of the reference for small angles before diverging. Even with an adaptive model, the linear approximation is flawed and will not be able to follow a circle reference. The same accumulating θ problem is still present as with the other linear model will become more inaccurate over time.

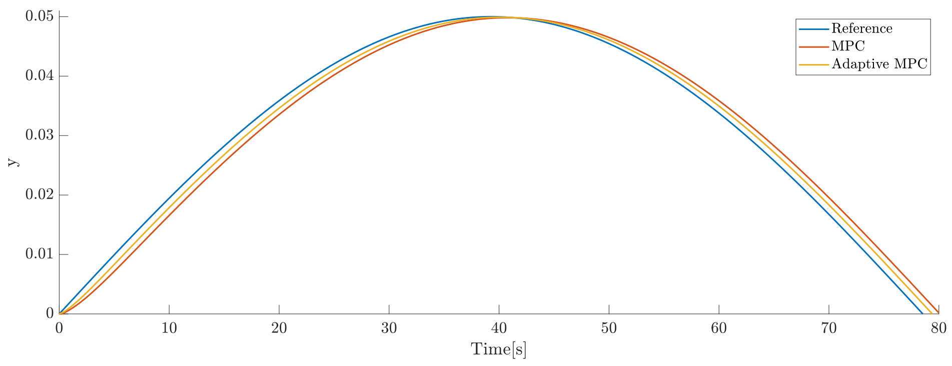


Figure 78: Sinusoid reference tracking using a adaptive MPC controller.

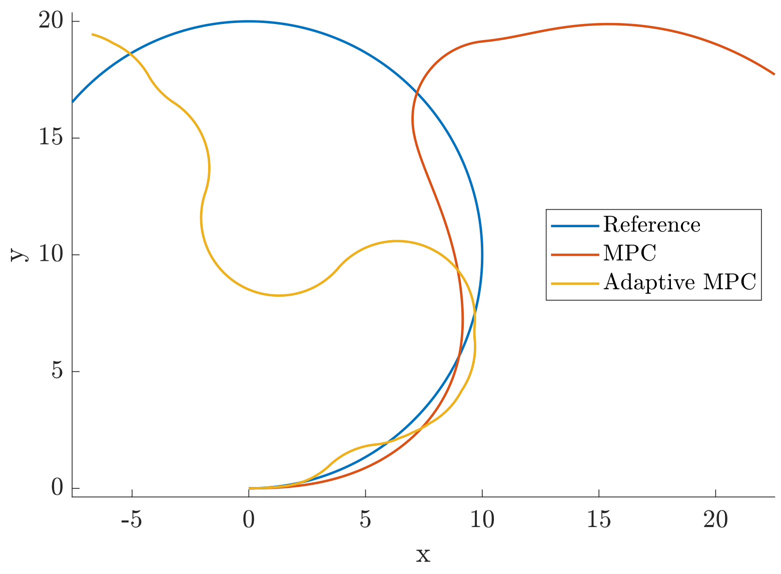


Figure 79: Circle reference tracking using a adaptive MPC controller for a duration of 40 seconds.

Nonlinear MPC

Because a linear MPC controller will not be able to follow the reference over time, a nonlinear MPC cotnroller is used. Nonlinear controllers will generally have a larger computational cost than linear and adaptive controllers. The nonlinear MPC controller in Simulink is not restricted to using a state-space model, allowing full flexibility in choice of model, constraints, and cost function.

The nonlinear MPC Simulink block accepts a continuous time nonlinear model. The model is therefore exactly equal to the system model described in Equation 79.

Simulink Implementation

The nonlinear MPC block takes the current state, last measured variable, and parameters. The last measured variable is by default used as the initial guess for the optimization. The signal from measured variables is delayed by a time-step to avoid short circuit. Parameters are values that are not directly controlled by the NMPC and are assumed constant for the prediction horizon. The params input will only accept a bus signal from the Bus Creator block.

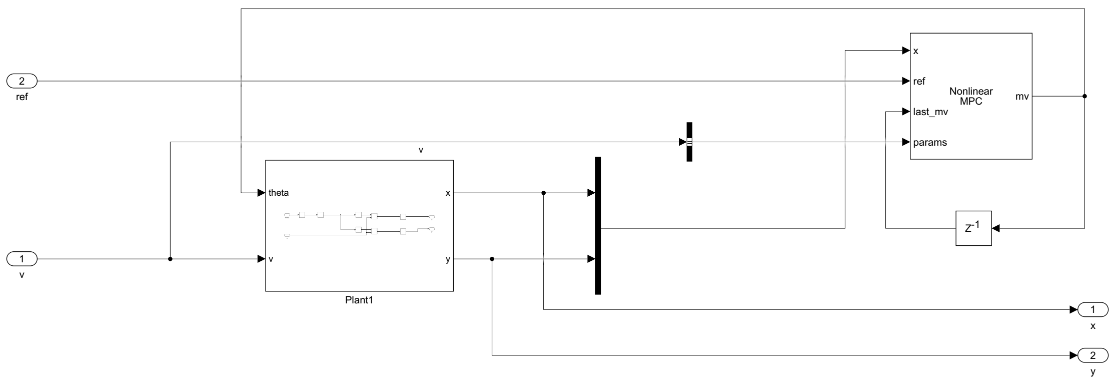


Figure 80: Simulink diagram of nonlinear MPC.

Sinusoid Reference

The nonlinear controller will quickly converge towards the sinusoid reference and is able to closely follow the reference. The control model is exactly equal to the system model and the time-step is small. Compared to the linear models the control performance is significantly better with a higher computational cost.

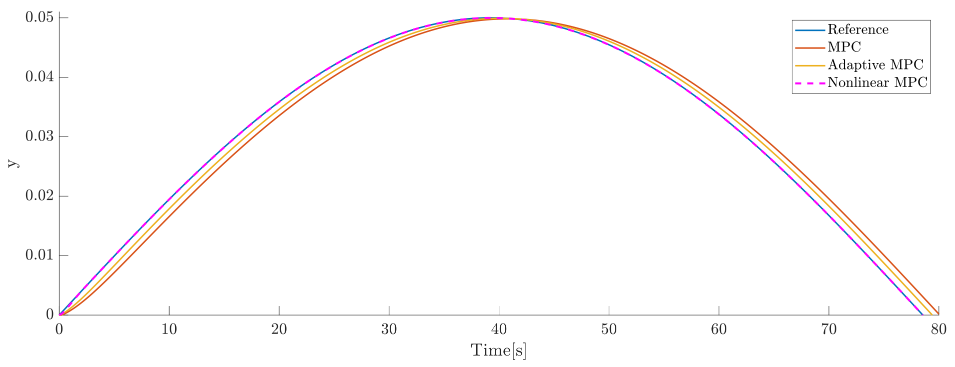


Figure 81: Sinusoid reference tracking using a nonlinear MPC controller.

Circle Reference

Like the sinusoid reference, the controller will quickly converge towards the reference and follow it accurately.

Input Comparison

Non-abrupt change to the inputs is a requirement for many physical systems. There is an upper limit to the maximum rate of change and weights will penalize the rate of change to encourage minimal changes to the controller input. A comparison of the inputs to the system is shown in Figure 83.

The nonlinear controller converges quickly towards a slowly changing input. The linear and adaptive MPCs will take longer time before converging to a similar input to the nonlinear controller.

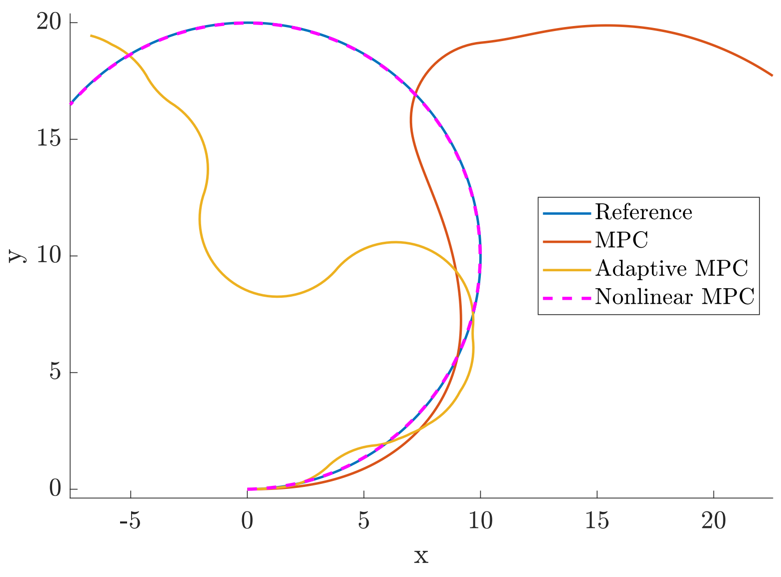


Figure 82: Circle reference tracking using a nonlinear MPC controller for a duration of 40 seconds.

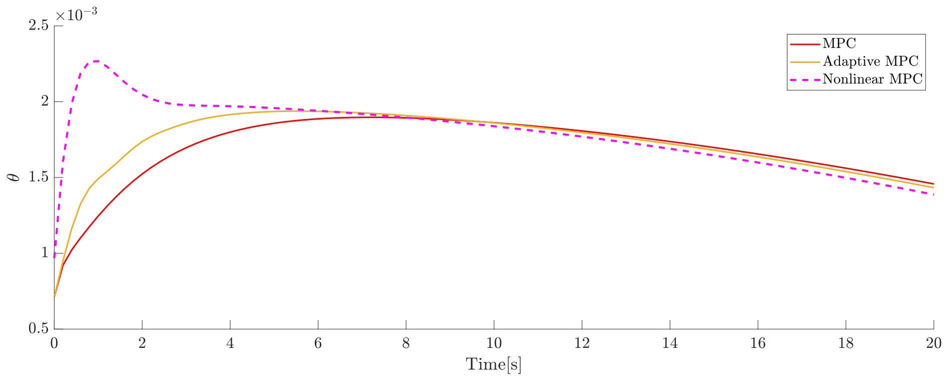


Figure 83: Inputs to the system with the different controllers.

Particle Frame MPC

Simulations with MPC controllers in the inertial frame suggests that linear controllers will not be able to control the particle from the inertial frame due to model shortcomings. Nonlinear MPC has a high computational cost, limiting the maximum prediction horizon and maximum time-step.

Changing from the inertial to the frame of the particle in the direction of the current velocity vector allows the model to have a more accurate linear approximation model and not require measured disturbance inputs. The inertial frame reference is translated to the frame of the particle using a rotation matrix.

$$R = \begin{bmatrix} \cos(\theta[k]) & -\sin(\theta[k]) \\ \sin(\theta[k]) & \cos(\theta[k]) \end{bmatrix} \tag{88}$$

The initial position and the orientation are hardwired to be zero in the controller as it moves relative to itself. The inertial reference must be translated based on current position displacement and orientation. The input provided by the controller is relative to the current orientation as shown in **??**.

$$\theta[k+1] = \theta_{controller}[k+1] + \theta[k] \tag{89}$$

Reference translation to the frame of the particle is found using a combination of position displacement and the rotation matrix in Equation 88. The coordinates for the circle reference coordinates are described in Equation 81.

$$\begin{bmatrix} x_{ref,inertial} \\ y_{ref,inertial} \end{bmatrix} = \begin{bmatrix} x - x_{ref} \\ y_{ref} - y \end{bmatrix} \tag{90}$$

The inertial reference coordinates are then multiplied with the rotation matrix so that the reference coordinates are given in the frame of the particle.

$$\begin{bmatrix} x_{ref,particle} \\ y_{ref,particle} \end{bmatrix} = R \cdot \begin{bmatrix} x_{ref,inertial} \\ y_{ref,inertial} \end{bmatrix} \tag{91}$$

Linear MPC

With the change of frame of the controller, the linear MPC controller gives the optimal input in its own frame based on a reference translated in position and orientation from the inertial frame.

The change of reference has implications for the linear model. With $\theta_0 = 0$ due to the change of frame, $\sin(\theta)$ can be approximated to zero, and $\cos(\theta)$ can be approximated to be one in the linear model. This simplifies the linearized model for the inertial frame controller described in Equation 82.

$$x_{pos}[k+1] = x_{pos}[k] + v[k] \cdot \Delta t \tag{92a}$$

$$y_{pos}[k+1] = y_{pos}[k] + v_{nom} \cdot \theta[k] \cdot \Delta t \tag{92b}$$

The new simplified model relies less on measured disturbances, but still uses a nominal velocity.

$$x_{pos,md}[k] = v[k] \cdot \theta[k] \tag{93a}$$

$$y_{pos,md}[k] = 0 \tag{93b}$$

Because there is no measured disturbance for the y state, the input vector will have one less value.

$$u = \begin{bmatrix} \theta \\ x_{pos,md} \end{bmatrix} \tag{94}$$

With the change of input matrix, the order or the B and D matrices can be reduced.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & \Delta t \\ v_{nom} \cdot \Delta t & 0 \end{bmatrix} \tag{95a}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{95b}$$

The new discretized state-space model can be implemented directly in the MPC Simulink block.

Simulink Implementation

A new MATLAB function block is added to transform the inertial frame reference to the frame of the particle described in Equation 90 and Equation 91. The measured output input to the MPC block is hardwired to a constant matrix block as the initial coordinates of the particle in its own frame will always be zero. The input transformation from Equation 89 is handled by an addition and memory block. The measured disturbance input must be a bus signal. The velocity signal is converted into a bus signal using a Bus Creator block.

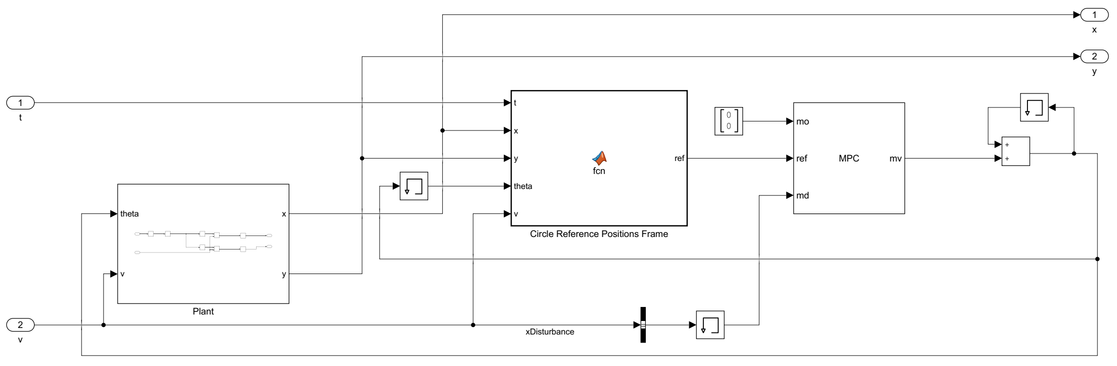


Figure 84: Simulink diagram of linear MPC with controller frame moved to frame of the particle.

Circle Reference

The new linear controller has very good tracking of the reference when the velocity is equal to the nominal velocity. There is no longer an issue with an accumulating θ value as with the inertial frame linear controller.

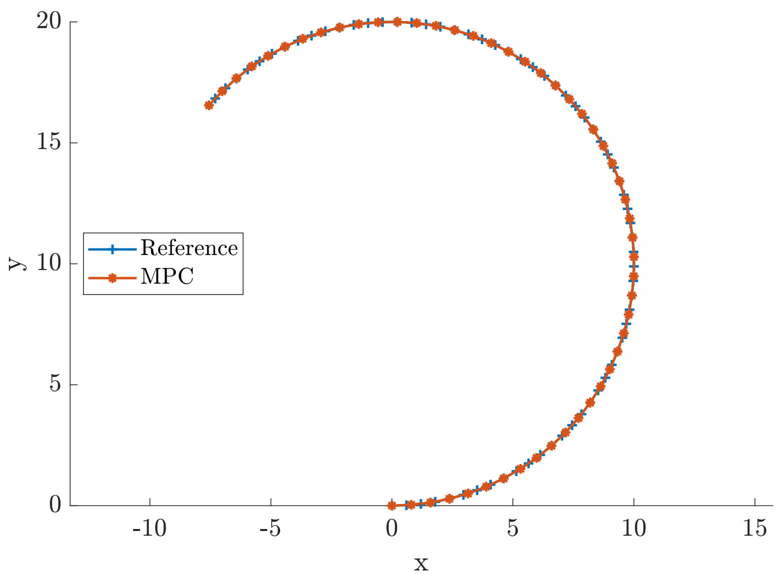


Figure 85: Circle frame reference tracking using a linear MPC controller for a duration of 40 seconds.

The model still relies on a nominal speed.

Adaptive MPC

A linear controller can follow the references if the particle is controlled from its own frame. The implementation of the linear MPC has a shortcoming of that the model cannot adapt to changes in the velocity. If the speed changes considerably the controller performance will be affected. Using adaptive MPC the controller model can be changed during runtime.

The adaptive model is equal to the linear model in Equation 96 with the exception that the nominal value of the velocity is replaced with the velocity at the current time-step.

$$x_{pos}[k+1] = x_{pos}[k] + v[k] \cdot \Delta t \tag{96a}$$

$$y_{pos}[k+1] = y_{pos}[k] + v[k] \cdot \theta[k] \cdot \Delta t \tag{96b}$$

The adaptive controller state-space model has similar alterations to Equation 95.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & \Delta t \\ v[k] \cdot \Delta t & 0 \end{bmatrix} \tag{97a}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{97b}$$

Simulink Implementation

The diagram is a combination of the inertial frame adaptive MPC controller and the particle frame linear MPC controller. The MATLAB Function block updating the model takes velocity as input and outputs the state-space model for the current velocity.

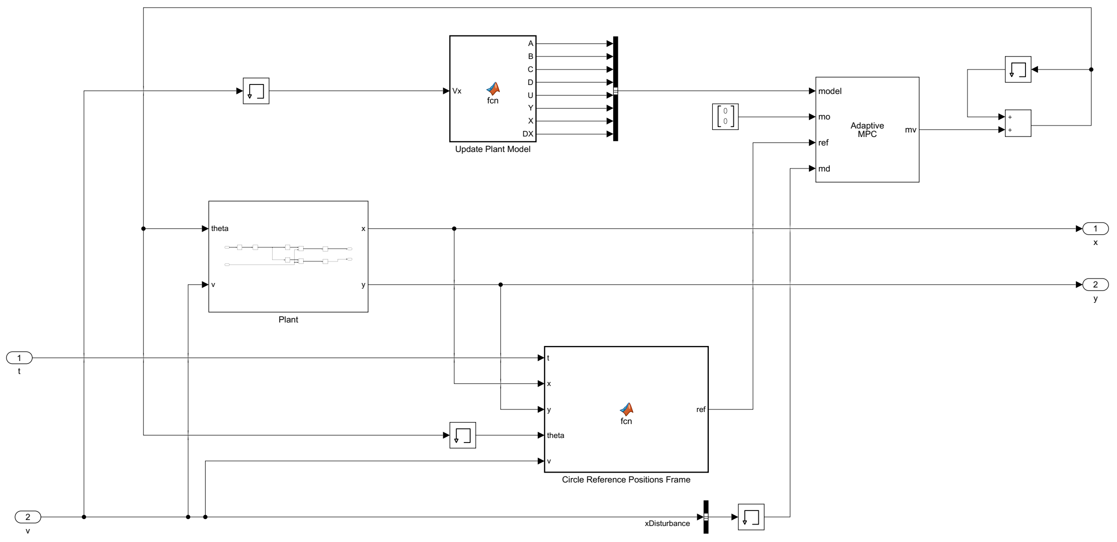


Figure 86: Simulink diagram of adaptive MPC with controller frame moved to frame of the particle.

Circle Reference

Using the nominal speed for the controllers, the linear and adaptive controllers had very similar performance. Both controllers have an equal model when the velocity is equal to the nominal velocity. The controllers have slight differences in Kalman filter implementations; however the states are hardwired to zero for both controllers.

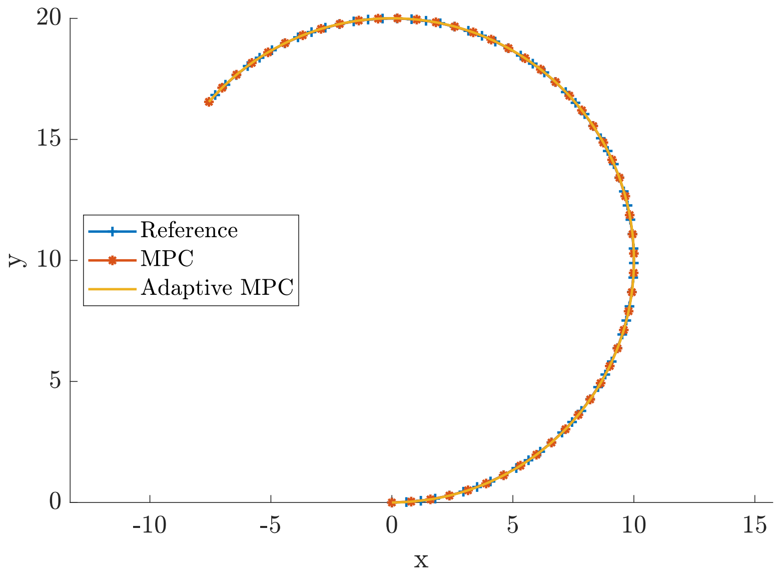
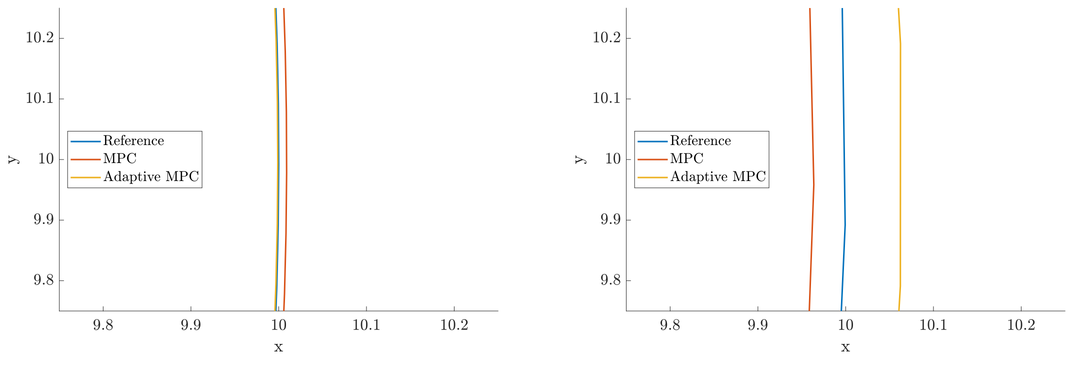


Figure 87: Circle frame reference tracking using a adaptive MPC controller for a duration of 40 seconds.

Changing the velocity shows the differences in the controllers. The adaptive controller has a slower turning rate than the linear when the velocity is greater than the nominal velocity, and a higher turning rate when the nominal velocity is smaller than the velocity.

An observation is that in the case of the double velocity, the tracking of the linear controller with the higher turning rate is closer to the reference. Reducing the velocity to half the nominal velocity effectively increases the sampling frequency to twice relative to movement length.



(a) Half the nominal velocity.

(b) Double the nominal velocity.

Figure 88: Comparison of nonnominal velocities for linear and adaptive MPC.

Nonlinear MPC

It is also possible to implement a nonlinear MPC in the frame of the particle. The nonlinear controller would normally not be used when there is good control performance with a linear as it is more computationally expensive. The controller was mainly implementation to compare the controller performance and have a functional implementation of a nonlinear controller in the frame of the particle.

The nonlinear MPC model is equal to that of the inertial frame in Equation 79. Current position and orientation is hardwired to zero, and outputs from the controller will be corrections to the input rather than the input used by the system compared to Equation 89.

Simulink Implementation

The Simulink model uses the same frame correction blocks as the linear and adaptive particle frame controllers.

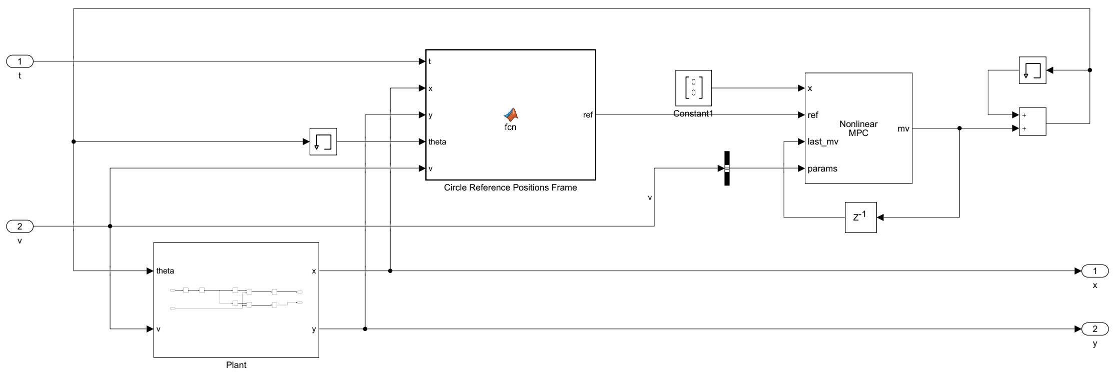


Figure 89: Simulink diagram of nonlinear MPC with controller frame moved to frame of the particle.

Circular Reference

The controller manages to follow the trend of the reference over time but has worse performance than the linear and adaptive controllers. Possible explanations for this could be the controller not finding the optimal solution, over-fitting of the reference points, and suboptimal tuning of controller parameters. The controller could potentially be tuned to achieve similar performance as the linear controllers.

Input Comparison

The input from the nonlinear controller is more fluctuating. Looking at the inputs in Figure 91, the inputs oscillate and is less consistent than both the linear controllers. This could possibly be improved with more tuning.

Conclusion

Linear, adaptive, and nonlinear MPC controllers were successfully implemented in Simulink for a simplified system model. The functional implementation will work as a template for implementing

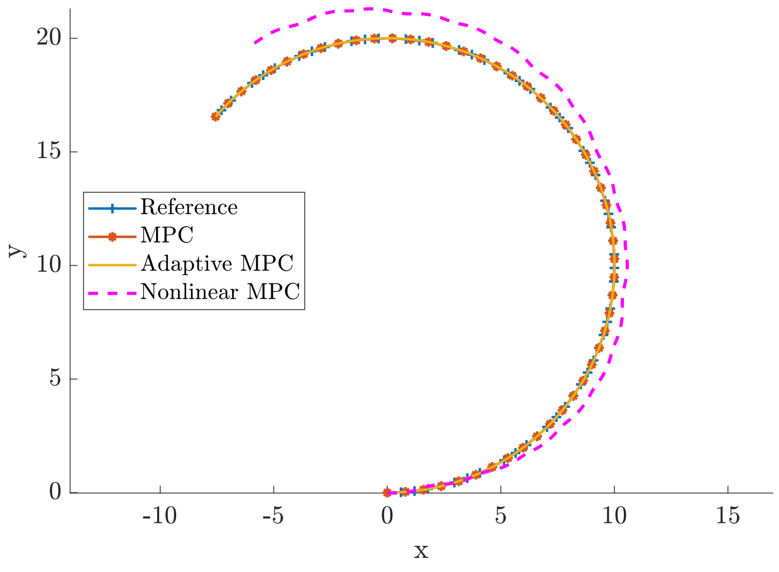


Figure 90: Circle frame reference tracking using a nonlinear MPC controller for a duration of 40 seconds.

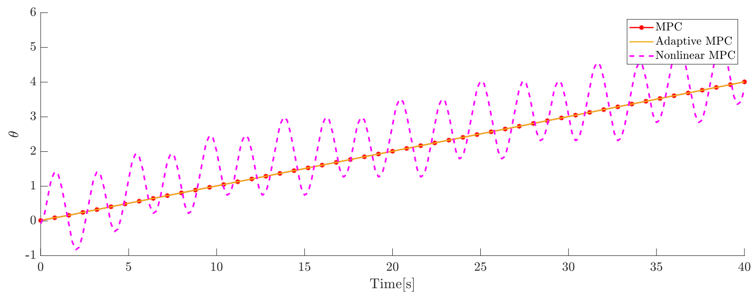


Figure 91: Circle frame input comparison.

a more advanced model later for controlling BHA trajectory. Moving the frame of the controller might be beneficial allow the usage of a less complex model for directional drilling control to reduce computational cost.

There was no experimentation with different solvers or fine tuning of prediction horizon, control horizon, weights, time-step, constraints, cost function, and parameters as the implementation was intended as an exercise to get familiar with the toolbox rather than trying to achieve the best possible controller performance.

B Sensor Housing Implementation Attempts

A total of 5 attempts were made at implementing the downhole sensor into the sensor housing with sufficient water seal.

First Attempt

For the first attempt the downhole sensor was covered in electrical tape along its side to prevent short circuits to the metal sensor housing. A small piece of electrical tape was placed on the microphone to protect the diaphragm from epoxy. Epoxy was then filled from the bottom with the sensor package was already inserted. Forcing high viscosity epoxy into the tiny gaps along the sensor proved difficult. The gap between the wire and upper stabilizer hole was too small to fill epoxy from the top. Because there was no way of resetting after the downhole sensor was covered in epoxy, it was attempted to continue with the suboptimal epoxy application to test if it was sufficient.

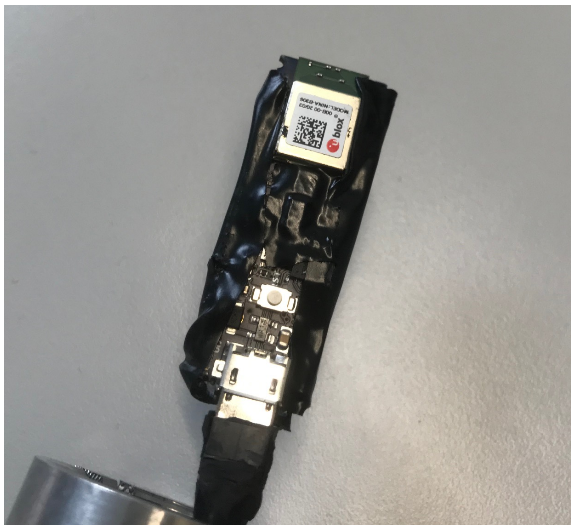


Figure 92: Arduino with electrical tape.

After letting the epoxy dry overnight, the Arduino was connected to a computer. The device was detectable in Arduino IDE and showed up as a BLE device in MATLAB. BLE communication was tested, confirmed that it was possible to communicate from inside the sensor sub. The magnetometer was tested to see if it was possible to accurately and reliable measure yaw from inside the metal enclosing. There were significant soft iron distortions differences compared to the sensor without enclosing. A calibration allowed the yaw to reliably and accurately be determined from magnetometer data.

The BHA was then submerged in a sink for 10 minutes to test if the epoxy had made a waterproof seal. When the Arduino was connected to a computer it was detected as an unrecognized device and did not broadcast its Bluetooth information when connected to power. Insufficient epoxy seal around the exposed micro USB connector and in the top of the sensor housing was assumed to be the cause. A heat gun was used to remove the epoxy and prepare for another attempt.

Second Attempt

The next attempt used a thin pre-applied layer of epoxy on the Arduino instead of electrical tape. The upper stabilizer was filled with epoxy before the wire was threaded through. Epoxy was then filled in the sensor compartment before inserting the Arduino. A heat gun was used during the process to reduce the viscosity of the epoxy. When the Arduino was fully inserted into the sensor

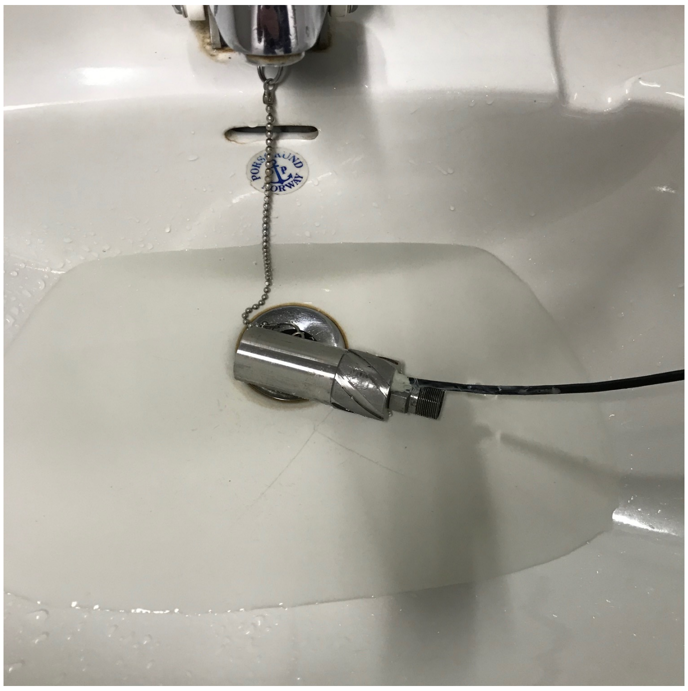


Figure 93: Submerging test of epoxy application.

compartment using a small flathead screwdriver, the bottom of the sensor housing was sealed with more epoxy.

The epoxy was set to dry overnight. When connected to the computer the device was not recognized, yet it broadcasted its BLE address and could send data wirelessly. At least one of the data transfer wires had likely been damaged inside the BHA and the device could not communicate over serial or be reprogrammed. Because this scenario was a possibility the device was programmed with the latest stable wireless communication software before being inserted. Device behaviour did not change after a submerging test.

Because the sensor package was still functional it was tested while drilling. BLE communication while drilling was possible but unreliable, especially at the later stages of the drilling operation as signal strength was reduced. The Arduino was used without issues for 3 drilling operations before unexpectedly becoming undetectable a while after a drilling operation without issues, even after multiple power cycles. Device failure was assumed to be caused by a small water leakage.

Third Attempt

To further increase waterproofing, the Arduino was but inside in a shrinking tube before epoxy was applied. The shrinking tube was intended as a protective layer around the connector and pins to reduce damage for small leakages.

A small shrinking tube was applied around the USB connector and a larger shrinking tube around the PCB. The shrinking tube failed to provide a complete seal at the top and bottom of the PCB as it did not deform as much as expected when heat was applied. Epoxy was filled into the chamber equally to the second attempt. Adding a shrinking tube increased the size of the sensor package, making insertion into the sensor housing more difficult.

The epoxy was set to dry over night before being connected to a power supply. The device did broadcast its BLE address and sent data, but frequently dropped the connection after around 10 seconds. Submerging test did not change functionality.

The issue was assumed to be caused by damaged connectors inside the BHA. Wires are forced into a small space and are exposed to stress in the top of the sensor housing and bottom of upper

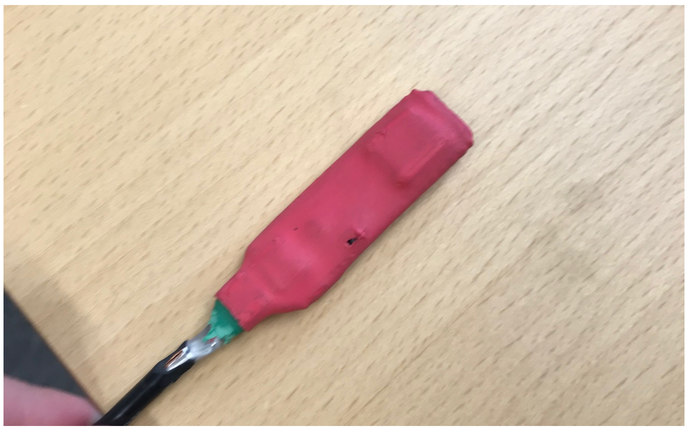


Figure 94: Arduino with shrinking tube.

stabilizer. It is assumed that one of the two power wires has been slightly damaged and is not able to provide sufficient maintained voltage causing brownouts.

Fourth Attempt

A new type of protective coating called Plasti Dip was tested. Plasti Dip is thin protective plastic layer applied by a spray can in multiple thin layers. Epoxy was applied as with attempt 2 and 3.

To reduce stress on the wiress, the hole in the bottom of the upper stabilizer was slightly widen. The Arduino was inserted into the sensor chamber with the minimal clearance to the bottom of the sensor housing that would allow a protective epoxy layer. The fourth attempt worked reliably until the cable was destroyed when the mount between the BHA and drill pipe loosened. The fifth attempt was a repeat of the procedure and was used until the end of the project.

Backup Solution

Multiple failed attempts at implementation and unexpected failures meant a backup solution was required. Instead of having the sensor package inside the BHA, a 3D printed plastic casing intended to be placed on top of the BHA was made. The plastic casing had an open compartment allowing the Arduino and cable to be directly inserted without resoldering the trimmed USB cable.

The casing had to be orientated in the opposite of the ideal orientation explained in subsection 9.11 to avoid contacting the well path while drilling a deviated well path. Magnetometer measurements had less soft iron distortion compared to sensors placed inside the BHA. The main drawback of the backup solution was increased displacement from the bit.

C Azimuth Motor EtherNet/IP Integration

Device communication in Simulink is highly resource demanding. If the azimuth motor could be configured for EtherNet/IP communication, the same Modbus commands used for the hoisting and top drive motor through the gateway could use aggregated commands explained in subsubsection 7.7.3 to reduce computational load drastically for Modbus TCP communication in the control system.

The EDS file for the azimuth drive was available at the product web page[65]. The file was imported in SYSCON.net software and added as another node on the EtherNet/IP subnet. The address space of the drive was added after the address spaces of the other devices. There were no provided examples for using EtherNet/IP in the documentation as with Modbus TCP. Based on the chapter describing EtherNet/IP it was assumed that SCL commands was to be used over the interface, and references to SCL was found in the EDS file. SCL commands is a basic set of text commands used to control the drive.

After configuring the drive for SCL command operation over Ethernet in M Servo Suite it was possible to read from the drive registers through the gateway. The gateway automatically mapped the EtherNet/IP values to Modbus registers. The EDS file has a name parameter for each accessible input and output.

It was first attempted to extract read values from the gateway through Modbus Explorer. The EDS file defined the datatype of all outputs to be 0xC4. The code is defined as a signed 32-bit integer. The number of bytes allocated in the gateway was 4 times the number of entries in the EDS-file, further verifying the expected data type. Because the hoisting and top drive read values was converted to input registers it was assumed that azimuth would use the same register type.

Azimuth values were available as 32-bit signed integer input registers as expected. Output values were large and byte order was switched to little endian. Little endian byte yielded sensible values. The drive was manually put in jog mode from the control panel on the physical drive to verify that the expected velocity value matches the register address. Addresses was then compared to their names specified in the CIP file relative to the velocity address. Mapping is shown in **??**.

Table 27: Azimuth EtherNet/IP to Modbus TCP Addresses.

Value	**Address**	**Data type**
StatusCode	33	int32
AlarmCode	35	int32
SupplyVoltage	37	int32
ActualCurrent	39	int32
DriveTemperature	41	int32
EncoderPosition	43	int32
AbsolutePosition	45	int32
PositionError	47	int32
ActualVelocity	49	int32
InputStatusExtended	51	int32
InputStatusMain	53	int32
OutputStatus	55	int32
AnalogInput1	57	int32
AnalogInput2	59	int32
Reg_1	61	int32
Reg_2	63	int32

No further explanations of states have been found in the driver manual or EDS file.

The final challenge was to send commands to the drive through the gateway. There was very little documentation and no examples on how to send commands through EtherNet/IP in the manual. Finding out how to send commands proved difficult with no indications rather than param names in the EDS file. The implementation was therefore not successful and further attempts discontinued after some of trial and error. Two commands that were tested with multiple different inputs were Current Command and SCL Command Letter. SCL is a command language that was successfully used to control the drive through a dedicated SCL client. It is probably possible to control the drive through the gateway and doing so would reduce the computational cost of communication. An intermediate step for figuring out the command structure would be to use a dedicated to EtherNet/IP client to figure out the native communication structure, however no free clients were found. If commands were to successfully control the drive through the gateway, unused write values would have to be unmapped in SYSCON.net to make use of aggregated commands described in subsubsection 7.7.3 as M Servo Suite does not allow EtherNet/IP values to be reconfigured as with Lenze Engineer.

D Downhole Data Filtering

High frequency data retrieval from the downhole sensor can be computationally demanding for the control system computer. Data can be sampled and transferred at 100Hz by the sensor package, while the high-level control system had difficulties receiving and processing data at 10Hz over a BLE connection.

IMU data is mainly used for position estimation. Raw data can be filtered downhole before being transferred to offload the high-level control system. Not transferring raw data to surface has the drawback of removing data that could have been used for post-analysis.

A median filter with extreme value exclusion was implemented. From the total of 10 samples, the minimum and maximum values are excluded and the average of the remaining 8 values transferred to surface. No arrays are needed if the sum of the non-excluded values is stored between samples.

```
// Define size of window and counter to keep track of current iteration
int averageWindowSize = 10;
int averageWindowIndex = 0;
float accXWindowMax;
float accXWindowMin;
float accXAverageWindowSum;
float accX;
...
void setup() {
    nextSampleTime = millis() + PERIOD;
}
void loop() {
    if(nextSampleTime - millis() <= 0){
        nextSampleTime += PERIOD;
        // Read from sensors
        ...
        // Showing only for accX to demonstrate concept
        if (averageWindowIndex == 0) {
            // Values will be both minimum and maximum at first iteration
            // First value will be added or removed from min max
            // Compensate by subtracting
            accXAverageWindowSum = -accX;
            // Discard the maximum and minimum value before taking average to
            ↪  reduce possibility of outliers
            // Initialize to maximum and minimum possible values
            accXWindowMax = accX;
            accXWindowMin = accX;
        }
        else {
            // AccX
            if (accX > accXWindowMax) {
              accXAverageWindowSum += accXWindowMax;
              accXWindowMax = accX;
            } else if (accX < accXWindowMin) {
              accXAverageWindowSum += accXWindowMin;
              accXWindowMin = accX;
            } else {
              accXAverageWindowSum += accX;
            }
        }
        averageWindowIndex += 1;
        if (averageWindowSize <= averageWindowIndex) {
            // Write filtered averages to buffer
            // -2 is used because max and min is removed
            buffer[bufferIndex] =
             ↪  round((accXAverageWindowSum/(averageWindowSize-2))*100);
            // Do same calculation for other values
             ...
            // Send values if buffer is full
            ...
            // Reset index
            averageWindowIndex = 0;
        }
    }
}
```

The filter was chosen as it was simple to implement, noise resistant, had a small computational cost, and only need to consider a small amount of samples.

It would have been possible to have the observer with orientation calculation downhole. Implementing and tuning observers were easier on surface using block diagrams. Sending calculated and filtered orientation values from downhole could have reduce the required transfer rate to surface multiple orders of magnitude.

E Directional Drilling MPC Jacobians

The computational cost of MPC optimization can be greatly reduced by providing analytic Jacobian matrices for the states and input. The Jacobian matrices are defined as.

$$A = \begin{bmatrix} \frac{\partial\dot\phi}{\partial\phi} & \frac{\partial\dot\phi}{\partial\theta} & \frac{\partial\dot\phi}{\partial\psi} & \frac{\partial\dot\phi}{\partial x} & \frac{\partial\dot\phi}{\partial y} & \frac{\partial\dot\phi}{\partial z} \\ \frac{\partial\dot\theta}{\partial\phi} & \frac{\partial\dot\theta}{\partial\theta} & \frac{\partial\dot\theta}{\partial\psi} & \frac{\partial\dot\theta}{\partial x} & \frac{\partial\dot\theta}{\partial y} & \frac{\partial\dot\theta}{\partial z} \\ \frac{\partial\dot\psi}{\partial\phi} & \frac{\partial\dot\psi}{\partial\theta} & \frac{\partial\dot\psi}{\partial\psi} & \frac{\partial\dot\psi}{\partial x} & \frac{\partial\dot\psi}{\partial y} & \frac{\partial\dot\psi}{\partial z} \\ \frac{\partial\dot x}{\partial\phi} & \frac{\partial\dot x}{\partial\theta} & \frac{\partial\dot x}{\partial\psi} & \frac{\partial\dot x}{\partial x} & \frac{\partial\dot x}{\partial y} & \frac{\partial\dot x}{\partial z} \\ \frac{\partial\dot y}{\partial\phi} & \frac{\partial\dot y}{\partial\theta} & \frac{\partial\dot y}{\partial\psi} & \frac{\partial\dot y}{\partial x} & \frac{\partial\dot y}{\partial y} & \frac{\partial\dot y}{\partial z} \\ \frac{\partial\dot z}{\partial\phi} & \frac{\partial\dot z}{\partial\theta} & \frac{\partial\dot z}{\partial\psi} & \frac{\partial\dot z}{\partial x} & \frac{\partial\dot z}{\partial y} & \frac{\partial\dot z}{\partial z} \end{bmatrix} \tag{98a}$$

$$B = \begin{bmatrix} \frac{\partial\dot\phi}{\partial\dot\psi_I} \\ \frac{\partial\dot\theta}{\partial\dot\psi_I} \\ \frac{\partial\dot\psi}{\partial\dot\psi_I} \\ \frac{\partial\dot x}{\partial\dot\psi_I} \\ \frac{\partial\dot y}{\partial\dot\psi_I} \\ \frac{\partial\dot z}{\partial\dot\psi_I} \end{bmatrix} \tag{98b}$$

The partials of the state Jacobian matrix are shown in Equation 98a. Zero value equations are excluded.

$$\frac{\partial\dot\phi}{\partial\phi} = \frac{DLS_{nom}\cdot\exp{(-\frac{\dot\psi_I^2}{q})}\cdot v_{nom}\cdot\cos\phi\cdot\sin\theta - \dot\psi_I\cdot\sin\phi\cdot\sin\theta}{\cos\theta} \tag{99a}$$

$$\frac{\partial\dot\phi}{\partial\theta} = \dot\psi_I\cdot\cos\phi + DLS_{nom}\cdot\exp{(-\frac{\dot\psi_I^2}{q})}\cdot v_{nom}\cdot\sin\phi + \sin\theta^2\cdot(\frac{\dot\psi_I\cdot\cos\phi + DLS_{nom}\cdot\exp{(-\frac{\dot\psi_I^2}{q})}\cdot v_{nom}\cdot\sin\phi}{\cos\theta^2}) \tag{99b}$$

$$\frac{\partial\dot\theta}{\partial\phi} = -\dot\psi_I\cdot\cos\phi - DLS_{nom}\cdot\exp{(-\frac{\dot\psi_I^2}{q})}\cdot v_{nom}\cdot\sin\phi \tag{99c}$$

$$\frac{\partial\dot\psi}{\partial\phi} = \frac{DLS_{nom}\cdot\exp{(-\frac{\dot\psi_I^2}{q})}\cdot v_{nom}\cos\phi - \dot\psi_I\cdot\sin\phi}{\cos\theta} \tag{99d}$$

$$\frac{\partial\dot\psi}{\partial\theta} = \frac{\dot\psi_I\cdot\cos\phi\cdot\sin\theta + DLS_{nom}\cdot\exp{(-\frac{\dot\psi_I^2}{q})}\cdot v_{nom}\cdot\sin\phi\cdot\sin\theta}{\cos\theta^2} \tag{99e}$$

$$\frac{\partial\dot x}{\partial\phi} = v_{nom}\cdot(\cos\phi\cdot\sin\psi - \cos\psi\cdot\sin\phi\cdot\sin\theta) \tag{99f}$$

$$\frac{\partial\dot x}{\partial\theta} = v_{nom}\cdot\cos\phi\cdot\cos\psi\cdot\cos\theta \tag{99g}$$

$$\frac{\partial\dot x}{\partial\psi} = v_{nom}\cdot(\sin\phi\cdot\sin\psi - \cos\phi\cdot\sin\psi\cdot\sin\theta) \tag{99h}$$

$$\frac{\partial\dot y}{\partial\phi} = -v_{nom}\cdot(\cos\phi\cdot\cos\psi + \sin\phi\cdot\sin\psi\cdot\sin\theta) \tag{99i}$$

$$\frac{\partial \dot{y}}{\partial \theta} = v_{nom} \cdot \cos\phi \cdot \cos\theta \cdot \sin\psi \tag{99j}$$

$$\frac{\partial \dot{y}}{\partial \psi} = v_{nom} \cdot (\sin\phi \cdot \sin\psi + \cos\phi \cdot \cos\psi \sin\theta) \tag{99k}$$

$$\frac{\partial \dot{z}}{\partial \phi} = -v_{nom} \cdot \cos\theta \cdot \sin\phi \tag{99l}$$

$$\frac{\partial \dot{z}}{\partial \theta} = -v_{nom} \cdot \cos\phi \cdot \sin\theta \tag{99m}$$

The non-zero partial equations of Equation 98b are.

$$\frac{\partial \dot{\phi}}{\partial \dot{\psi}_I} = \frac{\cos\phi \cdot \sin\theta}{\cos\theta} - \frac{2 \cdot DLS_{nom} \cdot \exp{(-\frac{\dot{\psi}_I^2}{q})} \cdot v_{nom} \cdot \dot{\psi}_I \cdot \sin\phi \cdot \sin\theta}{q \cdot \cos\theta} \tag{100a}$$

$$\frac{\partial \dot{\theta}}{\partial \dot{\psi}_I} = -\sin\phi - \frac{2 \cdot DLS_{nom} \cdot \exp{(-\frac{\dot{\psi}_I^2}{q})} \cdot v_{nom} \cdot \dot{\psi}_I \cdot \cos\phi}{q} \tag{100b}$$

$$\frac{\partial \dot{\psi}}{\partial \dot{\psi}_I} = \frac{\cos\phi}{\cos\theta} - \frac{2 \cdot DLS_{nom} \cdot \exp{(-\frac{\dot{\psi}_I^2}{q})} \cdot v_{nom} \cdot \dot{\psi}_I \cdot \sin\phi}{q \cdot \cos\theta} \tag{100c}$$

Jacobian matrices can be calculated using the Symbolic Math Toolbox in MATLAB[53] from symbolic model equations.

F Drillbotics Competition Guidelines

Society of Petroleum Engineers
Drilling Systems Automation
Technical Section (DSATS)
International University Competition
2020-2021

Drillbotics® Guidelines

Revised 30 September 2020

1. Introduction

This year marks the seventh competition for the title of Drillbotics® champion and a chance for students to learn about the drilling process from industry experts and for winning team(s) to travel and present a paper at the next SPE/IADC Drilling Conference and at an event organized by DSATS. The past years involved undergraduates, masters and doctoral students from a variety of disciplines who built innovative drilling machines and downhole tools while developing a deeper understanding of automating the drilling process. The university teams freely share lessons learned, which more rapidly advances the science of drilling automation. Everyone involved claims to have had a lot of fun while learning things that are not in the textbooks or published papers. Students also participated in related events at conferences, workshop meetings and networking with industry leaders in drilling automation. This year's contest promises to be just as challenging and hopefully as much fun.

This year's competition will be to create a virtual rig, including drill string/BHA and wellbore interaction, and to demonstrate the model using a control model developed by each team. Due to the COVID-19 pandemic, the Drillbotics challenge committee decided to focus this years' competition on a virtual rig in an effort to facilitate competition success while teams work together remotely. If school policies allow a team to design and build a physical rig and there are a sufficient number of teams capable of building a physical rig, the Committee will allow these teams to compete as a second group as mentioned below and referred to as Group B. The competition guidelines for the Group B competition is listed in Appendix C.

How did the competition first come about? (Florence et al., 2000). The origins began in 2008 when several SPE members established the Drilling Systems Automation Technical Section (DSATS) to help accelerate the uptake of automation in the drilling industry. DSATS' goal was to link the surface machines with downhole machines, tools, and measurements in drilling systems automation (DSA), thereby improving drilling safety and efficiency. Later, at an SPE Forum in Paris, the idea of a student competition began to take shape; a DSATS sub-committee was formed to develop the competition format and guidelines further. Several universities were polled to find out the ability of academic institutions to create and manage multi-

Version	Date	Section	Description
2021.01		All	Updated all sections to reflect virtual rig model+controls. Physical rig guideline is moved to appendix.
2021-02	23 Sept 2020	4.5	Removed restriction
2021-03	30 Sept 2020	Appx Objectives and 1.6 and All	Clarification Fix broken links and update Committee members

disciplinary teams. The Drillbotics committee began small in 2014-2015 to see if the format could succeed. With fine tuning, we continue along those lines as we start the 2021 process.

Competition Overview:

- The challenge requires teams to develop a full-scale drilling system model, including its corresponding control scheme, to virtually drill a directional well following a given trajectory.
- The teams will design a control system that will virtually control the full-scale drilling system model to test and demonstrate the automated system. The teams should incorporate virtual downhole sensors, in addition to surface sensors in their automation and controls scheme.

The DSATS technical section believes that this challenge benefits students in several ways. Petroleum, mechanical, electrical, and control engineers gain hands-on experience in each person's area of expertise that forms a solid foundation for post-graduate careers. They also develop experience working in multi-disciplinary teams, which is essential in today's technology-driven industries. Winning teams must possess a variety of skills. The mechanical and electrical engineers need to build a stable, reliable, and functional drilling rig. Control engineers need to architect a system for real-time control, including a selection of sensors, data handling, and fast-acting control algorithms. The petroleum engineers need an understanding of drilling dysfunctions and mitigation techniques. Everyone must work collectively to establish functional system requirements, often fully understood by each team member to accurately model the drilling issues and create an integrated package working seamlessly together.

The oil and gas industry today seeks lower costs through efficiency and innovation. Many student competitors may discover innovative tools and control processes that will assist drillers in speeding the time to drill and complete a well. This includes more than a faster ROP, such as problem avoidance for dysfunctions like excessive vibrations, stuck pipe, and wellbore stability issues. Student teams built new downhole tools using 3D printing techniques of designs that would be difficult, if not impossible to machine. They used creative hoisting and lowering systems. Teams modeled drilling performance in particular formations and adjusted the drilling parameters accordingly for changing downhole conditions. While they have a lot to learn yet about our business, we have a lot to learn about their fresh approach to today's problems. Good Luck!

Drillbotics® Committee

Challenge Team

Mike Attrell
Fred Florence
Salem H. Al-Gharbi
Jayesh Jain
Enrique Z Losoya
Bader Al-Otaibi
Reed Spencer
Shashi Talya

Judges

DSATS

Shashi Talya (Chair)	Duane Cuku	Jayesh Jain	Scott Petrie
Fred Florence (Co-Chair)	Dmitriy Dashevskiy	Mathew Keller	Tony Pink
Mike Attrell	Mohamed Ali Ibrahim Hassan	Enrique Z Losoya	Dimitrios Pirovolou
Vimlesh Bavadiya	Jana Hochard	Alex Ngan	Ritthy Son
Eric Cayeux	Oliver Hoehn	Nii Ahele Nunoo	Victor Soriano

DUPTS

Salem H. Al-Gharbi (Chair)
Bader Al-Otaibi

Contents

Objectives for the 2021 Competition

1.1. During the school year beginning in the fall of 2020, a team of students will organize themselves to solve a drilling-related problem outlined in Section 3 below. The team should preferably be a multi-disciplinary team that will bring unique skills to the group to allow them to design and construct hardware and software to demonstrate that they understand the underlying physics, the drilling issues and the usual means to mitigate the issues. We cannot stress enough the need to involve students with different technical training and backgrounds. They will need to develop skills to understand drilling dysfunctions and mitigation strategies, but they must also have the mechanical engineering and controls capabilities to model, design the rig/drilling package and develop the controls system. In past years, some entrants have not adequately considered the control network and algorithms needed for autonomous drilling. They have often misunderstood the need for calibrated sensors and fast, accurate data handling. All of this and more is needed to build and operate a complete automated drilling system.

1.2. The students could produce novel ideas leading to new drilling models, improved drilling machines and sensors, and the ability to integrate the data, models and machines that will hopefully create new, more efficient ways to drill wells in the future. Any such innovation will belong to the students and their university in accordance with the university's written policies. DSATS and SPE waive any claims to students' intellectual property.

1.3. The students, working as a multi-disciplinary team, will gain hands-on experience that will be directly applicable to a career in the upstream drilling industry.

2. Background

2.1. What is DSATS?

2.1.1. DSATS is a technical section of the Society of Petroleum Engineers (SPE) organized to promote the adoption of automation techniques using surface and downhole machines and instrumentation to improve the safety and efficiency of the drilling process. More information is available about DSATS at the DSATS homepage (http://connect.spe.org/DSATS/Home/).

2.1.2. The Drillbotics website at www.Drillbotics.com includes official updates to the competition guidelines and schedule, as well as FAQs, photos, and previous entrants' submittals and reports. Any updates to the guidelines posted on the Drillbotics website via blog entries from the Committee are considered to be an official revision to these Guidelines. Questions and suggestions can be posted there, or teams can email the sub-committee at 2021@Drillbotics.com.

2.2. Why an international competition?

2.2.1. DSATS, as part of the SPE, is a group of volunteers from many nations, connected by their belief that drilling automation will have a long-term, positive influence on the drilling industry. This diversity helped to shape the direction of the organization. The group feels that the industry needs to attract young professionals from all cultures and disciplines to advance drilling practices in all areas of the world. The winners of the Group A competition will receive a grant for economy class transportation and accommodations to attend the next SPE Drilling Conference and will present an SPE paper that will be added to the SPE archives of One Petro[1]. Winners of Group B will publicly receive recognition of their achievement and have the opportunity to publish an SPE paper that will be added to the SPE archives of One Petro. DSATS believes recognition at one of the industry's leading technical conferences will help encourage student participation. Also, the practical experience with drilling automation systems increases the students' visibility to the companies that are leading automation activities.

3. *Group A Competition Guidelines*

3.1. Challenge overview for the 2020-2021 competition: The Group A challenge requires teams to develop a drilling system model that represents a full-scale system and corresponding control scheme to virtually drill a directional well to a given trajectory. The Group A challenge does not involve building a rig or drilling system. The teams will design automation and control but will develop a virtual drilling system (i.e. computer models) to test and demonstrate the controls. Therefore, in addition to the guidelines specified below (Group A Competition Guidelines) in general, applicable guidelines from Appendix C (Physical Rig Considerations), should also be referred. While the teams will have to meet minimum competition requirements, any "above and beyond" work along the main theme will be rewarded additional points to encourage creativity and innovation.

3.2. Design: Since the challenge does not require rig construction, the scope of the design portion is limited. The teams are not expected to carry out detailed mechanical design of the rig but are expected to perform basic calculations for a realistic system. The scope includes selecting essential elements such as drive mechanism, drill pipe, BHA, surface systems for application of WOB and RPM, and other required components for the virtual system. Please refer to Appendix C for considerations for physical rig design.

[1] Publication is subject to the SPE program committee's acceptance of the abstract/paper. If the abstract is not accepted, DSATS will solicit other SPE events try to get the paper into OnePetro.

3.3. Modeling:

3.3.1. Rig model: The rig model will consist of a drum controlling the drawworks, a top drive controlling the torque and RPM. The RPM, Torque, and Hookload are measurements taken at the rig model and will be inputs into the Control System.

3.3.2. The downhole drilling system model should predict bit trajectory for given WOB, RPM, drive mechanism parameters (e.g. steering force, AKO angle), and rock strength – as a function of measured depth. While the teams are empowered to decide on the complexity of the simulation model, the minimum requirements are stated below.

3.3.3. Bit model: The bit model can be as simple as the equivalent model of Pessier et al. (1992) with appropriate framework for steerability such as bit anisotropy and bit tilt such as Menand et al. (2012). Effect of key parameters such as gage length, drilling efficiency (MSE-DOC relationship) should be included. Inclusion of bit wear effects is not mandatory. For the purposes of this challenge, the bit model will be provided.

3.3.4. Rock/wellbore: The rock model should be defined by rock type, UCS, and confining pressure. At each simulation step increment, the bit drills and extends the wellbore. While calculation of explicit contact forces with the wellbore are not mandatory, the build rate will still change due to newly formed wellbore geometry and changing rock strength. This phenomenon must be taken into effect accurately. Teams can assume a 2D wellbore and thus develop a 2D drilling propagation model.

3.3.5. BHA: The BHA must be modeled so that contact force at the bit and bit tilt are computed to be used in the steering model. Generally speaking, 100 ft. of the BHA within the wellbore needs to be modeled in order for correct bit side force and bit tilt computations. The resulting behavior of drive mechanism should be modeled. The BHA should also (virtually) measure certain parameters (such as inclination, RPM, vibration etc.) and return to the surface or the control system. The bit-to-sensor distance as well as measurement frequency (i.e. intermittent vs continuous survey) should be a configurable parameters in the design.

3.3.6. Steering Model: The steering model takes inputs from the Bit Model and BHA Model to predict trajectory. A Control System will also interface with the Steering Model and update parameters (such as pad force, AKO orientation, WOB, RPM, etc) accordingly.

3.3.7. Drillstring: The Drillstring may be represented by one or more models. These models will have to do the following:

1. Calculate torque and drag for a 3D survey, with hook load, mud weight, drill string/BHA dimensions, sheave friction and variable friction factors along the wellbore as inputs. Using this data, the model will be able to predict downhole WOB and available torque at the bit, which will be used as input to the Bit Models.
2. The Drillstring Model(s) must also calculate buckling conditions. Drilling ahead in simulation will not be allowed if the Drillstring is buckling at any point along the Drillstring.
3. The Drillstring Model(s) must be able to simulate torsional oscillations (slow ones, like stick slip). It must be made up of multiple torsional spring elements and have friction damping from wellbore contact. Bit behavior in different rocks and at different WOB/RPM settings will cause stick slip, and the Control System for the top drive must be able to counter act stick slip automatically when it appears.
4. Please do not attempt to model lateral vibrations of the Drillstring or BHA.

3.3.8. The directional bit behavior modeling assumptions should be clearly stated. The implementation (or sub-models) should be verified against published data such as Menand et al. (2012).

3.4. Controls: The control system may include the following elements

*3.4.1.*Drilling Optimization: Optimize set point commands for drilling parameters such as WOB, RPM, etc. such that drilling performance and steering are optimized (according to each team's definition of "optimized performance"). Such real-time optimization should be done automatically.

*3.4.2.*Trajectory Control: Steer the well according to the given well plan. The objective is both to minimize trajectory error and wellbore tortuosity. Virtual surveys should be acquired and be used as feedback for the steering control logic. Be prepared to model a push-the-bit RSS and a bent motor AKO. The steering model should include considerations for how often the survey is taken and how far from the bit the sensors are placed (e.g. projecting from the survey depth to the bit, and the control system using survey information to decide steering parameters).

*3.4.3.*Rig Display: Real-time display of the drilling parameters and wellbore positioning during the final testing is mandatory. End of well report immediately after the competition is mandatory.

*3.4.4.*Set Point Control: Although set point control, i.e. automatic control of drilling parameters as per optimal set points, is an integral element of the drilling systems, this competition does not make it mandatory to reduce complexity. It can be assumed that the surface parameters such as WOB and RPM reach the BHA, making quasi-static modeling sufficient. However, the teams are encouraged to go "above and beyond" and demonstrate set point control independent of trajectory drilling. For example, the WOB and RPM control could be implemented for the virtual drill rig with a suitable mechanism for applying WOB (e.g. dead weight and drawworks), RPM (e.g. top drive), etc. Characteristics for each sub-system could be assumed realistically (e.g. top drive motor characteristics with RPM-torque relationship). Other examples include slide/rotate mode control.

3.5. Coding:

3.5.1. The entire code should be written with a modular design with functions/subroutines for each sub-system. The drilling system model should be a separate application that interacts with the control system. Appropriate interfaces (APIs) should be developed for interoperability and deployment.

3.5.2. Teams are encouraged to share their code to promote the learning spirit. Such sharing can occur during or after the final presentations, or after securing any IP protection, at the discretion of the teams. However, release of codes is not mandatory and will not count towards the final score.

3.6. Evaluation:

3.6.1. The drilling plan will be presented to the teams on the day of competition. The rock properties will be provided as a function of true vertical depth or measured depth. The teams are given maximum of three hours to virtually drill the well. Students are allowed to debug/modify the code and use multiple attempts within the allotted time.

3.6.2. An RSS or AKO motor BHA will be specified on the day of the competition. Thus, the model should be capable of simulating both steering systems.

3.6.3. Drillbotics may provide data to calibrate sub-models such as the bit model. Additional details will be released during Phase II.

3.6.4. While sharing of code is not mandatory, the presentations should include the details of the control schemes. Organizers can be contacted in case of any confidentiality requirements.

3.6.5. Teams will be evaluated on a per model basis. Points will be given for having each model or control system present and functioning in a realistic manner. A team that predicts the trajectory the best but is missing a model of the rig will earn fewer points than a team that has all the models and control systems from bit to rig. The purpose is to model the entire system and have the sub-models behave realistically.

3.6.6. The set point control is not a mandatory item for the competition. Any demonstration of such capability will attract extra points in "above and beyond" category.

3.7. Deliverables:

3.7.1. Phase I: A detailed report containing detailed literature review, model assumptions, overall plan of the virtual system, including the system architecture, different layers (such as data layer, control layer etc.), mathematical framework for modeling and control schemes, a plan for implementation, and relevant details. It is preferable to include special section for the API, if other system need to interact with your system. Preliminary results from the virtual drilling rig model should be included, along with a discussion on the results.

3.7.2. Phase II: A deployable application that drills a directional well to a given trajectory plan using autonomous control of a virtual drilling system.

3.8. Useful resources:

3.8.1. Florence, F., Losoya, E., Drillbotics with Fred Florence and Enrique Losoya (2020, August 18), SPE Podcast, Link.

3.8.2. Pessier, R. C., & Fear, M. J. (1992, January 1). Quantifying Common Drilling Problems With Mechanical Specific Energy and a Bit-Specific Coefficient of Sliding Friction. Society of Petroleum Engineers. doi:10.2118/24584-MS

3.8.3. Menand, S., Simon, C., Gerbaud, L., Ben Hamida, M., Denoix, H. J., Cuillier, B., Sinardet, H. (2012, January 1). PDC Bit Steerability Modeling and Testing for Push-the-bit and Point-the-bit RSS. Society of Petroleum Engineers. doi:10.2118/151283-MS

3.8.4. Pehlivantürk, C., D'Angelo, J., Cao, D., Chen, D., Ashok, P., & Van Oort, E. (2019, March 4). Slide Drilling Guidance System for Directional Drilling Path Optimization. Society of Petroleum Engineers. doi:10.2118/194096-MS

3.8.5. Marck, J., Detournay, E., Perturbation to Borehole Trajectory across an Interface, ARMA-2014-7479, 48th US Rock Mechanics/Geomechanics Symposium, Minneapolis, Minnesota, June 1-4, 2014.

3.8.6. Zalluhoglu, U., Marck, J., Gharib, H., & Zhao Y. (2019) Borehole Propagation with Undergaged Stabilizers: Theory and Validation. ASME Journal of Dynamic Systems, Measurement and Control, vol. 141, no. 5: 051013. doi: 10.1115/1.4042380

3.8.7. Perneder, L., Marck, J. and Detournay, E., 2017. A model of planar borehole propagation. SIAM Journal on Applied Mathematics, 77(4), pp.1089-1114. doi: 10.1137/16M1094518

3.8.8. Zalluhoglu, U., Demirer, N., Marck, J., Gharib, H., & Darbe, R. (2019) Steering advisory system for rotary steerable systems. SPE/IADC Drilling Conference and Exhibition, 5-7 March, The Hague, The Netherlands. SPE-194090-MS, doi: 10.2118/194090-MS

3.8.9. Zalluhoglu, U., Gharib, H., Marck, J., Demirer, N., & Darbe, R. (2019) Steering advisory system for mud motors. SPE/IADC Drilling Conference and Exhibition, 5-7 March, The Hague, The Netherlands. SPE-194077-MS. doi: 10.2118/194077-MS

3.8.10. Franklin, G. F., Powell, J. D., Emami-Naeini, A., & Powell, J. D. (1994). Feedback control of dynamic systems, 3rd Edition, Reading, MA: Addison-Wesley.

3.8.11. Ogata, K. (2003). System dynamics, 4th Edition, Upper Saddle River, NJ: Prentice Hall.

3.8.12. Ogata, K. (2009). Modern control engineering, 5th Edition, Upper Saddle River, NJ: Prentice Hall.

3.8.13. Li, Y., Ang, K. H., & Chong, G. C. (2006). PID control system analysis and design. IEEE Control Systems Magazine, 26(1), 32-41.

3.8.14. Rawlings, J. B. (2000). Tutorial overview of model predictive control. IEEE control systems magazine, 20(3), 38-52.

3.8.15. Webinar: Machine Learning and Physics-based Solutions for Drilling Automation by SPE Distinguished Lecturer Prof. John Hedengren, Brigham Young University, YouTube Video.

3.8.16. Webinar: Drilling Automation and Downhole Monitoring with Physics-based Models. Link.

3.8.17. Video and Webinar Series: Understanding Control Systems by Mathworks. Link.

4. *Team Members*

4.1. DSATS envisions that the students would be at least senior undergraduate or Masters level, well versed in the disciplines needed for such a project. The maximum number of students per team is five (5) and the minimum shall be three (3). Any team that loses team members during the project can recruit a replacement.

4.2. At least one member of the team must be a Petroleum Engineering candidate with sufficient coursework completed to understand the physics relating to the drilling problems and the normal industry practices used to mitigate the problem.

4.3. Students with a background in mining, applied mathematics, mechanical and electrical engineering, as well as controls, mechatronics and automation or software development, are the most likely candidates, but students with any applicable background is encouraged.

4.4. A multi-disciplinary team simulates the working environment in the drilling industry today, as most products and services are produced with the cooperation of technical personnel from differing backgrounds and cultures.

4.5. Auniversity may sponsor more than one team in a group and may enter teams in one or both groups.

4.6. Students shall register their team not later than 1 November using the registration form on the Drillbotics website. Any changes to the team members or university supervisor over the course of the competition should be reported in the monthly reports.

5. *Expenditures*

5.1. Teams selected to advance to the second phase must limit the cost of the physical or virtual rig and materials to US$ 10,000 or its equivalent in other currencies. The students shall find a source of funding and report the source in the Phase I proposal. All funding and procurement should comply with university policy. These funds are intended to cover the majority of expenses for hardware, software and labor to construct and operate the team's equipment. DSATS shall not be liable for any expenditure other than DSATS provided material and specified travel expenses.

5.2. DSATS will assist when possible to obtain free PLCs or similar control devices from suppliers affiliated with the DSATS organization. Such "in-kind" donations shall not be included in the team's project costs.

5.3. Students and universities may use other "in-kind" contributions which will not be included in the team's project costs. Such contributions may include modeling software, laboratory equipment and supplies, and similar paraphernalia usually associated with university laboratory projects.

5.4. Any team spending more than US$ 10,000, or its equivalent in other currencies, may be penalized for running over budget.

5.5. DSATS reserves the right to audit the team's and university's expenditures on this project.

5.6. Any devices built for the project will become the property of the university and can be used in future research and competitions. Any maintenance or operating costs incurred after the competition will not be paid by DSATS.

6. *Other Considerations*

6.1. University coursework and credit: Each university will decide whether or not this project qualifies as a credit(s) towards any degree program.

7. *Project Timeline*

Phase I - Design:	Fall 2020
Submit monthly reports	On or before the final day of each month
Submit final design to DSATS	31 Dec 2020, midnight UTC
Submit an abstract to DSATS*	31 Dec 2020, midnight UTC

*DSATS will submit an abstract to the SPE that will include excerpts from the student abstracts by the conference paper-submittal deadline, typically in mid-summer, for consideration of a paper by the conference program committee.

Phase II – Model enhancement/testing and controls development	Spring 2021
DSATS to announce finalists	On or about 31 Jan 2021
Model & controls development/Construction	Spring 2021
Monthly reports	On or before the final day of each month
Final demonstration	The final demonstration will typically occur in late May or early June. Additional details on the logistics for the final demonstration will be shared in early 2021.

8. *Project report*

1.1.1. Starting in the fall term, the student team shall submit to DSATS a short monthly project report that is no more than one page in length (additional pages will be ignored) due on or before the last day of each month. Send it via email to 2021@Drilbotics.com. The monthly report should include:

1.1.2. Phase I

- Key project activities over the past month.
- Literature survey, rig modeling considerations, trade-offs, critical decision points etc.
- Cost updates
- Significant new learning, if any

1.1.3. Phase II

- Model enhancements, controls development updates.
- Preliminary results of exercising the drilling model and controls
- Other items of interest

1.1.4. Report content

1.1.4.1. To teach students that their work involves economic trade-offs, the monthly report should include at a minimum a summary estimate of team member labor hours for each step in the project: modeling, controls, testing etc. and a cost summary for software related expenditures. Also include labor for non-students that affect the cost of the project. Labor rates are not considered, as to eliminate international currency effects. Labor is not considered in the cost limits of item 6.1, but should be discussed in the report and paper.

1.1.4.2. Design reports must contain the following tables and place them in their design report appendices:

A. Student Biographies
   - Name
   - Previous degree attained – major
   - Current degree and expected graduation date (month/year)
   - Main area of contribution to the project
   - Other information as deemed appropriate by the team

B. Summary of Calculations, model details, controls algorithm etc.

9. Evaluation Criteria

9.1. DSATS will select an evaluation committee from its membership

9.2. Criteria/Weighting for Group A (see table below):

Criteria	Metrics	Weight
Drilling system model	Does steering model consider steering method, geometry (e.g. projection-to-bit algorithm), bit side force/tilt, new wellbore, etc.? Are string elasticity, wellbore friction modeled?	30
Control scheme	Does trajectory control algorithm use realistic constraints? Does it use realistic virtual-measurements? Does it consider surveying uncertainties and noise? Does the model utilize a re-planning to target process based on as-drilled surveys? Is basic drilling optimization algorithm implemented? Are rig controls simulated? (e.g. slide vs rotate)	30
The Virtual Drilling App	Features, modularity, and robustness of the app, real-time display, end of well report	20
Performance	Demonstration of the app and the degree to which drilling objectives are met	20
Bonus	Considerations above and beyond the minimum requirements that demonstrate thoroughness and creativity	10
	Maximum achievable score out of 100	110

9.3. Phase II Criteria/Weighting for Group B (see table below):

Criteria	Parameter	Weighting
Phase I:		
a. Safety	Safety: construction and operation	10
b. Mobility of rig	Rig up, move, rig down	5
c. Design considerations and lessons learned		10
d. Mechanical design and functionality, versatility		25
e. Simulation/Model/Algorithm		25
f. Control scheme	Data, controls, response times	25
	Total	100%
Phase II:		
a. Creative Ability	Analysis, concepts, development	10
B .Engineering Skills	Problem/Goal, design criteria, feasibility	10
c. Construction Quality		10
d. Cost Control		10
e. Performance		30
Various parameters such as:	ROP, MSE, Landing Bit, Inclination, and other	
Are these used within the control algorithms		
Accuracy of drilled wellbore trajectory (see Appendix "A" for details)	Proximity of drilled wellbore to required target X/Y coordinates and vertical depths	
f. Quality of wellbore	Tested using the Go-No-Go flexible 'Casing'	10
	Verticality, tortuosity, caliper, other	
g. Data	Data handling, data visualization, data comparison to judges' wellbore logs, and other	20
h. Downhole Sensor Data Used in Control Algorithm	Pass/Fail	Pass/Fail
	Total	100%
Intangibles	Additional score may be added or subtracted by the judges at their discretion	

9.4. *Group A Prizes*

9.4.1. The winning team of Group A will be sponsored by DSATS to attend the next SPE/IADC Drilling Conference to present a paper that explains their project in detail.

9.4.1.1. The program committee of the Drilling Conference awarded the Drillbotics subcommittee a permanent slot in one of the drilling sessions at the conference. As per SPE's customary procedures, the paper will be archived in OnePetro. In addition, SPE has agreed to furnish a booth in the exhibition area during the conference where the team can erect their rig and describe its operation to the conference attendees. This is an excellent opportunity for students to network with the industry.

9.4.2. Upon submittal to DSATS of a valid expense statement (typically a spreadsheet supported by written receipts) of covered expenses will be reimbursed by the treasurer of DSATS for the following:

9.4.2.1. Round trip economy airfare for the team and one university sponsor/supervisor to the gateway city of the next SPE/IADC Drilling Conference. Entrants should use the SPE approved carrier where possible to minimize cost. Airfares that exceed the SPE rate must be pre-approved by the committee or the reimbursement will be limited to the SPE rate. Information of reduced fare flights is available on the conference website. Please note that reservations must be made before the SPE published deadline. The departure point will be a city near the university, the student's home, or current place of work, subject to review by the Committee. Alternately, a mileage reimbursement will be made in lieu of airfare should the entrants decide to drive rather than fly to the conference. The reimbursement is based on current allowable mileage rates authorized by the US Internal Revenue Service.

9.4.3. One rental car/van at the gateway city for those teams that fly to the conference.

9.4.4. Lodging related to one hotel room per team member will be reimbursed at a rate not to exceed the SPE rate. Note that the room reservations are limited, so entrants must book their rooms early. Room and taxes for the night before the DSATS symposium, the night of the symposium and for the nights of the conference are covered. Charges for the room on the last day of the conference need to be pre-approved by the Committee as most conference attendees depart on the last day of the conference unless there are unusual circumstances.

9.4.5. A per diem will be pre-approved by the Committee each year, which will vary with the cost of living in the gateway city. The per diem is intended to cover average meals (breakfast, lunch and dinner) and incidentals.

9.4.6. ATCE registration will be reimbursed. Students should register for the conference at the student rate. Early registration is appreciated.

9.4.7. Individual award certificates will be presented to all participants upon request, with special certificates given to all finalists.

9.4.8. DSATS may provide additional awards, at its sole discretion.

9.4.9. The evaluation and all decisions on any matter in the competition by the DSATS judges and DSATS board are final.

9.5. Group B Prizes

9.5.1. The winning team of Group B may submit a SPE whitepaper that explains their project in detail. If the quality of the abstract is approved by the SPE Conference Program Committee, as per SPE's customary procedures, the paper will be archived in OnePetro

9.6. Other prize information

9.6.1. Individual award certificates will be presented to all participants upon request, with special certificates given to all finalists.

9.6.2. DSATS may provide additional awards, at its sole discretion.

9.6.3. The evaluation and all decisions on any matter in the competition by the DSATS judges and DSATS board are final.

10. Final report and paper

10.1. The finalists shall prepare a project report that addresses the items below. We suggest you use the format of most SPE papers. For reference, please see http://spe.org/authors/resources/

10.2. The winning team of Group A shall update the report as needed to comply with SPE paper submittal guidelines to write a technical paper for publication by the SPE at its Annual Drilling Conference. SPE typically requires that the manuscript is due in the fall following the Phase II test. While the Drillbotics committee will make every effort to have the paper presented during the Drilling Conference, the SPE Program Committee has authority over which papers will be accepted by the conference. If the paper is not accepted by the conference, the Drillbotics committee will endeavor to have it presented at the DSATS Symposium and will use its contacts to have the paper published via other related SPE conferences.

10.3. The report, paper and all communications with DSATS shall be in the English language. The presentation will be made by at least one member of the student team.

10.4. The timing for submittal of the abstract and paper will be the published deadlines per the call for papers and conference guidelines as posted on the SPE's website (www.spe.org).

10.5. The abstract must generate sufficient interest with the SPE review committees to warrant publication, although DSATS will help promote acceptance where possible

10.6. The paper should address at a minimum

10.6.1. The technical details of the drill system model, assumptions and architecture. Results of the model prediction and discussion of the model results.

10.6.2. Details of the controls scheme, including block diagram and control algorithm. Challenges and trade-offs associated with use of specific control schemes.

10.6.3. Results of the final demonstration and discussion on future work/enhancements.

10.6.4. Recommendations for improvements by DSATS of the competition guidelines, scheduling and provided material.

10.6.5. Areas of learning gained through the competition not covered in the university course material.

10.6.6. A brief bio or CV of the team members and their sponsoring faculty.

11. Terms and conditions

11.1. In no event will SPE, including its directors, officers, employees and agents, as well as DSATS members and officers, and sponsors of the competition, be liable for any damages whatsoever, including without limitation, direct, indirect, special, incidental, consequential, lost profits, or punitive, whether based on contract, tort or any other legal theory, even if SPE or DSATS has been advised of the possibility of such damages.

11.2. By entering this competition,

11.2.1. Participants and Universities agree to indemnify and hold harmless SPE, its directors, officers, employees and agents, as well as DSATS members and officers, and sponsors of the competition, from all liability, injuries, loss damages, costs or expenses (including attorneys' fees) which are sustained, incurred or required arising out of participation by any parties involved in the competition.

11.2.2. Participants and Universities agree and acknowledge that participation in the competition is an agreement to all of the rules, regulations, terms and conditions in this document, including revisions and FAQs posted to the DSATS and Drillbotics websites (see section 2.1).

11.2.3. Winning teams and finalists must agree to the publication of their names, photographs and final paper on the DSATS web site.

11.3. All entries will be distributed to the Drillbotics Committee for the purpose of judging the competition. Design features will not be published until after all teams have been judged and a winner is announced. Previous years' submittals, reports, photos and similar documentation will be publicly available to foster an open exchange of information that will hopefully lead to faster learning for all participants, both new and experienced.

11.4. DSATS and the SPE cannot provide funding to sanctioned individuals and organization per current US law.

11.5. Participants must comply with all local laws applicable to this contest.

12. Marketing

12.1. Upon request, DSATS will provide a link on its website to all participating universities.

12.2. If university policy allows, various industry journals may send a reporter to witness the tests and interview students to publicize the project.

12.3. Drillbotics is now a registered trademark. According to international law, the proper reference is to use Drillbotics® instead of Drillbotics™. The trademark reference is only needed the first time Drillbotics is referenced.

12.4. Any team that wishes to use the trademark on signs, tee shirts, technical papers or for other purposes may receive a no-cost license upon request. Send the request by email to the committee at 2021@Drillbotics.com. Upon completion of the license agreement, access to the files with the logo will become available.

Appendix

A. Directional Objective Requirements

The following attached pages describe the directional objectives as well as the data/deliverables requirements. Scoring for the directional competition objective will be primarily based on how accurately the directional targets are intersected by the calculated well trajectory.

<u>Objectives</u>

- Hit one or more targets at one or more vertical depth(s) and X/Y coordinates
- For the Group B competition, the starting directional plan to hit the targets will not require wellbore inclinations in excess of 30° from vertical, 15° change in azimuth, or 10" displacement (departure from the vertical axis at well center) The max displacement/inclination/azimuth are total/accumulated from the start to the end of the well path.
- Please note: Teams should be prepared to drill any given trajectory within the specified parameters, so the coordinates will not be provided in advance of the test.

<u>Automation Requirements</u>

- Drilling mode/survey mode switching must be automated (i.e. built-in survey interval and drill string movement for on/off-bottom, slide/rotation mode switching)
- Steering requirements (e.g. toolface direction, slide length) must be calculated autonomously
  - NOTE: Steering mechanism can still require human intervention for placement and/or retrieval (e.g. whipstock) but orientation of steering mechanism must be calculated by the system and shown on the rig floor display.
- Directional surveying process must be entirely autonomous
  - Survey qualification must be done autonomously, however secondary qualification/verification/override can be made by a human
- Dogleg severity required to hit target(s), distance/direction to plan must be autonomously calculated at each survey station and shown on the rig floor display

<u>Deliverables Requirements (Magnetic surveying)</u>

- All teams are required to provide a definitive directional survey (TXT, LAS, or CSV format) meeting the following minimum requirements:
  - Header info to include:
    - Team/school name
    - Directional Survey Date
    - Well Center Coordinates (WGS84 Latitude & Longitude)
    - True Vertical Depth Reference (in depth units above block level)
    - Grid Convergence
    - Geomagnetic model used (if applicable)
    - Magnetic declination applied (Geomagnetic model or in-field referenced)
    - Total Azimuth Correction
    - Magnetic field dip reference (Geomagnetic model or in-field referenced)
    - Total magnetic field strength reference (Geomagnetic model or in-field referenced)

        - Error model associated with well trajectory (ISCWSA/OWSG error model or otherwise)
            - If non-standard error model is being used (i.e. formulas being modified and/or coefficients being changed), error model description (using standard variable/coefficient naming conventions) and justification must be included in project design
    - Minimum Curvature calculated trajectory (using appropriate survey station interval to accurately represent the drilled wellbore position)
        - Each survey station is to include the following data:
            - Measured Depth
            - Inclination
            - Azimuth (referenced to "block north")
            - True Vertical Depth
            - Northing (from well center)
            - Easting (from well center)
            - Dogleg Severity
        - Final survey station is to be an extrapolation to total depth at the bit
- All teams are required to provide plan vs. actual plots containing the following minimum requirements:
    - As-drilled trajectory and original planned trajectory shown on same TVD vs. Vertical Section plot
        - Vertical section direction to be determined by well center-to-target bearing
    - As-drilled trajectory and original planned trajectory shown on same X/Y plot
        - Grid north reference to "block north"
        - [0,0] at well center
- All teams are required to provide directional survey raw data logs containing the following minimum requirements:
    - Each log entry is to include the following data:
        - Time stamp (containing year, month, date, hour, minute, second)
        - Sensor measured depth
        - Downhole sensor value(s) recorded
            - Sensor axes values
            - Calculated survey qualifier values
        - Accepted survey indicator (if log entry is an intended survey station)
            - If secondary (i.e. human) qualification is also used, both acceptance indicators must be shown

B. Safety

The team's safety plan should consider all foreseeable hazards and methods to mitigate them. Personal protective equipment is part of a safety plan but is far from sufficient. Teams must consider risks due to handling the rock, rotating machinery, electrical shock and others. How the team communicates with each other before and during rig operations is also important. Judges will grade each team on its comprehensive safety case.

Because most of the rigs have equipment spinning at high RPMs, some form of protective cover must be included in the team's rig design. A broken coupling, a loose screw or similar item becomes a projectile that can lead to serious injury to the team members, judges or visitors. Judges may decide to deny a team from competing if their design is unsafe.

The following links are a good starting point, but is by no means a comprehensive list of links:

- OSHA Pocket Guide, Worker Safety Series: https://www.osha.gov/Publications/OSHA3252/3252.html
- OSHA Checklist for General Industry: http://www.scosha.llronline.com/pdfs/genind.pdf

C. Group B Competition Guidelines

The Group B competition applies to the automation of a physical rig. This Appendix C 1.0 (physical rig) replaces Section 3.0 (virtual rig) for those teams who wish to compete in Group B.

1.0. Physical Rig Considerations: 2020-2021 High Level Challenge and Judging Changes

1.0.1. Directional steering is a more critical part of the competition for 2021. (see 3.6) The wellbore must be started vertically and then kicked off below a specified depth to hit multiple directional targets (at varying X/Y coordinates and vertical depths). Teams score more points based on how accurately each directional target is hit (see 9.3 for scoring details)

1.0.2. Downhole sensors are mandatory, and it is also mandatory to implement their data into the control algorithm of the rig. A severe penalty will be applied to teams who do not use downhole sensors. Closed loop control of the rig based on downhole data is mandatory in this year's competition and not integrating this data set into the control algorithm is considered a "F- Failing grade" in this year's competition.

1.0.3. A homogeneous sandstone Rock Sample will be provided by Drillbotics (see A 1.4)

1.0.4. DSATS to provide a new bit. The bit will be 1.5" diameter and 2" length. Students are permitted to use their own drill bit for the 2021 competition. (see A 1.5)

1.0.5. Additional information regarding the judging of the competition is detailed in section 1.16.

1.1. Two Project Phases

Fall Semester 2020

The first phase of the project is to organize a team to design an automatic drilling machine to solve the project problem. It is not necessary to build any equipment in this phase, but it is okay to do so. Design considerations should include current industry practices and the team should evaluate the advantages and shortcomings of today's devices. The design effort may be assisted by university faculty, but the students are encouraged to introduce novel designs for consideration. The design should also include consideration for downhole sensor and the control system to automatically control the drilling process. The level of student, faculty and technical staff involvement shall be reported when submitting the design. For returning teams, the Phase I Design should include an analysis of data and learnings from previous ("offset") wells drilled.

Spring Semester 2021

During the second phase, the finalist teams selected by DSATS proceed to the construction and drilling operation will use the previous semester's design to build an automated drilling machine. As per industry practices, it is common during construction and initial operations to run into problems that require a re-design. The team may change the design as needed in order to solve the problem subject to section 3.3. Teams may use all or part of a previous year's rig.

See section 6 for detailed timeline information.

1.2. Phase I – Design Competition

Design an automated drilling machine in accordance with the rules below.

1.2.1. DSATS envisions a small (perhaps 2 meters high) drilling machine that can physically imitate the functionality of full-scale rig machinery. (Since the winning machines will be presented at the SPE conference, there may be height restrictions imposed by the conference facility, so machines that are too tall may not be allowed on the exhibit floor.) The machine will be the property of the university and can be used in future research and competitions. New and novel approaches that improve on existing industry designs are preferred. While innovative designs are welcome, they should have a practical application to drilling for oil and gas.

1.2.2. The drilling machine will use electrical power from the local grid not to exceed 25 horsepower. Lower power consumption resulting from energy efficient designs will receive additional consideration.

1.2.3. The design must provide an accurate and continuous measurement of Weight-On-Bit (WOB), inclination, azimuth, and depth; as well as other drilling parameters (see Appendix "A" for directional surveying-specific data requirements), that should be presented as a digital record across the period of the test. All depth related measurements shall use the rig floor as the datum, not the top of the rock (the offset between the rock surface and the rig floor must be adequately processed within the control algorithms). Appropriate statistical measurements should be made at frequencies and with an accuracy and appropriate frequency content for the dynamics of the drilling system both at surface and downhole. Discussion of such choices should be included in the design report.

1.2.3.1. Distinguish in all data and documentation the difference between Weight-On-Bit and Hook Load; be specific when referring to these parameters

1.2.4. The proposed design must be offered in Phase I of the project, but changes are allowed in Phase II, as long as they are reported to the Committee via students' monthly reports. A summary of all significant changes, including the reason modifications were necessary, must be included in the students' final report.

1.2.5. Design submittal by the students shall include:

1.2.5.1. Engineering drawings of the rig concept, mechanical and electrical and auxiliary systems, if any

1.2.5.2. Design notes and calculations

1.2.5.2.1. All engineering calculations shall be included in the Phase I report, even if the rig is built using previous years' designs. This ensures that the 2021 team reviewed and understood the previous design assumptions and calculations.

Calculations should include each formula considered in the design, a reference that shows the origins of the formula, why is was chosen, what engineering assumptions were made, a definition of all variables and the values used in the calculation.

Example

Buckling limit	Euler's Equation	(1) cite a reference here or in the reference section of your design report

The critical buckling load, bcr, is calculated:

$$Pbcr = \pi 2 * E * I / (K * L)^2$$

$Pbcr$:	Critical buckling load
E:	Modulus elasticity of the aluminum drill pipe
I:	Area moment of inertia
L:	Length of the column
K:	Column effective length factor (explain how you chose the appropriate k or n factor)

1.2.5.2.2. The report should include a table that summarizes ALL calculations.

Example

Calculations	*Formula*	*Reference*	*Results*
Moment of Inertia	$I = \pi / 64\ (dp4 - idp4)$	*Thin wall approx. or ID/OD calc separately or other?* *List your reference*	*0.000546*$in4$
Buckling Limit	$Pbcr = \pi 2 * E * I / (K * L)^2$	*Euler's Eq*	*18.9* kg

1.2.5.3. Control system architecture. (The response time of measurements, data aggregation and control algorithms should be estimated.)

1.2.5.4. Key features for any models and control software.

1.2.5.5. Proposed data handling and display.

1.2.5.6. Specification for sensors, signal processing and instrumentation, (verifying their accuracy, precision, frequency response and environmental stability), including the methods planned for calibration before and after the Phase II testing.

1.2.5.7. Plan for instrumentation of sensors in the BHA, as well as a method to synchronize all measurements and utilize both the surface and downhole sensors for real-time control of the drilling process.

1.2.5.8. An explanation of the implementation of the output of the BHA sensors to improve the trajectory of the wellbore, drilling efficiency and other drilling concerns.

1.2.5.9. An explanation of the algorithm used to autonomously control the drilling rig based on the output of the BHA sensors

1.2.5.10. An explanation of the principles being applied to directionally steer the wellbore and hit the required targets (see Appendix "A") with the intent to score the maximum amount of points

1.2.5.11. Cost estimate and funding plan

1.2.5.12. A design summary video used to outline the design submittal not to exceed five (5) minutes in length. Videos shall be the property of the university, but DSATS shall have the rights to use the videos on its websites and in its meetings or events.

1.2.5.13. All design, construction and operation of the project are subject to the terms and conditions of section 10.

1.2.5.14. A safety case shall be part of the Phase I design (see Appendix "B"). Include a review of potential hazards during the planned construction and operation of the rig, and for the unloading and handling of any rock samples or other heavy items. An example of a safety case will be posted on the Drillbotics.com website.

1.2.5.15. The Phase I design report should include a discussion regarding the major design features proposed (mechanical and otherwise) - are they scalable to today's working rigs? If not, what would be needed to allow implementation?

1.2.5.16. The Phase I design report should include a discussion regarding the control scheme and algorithm - How is each individual measurement used in the control code? Are they all given equal weight, and if not, what criteria is used to assign importance? What is the expected response time of the control system's key components? How will this affect equipment selection? The teams are encouraged to perform control simulations to verify the control scheme.

*1.2.6.*A committee of DSATS members (the Committee) will review the Phase I designs and select the top five (5) teams[2] who will progress to Phase II of the competition.

*1.2.7.*DSATS shall also award a certificate of recognition and publication on its website for the most innovative design. The design video will also be shown at the DSATS automation symposium at SPE conferences.

*1.2.8.*DSATS will not fund any equipment, tools, software or other material, including labor, for the construction of the rig. Student teams are encouraged to find external funding from industry participants and suppliers.

1.3. Phase II – Drilling Competition

*1.3.1.*In the spring term of 2021, qualifying teams will build the rig and use it to drill rock samples provided by DSATS. Drilling a deviated well to hit the required targets (see Appendix "A"), efficiently through the sample while controlling drilling dysfunctions is the primary technical objective of the competition. Scoring of the directional drilling component will be primarily based on the horizontal distance from the target coordinate at which each target vertical depth was intersected. The use of both surface and downhole measurements to control the drilling process in real-time is mandatory, failure to do so will result in a failing grade. To avoid disqualification due to a downhole sensor failure, redundant or immediately replaceable items should be part of the design and implementation. Time to replace a sensor will be added to the drilling time for calculation of ROP.

1.3.2. The teams are to use manual control to pre-drill a vertical pilot hole not more than 1" deep measured from the rock's top face. This hole is to be drilled using the competition drilling rig. Location of this pilot hole will be marked on each sample by the committee at the intersection of two lines drawn from opposite corners of the rock sample.

1.3.3. Teams may use glue or use a mechanical fastener to attach a bell nipple or diverter housing to the top of the rock to allow connection of a flowline for return mud flow. The maximum allowable length of the bell nipple is 8 inches. If you use a fastener, be careful not to break the rock.

[2] The number of finalists could be increased or decreased by the DSATS Board of Directors.

1.3.4. When the competition drilling begins, teams will be required to continue to drill the pilot hole vertically to the kick off point. The kick off point may be at any depth greater than 4" below the surface of the rock.

1.3.5. Navigation shall be done autonomously

1.3.5.1. Manual intervention to add and/or remove a steering mechanism (e.g. whipstock) is permitted, however the determination/calculation of the orientation setting of the mechanism is required to be autonomous and must be shown on the rig floor display during each steering mechanism manipulation activity. The time to change orientation will affect the team's ROP calculation.

1.3.6. No lateral forces are allowed to be applied above the top face of the rock.

1.3.7. No forces are allowed to be applied external to the rock that will force the drill bit in a particular direction

1.3.8. External magnetic field effects from the drilling rigs will be present on the directional sensors used to drill the wellbore. The industry has accepted practice of magnetic ranging. This may be a technique worth investigating to improve the signal to noise of magnetic measurements

1.3.9. Once drilling commences, the test will continue until the drill bit exits the rock sample, or three (3) hours, whichever comes first

1.3.10. Drilling performance will be observed and measured by Drillbotics judges invited to attend and witness the test. This could be a virtual event depending on travel restrictions. The details will be announced in early 2021.

1.3.11. DSATS will judge the competitors primarily on their ability to hit the required targets as accurately (i.e. as close to target center at the given target vertical depth) as possible (see Appendix "A" for details)

1.3.12. The final test will be scheduled late in the school year.

1.4. Rock Samples

1.4.1. DSATS will prepare a set of nearly identical homogeneous sandstone samples appx. 12"W x 24"L x 24"H (30 x 60 x 60 cm) for the final demonstration

1.4.2. The rock sample will be homogeneous sandstone, and rock compressive strength values will be provided for the sandstone samples furnished by DSATS in early 2021. The Drillbotics committee will mark the surface of rock to indicate the well center where drilling will start. It will be located at the intersection of two lines drawn from opposite corners of the rock sample.

1.4.3. The university and/or students may acquire or produce at their own cost rock samples as needed to verify the design and allow students to practice using their machine prior to the test. Drilling of any samples provided by DSATS prior to Phase II testing is not allowed and could lead to disqualification, except for the pilot hole to be drilled at the test location.

1.5. Bits

1.5.1. Upon request, DSATS will send a bit to the finalist teams for use in Phase II. It is expected that the BHA and pipe will cause some difficulty, both for initiating drilling dysfunction and for sensor integration and data telemetry. The judges will look for creative concepts supported by sound reasoning showing an understanding of how the BHA, bit and drillstring function together, and how the downhole system measures, samples and transmits the drilling data.

1.5.2. Upon request, the bit shall be returned to the Committee following Phase II testing for reconditioning for use in future competitions.

1.5.3. One (1) PDC bit will be provided by DSATS to be used during the Phase II tests. For 2020-2021 the bit will be:

1.5.3.1. A micro-bit 1.5" in (38.1 mm) diameter and 2.0” in total length.

1.5.3.2. Low axial aggressiveness and high side aggressiveness (i.e. high bit anisotropy).

1.5.4. Students are encouraged to consider bit wear prior to the final test and its impact on drilling performance during the onsite testing. Based on prior competitions, bit wear should be minimal but some cutter damage is always possible.

1.5.5. Student teams may build or buy similar drill bits to test their design with the rock samples they sourced. The students must not engage any third parties or receive professional assistance in designing their own bit, however manufacturing can be performed by a third party.

1.5.6. For the final competition, the students may use the directional drill bit provided by DSATS or use their own bit design. However, the dimensions of their bits must not exceed 1.5 inches in diameter and 2 inches long. This provision is made to enable students to fully optimize the bit design for their specific directional system.

1.6. Drillpipe

1.6.1. Preliminary typical tubing specifications for aluminum tubing are listed below to assist with the mechanical and electrical design of the rig. Stainless steel tubing or aluminum tubing is permitted for the competition but must use the same dimensions as below, or the nearest metric equivalent. Teams may choose durability over flexibility and shall explain their choice in the design report.

1.6.2. The drill pipe specifications for the 2020-2021 competition are subject to change, but should be:

1.6.2.1. Round Aluminum Tube 3/8 inch diameter x 36 inches long; 0.049 inch wall or equivalent

1.6.2.2. The material from KS Precision Metals is a typical low alloy material: "Our Aluminum tubing with wall thickness of .035 or .049 is 6061 T6"

1.6.3. DSATS will not be providing tubing to the competition teams.

1.6.4. The use of a metric equivalent of the tubing is permitted.

1.6.5. Tubing is usually available from various hobby shops such as K-S Hobby and Craft Metal Tubing and via Amazon and other suppliers.

http://www.hobbylinc.com/htm/k+s/k+s9409.htm

1.7. Tool joints

1.7.1. Students may design their own tool joints as long as the design concept is included in the Phase I proposal.

1.7.2. Alternately, students may use commercially available connectors/fittings attached to the drillpipe using threads, epoxy cement or other material, and/or may use retaining screws if desired, as long as the design concept is included in the Phase I proposal.

1.7.2.1. A fitting used somewhat successfully in 2017 is available from Swagelock. In 2018, the winning team used a fitting from Vertex.

1.7.2.2. A fitting used successfully in 2016, but which did not work well in 2017, is available from Lenz (http://lenzinc.com/products/o-ring-seal-hydraulic-tube-fitting/hydraulic-straight-connectors) that uses a split-ring to allow a torque transfer across the fitting.

1.7.3. Students must state WHY they choose a tooljoint design in the Phase I proposal.

1.8. Bit sub/drill collar/stabilizers

1.8.1. It is expected that each team will design and build their own bit sub. Instrumentation of the bit sub is ideal for directional sensors.

1.8.2. Additional weight may be added to the bit sub, or surface weight/force (above the rock sample) may be applied to provide weight on bit and drill pipe tension

1.8.3. Stabilizers are permitted but will be limited in length at the discretion of the Challenge Committee. Advise the committee of your choice and why and include this in the Phase I design for committee consideration.

1.8.4. Students must add sensors to the drillstring but are not permitted to instrument the rock samples. They must have a smaller diameter than the stabilizers and bit by at least 10%. Please include design concepts in the Phase I design.

1.8.5. The addition of along-string sensors to measure vibrations, verticality and/or tortuosity or other parameters will receive extra consideration. They must have a smaller diameter than the stabilizers and bit by at least 10%.

1.9. Automated Drilling

1.9.1. Drilling automation should be considered a combination of data, control AND dynamic modeling so that the control algorithm can determine how to respond to differences between the expected and actual performance. Process state detection can often enhance automation performance. Refer to documents posted on the DSATS website for more information.

1.9.2. Once drilling of the sample commences, the machine should operate autonomously. Remote operation and/or intervention is not allowed.

1.9.3. All directional control operations should be autonomously controlled by the drilling rig

1.9.3.1. Manual intervention to add and/or remove a steering mechanism (e.g. whipstock) is permitted, however the determination/calculation of the orientation setting of the mechanism is required to be autonomous and must be shown on the rig floor display during each steering mechanism manipulation activity

1.9.3.2. Length and timing of drilling modes (e.g. switching from slide drilling to rotational drilling, initiating the directional surveying procedure at the appropriate survey interval), must be autonomously determined/calculated and controlled

1.9.3.3. Directional surveys acquired by the system need to be used as feedback for the steering control (and/or calculation of the steering requirements) logic.

1.9.4. Set-point commands for drilling parameters (WOB, RPM, ROP, etc.) should be optimized such that drilling dysfunctions are avoided, and drilling can be completed within the given time frame. Real-time optimization should be done automatically. The controllers need to ensure that the drilling parameters respond once the set points are altered.

1.10. Sensors

1.10.1. The team may elect to use existing oilfield sensors or may look to other industries for alternate sensors.

1.10.2. The team may develop its own sensors if so desired.

1.10.3. Sensor quality differs from data quality. Both are important considerations in this competition.

1.10.4. The final report shall address which sensors were selected and why. The sensor calibration process shall also be explained.

1.11. Data collection and handling

1.11.1. The team may elect to use standard data collection and recording techniques or may develop their own. Data handling techniques and why they were chosen should be described in the Phase I submittal.

1.11.2. The final report shall address which data systems were selected and why.

1.11.3. The observed response time of measurements, data aggregation and control algorithms should be compared to the Phase I estimates and published in the final report.

1.11.4. Describe how data is measured, aggregated, stored and retrieved. Describe calibration and data validation techniques used.

1.12. Data visualization

1.12.1. Novel ways of presenting the data and progress of drilling in real time while drilling will receive particular attention from the judges.

1.12.2. Visualization of the processes (automation, optimization, drilling state, etc.) should be intuitive and easily understood by the judges, who will view this from the perspective of the driller operating a rig equipped with automated controls.

1.12.3. Data must be presented in a format that allows the judges to easily determine bit depth, elapsed drilling time, ROP, MSE, verticality/inclination, vibration, and any other calculated or measured variable used to outline the drilling rigs performance to the judges. Lack of an appealing and usable Graphic User Interface (GUI) will be noted to the detriment of the team.

1.12.4. All depths shall use the industry-standard datum of rotary/kelly bushing interface (RKB), which should be the top of the rig's "drill floor."

1.12.5. An End of Well (EOW) report should be provided to the judges at the conclusion of drilling.

1.12.6. See Appendix "A" for directional surveying-specific data visualization requirements

1.13. Measure and analyze the performance

1.13.1. The drilling machine should react to changing "downhole" conditions to select the optimal drilling parameters for improved performance, as measured by the rate of penetration (ROP), mechanical specific energy (MSE), verticality, cost per foot or meter, and other standard drilling measures or key performance indicators. Adding parameters such as MSE, or similar features, to the control algorithms will receive special attention from the judges.

1.13.2. Design limits of the drilling machine shall be determined and shall be incorporated in the programming of the controls during the construction phase.

1.13.3. Downhole measurements from directional sensors are to be used for adjusting drilling parameters and control of drilling machines used to aid in directional drilling

1.13.4. The final report (see Clause **Error! Reference source not found.**) shall outline drilling performance and efficiency criteria and measured results.

1.13.5. One of DSATS' goals is to promote plug and play capability to accelerate the implementation of drilling automation. A DSATS committee is preparing definitions and examples of proposed data communication protocols and interfaces. Once this is available, the Drillbotics competition will require the use of these standard protocols. This will not be a requirement for 2021 but it will be included in future competitions. Links to these standards will be added to the Drillbotics.com website when they are published.

1.14. The test well:

1.14.1. The location and logistics for the final demonstration will be announced in early 2021.

1.14.2. Prior to the commencement of the test, teams will attach a bell nipple (per 1.33). They will then manually drill the pilot hole not to exceed 1" deep.

1.14.3. When the test begins, the teams will start drilling autonomously by continuing to drill the pilot hole, keeping the wellbore as vertical as possible until reaching the kick-off point. All rigs start the drilling competition at the same time.

1.14.4. The teams will kick off from vertical at any depth below the 4” vertical surface hole

1.14.5. The teams will attempt to hit multiple targets (varying X/Y coordinates and vertical depths) by following a provided directional plan/trajectory. Directional objective scoring will be based on the accuracy of the target depth intersection (i.e. horizontal distance from the target coordinate at the given target vertical depth). Refer to Appendix “A” for additional directional objective details.

1.14.6. No lateral forces may be applied above the rock or to the rock.

1.14.7. Drilling will stop at 3 hours or when the last team exits the rock sample.

1.14.8. A closed-loop fluid circulation system is not required, but could be of advantage for directional drilling, the bit and machinery should be cooled with air or fluid/water if needed. The design of the fluid system, if any, should be included in the Phase I design.

1.14.9. The rock sample will be homogeneous and will be capable of aiding in closed-loop fluid circulation. Note that the rock samples will leak once the drillbit punctures a rock face, so a rig design that includes a containment system is required.

1.14.10. Casing must fit in the directional wellbore. The ability to “run casing” is the secondary judging metric. Judges will run a “flexible” casing used as a gauge of borehole quality

1.14.11. A rig move, walking or skidding is not required, but the mobility of the rig will be considered in the design phase. The chargeable weight of the rig is an important consideration by the judges. See C 1.18.4.2 F regarding how this is calculated.

1.15. Not included in the 2020-2021 competition

1.15.1. The drilling will not include automating the making or breaking of connections. If connections are necessary due to the rig and drillstring design, connections should be made manually, and the time involved with the connections will be included with respect to its effect on drilling performance (rate of penetration reduction).

1.16. Other Considerations

1.16.1. The design concepts shall be developed by the student team under the supervision of the faculty. Faculty and lab assistants should review the designs to ensure student safety (see Appendix B).

1.16.2. Construction of the equipment shall be supervised by the student team, but may use skilled labor such as welders and lab technicians. The use of outside assistance shall be discussed in the reports and the final paper. DSATS encourages the students to gain hands-on experience with the construction of the rig since this experience will be helpful to the career of individuals in the drilling industry.

1.17. Presentation to judges at Phase II Testing

1.17.1. The students will present a BRIEF summary of their final design, highlighting changes from their Phase I design, if any. Include an explanation of why any changes were necessary, as this indicates to the judges how much students learned during the design and construction process. Explain what measurement and control features have been deployed. Describe novel developments or just something learned that was worthwhile. Also include how actual expenses compared with the initial estimate. (Previous teams used a short PowerPoint presentation of about ten slides or so. Use any format you like.) At some time during your talk, let us know who the team members are and what background they have that pertains to the project. Be sure to include all your team members as presenters, not just one spokesperson. The committee wants to see if all team members have a good understanding of key issues.

1.17.2. Judges will ask questions to ascertain additional details about the design and construction process and to see if all team members have a reasonable understanding how all the various disciplines used for the rig design and construction fit together.

1.17.3. All teams may sit in for the presentations and Q&A of the other teams. The order of presentation will be determined by drawing lots.

1.18. Project report

1.18.1. Starting in the fall term, the student team shall submit to DSATS a short monthly project report that is no more than one page in length (additional pages will be ignored) due on or before the last day of each month. Send it via email to 2021@Drilbotics.com. The monthly report should include:

1.18.2. Phase I

- Key project activities over the past month.
- Rig design criteria, constraints, tradeoffs, and how critical decisions were determined
- Cost updates
- Significant new learning, if any

1.18.3. Phase II

- Construction issues and resolution
- Summary of recorded data and key events
- Drilling parameters [such as WOB] and how they impact the test
- Other items of interest

1.18.4. Report content

1.18.4.1. To teach students that their work involves economic trade-offs, the monthly report should include at a minimum a summary estimate of team member labor hours for each step in the project: design, construction, testing, reporting, and a cost summary for hardware and software related expenditures. Also include labor for non-students that affect the cost of the project. Labor rates are not considered, as to eliminate international currency effects. Labor is not considered in the cost limits of item 6.1, but should be discussed in the report and paper.

1.18.4.2. Design reports must contain the following tables and place them in their design report appendices:

B. Student Biographies

- Name
- Previous degree attained – major
- Current degree and expected graduation date (month/year)
- Main area of contribution to the project
- Other information as deemed appropriate by the team

C. Summary of Calculations (list these at a minimum, list other is a similar format)

Parameter	Symbol	Calculated Results		Safety Factor	Max Allowable		Reference	(Other as needed)
		Field Units	Metric Units		Field Units	Metric Units		
Critical buckling load								
Burst limit								
Torque limit								
… Other								

D. Power Consumption (rename devices as appropriate)

Device	Voltage	Current	Estimated		Single or Three ϕ	(Other as needed)
			HP	Watts		
Rotation						
Hoist						
Pump						
… Other						
Controls						
Displays						
…						
Total						

E. Diagram showing maximum dimensions of rig when operational (Include all auxiliaries) [Needed to determine size of display area as the Drilling Conference and confirm the height is within the limits imposed by the conference organizers]

F. Chargeable Weight of Rig (include shipping crates/boxes for rig and auxiliaries)

- The Chargeable Weight of Freight shipments are calculated as the Actual Weight (Gross Weight) or the Volumetric (also called Volume or

Dimensional) Weight of the shipment, whichever is the greater. This uses an estimated weight that is calculated based on the dimensions (length, width and height) of a package (shipments are always shown in the order of L x W x H). Typically, large items with a light overall weight take up more space on an aircraft than a small, heavy item. That's why the shippers charge according to Chargeable Weight.

- Multiply the length by the width by the height (L x W x H) in inches to obtain the cubic inches, then:
- To obtain the dimensional weight in pounds using inches, divide the cubic inch result by 166
- To obtain the dimensional weight in kilograms using inches, divide the cubic inch result by 366
- Using Dimensions in Centimeters: To obtain the dimensional weight in kilograms using centimeters, multiply the length by the width by the height (L x W x H) in centimeters and divide the result by 6000

1.18.5. File naming convention

1.18.5.1. To avoid extra work by the committee to rename all files, please use this convention for:

1.18.5.1.1. Monthly reports
Year-Month# University Name (abbreviated)
(note this is the competition year (spring term))
Example 2021-09 UDC

1.18.5.1.2. Design reports
Year University Name (abbreviated)
(note this is the competition year (spring term))
Example 2021 University of Drillbotics Competition

1.19. Final report and paper

1.19.1. The finalists shall prepare a project report that addresses the items below. We suggest you use the format of most SPE papers. For reference, please see http://spe.org/authors/resources/

1.19.2. The winning team of Group B is encouraged to update the report as needed to comply with SPE paper submittal guidelines and to submit a technical paper for publication by the SPE at its Annual Drilling Conference. SPE typically requires that the manuscript is due in the fall following the Phase II test.

1.19.3. The timing for submittal of the abstract and paper will be the published deadlines per the call for papers and conference guidelines as posted on the SPE's website (www.spe.org).

1.19.4. The abstract must generate sufficient interest with the SPE review committees to warrant publication, although DSATS will help promote acceptance where possible

1.19.5. The paper should address at a minimum

- *1.19.5.1.* The technical and economic considerations for the control system, rig, and BHA design, including why certain features were chosen and why others were rejected.
- *1.19.5.2.* The setup of the experimental test, the results and shortcomings.
- *1.19.5.3.* Recommendations for improvements to the design and testing procedures.
- *1.19.5.4.* Recommendations for improvements by DSATS of the competition guidelines, scheduling and provided material.
- *1.19.5.5.* Areas of learning gained through the competition not covered in the university course material.
- *1.19.5.6.* A brief bio or CV of the team members and their sponsoring faculty.