Bjørn Kristian Eid
Thomas Overen
Stine Olsen

Bachelor's thesis

Design and implementation of full system monitoring for a prototype Hyperloop pod

with custom designed data acquisition PCB

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2021

NTNU
Norwegian University of
Science and Technology

Bjørn Kristian Eid
Thomas Overen
Stine Olsen

Design and implementation of full system monitoring for a prototype Hyperloop pod

with custom designed data acquisition PCB

Bachelor's thesis
May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU

Institutt for elektroniske systemer

Bacheloroppgave

Project title: Design and implementation of full system monitoring for a prototype Hyperloop pod		**Gitt dato:** 16.01.2021 **Innlevingsdato:** 20.05.2021 **Gradering** **[X] åpent** **[ ] lukket** **[ ] åpent fra** _____
Gruppedeltakere: Bjørn Kristian Eid – 41386323 – bjornke@stud.ntnu.no Stine Olsen - 41311984 – stine.olsen@ntnu.no Thomas Overen – 98823488 – thomov@stud.ntnu.no		**Antall sider/bilag** 47 sider rapport / 45 sider vedlegg
Studieretning: Industriell Instrumentering / Elektronikk		**Prosjektnummer:** E2111
Oppdragsgiver: Shift Hyperloop	**Kontaktperson hos oppdragsgiver (navn/tlf.):** Bendik Nyhavn 99735950 bny@shifthyperloop.com	**Veileder internt (navn/email/tlf.):** Dominik Osinski 73559609 Dominik.osinski@ntnu.no

Sammendrag

The purpose of this project is to design and implement full system monitoring for a prototype Hyperloop pod in collaboration with the student organization Shift Hyperloop. Due to halted progress because of the Covid-19 pandemic in the previous project year, such a system did not yet exist other than a crude revision 1 of a custom data acquisition circuit board. Not much testing had been done, and any gathered data and most documentation had been lost in a storage server crash. Individual members were therefore the primary source of information, so much work was needed to build a solid foundation for future improvements and iterations. Some sensor selection choices were kept because those sensors were acquired, but not used. In terms of printed circuit board (PCB) design, interface circuitry, and code, everything was designed from scratch for this thesis.

The project group began with constructing the overall concept for the system, an interdisciplinary process which meant discussing needs and requirements of other technical systems on the pod. A requirement for all electronic systems was that we had to use the Atmel SAMV71Q21 microcontroller unit (MCU) and the system had to communicate via CAN-bus. It was decided on a solution with two separate custom PCBs for data acquisition and signal processing to handle the expected number of sensors, and to reduce the length of signal cable routing.

Design progression continued with circuit simulations and PCB design. Measurement needs of the subsystems kept changing throughout the process, therefore specific sensor selections and interface circuitry were not ready until revision 2. Because of this, there will be less focus on revision 1 in this report.

There have been little time for testing of revision 2, as PCB components for all systems were ordered very late. Preliminary testing have shown promising results, as most interface circuits are working and most of the code will run on our customized MCU board. Each individual sensor have been tested for function and calibration, but there have not been time to test sensors in combination with custom PCBs and code. Further testing will be necessary to verify the functionality of the complete system, however, this is considered acceptable since the progress of the main project is in sync with this progress. With the project group members being a part of the Shift Hyperloop project as well, we will continue working on the system towards the European Hyperloop event in Valencia, Spain, this July.

Stikkord:	Keywords: Hyperloop, PCB, sensors, data acquisition

Abstract

The purpose of this project is to design and implement full system monitoring for a prototype Hyperloop pod in collaboration with the student organization Shift Hyperloop. Due to halted progress because of the Covid-19 pandemic in the previous project year, such a system did not yet exist other than a crude revision 1 of a custom data acquisition circuit board. Not much testing had been done, and any gathered data and most documentation had been lost in a storage server crash. Individual members were therefore the primary source of information, so much work was needed to build a solid foundation for future improvements and iterations. Some sensor selection choices were kept because those sensors were acquired, but not used. In terms of printed circuit board (PCB) design, interface circuitry, and code, everything was designed from scratch for this thesis.

The project group began with constructing the overall concept for the system, an interdisciplinary process which meant discussing needs and requirements of other technical systems on the pod. A requirement for all electronic systems was that we had to use the Atmel SAMV71Q21 microcontroller unit (MCU) and the system had to communicate via CAN-bus. It was decided on a solution with two separate custom PCBs for data acquisition and signal processing to handle the expected number of sensors, and to reduce the length of signal cable routing.

Design progression continued with circuit simulations and PCB design. Measurement needs of the subsystems kept changing throughout the process, therefore specific sensor selections and interface circuitry were not ready until revision 2. Because of this, there will be less focus on revision 1 in this report.

There have been little time for testing of revision 2, as PCB components for all systems were ordered very late. Preliminary testing have shown promising results, as most interface circuits are working and most of the code will run on our customized MCU board. Each individual sensor have been tested for function and calibration, but there have not been time to test sensors in combination with custom PCBs and code. Further testing will be necessary to verify the functionality of the complete system, however, this is considered acceptable since the progress of the main project is in sync with this progress. With the project group members being a part of the Shift Hyperloop project as well, we will continue working on the system towards the European Hyperloop event in Valencia, Spain, this July.

Sammendrag

Målet med prosjektet er å lage et komplett overvåkingssystem for en Hyperloop prototype fartøy i samarbeid med studentorganisasjonen Shift Hyperloop. På grunn av Covid-19 pandemien ble fjorårets system ikke ferdig, og alt som ble produsert var en grov revisjon 1 av et kretskort for datainnsamling. Det ble ikke gjennomført mye testing, og det lille av data som var samlet inn gikk tapt da organisasjonens lagringsserver krasjet. Det meste av dokumentasjon gikk også tapt som følge av dette. Hovedkilden til informasjon ble derfor enkeltmedlemmer, så mye arbeid var derfor nødvendig for å danne et solid grunnlag for videreutvikling. Enkelte sensorer som allerede var anskaffet, men forblitt ubrukt ble vurdert og inkludert i prosjektgruppens design. Kretskortdesign, valg av analoge grensesnitt, kode og valg av tilleggssensorikk ble gjort fra bunnen av for denne oppgaven.

Prosjektgruppen startet med å konstruere det generelle konseptet for systemet, en prosess som innebar å kartlegge hvilke systemer som trengte overvåking og hvilke krav de stilte. Dette inkluderte både mekaniske og elektroniske sytemer. Felles krav for alle elektroniske systemer var at de måtte bruke Atmel SAMV71Q21 mikrokontolleren, og systemet måtte kommunisere via CAN-bus. Prosjektgruppen valgte en løsning med to separate kretskort for datainnsamling og signalbehandling. Dette for å håndtere det anslåtte antall sensorer og for å redusere lengden på analoge signalkabler.

Videre jobbing med design inkluderte kretssimulering og kretskortdesign. Sensorbehovene til flere tekniske systemer endret seg gjennom hele prosessen, noe som førte til at sensorvalg og design på analoge grensesnitt ikke ble bestemt før revisjon 2. Grunnet dette vil det være lite fokus på revisjon 1 i denne rapporten.

Komponentene til de ulike elektriske systemene ble bestilt svært sent, så det ble lite tid til testing av revisjon 2. Testene som har blitt foretatt hittil har vært lovende, da de fleste analoge grensesnittkretsene fungerte som forventet og store deler av koden kjører på mikrokontrolleren. Hver sensor er blitt testet enkeltvis, men det har ikke vært tid til å teste i kombinasjon med våre egne kretskort og kode. Ytterligere testing vil være nødvendig for å verifisere funksjonaliteten til det fulle systemet. Ettersom dette skyldes forsinkelser i hovedprosjektet så anses dette som en akseptabel mangel i denne rapporten. Prosjektgruppens medlemmer er også medlemmer i Shift Hyperloop og vil fortsette og jobbe for hovedprosjektet med sensorsystemet fram til European Hyperloop Week, som vil finne sted i Valencia i midten av juli.

Preface

The aim of this project is to design and implement a fully working monitoring system for a prototype Hyperloop pod. The project is in collaboration with Shift Hyperloop, a student organization working with creating a prototype pod. Shift Hyperloop has the goal to create a fully functioning pod in order to compete in European Hyperloop Week, 2021, a Hyperloop conference and competition in Spain. being able to create and work on a product until completion has been an essential part of this project.

The motivation to write and work with this project has been an interest in sensors, designing printed circuit boards and the desire to create a product. The group got together in August and decided to write our bachelor thesis together about a sensor system. All group members became a part of Shift Hyperloop, sent in the thesis to NTNU and reserved it. This thesis have therefore been in the works since September.

Along the way we experienced some challenges. One of our main challenges was the COVID-19 pandemic, as this affected both access to campus and group workshops. Another challenge for the project was the arrival time of components, as testing the second revision of PCBs was done very closely to this thesis submission deadline.

The project group consists of Thomas Overen, Bjørn Kristian Eid and Stine Olsen. We began working on the project in September 2020, and have worked steadily throughout the whole school year with common workshops in Shift and group work.

We would like to thank our adviser, Dominik Oskinski for his inspiring enthusiasm, insightful guidance and reassurances when we felt overwhelmed by the scope of the project.

In addition, we would like to thank Shift Hyperloop for the opportunity to write this thesis and all the help throughout the working period. A special thanks to our chief technical officer, Bendik Nyhavn, for all his input and help throughout the project. Additionally, we would like to thank Magnus Kolnes Oddstøl for all his help, guidance and gleeful cheers at the office whenever he achieved a breakthrough.

Trondheim. May 20th, 2021

Bjørn Kristian Eid

Thomas Overen

Stine Olsen

Contents

List of Figures

List of Tables

Acronyms and explanations

ADC - Analog to digital converter
ASF - Advanced Software Framework
CAN - Controller area network
CHSR - California High-Speed Rail
EHW - European Hyperloop Week
FDD - Final Design Documentation
FMECA - Failure Mode, Effects, and Criticality Analysis
I2C - Inter-Integrated Circuit
IDE - Integrated Development Enviroment
IMU - Internal measurement unit
IN-AMP - Instrumentation amplifier
LED - Light-emitting Diode
MCU - Microcontroller Unit
MiT - Massachusetts Institute of Technology
NTC - Negative temperature coefficient
NTNU - Norwegian University of Science and Technology
OP-AMP - Operation amplifier
PCB - Printed circuit board
PSU - Power Supply Unit
SAMV71 - ATSAMV71Q21B, the specific microcontroller that is being used
VCU - Vehicle Control Unit

1 Introduction

1.1 Background and motivation

This project is a collaboration between the student organization Shift Hyperloop and the project group members. The initiative was taken by one of the group members, who had already been part of the Shift team for three semesters. The specialization Industrial Instrumentation focuses on the functionality of sensors and how they interact with electronic circuitry. A complex sensor system such as the one needed for a prototype Hyperloop pod was therefore a highly relevant task for a bachelor thesis. The use of design tools such as Altium Designer and Microchip Studio also make it highly relevant for the specialization of Electronics. The details and scope of the thesis was determined by the project group in collaboration with technical management in Shift, and required the students to join the organization and be part of the main project at the start of the fall semester 2020.

Shift Hyperloop is a young organization, as it was founded in the spring of 2019. During the 2020 season the entire progress was cut short due to the Coronavirus outbreak, and few technical systems were developed beyond early first revision testing. In addition, the server Shift used to store documentation had a malfunction during the summer of 2020, which caused much documented progress to be lost. Providing a functioning and well documented data acquisition platform which could be further developed by future teams was therefore a primary motivation for this thesis. For the project group it would provide valuable experience as the project would include both sensor theory, complex circuit board design, embedded code and implementation of hardware. Since the main project and the thesis project have slightly different timelines, it was not possible to include the entire scope of our work in this report. Primarily this means thorough testing on finished products, and the final implementation on the actual prototype pod will not be included in this thesis.

1.2 Shift Hyperloop

Shift Hyperloop is a non-profit student organization working with designing and constructing a prototype Hyperloop pod. They are operated exclusively by students currently enrolled at the Norwegian University of Science and Technology (NTNU). The organization was founded in 2019 by ten students inspired by Elon Musk's whitepaper about the Hyperloop concept. Today Shift Hyperloop consists of around eighty students from several different disciplines, including geologists and architects who work on full-scale integration and even a philosophy student studying attitudes towards this new mode of transportation. The previous team had the goal of competing in the SpaceX Hyperloop Pod Competition, but did not get the opportunity because of the Covid-19 pandemic. This years team are expected to compete in the European Hyperloop Week (EHW), a new conference and competition in Valencia, Spain.

This thesis is written by members of the Sensor System. It is a subsystem in the Electronics Group. The Electronics Group also consist of the systems Embedded, Master Library, State Indication, Telemetry and Vehicle Control Unit (VCU), typically with two members responsible for each system. Other groups in the Shift Hyperloop organization for the 2021 season are; Battery, Concept, Levitation, Mechanical, Powertrain, Relations and Software. In addition there is a Board that is made up of a CEO, CFO, CTO electrical, CTO mechanical, CMO and CCO (concept).

1.3 Thesis structure

This thesis will begin with an introduction into the Hyperloop concept, where the reader will be acquainted with what Hyperloop is and the current status of some commercial development projects. Following this we will present theory about sensor systems and some of the sensors used

in this project, in order for the reader to understand the choices made later on.

We will then go on explaining the different phases of the project and our method of working. In this section the reader will be walked though everything from planning to designing and testing the system. Then all the final design choices are presented, this includes sensor selection, circuitry, hardware design and code.

Following this, the reader will be walked through the functionality testing performed of the different sensors, which were tested separately with minimal additional circuitry to minimize sources of errors.

In the next section we will present our results from testing our custom sensor suites. These results will be discussed further in the next section, alongside a discussion about our method and design choices. Lastly, we will present our conclusion as well as suggestions for further work on the project.

2 Hyperloop concept

The idea of magnetically levitating modes of transportation have been circulating since near the start of the 20th century. Inventors and electricians such as Robert H. Goddard and Èmile Bachelet each held patents for various designs of magnetically levitated transportation methods, with or without the inclusion of vacuum tubes [1].

Entrepreneur and Tesla-founder Elon Musk published in 2013 a whitepaper detailing the design of the Hyperloop Alpha (Figure 1), a modern version of what had previously been called a vactrain. The Hyperloop Alpha came as a response from Musk to the 2008 approval of the California High-Speed Rail (CHSR). Musk felt the CHSR was in no means a good alternative to existing modes of mass transport in the region, especially given the cost of investment versus the gained benefits. Musk felt the CHSR failed to improve sufficiently when it came to operating cost, travel time and passenger safety. [2]

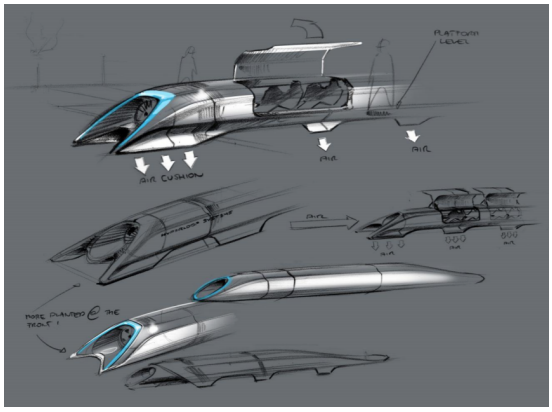


Figure 1: Design sketches of the Hyperloop Alpha prototype.

Source: Tesla

Musk's concept intended to rely on pods running in a low pressure tube instead of complete vacuum. A compressor fan would be installed at the front of each pod. This compressor would serve as the method of removing the build up of air in front of the pod, as well as pressurizing the air so it could be used as an air cushion for levitation. For propulsion, the tubes would have linear induction motors placed along the length and the pods would have fixed aluminum stators mounted underneath. With this design, Musk theorized the Hyperloop Alpha to be able to reach a cruising speed of up to 1220 km/h, thus covering the 560 km long San Francisco to Los Angeles stretch in approximately 35 minutes. [2]

As Musk did not have time to pursue the Hyperloop concept to a real life model, the design was therefore open sourced. Multiple companies were started to continue the work with the Hyperloop design, but it is Virgin Hyperloop that have come the furthest (as of May 2021). Virgin Hyperloop (formerly known as Hyperloop One and Virgin Hyperloop One) was founded in 2014, and have since then worked with building the first Hyperloop [3]. In November 2020, they had the first passenger test. Two passengers traveled 500m, with a speed of 172km/h or 42m/s. Virgin Hyperloop is continuing their work, in order to reach their goal of more than 1000km/h. [4]

2.1 Other Hyperloop teams

SpaceX and founder Elon Musk created in 2015 a design competition for the purpose of accelerating the development of a functional prototype, and to encourage student innovation [5]. The competition has been held five times over a total of four years, 2016-2019 **Wiki˙hyperloop˙comp**, and some of the competing teams have shared their design reports from earlier pods. These designs were looked at and considered by Shift Hyperloop as part of the initial research during the Concept phase. Here we will present a short summary from the available reports, focusing on sensor system related information.

2.1.1 MIT Hyperloop

Massachusetts Institute of Technology (MIT) designed their 2017 pod with more than 35 sensors distributed over five customized PCBs. Each PCB had an Arduino Due attached on top of it through header pins, and every PCB was designed the same way. A sixth custom PCB was designed to host an Odroid C1+ unit to act as the master computer. The custom PCBs all communicate via RS485 protocol. Each Arduino and the Odroid were also connected via an onboard USB hub, to allow for flashing of code.

The MIT pod was passive, which means it did not have a propulsion system. Instead the pod was designed to "glide" on arrays of magnets, using a launch system provided by SpaceX in the early years of the competition. Sensor data was used to monitor positioning relative to the track, to the tube and status of a low-speed wheel system and braking [6].

2.1.2 HypED

The team *HypED* from the University of Edinburgh had a total of 25 sensors on their pod "Poddy McPodface" in 2017. They used five Raspberry Pi units in a master-slave configuration to control and process sensor data. They had four slave modules and one master, all connected through an onboard ethernet switch. Each slave had up to four sensors connected to them. Some sensors went right into the master Raspberry Pi. The master performed all the on-board calculations.

The HypED pod has an active propulsion- and a passive levitation system. The pod will float on magnetic Hallback arrays while being propelled by a high-friction wheel driven by a rotary electric motor. The sensors will monitor pod position and status, and multiple data values are used in real-time as active control over the pod [7].

2.1.3 CU Hyperloop

The team CU Hyperloop comes from the University of Colorado. In 2018 they posted a report showcasing their pod design. More than 14 sensors are connected to a TIVA C unit, some through an external ADC. The ADC sends data to the TIVA using SPI. Other sensors used various digital communication protocols and were connected directly to the TIVA.

Their pod would be propelled by an initial sling, and further propelled with nozzled tanks of pressurized Nitrogen. Wheels would be used for contact with the track. Sensor data would therefore monitor pod positioning relative to track and tube, as well as pressure system status and temperatures. Most data would be used in real-time as pod software state-machine input [8].

3 Sensor theory

While the complexity may vary, measuring systems are usually based around the same building blocks. It all starts with a physical variable such as temperature or distance, which somehow has to be made compatible with electronics. Once this is achieved, typically through a sensor element or transducer, this measurement signal has to be adapted again to meet specific criteria of digital circuitry, which is most common these days. After digitization of the signal, further processing is required to convert the signal to easily recognizable values, e.g °Celsius for temperature or mm for distance. Below we will elaborate on each of these steps of the process.

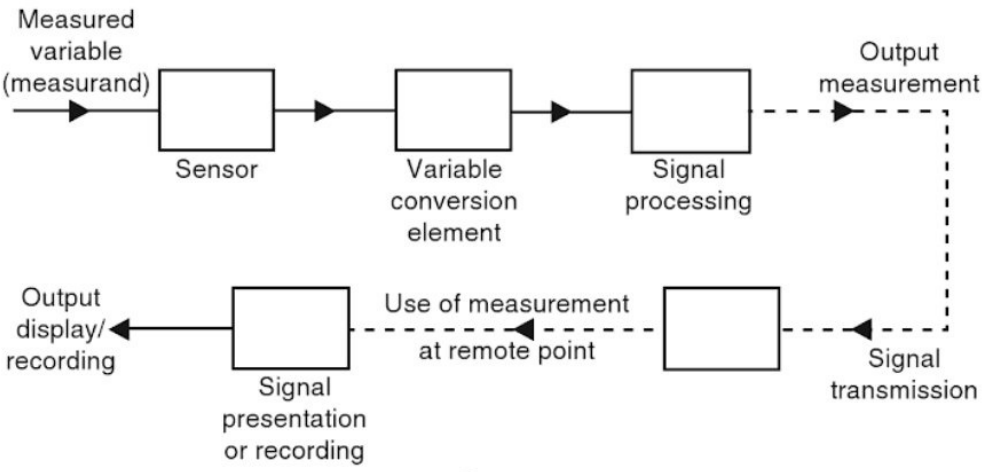


Figure 2: Example diagram of typical elements of a measuring system.

Source: Measurement and instrumentation

First up is the *sensor*, or sensing element. This is the actual transforming part, which take the physical *measured variable* such as temperature or chemical composition and transform it into a different medium. This output medium is in the form of resistance, voltage, current, capacitance, inductance or similar electrical characteristics. Examples of sensing elements are strain gauges or thermistors, which transform mechanical force or temperature into resistance, respectively.

Electronic devices are not equipped to directly read variables such as the ones mentioned above. Even in a multimeter used to measure a resistor there are several internal steps between the *measured variable* and the resistance displayed on screen. This is the function of the *variable conversion element*. Commonly known as interface circuits, which we will call them through this thesis, their purpose is to convert inconvenient variables into a manageable form. Such a form is typically a voltage or current output, but the function of the interface circuit depends on the sensor output and the monitoring device input. More on interface circuits in subsection 5.2.

When a sensing element and conversion element is combined into one device it is referred to as a transducer [9]. Some of the sensors used for this thesis are transducers, such as the OMT300 LED distance sensor and the HDA4400 hydraulic pressure sensor.

Next there is *signal processing*. This stage is for the most part to improve the output signal. This could be to filter out noise or to amplify the signal. Amplification is essential when using sensors such as thermocouples where the measurement signal is only a few millivolts [9, p. 7] and most input devices operate in the range of 3.3-5V. Signal processing can also include digitization of analog signals for communication with microcontrollers etc. When digitized, additional processing of the signal is possible through software, depending on the application and equipment. Analog to digital conversion will be further explained later in this chapter, in subsection 3.6.

Signal transmission serve as the sole purpose of transmitting the *output measurement* to the desired receiver. The receiver is represented by the block *Signal presentation or recording*, and can be a speaker, a screen, a computer etc. In many cases the presentation is omitted altogether, as the signal is only used internally as feedback in a control loop. The transmission medium is traditionally appropriate wiring, often screened to reduce effects of induced electrical noise [9, p. 7].

3.1 Photoelectric sensors

For the entire length of the competition track there will be poles with retro-reflective tape every 15 meters. These will act as navigational markers and have to be detected to determine the pod position along the track. Photoelectric sensors emit light from an light emitting diode (LED) or laser and detects if an object pass in front of it or there is a change in contrast. The object or changing contrast cause a change in reflection and is detected by a receiver. They have become quite common in industry and production factories, and can be used for multiple different means/systems. Therefore there is three primary modes for detection: diffuse-reflective, retro-reflective and through-beam (see Figure 3).

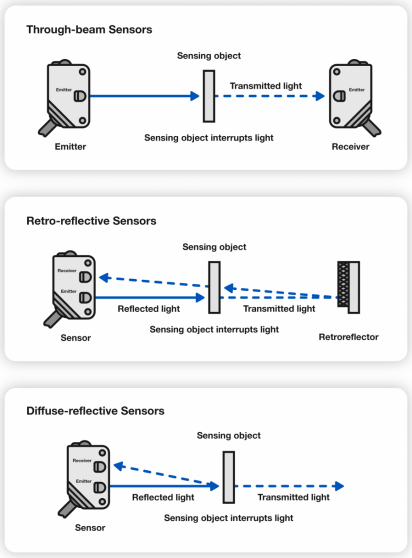


Figure 3: Illustration of photoelectric sensing modes.

Source: RS-online

The *Tri-Tronics XP10R5* uses retro-reflective mode. In retro-reflective mode both the transmitter and receiver are in the same casing. The transmitter sends a light out and if a reflective object blocks the light than it will reflect back to the sensor and be detected by the receiver. As this sensor will be used with red reflective tape it is crucial that the sensor will not detect anything else in the background, but only the red tape.

As the sensor will have to detect objects 1.5m away it needs to have a good range. With retro-reflective sensing mode there is no difference for the sensor whether the tape is red, yellow or gray, and the tape color does not affect the sensing range as the reflector reflects light back to the sensor better than other objects.

In this project there will be used a *R5* lens/block. The R5 block is a polarized anti-glare retro-reflective optical block used for reducing glare from shiny objects near the sensor. Polarization filter allows light from certain angles to be detected. This is because light reflected from the reflectors shift the phase of the light 90 degrees, whereas light reflected from a shiny target does not (see Figure 4). With this polarizing filter, only the light waves that are rotated will pass through, Polarized retro-reflective sensors need to have receiving light waves rotated, and this is typically done with corner cube reflectors, additionally this means that the target will be reflected parallel to the transmitted light. [10] [11]

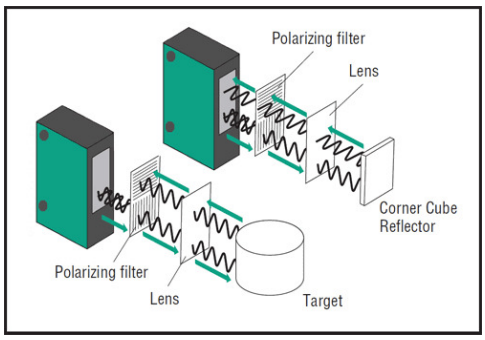
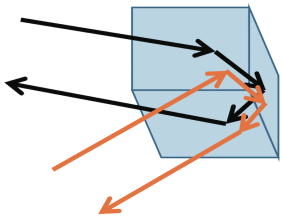
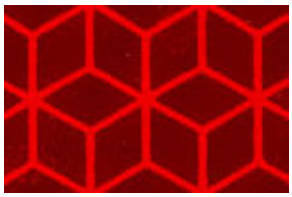


Figure 4: Polarized retro-reflective mode for photoelectic sensor.

Source: Automation



(a) Incoming light reflects back in parallel
Source: Researchgate



(b) Red tape with corner cube pattern
Source: ORAFOL

Figure 5: The workings of corner cube elements

The target for our sensor is *ORALITE VC104+* tape, is composed of corner cube elements bonded to a flexible, smooth surface [12]. The corner cube pattern on the tape can be seen in Figure 5b.

3.2 Optical distance sensors

In order to measure the distance between pod and track (I-beam) without touching the beam, there is need for a non-contact distance sensor. The most common non-contact distance sensors are ultrasonic and optical. As this is a pod that hopefully will travel in high speed, ultrasonic sensors will be too slow and there is no need to measure great distances. In this project a distance sensor from Pepperl+Fuchs will be used, model *OMT300-R200-IEP-IO-V1*. [13]

Triangulation is a method of measuring distance, where the sensor has one transmitter that sends out light (often as laser or LED) and one receiver that will measure the angle the light is reflected. Triangulation is based on that the length of one side and two angles are known. In an optical sensor the distance between the light source and the receiver is known, this is known as baseline [14]. The angle between the light source (laser or LED) and the detector is 90 degrees. The last angle we need to know is measured with the detector and lens. When the angle between the light source and reflected light is measured, the sensor can calculate the distance.

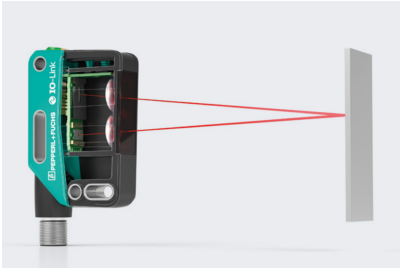


Figure 6: Triangulation for optical sensors

Source: Pepperl+Fuchs

3.3 IMU - Inertial measurement unit

To determinate how the pod moves during a run there will be placed four accelerometers inside, one in each corner of the pod. Additionally, the pod will be supplied with one IMU, or Internal Measurement Unit, near the center of mass. The IMU will also be used in order to determine the speed of the pod, as it is a requirement from EHW to have at least two independent speed measurements.

IMU is a sensor which uses gyroscopes, accelerometers, magnetometers in order to measure multiple parameters. Some of these parameters are angular rates (yaw, pitch, roll), force and acceleration in x-, y-, z-axis and magnetic field around the IMU. Additionaly, an IMU can often measure temperature and pressure. An IMU works best if it is placed at the center of mass of the product it will measure. [15]

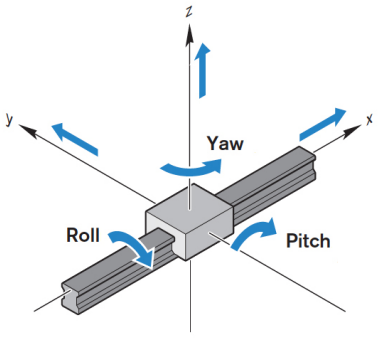


Figure 7: Yaw, pitch, roll movement

Source: Linearmotiontips

Accelerometers measure the acceleration. Acceleration is the rate of change of velocity with time [16].

$$\overline{a} = \frac{\Delta v}{\Delta t} \tag{1}$$

Or if we want instantaneous acceleration

$$a = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} = \frac{dv}{dt} \tag{2}$$

There are three main types of accelerometers, but the general concept that often repeats itself is that they use piezoelectric material. When stress or force is applied, the voltage or resistance increases, and can therefore estimate the acceleration. This can be done in multiplie axis, an IMU generally measures in x-, y- and z-axis.

With the acceleration from the accelerometer, the speed can be determined. Acceleration is the derivative of velocity with time, as seen in Equation 2. In order to find the velocity or speed we can integrate the equation.

Gyroscopes senses angular velocity ω, in other words the speed of rotation in rad/s or deg/s. The gyroscope in the IMU measures angular velocity in three directions (3-axis). [17]

Magnetometers measures the magnitude and direction of the Earth's magnetic field, and are used in multiple different applications. Some of the applications magnetometer often are used are in geophysical measurements, but also in planes, drones, boats, cars and metal detectors. One of the most common uses for magnetometers are in compasses. Magnetometers are not the most accurate sensors (0.5° - 3.0°) because of Earth's magnetic field changing with location and time. [14]

3.4 Thermocouples

To measure temperature in the motor thermocouples will be cast into the same block of epoxy as the motor. Thermocouples are one of the most applied and versatile means of measuring temperature, especially for industry applications. This is in part due to their large sensing span, fast response time and physical versatility [18].

Thermocouples consist of two leads of different metals with different heat coefficients that are welded together in one end, known as the hot junction. This point represent the sensing part of the thermocouple, whereas the opposite ends of these leads are referred to as the cold junction. The temperature is measured as the difference between the hot and cold junction points, and this calls for the temperature at the cold junction to be known so it can be compensated for, a process called cold-junction compensation. As the metals in the hot junction experience changes in temperature, the number of electrons being forced towards the cold end due to the kinetic energy caused by the heating process is what causes the cold junction point to experience an increase in electrical charge. As the two metals have different heat coefficients they will allow for different amounts of electrons to migrate at the same temperature, causing a difference in voltage potential between them.

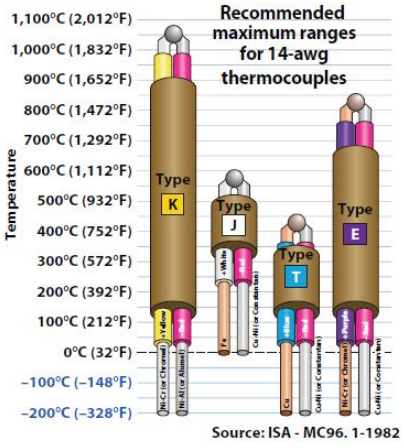


Figure 8: Recommended ranges for the four most common types of thermocouples; K, J, T and E

Source: Phoenix Contact

This voltage potential is in the range of millivolts and is therefore in need of amplification, but is also more exposed to electromagnetic interference. The type T thermocouple is shown to be the least susceptible type to magnetic fields when compared to type J and K [19]. This is because the copper and constantan metals it is made from are non-magnetic.

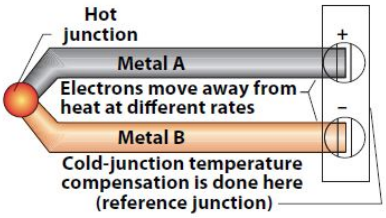


Figure 9: Construction of a thermocouple

Source: Phoenix Contact

3.5 Thermistors

To measure temperature for the brake pads and inside the inverter casing there will be multiple thermistors. Thermistors are often chosen due to the small size and low price.

Negative temperature coefficient (NTC) thermistors are resistors with a negative temperature coefficient, in other words the resistance decreases when temperature increases. NTC thermistors has often a range between -55°C to 200°C and are often very cheap. One of the main problems with thermistors is their non-linearity as this can provide some challenges.

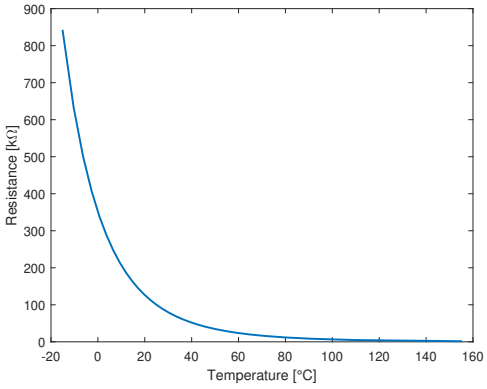


Figure 10: NTC thermistor, 100kΩ, β=4092 K

A NTC thermistor are in reality a third order polynomial

$$\frac{1}{T} = A + B * lnR_T + C * (lnR_T)^3$$

where A, B and C are Steinhart-Hart coefficents. These can be found from measuring the thermistor at three different temperatures and then use the Steinhart-Hart coefficient equations to find A, B and C.

To find the relation between temperature and resistance the Steinhart-Hart equation is often used, as Steinhart-Hart equation is typically accurate to 0.15 degrees in the range -50°C to 150°C. This is more accurate than most projects with a broad temperature range need. The temperature in the pod does not need to be that accurate, and instead of the Steinhart-Hart equation there will be used a simplified equation - the beta equation (Equation 14). [20]

3.6 ADC

Analog to digital converter, or ADC for short, converts the voltages that represents analog values (voltage, pressure, temperature etc.) to digital signals (bit-values) that the microcontroller can process. In the custom PCBs there are 12-bit ADCs which means that the voltages can be converted to bit-values between 0 and 4095 ($2^{12} = 4096$).

The bit-value we will get from an ADC can be found with the formula:

$$ADC_{OUT} = \frac{Analog\ input}{Reference\ input} * ADC\ Resolution \tag{3}$$

The ADC will convert voltages between $V_{Negative\ reference}$ and $V_{Postive\ reference}$, or shortened V_{ref}. In most cases (including this one) $V_{Negative\ reference}$ will be zero/GND. If the ADC reads an

analog input at the same value as V_{ref}, it will give out the highest bit-value possible (same as bit-resolution). For a 12-bit ADC the highest bit-value would be 4095. The measurement resolution with an ADC could be crucial in some systems, to find the resolution, Equation 4 can be used [14, p 351][21].

$$Measurement\ resolution = \frac{Measurement\ range}{ADC\ resolution} \tag{4}$$

If we would use a 12-bit ADC and have a measuring range at for example 200mm, the best possible measurement resolution is:

$$\frac{200\ mm}{2^{12}} = 0.0488 \frac{mm}{increment}$$

Another thing to consider when choosing ADC is how big each step should be. If a step is how much the voltage would change for the bit-value to increase by one bit [14], [22].

$$\Delta V = \frac{V_{ref}}{2^N} \qquad \text{Where N is the bit resolution.}$$

4 Method

The way the group have been working was deeply influenced by the working structure of the employer Shift Hyperloop. The year is divided into three phases; concept, design and system verification. The Concept Phase is the period from the beginning of the main project until Concept Review in October, where all systems presents their ideas to the rest of the organization. Before presenting their system to the organization everyone has to get to know their system. In this phase you go through everything done last year, and sort out what worked and what did not. Following this, the group needs to do research on everything so that all changes can be argued.

Following the Concept Review was the design phase. This phase can be divided into multiple sections. The designing of the first circuit board revision and testing is separated by design review, organized as a workshop where other subgroups are criticizing and questioning your design in the purpose of limiting mistakes. In late January this first revision is soldered and tested in order to find errors and fix them in the second revision. After design phase the main project will step into a system verification phase, but this phase is not a part of this thesis as the phase begins after the thesis deadline.

In addition to the other phases we have worked with code throughout the whole spring semester, beginning with pseudo code and then writing and testing code with development boards. At this point we split up and each group member got responsibility for a main task: thesis, code and hardware.

The group have additionally tested each of the sensors separately, without the custom PCBs. Both method and results for these tests are explained in section 6.

As we are a part of Shift Hyperloop and have been following their structure, the timing of certain tasks have been a little later than wanted. An example of this is the soldering and testing of revision 2, which began in the beginning of May.

4.1 The Concept Phase

The Concept Phase began with deciding what the goal of our system is, what the system needed, and how we could fulfill those needs. The goal was decided to measure all parameters needed by other systems, such as temperature, pressure, movement, speed and distance to track. This was decided in order to verify the safety of the pod and to accommodate the rules set by EHW. Besides this we also decided that we wanted to add some sensors for analysis, and provide useful data for next years pod.

Following this we looked at what other teams had done in order to get an idea of what had previously been approved by the SpaceX competition. We also looked into what was done the previous year by Shift Hyperloop and discussed with the responsible members. We did this so we could get a starting point for the Concept Phase, but as they did not have the opportunity to create a product last year we mostly discussed what was planned. After we had read up on other teams and talked with last years team members, we decided to put everything away and begin from scratch.

Getting a complete list of all necessary parameters to measure proved difficult, as most of the other systems around the pod did not know themselves. The original list of what should be measured was: position indicators, temperature brakes, temperature inverter, temperature battery, temperature motor, position of brakes, distance to track, compression of suspension, acceleration, speed, position, vibration, hydraulic pressure, obstacles/upcoming turn.

We were provided with a few sensors from Shift Hyperloop at the beginning of the project, and wanted to implement them in our system. The sensors already provided was an optical distance sensor, one IMU and three photoelectric sensors. In addition to these, we needed to decide the

remaining sensors based on the criteria given to us from other systems. When all sensors were decided, we found it made sense to create two sensor suites PCBs, where we could distribute the different sensors based on their position on the pod.

4.2 The Design Phase

Following the Concept Review was the Design Phase. This is where concepts and ideas would be realized as tangible plans and designs. During this phase the project became even more interdisciplinary, as multiple designs developed in parallel. Mechanical restrictions in regards to load carrying capabilities and available space suddenly affected choices for the Electronics Group. Ideas of using certain software or measuring any given parameter had to be put into action and manifested in plans that could be simulated, produced and tested for functionality and compatibility with other systems. Competition rules are scrutinized even deeper to make sure all systems would be approved by the jury when the Final Design Documentation (FDD) report was due the 14th of March.

4.2.1 Designing PCB revision 1

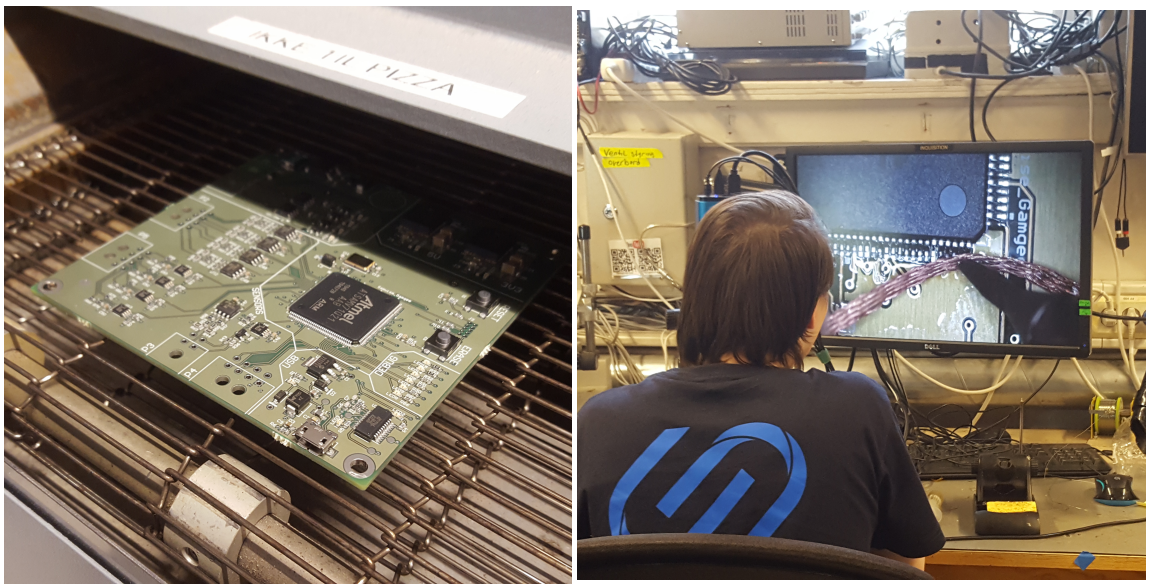
PCB design was done using Altium Designer. This software was provided by Shift and used by all groups. PCB design began with drawing circuit schematics, for which our revision 1 only included one interface circuit, connector headers and signal inputs to the Atmel SAMV71Q21 microcontroller unit (MCU). Other necessary schematics had been made by the Master Library system and were common for all systems in the Electronics Group. This included schematics for input protection, power supply units (PSUs), CAN transceivers, USB to universal asynchronous receiver-transmitter (UART) interface, and various headers and debug LEDs. Schematics for revision 1 are attached in Appendix A and B, but does not include Master Library schematics, as we had no say in the design of these.

The schematics were then imported to the PCB layout file. Instructions had been given by Shift on board dimensions and other design requirements, and a set of design rules were imported into our Altium projects to control the layout design. These specifications are described in detail in subsection 5.3.

All systems under the Electronics Group had the same deadline for submitting designs for revision 1. This deadline was set to one week after Design Review, so there was time to make changes based on feedback. Once final design files were reviewed by management and production files were generated from Altium, they were sent to a main project sponsor, NCAB, for production.

4.2.2 Production and testing PCB revision 1

Our PCB design was influenced by the opportunity to use the re-flow solder oven available at Omega Verkstedet. This allowed for use of smaller components and tighter placement with little to no risk of damage to components or the PCB itself. Hand soldering small components in tight spaces can be risky when done by unskilled students. Before soldering SMD (surface mounted device) components we used a microscope, pincers and a pneumatic solder paste dispenser to apply solder paste to pads and place components. Every component on the top layer that was compliant with re-flow soldering was mounted this way. Pictures of this process can be seen in Figure 11. Components on the bottom layer and through-hole components were soldered manually after the re-flow process was completed. Only one of the sensor suites, Suite 2, was completed this way for revision 1, as campus shut down due to national Corona-guidelines before we had the opportunity to solder both suites.



(a) PCB with components placed on solder paste, about to go into the re-flow oven at Omega Verksted.

(b) After re-flow soldering, excess solder paste was manually removed for some components.

Figure 11: Pictures showing two stages of the soldering process. These pictures show a revision 2 PCB, but the process was the same as for revision 1.

A general test plan for PCBs had been made by the project group and the VCU system. This plan is attached in Appendix H. Before doing any soldering we probed each PCB for continuity, making sure there were no short circuits from the production. This way we could rule out such errors from later stages of testing. After soldering, the same continuity tests were performed again.

When proved free of faults we powered up the board. Not via the main input, but at header pins placed in each of the two power planes 3.3V and 5V. Mistakes in the PSU layouts had been discovered and to not risk damage to the components on the board it was decided to bypass them. Voltage levels were measured for inputs on the MCU and other ICs on the board. Another error was discovered at the CAN transceivers, so we removed them from the PCB so they would not disrupt other circuits. Other voltage levels measured ok. At this stage we did not have any code ready for the Atsam MCU, so we modified an example code for an Xplained Ultra dev-kit and uploaded it to the MCU with an Atmel-ICE debugger. We managed to flash toggling of the debug LEDs.

Any further testing was limited to the transimpedance interface circuits. This was done by connecting a power supply with current limiting to the input and measuring the output of the operation amplifier (OP-AMP). The circuit behaved as in simulation.

4.2.3 Simulating interface circuits

To make sure we had the best possible chance of success we decided to simulate our analog signal interface circuits before implementing them into our PCB design. For revision 1 we only simulated one type of transimpedance amplifier.

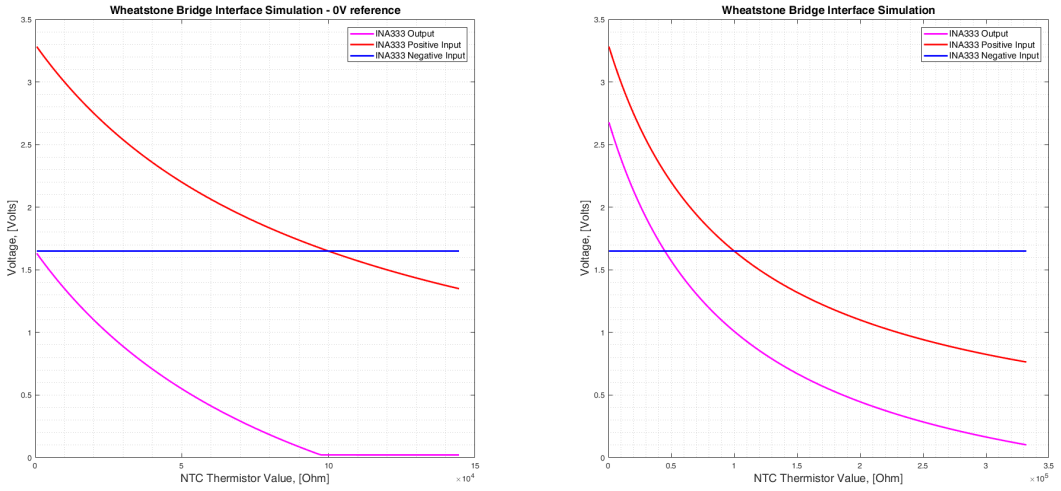
The target values we sat for each circuit are found in Table 1. They are based on requirements from the subsystems each sensor were to monitor. Component values and simulation settings for each circuit were calculated and chosen from these criteria. Details are described in subsection 5.2. LTspice schematics showing the circuits and settings are found in Appendix I.

Sensor	Measuring range	Unit	Sensor output	Unit	IC output	Unit
Hydraulics pressure	0-400	bar	4-20	mA	0,66-3,3	Volt
Distance	100-300	mm	4-20	mA	0,66-3,3	Volt
Thermocouple	0-200	°C	4-20	mA	0,66-3,3	Volt
Thermistor	0-200	°C	552,99-332,4k	Ohm	0-3,3	Volt
Photoelectric	Low/High	-	0 / 24	Volt	0 / 3,3	Volt

Table 1: Interface circuit requirements for successful simulation.

Results for the simulations are presented in Figure 12 and 13. With the exception of the Wheatstone circuit, only simulations for final design is presented. The process did include testing different integrated circuits, such as non rail-to-rail OP-AMPs or different instrumentation amplifiers (IN-AMPs). During this trial and error we gained more knowledge of which specifications to look for in component data sheets and how each specification would affect simulation outcome.

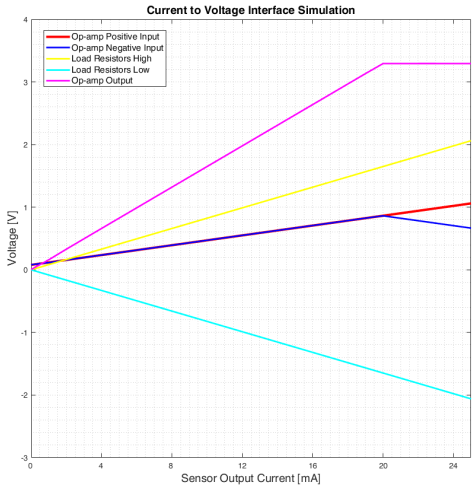
As can be seen in Figure 12a the instrumentation amplifier, the INA333, we chose for the Wheatstone interface could not handle the full range of the NTC thermistor without modification. In subsubsection 5.2.3 we describe the details of how this was solved to get the result seen in Figure 12b.



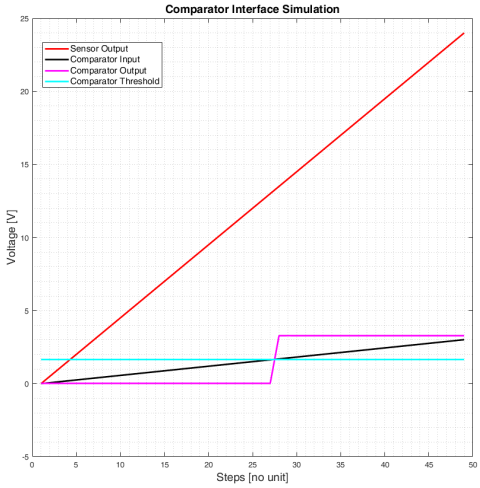
(a) First simulation without reference voltage to INA333.

(b) Simulation with added reference voltage to INA333.

Figure 12: Difference in Wheatstone interface circuit behavior without (a) and with (b) an offsetting reference voltage.



(a) Simulated characteristics for the current loop to voltage conversion circuit. Voltage over load resistors vs differential input to op-amp vs output signal.



(b) Simulated behavior for the comparator circuit. Output goes high when input reaches set threshold.

Figure 13: Simulation results for the current to voltage converter (a) and the comparator interface circuit (b).

4.2.4 Designing PCB revision 2

The PCB design for revision 2 began with making Altium schematics for new interface circuitry. The designs were transferred from the simulated circuits. This meant updated circuit design for the current to voltage converter and added circuits for comparators and instrumentation amplifiers. New integrated circuits also meant adding new components to the Altium library. Footprints and schematic symbols were custom made by us for the TLV7012 comparator and the INA333 instrumentation amplifier. Complete schematics for revision 2 can be found in Appendix C and D. Schematics for Master Library circuits are again not included.

Next we sketched up suggestions for the PCB layouts. This was done to visualize the schematic blocks in relation to each other and see how signal paths and available space were affected. We had not done this for revision 1, which led to an untidy layout. With the experience we had gained from this we were able to streamline layouts and utilize the space on the boards more effectively. Simultaneously it was possible to focus more on good practices when it came to signal paths and noise considerations.

Additionally, the group sat down and wrote a failure mode, effects, and criticality analysis (FMECA). The FMECA which can be found in Appendix J was created to identify and eliminate potential problems, and was a requirement to include in the FDD.

4.2.5 Production and testing PCB revision 2

After soldering all component onto the PCBs, the first thing that was done was to probe the entire PCB without connecting it to power, in order to check for shorts.

Signal	Suite 1	Suite 2
GND	ok	ok
5V	ok	ok
3.3V	ok	ok
VDDcore	ok	ok
Signal paths	ok	ok

Table 2: Checklist for testing continuity without power.

We probed ground, 5V, 3.3V, VDDcore and some sensor signal paths as seen in Table 2. These were probed to check if the signals were connected to the places they are being used, and to check that none of these have shorted to other signals. After having successfully probed the different signals it was time to power the two PSUs to check if they were functioning properly. First the 3.3V PSU was connected to power, and then we moved on to check if the places that require 3.3V actually received said voltage. After having checked the 3.3V PSU, we then performed the same check with the 5V PSU. While powering the 5V PSU then the LED of the 3.3V PSU also lit up. Powering one of the PSUs should not power the other, both of them should only be powered simultaneously when the board is powered through the input protection. After some inspection we figured out that two resistors have been soldered on vertically instead of horizontally by the 3.3V PSU.

When everything was being powered properly, next step was to try flashing code to the microcontroller. This was done by flashing a code that would test whether the LEDs used for debugging were functioning or not. After flashing and running a simple code that should toggle the pins connected to the LEDs on and off, we could quickly see that they were not functioning. While using the code from Figure 14, neither of the LEDs turned on, and through discussion we quickly figured out that it was caused by a hardware problem. The debug LEDs on both PCBs were soldered in the wrong direction. This was corrected and the test performed successfully.

```
while(1) {
    ioport_toggle_pin_level(LED_A);
    ioport_toggle_pin_level(LED_B);
    ioport_toggle_pin_level(LED_C);
    ioport_toggle_pin_level(LED_D);
    ioport_toggle_pin_level(LED_E);
    ioport_toggle_pin_level(LED_F);
    delay_ms(1000);
}
```

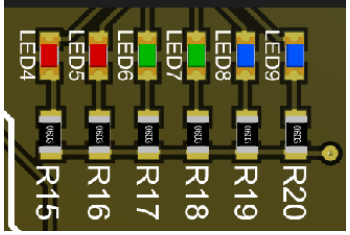


Figure 14: Code and PCB layout for the debug LEDs

For testing the different circuits containing an OP-AMP, we tested only one circuit because the rest are exact copies and should behave similarly. We ended up using the circuit connected to the 4-20mA signal from the distance sensor for testing. The circuit can be seen in Figure 15 below. By measuring the output from the OP-AMP, and comparing it to the simulated output, made us able to determine if the circuit was functioning properly. After having checked the functionality of the circuit we would need to test it with the ADC and a terminal monitor, to compare ADC output values against expected values.

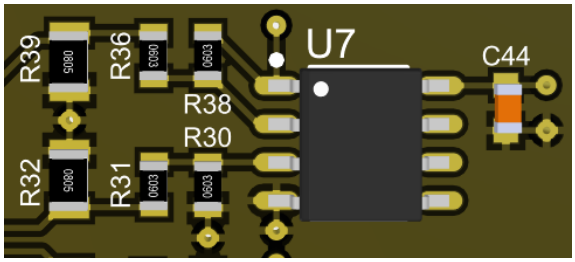


Figure 15: PCB layout of the OP-AMP circuit

Same methodology was used when performing the tests with the instrumentation amplifier circuits.

Every IN-AMP circuit is an interface circuit for thermistors, therefore testing was performed with a thermistor connected. By measuring the resistance of the thermistor and having simulated the circuit, we could then predict the output voltages of the IN-AMP. However, the output voltages we measured did not resemble the simulated output voltages. During troubleshooting and performing measurements, we also noticed that the presumed offset reference voltage pin of the IN-AMP had 2.5V connected to it. This voltage did not make sense since the voltage divider should have supplied the offset reference pin with 1.03V. From further inspection it became clear that the problems were caused because of an error with the footprint in the Altium schematic library for the IN-AMP.

Since a photoelectric sensor is used in combination with the comparator circuit, and it outputs either 24V or 0V, we decided to just use a power supply to replicate the sensor output. By measuring the voltage being supplied to the non-inverting input of the comparator, we could determine if the preceding voltage divider was functioning as assumed. After having checked the supplied voltage, we could move on to measuring the voltage output from the comparator in order to confirm proper behavior. After having tested the interface circuit, we could then test the circuit in combination with the microcontroller.

When it came to testing the Suite XS PCB we started off by checking if any of the copper pads had been short circuited during soldering. When having checked for shorts we moved on to checking whether the data pin for both sensors were connected to the data line exiting the PCB. Same check was performed with the clock pin respectively. For the next steps we used two different development kits, the SAM V71 Xplained Ultra for the microcontroller, and Würth Elektronik Sensor FeatherWing for the sensors. The reason for using development kits is to minimize our sources of error, and because of uncertainty surrounding communication with Two-Wire Interface. First step was to make sure communication was established with the sensors. This was done by separately reading the device ID register of both sensors. The second step was to try writing to the control registers of the sensor, in an attempt to configure the sensors. The last step in the testing process was to execute code that would read sensor data, by reading the temperature from the WSEN-ITDS sensor for example.

4.3 Writing code

The choice of which Integrated Development Environment (IDE) to use was decided by our employer. Since we were using a microcontroller made by Microchip Technology and had access to several Atmel-ICE then the choice became Microchip's own Microchip Studio. Microchip Studio in combination with their physical Atmel-ICE debugger and programmer, allows us to write, flash, and debug our code with one program. Using Microchip Studio also gives us access to their own Advanced Software Framework (ASF) which includes libraries and drivers that aids in development of code for SAM based microcontrollers [23]. Microchip studio has an integrated terminal emulator, however, based on personal preference we instead chose to use Tera Term. The terminal emulator would be used for printing useful messages during debugging of code. An example would be when having to debug the ADC, a terminal monitor then becomes useful for printing ADC values in real time.

4.3.1 Converting ADC values to desired parameter

In order for the microcontroller on the custom PCB to process the data from the distance sensor it has to be connected to an ADC. The ADC converts voltage to binary values that we will use in our code.

First we need to find the ADC value at 3.3V, that corresponds with the range we will measure. In

order to do this we can use Equation 5:

$$\begin{aligned} x &= \frac{ADC\ resolution}{3.3V} * 3.3V \\ x &= \frac{4095}{3.3V} * 3.3V \\ x &= 4095 \end{aligned} \tag{5}$$

Then we will find the ADC value for 0.66V.

$$\begin{aligned} x &= \frac{ADC\ resolution}{3.3V} * 0.66V \\ x &= \frac{4095}{3.3V} * 0.66V \\ x &= 819 \end{aligned} \tag{6}$$

If the sensor is linear we can from here determinate the linearity/function for the relation between ADC-values and the measurement parameter.

If the sensor is linear, it is known that
20mA → 3.3V (current to voltage circuit) → 4095 (ADC) → max range value.
4mA → 0.66V (current to voltage circuit) → 819 (ADC) → min range value.

Find the linearity/function for the relation between ADC-values and distance.

$$y = ax + b \tag{7}$$

$$a = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_{max} - y_{min}}{range_{max} - range_{min}} = \frac{4095 - 819}{range_{max} - range_{min}} \tag{8}$$

$$4095 = a * range_{max} + b \quad \rightarrow \quad b = 4095 - a * range_{max} \tag{9}$$

Optical distance sensor (OMT300), current to distance First we follow the steps in Equation 5 and Equation 6, we now have the ADC-values that correspond with the measurement range for the sensor. In this case 10-30cm. We know that
20mA → 3.3V → 4095 (ADC) → 30cm.
4mA → 0.66V → 819 (ADC) → 10cm.
From there we use Equation 7, Equation 8, Equation 9 to determinate the linear function for the sensor, from ADC-values to distance (in cm). It is important to remember that $range_{max}$ is 30cm and $range_{min}$ is 10 cm for the optical distance sensor.

After following these steps we end up with

$$\begin{aligned} y &= ax + b \\ a &= \frac{4095 - 819}{30 - 10} = 163.8 \\ b &= 4095 - (163.8 * range_{max}) = -819 \\ Distance\ in\ cm &= \frac{ADC\ out + 819}{163.8} \end{aligned}$$

As we want to measure small distances we converts this function into distance in mm (this is done by multiplying with 10).

$$Distance\ in\ mm = \frac{ADC\ out + 819}{16.38}$$

Thermocouple, current to degrees in Celsius We use the same steps as distance, but here $range_{max} = 200°C$ and $range_{min} = -75°C$. From Equation 5 and Equation 6, we get the ADC-values that matches the measurement range for the sensor. We know that
20mA $\rightarrow$ 3.3V $\rightarrow$ 4095 (ADC) $\rightarrow$ 200 degrees.
4mA $\rightarrow$ 0.66V $\rightarrow$ 819 (ADC) $\rightarrow$ -75 degrees.
From there we use Equation 7, Equation 8, Equation 9 to determinate the linear function for the sensor, from ADC-values to temperature.

$$y = ax + b$$
$$a = \frac{4095 - 819}{250 - (-75)} = 10.8$$
$$b = 4095 - 10.8 * 250 = 1395$$
$$Temperature\ in\ °Celsius = \frac{ADC\ out - 1395}{10.8}$$

Pressure, current to bar We use the same method as distance and thermocouples. We know that $range_{max}$ is 400 bar, and that is the value at 3.3V (4095 bit from ADC). Additionally we know that $range_{min}$ is 0 bar, and that the value is 0.66V (or 819 bit from ADC).

Find the linearity/function for the relation between ADC-values and temperature.

$$y = ax + b$$
$$a = \frac{4095 - 819}{400 - 0} = 8.19$$
$$b = 4095 - 8.19 * 400 = 819$$
$$Temperature\ in\ °Celsius = \frac{ADC\ out - 819}{8.19}$$

Thermistor, voltage to temperature Math for thermistors is a bit trickier. We will start with finding the voltages over the Wheatstone bridge, $V_{out} = V_1 - V_2$. In the Wheatstone bridge the left side (V_2) is constant, $V_2 = 1.65V$. The output voltage is then $V_{out} = V_1 - 1.65V$.

Looking at the right side of the bridge, the thermistor resistance could be found by using voltage divider.

$$V_1 = 3.3V * \frac{100k\Omega}{100k\Omega + R_T}$$
$$\frac{V_1}{3.3V} = \frac{100k\Omega}{100k\Omega + R_T}$$

$$R_T = \frac{3.3V * 100k\Omega}{V1} - 100k\Omega$$
$$R_T = \frac{3.3V * 100k\Omega}{V_0 + 1.65V} - 100k\Omega \tag{10}$$

Add output voltage from Wheatstone bridge and offset voltage from circuit.

$$V_{ATSAM} = V_0 + 0.99$$

Insert V_{ATSAM} in R_T function (Equation 10)

$$R_T = \frac{3.3V * 100k\Omega}{V_{ATSAM} + (1.65V - 0.99V} - 100k\Omega = \frac{3.3 * 100k\Omega}{V_{ATSAM} + (0.66V} - 100k\Omega \tag{11}$$

Converts voltage into bit-value as that is what is the output from the ADC

$$V_{ATSAM} = \frac{ADCoutput * SystemVoltage}{ResolutionADC} \tag{12}$$

Replaces V_{ATSAM} from Equation 11 with Equation 12

$$R_T = \frac{330k}{\frac{ADC_{out}*3.3}{4095} + 0.66} - 100k\Omega \tag{13}$$

Convert from resistance to temperature with the beta parameter equation

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{\beta} * ln(\frac{R_T}{R_0}) \tag{14}$$

Where $R_0 = 100k\Omega$, $T_0 = 25°C = 297.15K$ and $\beta = 4092$. R_T is Equation 13

$$T = \frac{1}{\frac{1}{T_0} + \frac{1}{\beta} * ln(\frac{R_T}{100k\Omega}} - 273.15 \tag{15}$$

5 Design

In this section we will present the final decisions for sensor selections, hardware design and code functionality. PCB design for revision 1 will briefly be described, as this lay the foundation for revision 2. Overall design choices and design specifications will be described in detail with relevant calculations and reasoning. First we will present final choices for sensors and their respective function and placement on the pod. Next is a detailed description of our specific interface circuits. Both revisions of the sensor suite PCBs will be presented next, including the Suite XS PCB. Lastly a description of how we structured our code and what functionality it contains.

5.1 Sensor selection and placement

As the pod need a lot of data to function and run safely, there is a great need for different sensors. When choosing each sensor we needed to take into account the most important characteristics of the parameters we would measure, such as measurement range, frequency and accuracy. A rough overview of sensor placement can be found in Appendix K.

Different types of sensors were considered for the various purposes, but were ruled out based on characteristics needed for measurement, complexity of implementation, size and cost.

5.1.1 Photoelectric sensors

The pod will contain three photoelectric sensors that shall be used to register some navigational markers placed along the track, provided by EHW. The navigational markers are made of red reflective tape and are placed every 15m [24]. The photoelectric sensor provided by the employer is the *X-PRO XP10R5* from Tri-Tronics.

Two of the photoelectic sensors will be placed in the front, one on each side. The last sensor will be placed in the rear on the right hand side. The two sensors on the right side of the pod will be used to estimate speed by dividing the known distance between them to the time it takes between registration. The sensor on the left side will register markers in order to verify that all markers are registered by checking if front left and front right has the same marker count. The photoelectric sensor will be used to determinate the distance travelled as the navigation markers are placed every 15m.

5.1.2 Optical distance sensors

There will be placed two optical distance sensors on the pods left hand side, one in the front and one in the rear. The sensor model is OMT300, and is from *Pepperl+Fuchs*. The reason for these two sensors were to make sure that the motor did not come within the keep-out zone, a zone determined by EHW. As seen on Figure 16, the keep-out zone is located on the top and bottom of the track [24]. If the distance from the motor to the track was less than a safe distance, the pod have to stop.

Due to the physical barriers the pod has, these measurements will this year be used for the most part for analysis.

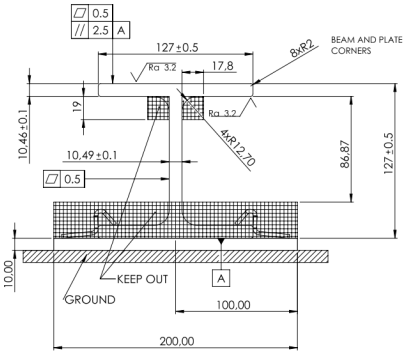


Figure 16: Track specifications provided by EHW

5.1.3 Thermistors

Two thermistors will monitor the temperature of the brake calipers/pads. There will be one thermistor placed on each caliper as close to the pad as possible. These thermistors will monitor any excessive increase in temperature during braking, and will primarily be used for after run analysis.

The temperature of the two inverters will be monitored with thermistors. Four thermistors will be placed inside the cooling blocks (two in each) and two thermistors will measure ambient air temperature inside the casings (one in each).

5.1.4 Thermocouples

The motor temperature will be measured using a total of four thermocouples type T. Each thermocouple will be connected to a dedicated transmitter which converts the induced thermocouple voltage to a 4-20mA output. The transmitter that will be used is a programmable transmitter from RS Pro (type: 192-6538). This transmitter has a galvanic isolation rating input-output of 1500V AC, which will keep the sensor PCBs from being electrically in touch with the motors through the thermocouples. The motor temperature will be used as part of the safety protocol for the pod so that if the temperature reaches a specified limit, the melting point for the epoxy surrounding the motor, then the pod will stop.

5.1.5 Hydraulic pressure sensors

To measure the pressure in the hydraulic brake systems there will be one pressure sensor in each of the two separate systems, both front and rear. The sensor chosen was HDA4446 from Hydac, as they produced the brake system for the pod. The sensors will be used to detect possible leakage from the fluid accumulators and whether the valves engaging the brakes are opened. This sensor is part of the safety protocol for the pod.

5.1.6 Accelerometers

Four accelerometers will be placed on the pod, close to each corner. These accelerometers, WSEN-ITDS accelerometers from Würth Elektronik, are surface mounted on custom designed PCBs (Suite XS). The four accelerometers are meant to help analyze how the pod is moving along the track, and will not be used for anything vital during a run. We will be using Two-Wire Interface, also known as Inter-Integrated Circuit (I2C), as our method of communicating with the sensors on the Suite XS PCB.

5.1.7 Ambient pressure sensors

The ambient pressure of the pod will at all times be measured for analysis purposes. We will use the WSEN-PADS pressure sensors from Würth Elektronik, and these are placed on Suite XS alongside the accelerometers. This sensor will also measure ambient temperature with an integrated temperature sensor. The data from this sensor will not affect anything while running.

5.2 Interface Circuits

Our MCU have specific requirements when it comes to input signals to the analog pins. It can only read signals up to 3.3V, and cannot handle current loops directly. We have therefore designed

necessary interface circuits for some of our sensors in order to supply the MCU with a compatible input format.

In the industry there are often ready-made solutions available as interfaces for any sensor need a manufacturer or customer might need. These are usually stationary and intended for factories, laboratories or other applications where they can be permanently installed without much regard for size or mobility.

For our Hyperloop pod application we therefore needed to build our own interface circuits to condition the signals from each of our various types of sensors. Recommendations for such circuits can be found, but each circuit have to be tailored to each specific application.

5.2.1 Comparator circuit

Our photoelectric sensors, the X-PRO XP10RR5 have an output configuration that is either 0V or 24V. The output is nominal at 0V, but when the sensor sees a reflective marker, the output goes high to 24V. None of the input pins on the SAMV71 chip can handle 24V, so we designed a comparator circuit, involving a voltage divider, to convert the 24V sensor output down to a 3.3V signal. A comparator compares two voltages, input and threshold. When the input voltage exceeds the threshold, the output will go from 0V to the comparator supply voltage. [25, p. 90]

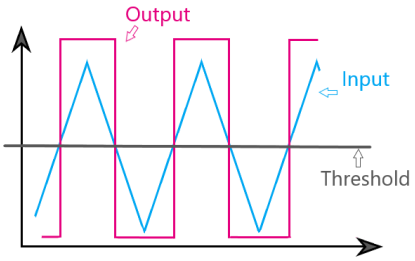


Figure 17: Comparator functionality graph. When the input level exceeds a threshold, the ouput switches to high.

We use the TLV7012 chip from Texas Instruments and supply it with 3.3V. The maximum input voltage for this chip is 6.5V so we implemented a voltage divider and a 3.3V Zener diode to protect the chip from the 24V sensor output. The divider values were decided so the input voltage to the comparator would be 3V, just below the Zener voltage. 24V divided by 3V is eight, so a 1:7 ratio was required. From standard E96 resistor values we tried 10k and 69.8k in the formula for voltage dividers:

$$V_{div} = 24V * \frac{10k}{10k + 69.8k} = 3.0V \tag{16}$$

We used two 10k resistors in a voltage divider for a threshold voltage of 1.65V. The complete circuit as used in revision 2 can be seen in Appendix C and D.

5.2.2 Current to voltage circuit

Many of our chosen sensors have a standard 4-20mA analog output signal. We chose this where we had the option, as this is widely recognized as being more robust to noise interference than traditional voltage outputs. Adapting this signal to our microcontroller's ADC input specifications required another interface circuit.

The design for this, as seen in Figure 18, was taken from an application note by Dialog Semiconductor [26] and adapted for our desired volt-

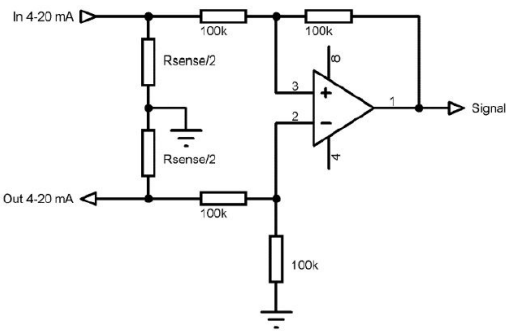


Figure 18: Interface circuit for a 4-20mA loop

age levels. Our final design for revision 2 can be seen in Appendix C and D.

Load resistors R_{load} are placed in series with the 4-20mA loop. The measured voltage across these becomes the input to a differential op-amp in unity-gain configuration. The value of the load resistors was calculated using Ohms Law as seen in Equation 18, based on the maximum desired voltage level and the maximum current in the loop.

$$R_{load} = \frac{V_{max}}{I_{max}} = \frac{3.3V}{20mA} = 165ohm \tag{17}$$

The result was divided by two due to the circuit design with two load resistors, and this matched the E96 resistor standard of 82.5 Ohm. Now we could use Ohms Law again to find the minimum voltage for the input, based on the load resistors and expected minimum current:

$$V_{min} = R_{load} * I_{max} = 165ohm * 4mA = 0.66V \tag{18}$$

Another advantage of the 4-20mA loop is the live zero value of 4mA, which represent zero percent of the measured value [27]. If the current in the loop disappears or falls below 4mA the connected ADC input can be programmed to detect voltage levels below the calculated 0.66V and trigger an error message.

The OP-AMP, Texas Instruments LME4972MA, was already present in the Shift Altium libraries. After reviewing the electrical properties we found it would fit our needs. It is both capable of handling the low differential voltage of 0.66V and it is rail-to-rail single supply powered capable of a 3.3V supply. Rail-to-rail powered means the output port can be pulled as low as 0V and as high as 3.3V, giving the op-amp a greater dynamic output range than non rail-to-rail designs [28].

5.2.3 Wheatstone-bridge

To find the relationship between output voltage from a variable resistor and the resistance of the variable resistor a Wheatstone bridge is often used. A deflection-type DC bridge is a variant of Wheatstone bridge which uses three resistors and one variable resistor, R_T. See Figure 19. When R_T changes, the balance of the right side of the bridge is shifted, so the differential output voltage V_O will change too. The relationship between R_T and V_O is:

$$V_O = V_I(\frac{R_1}{R_1 + R_2} - \frac{R_T}{R_T + R_3}) \tag{19}$$

This bridge configuration also compensates for minor changes or ripple in the supply voltage V_I, as both sides of the bridge would vary equally, and the differential gap between them remains the same.

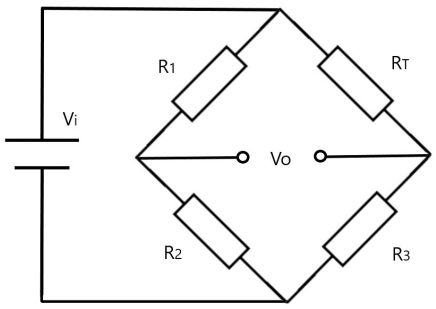


Figure 19: The basics of a deflection-type DC Wheatstone bridge

For our NTC thermistor sensor circuits, this is the type of interface we will use. R_T represents the thermistor, a 100k ohm NTC from TDK, and V_O will be connected to an INA333 instrumentation amplifier from Texas Instruments. The INA333 will be used in unity gain mode. It can handle the 3.3V single supply on our PCB and is rail-to-rail which gives full range on the output port. The INA333 can also be offset, which was necessary due to the wide resistance span of the thermistor over the required measuring range of 0-200°C. In ohms, this translates to a span from 332k down to 553 ohms, respectively. Without an offset voltage the circuit could not handle resistances

>100k, due to the differential voltage wanting to draw the INA333 output <0V. See the section on simulations for further details. The necessary offset voltage was found by inserting the maximum R_T value into Equation 20.

$$V_O = 3.3V(\frac{100k}{100k + 100k} - \frac{332k}{332k + 100k}) = -0.886V \tag{20}$$

Since we had more headroom in the upper voltage range, we made the voltage divider from a 10k and a 22k resistor to get a reference voltage of approximately 1.0V:

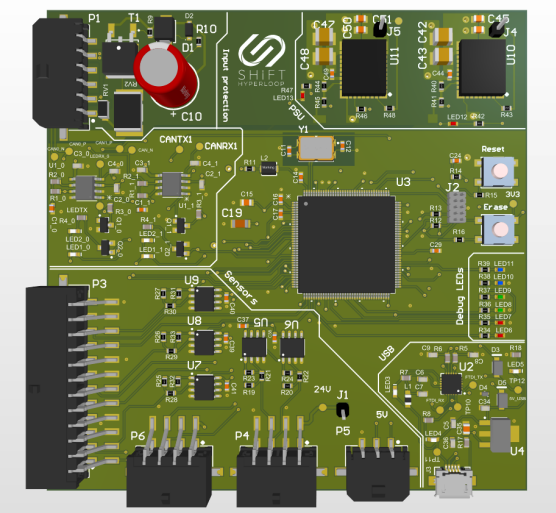
$$V_{div} = 3,3V * \frac{10k}{10k + 22k} = 1,03V \tag{21}$$

5.3 PCB Revision 1

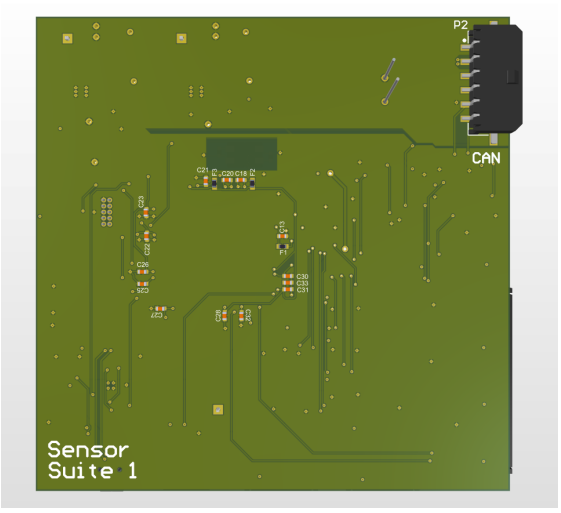
The PCB design for revision 1 was meant to function as a test for both interface circuitry and space on the board itself. We had specific dimensions to meet, as well as some restrictions to which sides we could put connector headers. These restrictions came from the intended common electronics box, an aluminum box to house all the PCBs made by the electronics group. These dimensions were a 10x10cm board, in which only two sides were available for connector headers. The board needed to have four mounting holes, one in each corner. Other design restrictions were given by the manufacturing capabilities of our PCB supplier, NCAB. Such restrictions include, but are not limited to; trace width and geometry, solder mask gap distance, via hole size, silk layer print sizes. All PCBs were four layers, for which bottom and top were intended for signals and the middle layers for power and ground. The remaining area of both top and bottom layers were poured with copper to act as additional ground planes. Between signal traces and components there can be stray capacitance which causes unwanted loops. Noise can generate because of such loops, but with ground planes covering the area between components and traces, the capacitance is lost in the low impedance path to ground.

Non-sensor specific circuitry were designed by other sub systems within the Electronics group. Input protection, PSUs, surrounding components for SAMV71, CAN tranceivers and the USB interface were all designed by other members and copied into each sub systems specific Altium projects. This was restricted to Altium schematics and component values. With the exception of the PSUs, we had to decide on the PCB layout ourselves for these circuits also.

PCB Revision 1 had an internal design freeze before most sub systems had decided on parameters they wanted measured. This meant that we only had one interface circuit we knew we needed at this point in time - the 4-20mA current loop converter. The schematics for this can be seen in Appendix A and B. It was based on a transimpedance amplifier, as this was the design we had been able to verify through simulation at this time.

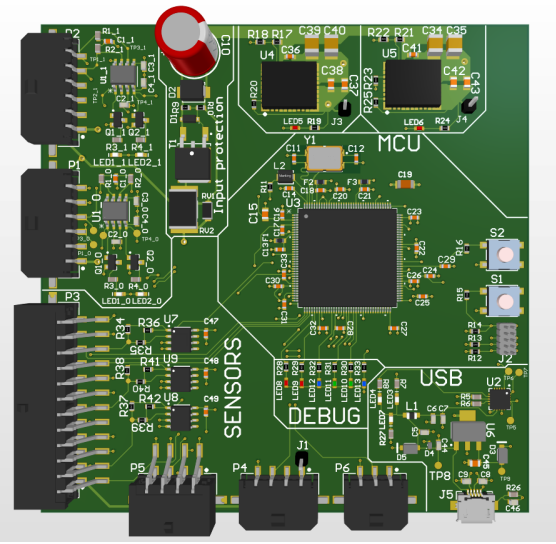


(a) 3D top view

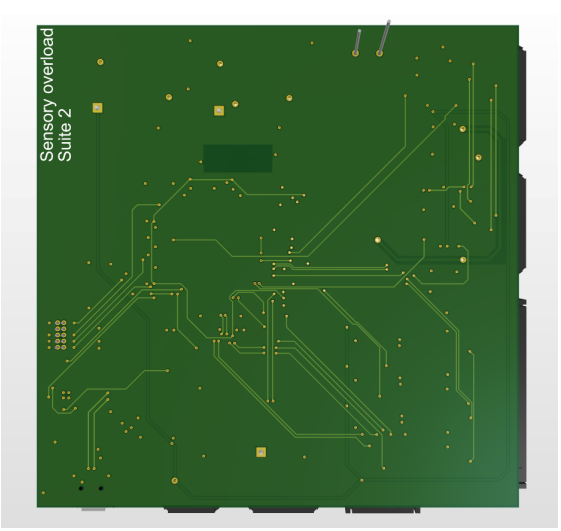


(b) 3D bottom view

Figure 20: 3D models of Sensor Suite 1, revision 1



(a) 3D top view



(b) 3D bottom view

Figure 21: 3D models of Sensor Suite 2, revision 1

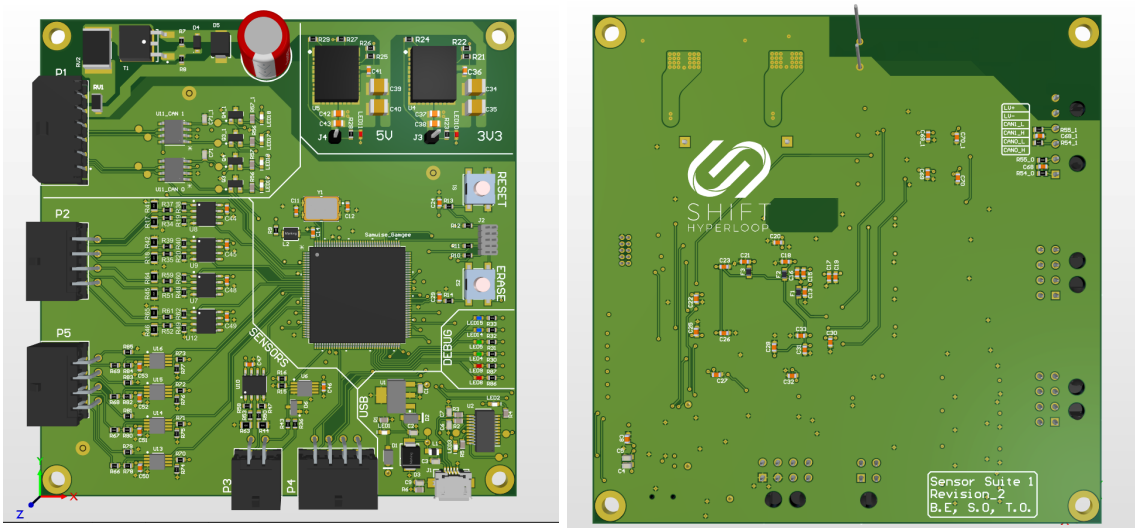
5.4 PCB Revision 2

For revision 2 we had been able to simulate several interface circuits beforehand, and we knew exactly which sensors we needed to spec them to. Unfortunately we did not have much useful test results from revision 1, instead we focused on streamlining the layout and keeping certain recommended practices more in mind when designing revision 2.

Important changes in revision 2 include:

- Streamlining several circuits, especially CAN and input protection. This to keep the signal moving primarily in one direction.

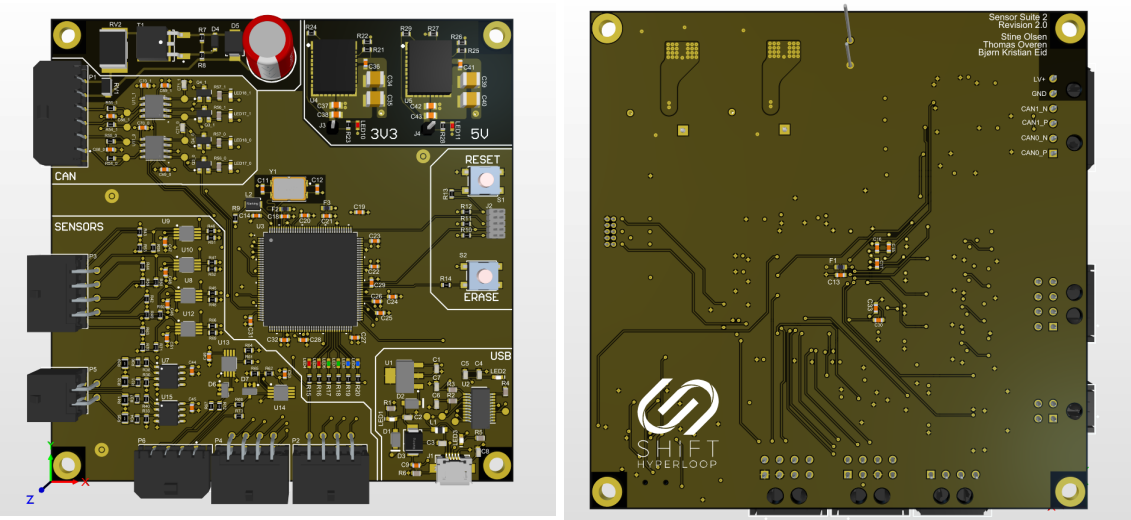
- Adapted SAMV71 input pin locations in relation to interface circuits and sensor inputs on connector headers. This eliminated signal paths crossing each other and meant we could for the most part route all signals on the top layer and close to the ground plane.
- Less broken up signal traces due to crossings meant overall cleaner signal paths with less interruptions in regards to current return paths.
- Fixed sharp angles on some traces to avoid chemical build-up or mechanical stress.
- CAN tranceivers on each PCB was changed to endpoints of stubs. This meant we could remove one connector header as there was no need to pass the CAN signal and power lines on to the next PCB. This was a joint decision for all sub groups of Electronics.
- Rotated PSUs on both Suites to get better access to the output.
- Changed places of PSUs (5V and 3.3V) on Suite 1 to better fit in the power plane.
- Instead of supplying the ICs (op-amps, instrumentation amplifiers and comparators) in interface circuitry with 5V, we chose variants with rail-to-rail functionality and supplied them with 3.3V. This gave us less theoretical range on the outputs, but meant we did not need to downscale 5V to 3.3V for the SAMV71 ADC inputs (their absolute max input voltage is 3.6V [29]. Instead we could use the full range of 3.3V and not have to add extra circuits to protect the ADC inputs.
- Removed everything relating to 24V from the PCBs, as this would be supplied from an external power supply. This was an executive decision, not ours.



(a) 3D top view

(b) 3D bottom view

Figure 22: 3D models of sensor Suite 1, revision 2



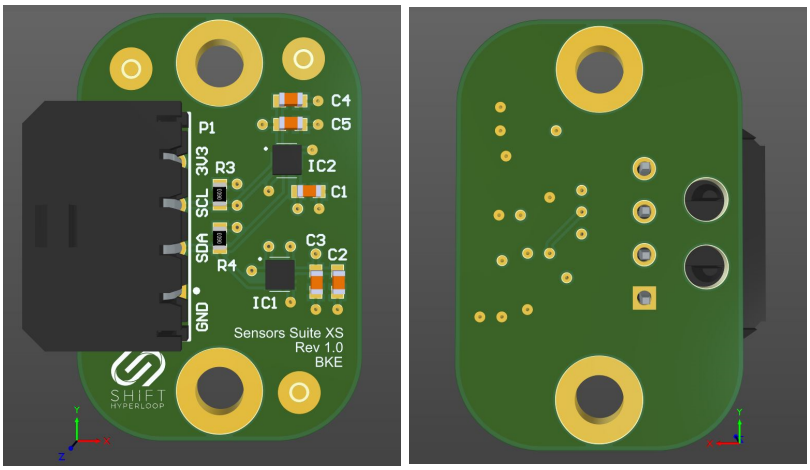
(a) 3D top view

(b) 3D bottom view

Figure 23: 3D models of sensor Suite 2, revision 2

At the time of revision 2 we had included another custom PCB to hold the four accelerometers and ambient air pressure sensors. These were called Suite XS. One accelerometer and one pressure sensor was placed on each Suite XS, which led to a total of four XS suites. Rest of the design was based on design examples from the user manuals of the sensors. By merging the data lines of the sensors, and also merging the clock lines separately, then the two sensors could act as slaves on the I2C line.

A problem that emerged when trying to create Suite XS was that the sensors being placed on the PCB had less than 0.2mm distance between the copper pads (actually 0.15mm between pads). This would be difficult for NCAB Group to create, therefore some manual changes to the footprint were made. The solder mask between the pads was removed to help NCAB be able to easier create the copper pads correctly. The solder mask is important because it stops the tin to flow freely between the pads, and possibly create a short. Removing the solder mask would therefore make the soldering of the sensor to the PCB more difficult because of heightened shorting possibilities.



(a) 3D top view

(b) 3D bottom view

Figure 24: 3D models of sensor Suite XS, revision 1

5.5 Code

Figures 25 - 28, as seen below, provide a visual walkthrough of the code structure in the form of flowcharts. There are a total of four flowcharts, one for each of the main sensors. The code in its entirety can be seen in Appendices N - P.

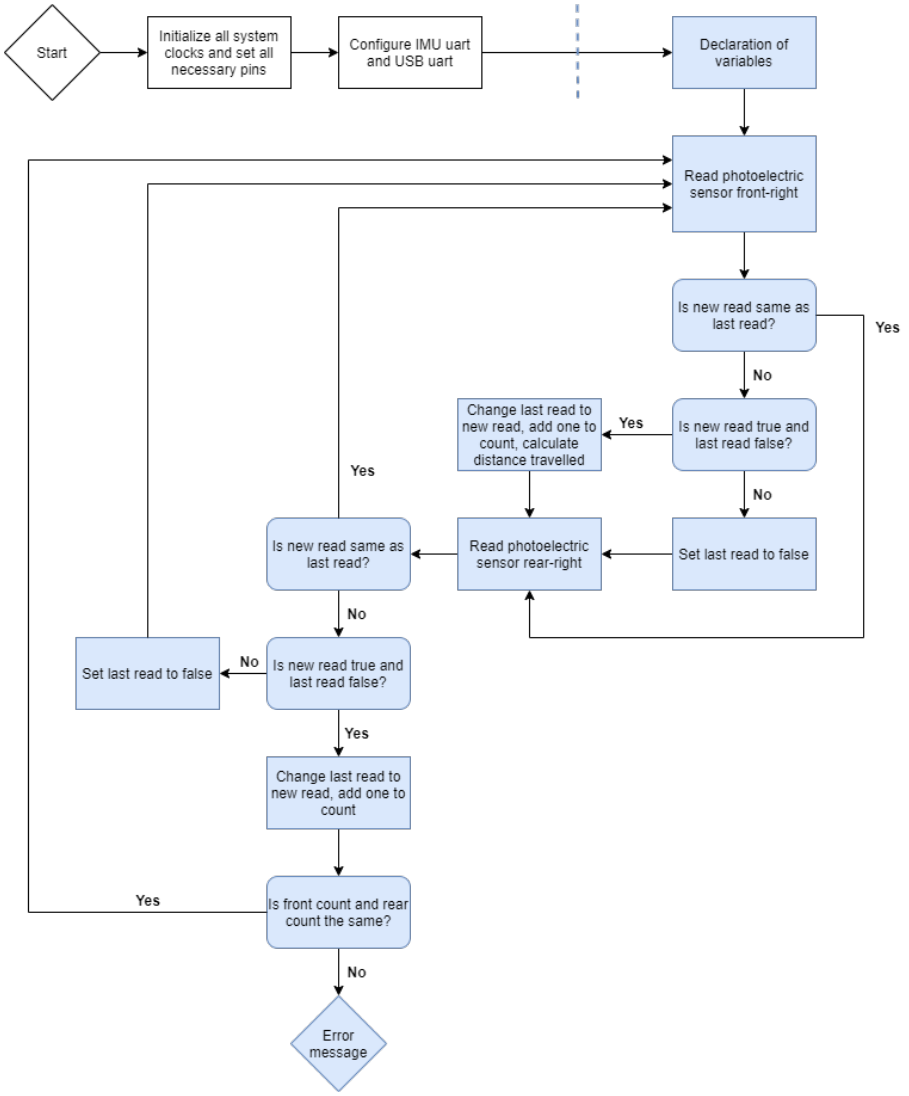


Figure 25: Flowchart for Photoelectic sensors

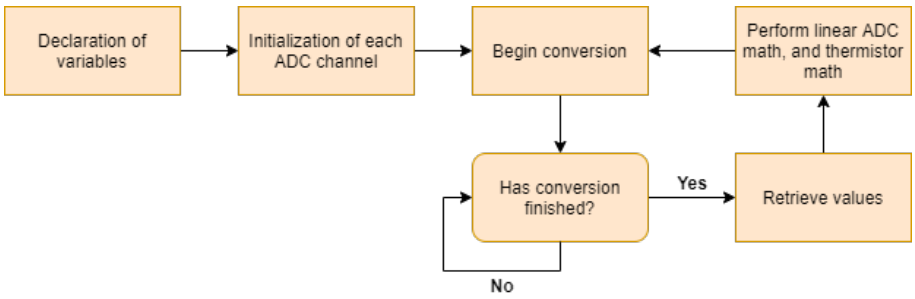


Figure 26: Flowchart for ADC

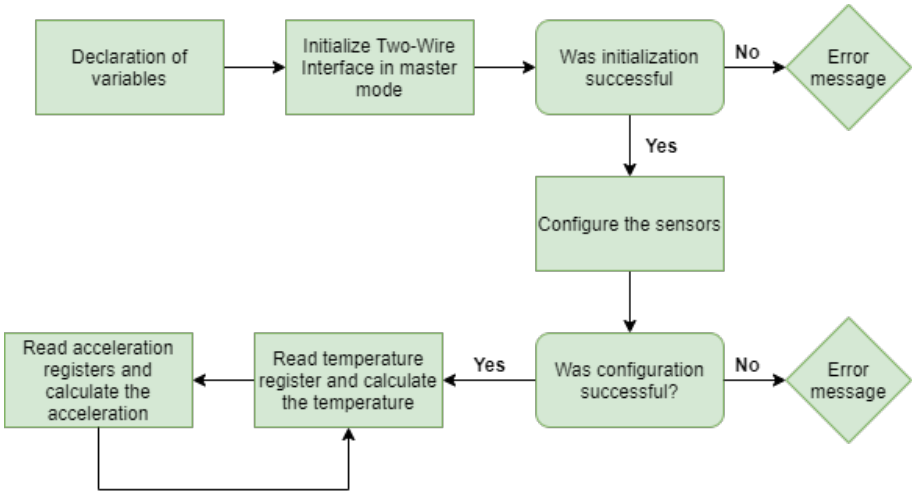


Figure 27: Flowchart for TWIHS

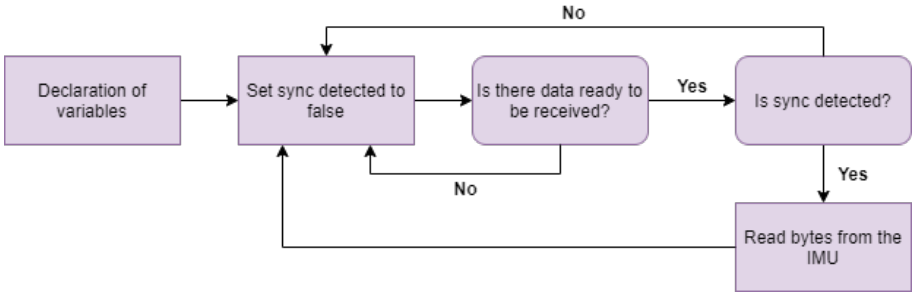


Figure 28: Flowchart for IMU

6 Testing sensors

In order to make sure that all parts of the system works and to limit sources of error, we have tested all sensors separately, without custom PCBs. We needed to test if they worked in their measuring range and for some of the sensors - their linearity or selected output signal. The tests were done either at the office or at home, depending on the necessary equipment.

All test used measuring tool such as ruler and/or protractor the parameter measured. As these tools use eye measurement there will be a uncertainty of 0.5mm and 0.5°. For the temperature tests there was used a thermometer, this thermometer has an uncertainty of 0.5°C.

6.1 IMU, VN100

In order to test the IMU *Vector Nav VN100* we had to setup a test rig and get necessary measuring equipment. The equipment we used for our test was a digital laser leveler, protractor, VN100 and tripod.

We began the test with placing the sensor on a custom designed tripod with movement in x-, y- and z-axis, that way we would be able to move it in multiple angles. The IMU can measure in three axis: yaw, pitch and roll. Then we used a digital laser leveler to make sure that the IMU was completely level with regards to yaw, pitch and roll. This was done as we needed to make sure that all angles was at zero degrees when beginning the test.

The IMU was connected to an Arduino Mega using the Mega2560 chip, as we needed an Arduino with multiple serial pins. Once the Arduino and cables were set up, we moved the IMU to different angles. We compared measured angles using a protractor to angles given as output through code to a terminal monitor. Because we were using a protractor and reading of it ourselves the uncertainty will be ± 0.5°. The protractor had the tendency to be stuck at certain angles, and is therefore an additional source of error.

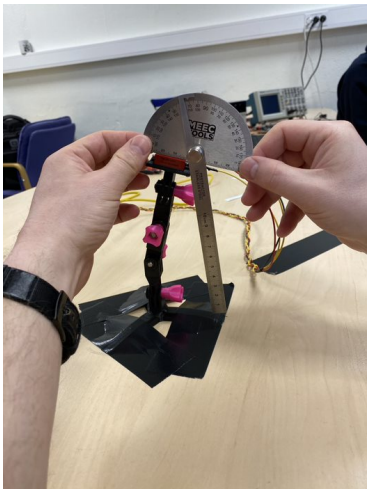


Figure 29: Test setup for testing IMU.

Pitch:

Protractor (in °)	Code (in °)
50	49.3
25	25.7
90	90
50	50.17
10	10.66

Table 3: Pitch test for IMU, VN100.

Roll:

Angle meter (in °)	Code (in °)
33	27
26	27.06
50	40 (complementary angles)
60	29.5 (complementary angles)
5	4.61

Table 4: Roll test for IMU, VN100.

As shown in the Table 3 and Table 4 the VN100 is more accurate than the protractor, but is still close on every measurement. The VN100 has a angular resolution at 0.001° [30], while the protractor is limited to eye measurements.

6.2 Photoelectric sensor, XP10

To test the photoelectric sensor we needed to create a test rig which included a leveler, ruler, and a tripod with the sensor fastened. Additionally, we needed a navigation marker with the red reflective corner cube tape, which is the same as will be used in competition. To make sure we are testing with the same navigation markers as in competition, we made our own navigation marker with 10cm wide red reflective tape given to us by Omron. The tripod and navigation marker was placed with a distance between them, the distance began with 1.5m and where changed during the tests.

We connected the sensor to a power supply, oscilloscope and a load resistor. We wanted to test the sensor with different distances and angles, and make sure the circuit gives out 24V each time the tape is in front of the sensor. As the distance markers are planned to be set 1.5m from the track, we needed to check if the sensor reacts around that distance. The sensor combined with an R5 lens have a range up to 91.5cm, since we need to measure at 150cm we contacted Tri-Tronics about this. The response was that the sensor would work if we got 10 on the built-in LED contrast indicator, and we therefore needed to verify this. The sensor has a built-in LED contrast indicator (Figure 30, so all tests need to score over 10 on the contrast indicator in order to be sure it works.

Figure 30: Contrast scale for the photoelectric sensor.

We tested if we would receive 24V output when the tape is placed in an angle. It is unlikely that we would ever have the tape in an angle, but at 45 degrees we got 24V out and a 7 on the contrast scale.

This test was executed inside Shift Hyperloop's office. As the photoelectric sensor will detect navigation markers outside, there will be done tests outside in daylight at a later point. These tests will not be included in this thesis.

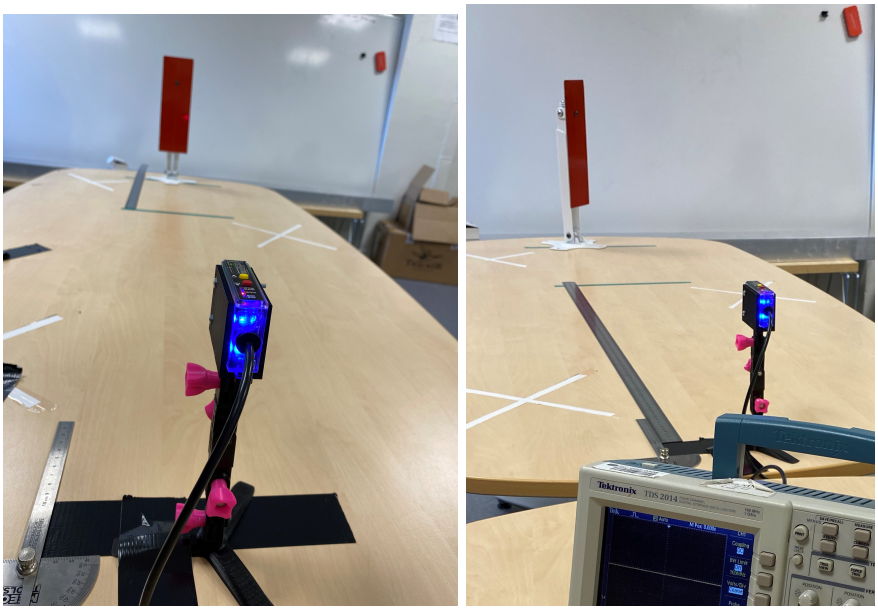


Figure 31: Test setup of photoelectric sensor, parallel and 45 degrees.

6.3 Distance sensor, OMT300

When testing the distance sensors we needed a test rig containing the sensor, tripod, ruler, aluminium plate, power supply and oscilloscope. We fastened the sensor to the tripod and placed it beside the ruler. On the other end of the ruler we placed the aluminium plate. We needed to use an aluminium plate as that is what the track is made of. The sensor was then placed perpendicular to the table and perpendicular to an aluminium plate. The distance between the sensor and the aluminium plate began at 10cm, then moved in increments of 2 cm each run. In order to do this as accurate as possible we had placed the ruler next to the aluminium plate.

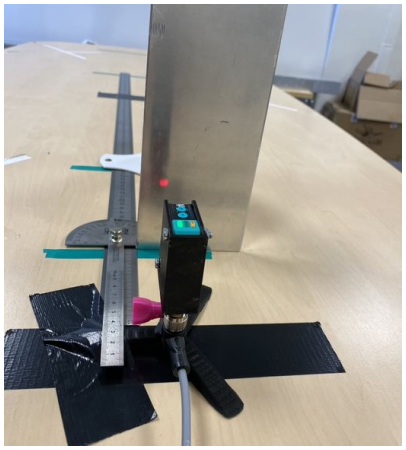


Figure 32: Test setup for optical distance sensor.

Before we could test the sensor we had to calibrate it to fit out distances from 10-20cm, and make sure they fit the 4-20mA output signal. The calibration was done with an aluminum plate and ruler.

We began with connecting the sensor to a power supply with load resistor (178.5 Ω) and an oscilloscope. When everything was connected, we read the voltages of the oscilloscope and wrote them down. For each measurement we moved the aluminium plate 2 cm, all measurements can be found in Table 5.

Distance (cm)	Squares on oscilloscope	Vertical scale value	Voltage
Less than 10cm	7.2 squares	100 mv/div	0.7V
10	7.2 squares	100 mV/div	0.72 V
12	4.85 squares	200 mV/div	0.97 V
14	6.8 squares	200 mV/div	1.36 V
16	3.4 squares	500 mV/div	1.7 V
18	4 squares	500 mV/div	2.0 V
20	4.5 squares	500 mV/div	2.25 V
22	5 squares	500 mV/div	2.5 V
24	5.6 squares	500 mV/div	2.8 V
26	6.2 squares	500 mV/div	3.1 V
28	6.9 squares	500 mV/div	3.45 V
30	7.2 squares	500 mV/div	3.6 V
More than 30cm	7.2 squares	500 mV/div	3.6 V

Table 5: Distances tested and voltages from oscilloscope for distance sensor test.

We tested the sensor with multiple distances in the sensors
measurement range and measured the voltages with an oscilloscope. We then compared the measured voltages to expected voltage values. The comparison are represented in Figure 33.

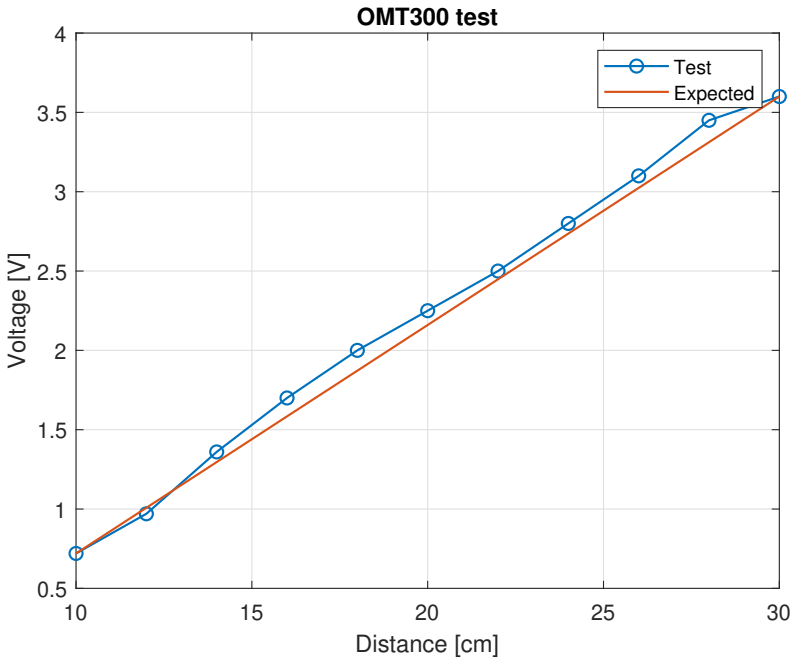


Figure 33: Comparisons of measured and expected voltages for distance sensor test.

6.4 Thermistors

For the thermistor test we needed a thermistor, Arduino, breadboard, load resistors, multimeter, water, ice cubes, cooktop and pot.

We tested one thermistor in known temperatures, such as ice water and boiling water. First we checked what the thermistor resistance, R_{ref}, was at room temperature. Additionally, we measured all the resistance values in the Wheatstone bridge as we used resistors with higher tolerance than

the ones on the PCBs. In addition to this we measured the input voltage, all values can be found in Table 6.

	Measured value	Unit
V_{in}	3.298	Volt
R_{ref}	118.2k	Ω
R_2	99.2k	Ω
R_3	99.8k	Ω
R_4	99.3k	Ω

Table 6: Test setup for Wheatstone bridge with values and units.

Then we could test and measure thermistor resistance and voltage in boiling water, as that has a known temperature of 100 degrees.

	Measured value	Unit
R_{100}	6.2k	Ω
V_{100}	1.432	Volt

Table 7: Resistance and voltage from testing thermistor at 100°C.

When we finished testing with boiling water, we moved on to testing the thermistor in ice water. Due to the uncertainty of the temperature in ice water we decided to add a thermometer to the tests. We measured the temperature with both thermistor and thermometer in order to check if the measured resistance matches the expected temperature. When we knew the resistance we could find the estimated temperature using a graph in Matlab. The graph, Figure 34, was created based on the beta formula for thermistors (Equation 14).

	Measured value	Unit	Temperature (°C) from graph
$R_{1.8}$	320k	Ω	1.7
$R_{2.7}$	301k	Ω	2.8
$R_{1.9}$	316k	Ω	1.9
$V_{1.9}$	Compromised, thermistor died		

Table 8: Temperature, resistance and voltage from testing thermistor in ice water.

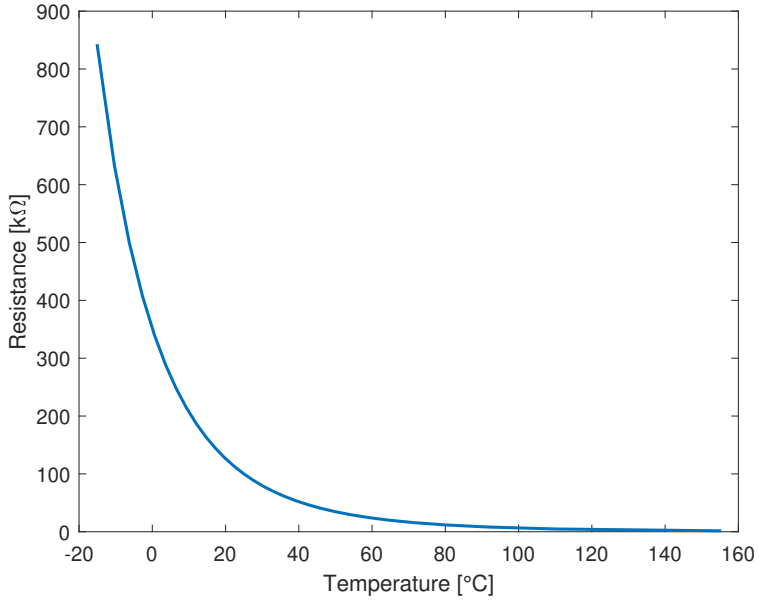


Figure 34: Thermistor graph, Resistance and temperature.

6.5 Thermocouples

In order to test the thermocouples we needed a thermocouple type T, temperature transmitter, breadboard, load resistor, 24V power supply, multimeter, water, ice cubes, cooktop and pot. We tested with known temperatures in boiling and ice water, in a freezer and in room temperature. This test was intended to verify the functionality of the transmitter, and whether it was functional with default settings or had to be programmed. The transmitter was connected as instructed in the manual, see Figure 35. The thermocouple tip was then exposed to various temperatures, as are shown in Table 9

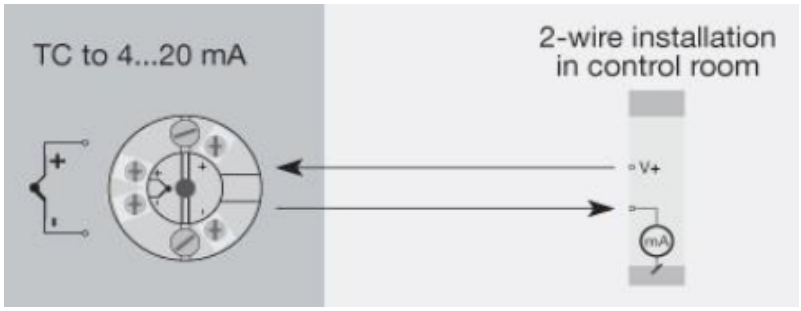


Figure 35: Schematic of the transmitter connections. The circled mA symbol represent the load resistance. Transmitter is a RS-PRO 2-wire programmable, type 192-6538

Reference voltage values for each temperature were read off a standard chart for type T thermocouples. Voltages over the load resistance were measured and to calculate the corresponding output current we used Ohms law $I = \frac{V}{R_L}$, where $R_L = 178.6\Omega$. Uncertainty arose during testing due to a lack of information about the connection between thermocpouple output voltage and the transmitter output current.

Freezer (-18°C):		
V_0	-0,683	mV
V_{RL}	678.3	mV
I_{RL}	3.7979	mA
Ice water (1°C):		
V_0	0.039	mV
V_{RL}	735	mV
I_{RL}	4.115	mA
Room temperature (22°C):		
V_0	0.87	mV
V_{RL}	816.7	mV
I_{RL}	4.5728	mA
Boiling water (100°C):		
V_0	4.279	mV
V_{RL}	1.15	V
I_{RL}	6.4390	mA

Table 9: Results from thermocouple test with an un-programmed transmitter. V_0 are reference values from Thermocoupleinfo.com

As we can see in Figure 36, the transmitter output current is nearly linear with temperature changes. The deviation could be from the multimeter or the natural non-linearity of the type T thermocouple.

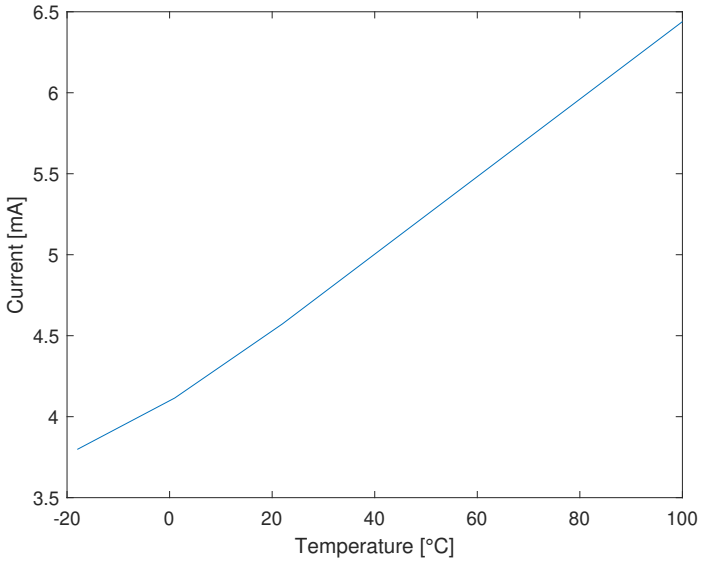


Figure 36: Test thermocouple

Following the first test, another test was conducted after the transmitters were programmed properly. PR Electronics provide software for this purpose, and the version used was the PReset 5300-OEM 8.05.1004, downloaded from their website. The necessary hardware, the Loop Link 5909 programming unit was lent to us by Kebony AS. The transmitters were programmed with the settings seen in Table 10. The built-in linearization function of the transmitters were also enabled during this step. This test was not as extensive since so much time had passed and we needed to focus on other tasks. Additional tests will be conducted before EHW, but for now we will just argue why the secondary test results seem valid.

Port	0%	100%	Unit
Input	0	250	°C
Output	4	20	mA

Table 10: Input and output settings programmed into the RS-PRO transmitter.

In Table 11 the results from each of the four transmitters are presented. The transmitter output was connected to a load resistor and we measured the voltage over it with a multimeter. The current is calculated using Ohm's law.

Transmitter #	Measured Voltage	Unit	R_{load}	Unit	Calculated Current	Unit
1	1,165	V	216,7	Ω	5,376	mA
2	1,165	V	216,7	Ω	5,376	mA
3	1,168	V	216,7	Ω	5,389	mA
4	1,168	V	216,7	Ω	5,389	mA

Table 11: Test in unknown room temperature with programmed transmitters.

This test was performed at our office, in an unknown room temperature. Based on the programmed settings for the transmitters and measured values during the test we can calculate the room temperature. Since the range of the 4-20mA signal start on 4mA, this is subtracted from the circuit current to get the realistic percentage of the range. Equation 22 show the formula used for finding the room temperature $temp_{room}$.

$$temp_{room} = \frac{I_{out}}{I_{range}} * temp_{range} \tag{22}$$
$$temp_{room} = \frac{5,376}{16} * 250 = 21,5°C$$

A room temperature of 21,5°C is a believable result, and no further verification of the transmitters will be conducted for this thesis. The difference between two of the four transmitters might be down to component tolerances, but the deviation was so small that we will likely correct for this in software, if at all.

6.6 Hydraulic pressure sensor, HDA4400

The test of the hydralic pressure sensor was limited as we could not test with a pressurized system. The goal of the test was to check if we would get around 1 atm = 1.01 bar. The sensor has a range from 0 to 400 bar, and as it is linear we expect a value close to 4mA as output.

For this test we needed a 24V power supply, multimeter, breadboard and load resistor.

We started with connecting the sensor to a power supply with a load resistor. We used the same load resistor as for the other 4-20mA current loop tests. Supply voltage and load resistor was controlled to be: $V_{in} = 24.05V$ $R_L = 178.6\Omega$

With everything connected and power switched on, the voltage over R_L read V = 707 mV. The current was calculated using Ohms law:

$$A = 707 * \frac{10^-3}{178.6} = 3.9586mA$$

The current 3.96mA is lower than what the sensor should be able to produce. A potential source of this error is a poorly calibrated multimeter.

7 Results

This section will present the results from testing our two custom PCBs; the sensor suites. We will limit this section to include results from testing revision 2 of Suite 1 and 2, and revision 1 of Suite XS. Some early results from revision 1 testing are described in subsubsection 4.2.2.

7.1 General functions

Suite 2 was having some problems with the 3.3V PSU being powered when only the 5V PSU was supposed to be powered. After inspecting the Altium PCB layout for suite 2, we noticed that both resistor R22 and R21 were soldered on vertically instead of horizontally. From Figure 37 one can see how it was supposed to be soldered on. After having resoldered the resistors on properly the PSUs started functioning as supposed to.

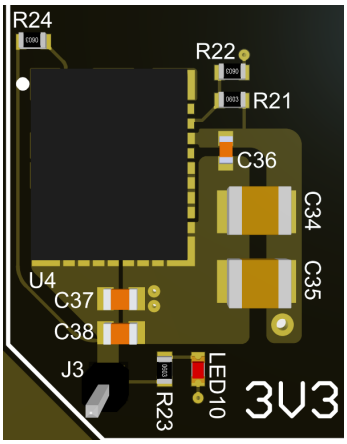


Figure 37: PCB layout of the 3.3V power supply circuit

After the board could be powered correctly, we moved on to flashing code to the microcontroller. Flashing of the microcontroller went smoothly because Microchip Studio instantly detected the Atmel-ICE debugger, and was able to read the device ID of the microcontroller. The test code that was used for flashing was supposed to toggle the debug LEDs, however, this did not work. It turned out that all the debug LEDs were soldered on in the wrong direction. After resoldering the debug LEDs on correctly, then the LEDs started toggling.

7.2 4-20mA signal circuits

All tests were performed on revision 2 of the Suite 2 PCB. During the first round of tests with a distance sensor connected to the ADC through its interface circuit, some problems were discovered. When the distance sensor was unconnected the terminal monitor showed expected values of close to zero. However, when the distance sensor got connected, and placed such that the actual distance measured would be close to 0cm, then the values started fluctuating as seen in Figure 38. With the sensor placed at that distance the ADC decimal value should have been 819. We measured the voltage directly from the output of the operational amplifier, and with distance of 0cm we measured close to 0.66V, which was as expected. After changing the distance to 20cm we measured 3.3V output, which again is as expected. This meant that the circuit was working properly, and the issue must be software related.

```
ADC value: 2103
ADC value: 2099
ADC value: 1269
ADC value: 2304
ADC value:  325
ADC value:  519
ADC value: 1252
ADC value:  354
ADC value: 1918
ADC value: 2048
ADC value: 2139
ADC value: 1447
ADC value:  650
ADC value: 2180
ADC value:  560
ADC value:  980
ADC value: 1193
ADC value:  677
ADC value: 2145
ADC value: 1271
ADC value: 2241
ADC value: 1073
ADC value: 2196
```

Figure 38: Fluctuating ADC values

After having tested the OP-AMP circuit, we moved on to testing the instrumentation amplifier circuit ended up being ruled invalid because of an error during the PCB design phase. The error had to do with the Altium schematic library for the IN-AMP footprint. The footprint had to be manually created, and was wrongly created because of human error. After inspecting the schematic library we noticed that the pin labels differed from the actual pins in the datasheet. One can see the differences from Figure 39, where the actual pin layout is on the left and the schematic on the right.

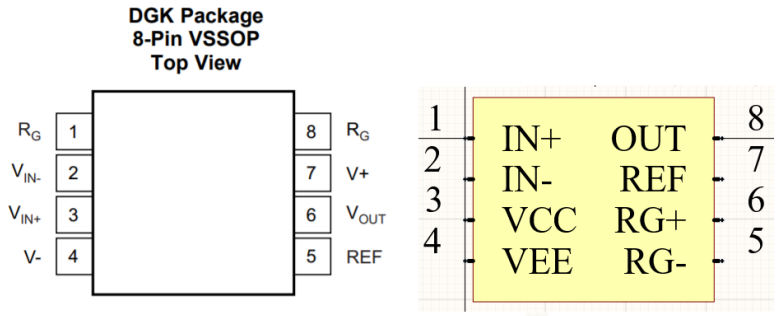


Figure 39: INA333 pin layout from datasheet [31], and custom pin layout from Altium schematic library.

The error caused every IN-AMP to be traced incorrectly, and in return become nonfunctional. Even though the IN-AMP did not work, we could still test the Wheatstone bridge. We measured the resistance of the thermistor to be 95k ohm before connecting it to the circuit. After connecting the thermistor to the circuit, we measured the voltage being supplied to the non-inverting input to be 1.67V. This is not far from the simulated voltage of 1.69V.

7.3 Comparator circuit

While testing the comparator some issues emerged. The 24V sensor output signal is connected to the comparator through a voltage divider. This way, the comparator will receive a maximum voltage of 3.3V to the input pin. With the reference voltage of the comparator being set to 1.65V, then an input of 3.3V should result in an output that equals a digital high read by the microcontroller. However, the value read by the microcontroller and then printed to a terminal monitor was still a digital low. We speculated that it was caused by the 24V having ground reference in the power supply and the microcontroller having a different reference to ground. Instead of using the 24V being supplied from the power supply, we decided to try supplying the input of the comparator with 3.3V locally sourced from the PCB. Since each of the PSU outputs on the PCB have a header pin soldered to them, we could easily connect a wire probe to this header pin and manually supply the input pin of the comparator with it. This trial solved the problem. By using a locally sourced 3.3V the microcontroller managed to read a digital high. After that we tried connecting the ground of the PCB to the ground of the power supply, and also powering the PCB with the power supply. By using a dual channel power supply, it allowed us to power the PCB with 8V, and at the same time power the comparator with 24V. This trial ended up working as well, and at the same time proving our suspicion that a different ground reference could cause problems.

Another minor problem that was noticed during testing of the comparator was that counters in the code got reset after a certain amount of time. We first believed it was caused by some issue regarding printing to the terminal monitor or integer overflow. However, neither of these believed causes were true. Because the reset had no correlation with how far the counter had reached, nor did changing the printing method from integer to decimal correlate. After adding some delay functions to control the time, it became clear that the reset happened on a set timer. Through conferring with another group member in Shift, it came to our attention that the Watchdog ASF might be enabled. Watchdog is an ASF created by Microchip Technology in order to have a system that can reset the microcontroller on a set timer, unless the Watchdog counter is manually reset periodically [32]. No more resets happened after disabling this feature.

7.4 Two-Wire Interface

During testing of the Suite XS PCB both hardware and software problems were discovered. We had previously managed to read the device ID of the sensor while using the SAM V71 Xplained Ultra Evaluation Kit and the Sensors FeatherWing. Being able to read the device ID means that communication is established and working. While using the aforementioned development kits to read actual sensor data from the sensors, the code entered a timeout loop, and was therefore unable to successfully read sensor data. We also tried using the Xplained Ultra evaluation kit in combination with the sensors on Suite XS. With this setup we were unable to read the device IDs of the sensors. We were also unable to write to the control registers of the sensors in both of the attempts. While trying to write to the control registers, the code got stuck in a status check loop. In figure Figure 40 one can see that the status variable gets set from the status register. The problem is that the status variable is neither NACK, nor TXRDY, and that causes it to get stuck in the while loop. The TXRDY means that there are bits remaining to be sent, and the NACK is when the slave responds by saying it did not acknowledge the transmission.

```
369     while (1) {
370         status = p_twihs->TWIHS_SR;
371         if (status & TWIHS_SR_NACK) {
372             return TWIHS_RECEIVE_NACK;
373         }
374
375         if (status & TWIHS_SR_TXRDY) {
376             break;
377         }
378     }
```

Figure 40: Status check loop in Two-Wire Interface code

8 Discussion

A major factor for this thesis has been the time schedule as a part of Shift Hyperloop. Because we were following the same timeline as the main project, certain steps in the process could not be executed as thoroughly as needed or as quickly as wanted. PCB revision 1 was heavily affected by this as by the time the Electronics Group had to submit designs, few of the affected sub systems had decided on parameters they wanted or needed monitored. Revision 1 was created based on the little we did know, but as there were so many changes between revisions 1 and 2, the first revision was more or less useless. What we did manage to get from revision 1 was valuable experience with Altium Designer and specific ideas for how to improve the process and product for revision 2. One way we could have exploited this better would be to research several different interface circuits earlier and included one of each on the PCBs, instead of including several of the same. This we could have done regardless of knowing if they would be used, because revision 1 would never be used on the finished pod. In general, designing revision 1 with added flexibility for trial and error could have been beneficial for the next stages of circuitry design. The testing we were able to perform on the transimpedance amplifier circuit did verify the simulations we had done in LTspice, which was promising for further development.

However, the lack of thorough testing and relevant designs in revision 1 meant that revision 2 was designed almost blindly. We took care to plan the design in a more structured manner and thus were able to implement general recommendations for PCB layouts at an earlier stage. LTspice proved a valuable aid in simulation of interface circuits, and allowed for trial and error with various ICs with different electrical characteristics before finding a suitable component.

As we had to follow the same timeline as the other systems in the electronics group, we have experienced some difficulty with the timeline. This was due to components being ordered late and we could therefore not solder the PCBs until close to the thesis deadline. As a result we could start testing revision 2 of the system only a week before the thesis deadline. We did not have time to connect all signals, measure noise or do a full system test for the monitoring system. This also left us with not enough time to properly test the code and make it function without problem. If the components were ordered earlier, and we could solder our PCBs earlier, we would not have this heavy time constraint. This would allow for more proper testing of the system as a whole, instead of having to rush through testing.

We were unable to verify whether we had a better product for revision 2 than revision 1. This was because revision 1 was crude and had only the bare minimum required to function. The interface circuit from revision 1 was not kept for revision 2, because we were only able to test one circuit, and revision 2 also required new circuits. Our employer also recognized revision 1 as more of an experiment meant for learning the Altium Designer, than being close to a finished product. Being able to spend time with Altium Designer proved useful for all the systems, since we could spend most of that time learning and preparing for revision 2. The benefit from revision 1 was that we were able to test the shared circuitry designed by the Master Library system. The most visible non-verifiable improvement is the PCB design. We can clearly see our efforts of properly planning the layout compared to revision 1.

As Shift Hyperloop did not have any existing, finished product from previous years, we did not have any data or reference we could use when planning or designing our system. Most transfer of knowledge came from talking to the members who had previously worked on the sensor system, taking notes of ideas and struggles they encountered. No code existed and revision 1 of their PCB was soldered, but never tested for functionality. No design existed for interface circuitry. One key motivation for this thesis was to make sure Shift was provided with this foundation, and even at this stage we have achieved this. With even further testing and work following this thesis and a system installed on a finished pod, there will be extensive documentation and valuable experience for future teams to study during their concept phase and improve or change as needed.

When it comes to components, there are some changes that could have been made. We could have chosen a Zener diode with higher Zener voltage, since the comparator withstands a voltage as high

as 6.5V. This would have given the voltage divider more headroom. Since no proper testing of the functionality has been performed, this remains speculation. Overall, we benefited from the LTspice simulations, due to how easy it was to change component values whenever a value was out of stock or otherwise unavailable. We could easily run a new simulation of the circuits with new values and confirm the circuit would still perform satisfactory with the alternative component(s).

Right after revision 1 was finished, there were talks of implementing a 24V power supply circuit on our PCBs. This was mainly because most of our sensors needed 24V to be powered. However, after some discussion with the representative from our employer, it was concluded with buying an off the shelf solution for the 24V power supply. This was a choice made by the employer and our opinions would not affect the outcome. Issues regarding voltage potential could be caused by having the 24V that powers the sensors originate from a different 0V reference than the PSUs on the circuit boards. The chosen alternative solution was non-isolated (A and B side share 0V potential), and as this will be powered by the same DC/DC as the 3.3V and 5V PSUs, we are almost certain this will not cause real issues on the final pod. The test performed with the comparator circuit hinted at this, but we cannot say for sure.

We did not have the opportunity to look into electromagnetic interference (EMI). Some of the sensors will be placed close to the motor, and could be affected by EMI. As neither the motor nor the inverter exists at the moment, we could not test if the different sensors would be affected, and to which extent. Because of this, most of our choices were made based on basic theory. We chose current signals, 4-20mA, as they are typically more robust. We do not know whether this is enough to prevent the signals from being disrupted or not. As we did not have the opportunity to test the PCBs with the motor, it is difficult to predict how the circuits would behave, and therefore difficult to create the best and most effective filters. To further workings with the sensor system, we could recommend looking into how to reduce possible noise and interference.

One member in the project group experienced regular panic attacks due to a lack of implemented analog filters on revision 2, but was time and time again reassured by the reminder that it would be irresponsible to throw filters at a signal we do not know the characteristics of.

Our whole project regarding the Suite XS ended up not working. We were unable to determine exact reason for the Suite XS not working, but we have some ideas. First one is that it was difficult to actually produce, and therefore the soldering procedure failed. Another one could be that the uncertainty regarding Two-Wire Interface, made the code be unusable. Last is that it is a combination of both cases. For future endeavours it is important to think about how possible is it to actually produce the solution, before going through with it as the only option. We thought of using linear potentiometers or lasers sensors to monitor suspension at first, but these options would break the budget. We instead decided to go with the more affordable option, and with optimism that we could do it. The affordable option was also chosen because monitoring suspension was not deemed critical, and the whole system could work without it. The problem is that we would have no information whether the suspension is working or not.

The results from testing the different interface circuits were satisfactory, they behaved close to simulated results. Even though we did not get to test the complete IN-AMP circuit, we are confident that if the IN-AMP had been traced correctly, then the circuit would also function as expected.

Most efficient use of time is to continuously test code while it is being written. This also allows for debugging at the same time. Since we did not have custom PCBs that could be used for testing functionality of the code, we instead chose to use an Xplained Ultra evaluation kit. All of the code was tested and verified to be working with the evaluation kit, but when it came to testing with a custom PCB, then most of the code did not work. One of the reasons for this could be that we are using the same microcontroller initialization as the evaluation kit. This could also include certain configurations which are specialized for the evaluation kits. We did notice that the evaluation kit used a different amount of clock cycles for the ADC startup time, than what is suggested in the user manual for the ATSAMV71Q21. This led to the suspicion that conversion times may also be

different, thus somewhat explaining the fluctuating ADC values from Figure 38 in subsection 7.2. It was also reaffirmed by other groups that they were having similar problems, code working with evaluation kit, but not with custom PCB.

Our test setups and rigs for preliminary calibration of the distance sensors suffer from a lack of accuracy. We did not have access to proper measuring instruments. Additional calibration of sensors will likely be necessary, but some issues might be solved by compensation in software. The temperature measurements require the least accuracy, and the stated accuracy of the thermometer used for control during testing of +/- 0.5°C is acceptable. When calibrating the LED distance sensor we were only able to achieve an accuracy of +/- 0.5mm with a 1mm ruler and visual readings, which is not acceptable. The test setup was even less accurate, so another calibration and test have to be conducted with more precise instruments. The key will be to measure the distance closer to the point of light emitted from the sensor and where it reflects from the detected object, with a certainty of a perpendicular relation between sensor surface and detected object. The photoelectric sensors have a built-in contrast gauge which maxed out on 10 for our tests of up to 2 meters in indoors ambient lighting. The sensor switches the output already at a contrast level of 6, and the realistic distance during competition will be <1.5m. This is promising, but further testing will be necessary in daylight. Because of the polarized lens on this particular model, however, this is not expected to be a problem. The IMU suffered from similar circumstance during testing, but we expect we will implement functions in code to compensate for any offset or calibration issues. One way to solve this might be to do a reading at the start of each pod run, when the pod is stationary on the track, and do an onboard variable calibration before each run. An example would be to set variable values as ”0”, then simply use the delta values in either direction as measurements. This would need to be handled carefully, as a check of nominal values is also part of the pod startup verification process, so a valid range would have to be determined.

9 Conclusion

The primary goal of this project was never to create a perfectly optimized product, but rather create a working platform that the team for next year could use as a baseline and improve upon. Through designing and testing our second PCB revision we feel this is something we have achieved. Interface circuits for converting 4-20mA current loops to a 0.66-3.3V signal and the comparator used to adapt a 24V output to a 3.3V input were both proven successful. Even though the footprint for the INA333 was incorrect, and code related to ADC and Two-Wire Interface, we have a solid baseline for further work. We have strong reasons to believe the INA333 circuit will be successful in coming revisions, based on the success with simulations. Our code works well on Microchip evaluation kits with the Würth Elektronik Sensor Featherwing, and we have no reason to believe it will not work with a fully customized setup after more work. Further testing, troubleshooting and tuning will occur by the group members leading up to EHW, as this will grant us and future teams with more knowledge and valuable experiences.

At the time of this report being written, it is known that a revision 3 is necessary because of the need for additional sensors and some design mistakes made during design of revision 2. Current schematics for revision 3 are included as attachment (Appendix F and G), but will not be discussed. For anyone that will recreate or improve on this project, we recommend using those schematics rather than revision 1 and 2.

Additionally, the lack of documentation from previous years have been a great motivation during this project. Through working on this report we feel we have created a solid baseline for further development, and satisfactory documented what have been done and the results achieved during this project.

Bibliography

[1] R. Miller, "The scifi story robert h. goddard published 100 years ago," *Gizmodo*, 2014. [Online]. Available: https://io9.gizmodo.com/the-scifi-story-robert-h-goddard-published-100-years-a-1494959842.

[2] E. Musk, *Hyperloop alpha*, 2013. [Online]. Available: https://www.tesla.com/sites/default/files/blog_images/hyperloop-alpha.pdf.

[3] (). "Virginhyperloop.com," [Online]. Available: https://virginhyperloop.com (visited on 05/19/2021).

[4] Z. Kleinman, "Virgin hyperloop pod transport tests first passenger journey," *BBC.com*, 2020. [Online]. Available: https://www.bbc.com/news/technology-54838982.

[5] *Spacex hyperloop pod competition*, Jun. 2015. [Online]. Available: https://web.archive.org/web/20150714105843/http://www.spacex.com/sites/spacex/files/spacex_hyperloop_pod_competition.pdf.

[6] "MIT Hyperloop Final Report," MIT Hyperloop, Tech. Rep., 2017. [Online]. Available: http://web.mit.edu/mopg/www/papers/MITHyperloop_FinalReport_2017_public.pdf.

[7] "Poddy McPodface Final Design Briefing," HYPED, Tech. Rep., 2017.

[8] "CU Hyperloop 2018 Technical Report," CU Hyperloop, Tech. Rep., 2018.

[9] A. S. Morris and R. Langari, *Measurement and Instrumentation: Theory and Application*, 2nd ed. Academic Press, 2015.

[10] E. M. Gary Frigyes and J. Allison, "Fundamentals of Photoelectric Sensors," *Automation.com*, 2010. [Online]. Available: https://www.automation.com/en-us/articles/2014-1/fundamentals-of-photoelectric-sensors.

[11] "Retroreflectors," *RP Photonics Encyclopedia*, [Online]. Available: https://www.rp-photonics.com/retroreflectors.html.

[12] *Technical Datasheet ORALITE VC 104+*. [Online]. Available: https://www.orafol.com/products/europe/en/technical-data-sheet/oralite-vc-104plus-rigid-grade-id7446-technical-data-sheet-europe-en.pdf.

[13] "Comparison between Ultrasonic Sensors and Optical Sensors," *Keyence*, [Online]. Available: https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/comparison/.

[14] J. G. Webster and H. Eren, *Measurement, Instrumentation, and Sensors Handbook*, J. G. Webster and H. Eren, Eds. CRC Press, 2014, ISBN: 978-1-4398-4888-3.

[15] "What is an IMU," *Vector Nav*, [Online]. Available: https://www.vectornav.com/resources/what-is-an-imu.

[16] T. P. Hypertextbook, *Acceleration*. [Online]. Available: https://physics.info/acceleration/.

[17] "Gyroscopes," *Farnell*, [Online]. Available: https://no.farnell.com/sensor-gyroscope-technology.

[18] N. Nager, "Hot tips on THERMOCOUPLES," *Phoenix Contact USA*, [Online]. Available: https://www.phoenixcontact.com/assets/downloads_ed/local_us/web_dwl_promotion/HotTipsonThermocouples.pdf.

[19] S. Beguš, J. Bojkovski, J. Drnovšek, and G. Geršak, "Magnetic effects on thermocouples," *IOP Publishing Ltd*, 2014. [Online]. Available: https://iopscience.iop.org/article/10.1088/0957-0233/25/3/035006.

[20] "NTC Thermistor," *EE Power*, [Online]. Available: https://eepower.com/resistor-guide/resistor-types/ntc-thermistor/#.

[21] "Analog to Digital Conversion," *Analog Devices Wiki*, 2021. [Online]. Available: https://wiki.analog.com/university/courses/electronics/text/chapter-20.

[22] "Analog to Digital Converter (ADC) – Block Diagram, Factors Applications," *Electrical Technology*, [Online]. Available: https://www.electricaltechnology.org/2019/02/analog-to-digital-converter-adc.html.

[23] "Advanced software framework (asf) for sam devices," *Microchip Technology*, [Online]. Available: https://www.microchip.com/en-us/development-tools-tools-and-software/libraries-code-examples-and-more/advanced-software-framework-for-sam-devices.

[24] "EuropeanHyperloopWeek 2021 Rules and Regulations Version 1.1," EuropeanHyperloopWeek, Tech. Rep., 2020.

[25] N. K. Rossing, *Grunnleggende elektronikk og sensorteknikk*. NTNU, 2014. [Online]. Available: https://www.ntnu.no/documents/2004699/12108297/Grunnleggende%20elektronikk%20og%20sensorteknikk%208.0.pdf/f743226b-3698-40e4-b123-8b105685893f.

[26] "Implementing 4-20 mA Sensor Interface," *Dialog Semiconductor*, 2018. [Online]. Available: https://www.dialog-semiconductor.com/sites/default/files/2021-03/AN-CM-229%20Implementing%204-20%20mA%20Sensor%20Interface.pdf.

[27] J. Heath, "Why do industrial sensors measure in 4-20ma to programmable logic controllers?" *Analog IC Tips*, 2016. [Online]. Available: https://www.analogictips.com/faq-industrial-sensors-measure-4-20ma-programmable-logic-controllers/.

[28] "Use of rail-to-rail operational amplifiers," *Texas Instruments*, [Online]. Available: https://www.ti.com/lit/an/sloa039a/sloa039a.pdf?ts=1620046145450.

[29] "Sam e70/s70/v70/v71 family," *Microchip*, [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-E70-S70-V70-V71-Family-Data-Sheet-DS60001527E.pdf.

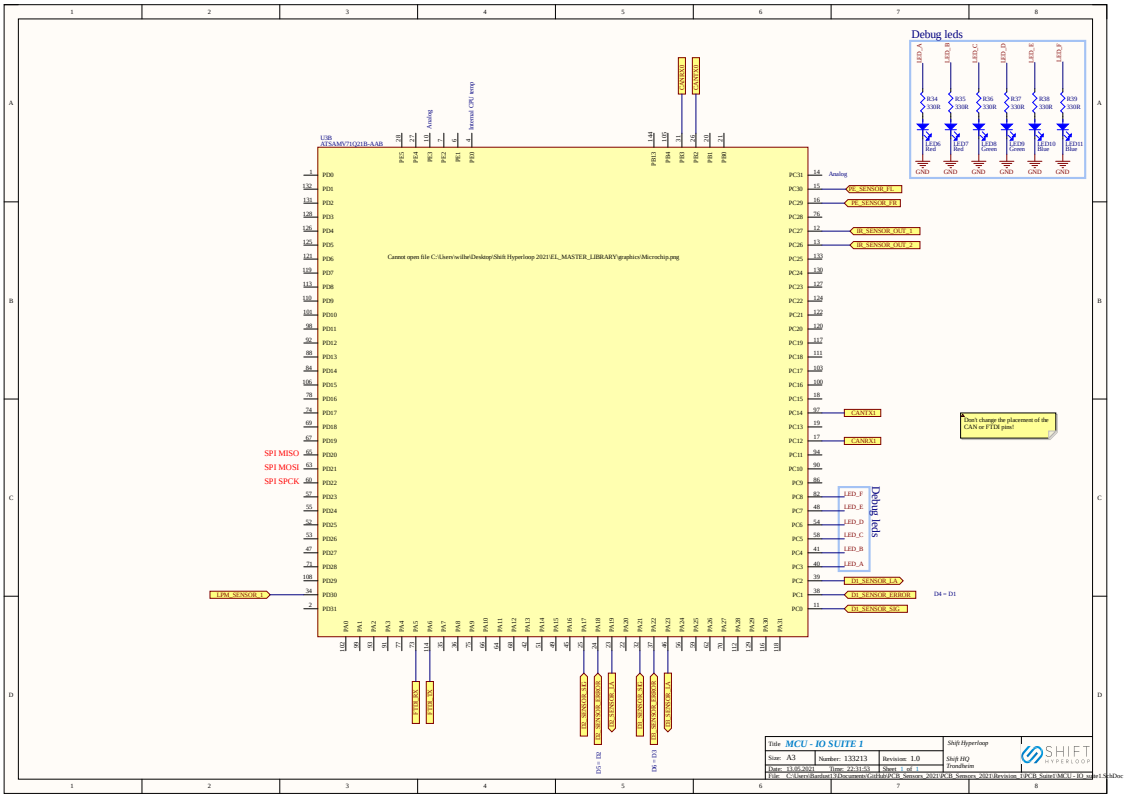
[30] "VN100," *Vector Nav*, [Online]. Available: https://www.vectornav.com/products/vn-100.

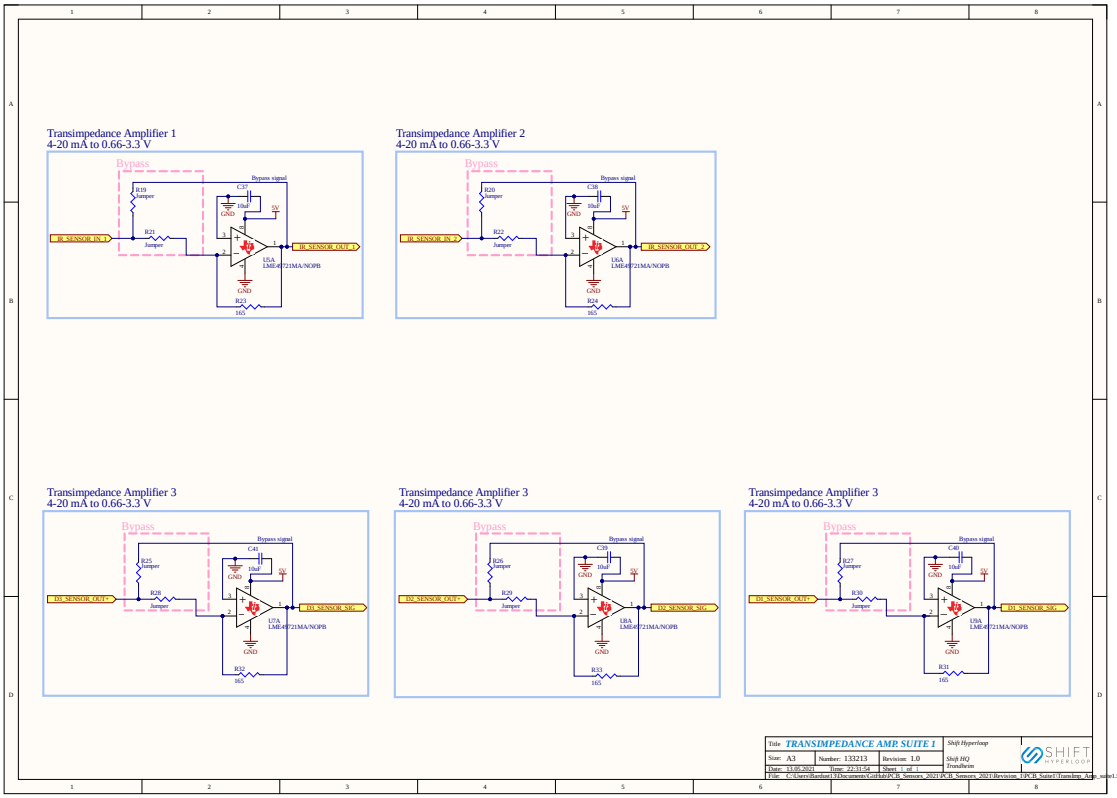
[31] "Ina333 micro-power (50a), zerø-drift, rail-to-rail out instrumentation amplifier," *Texas Instruments*, [Online]. Available: https://www.ti.com/lit/ds/symlink/ina333.pdf?ts=1621429256269.

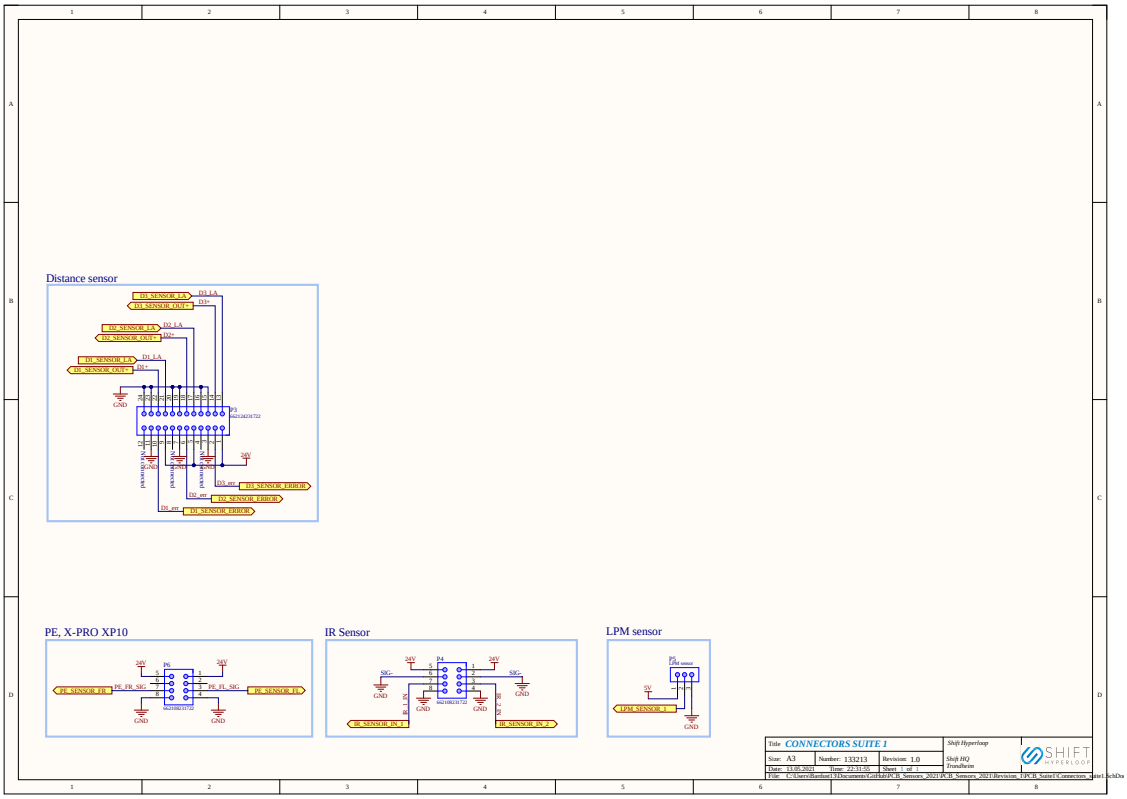
[32] "Quick start guide for wdt - basic," *Microchip Technology*, [Online]. Available: https://asf.microchip.com/docs/latest/sam0.drivers.wdt.unit_test.saml21_xplained_pro/html/asfdoc_sam0_wdt_basic_use_case.html.

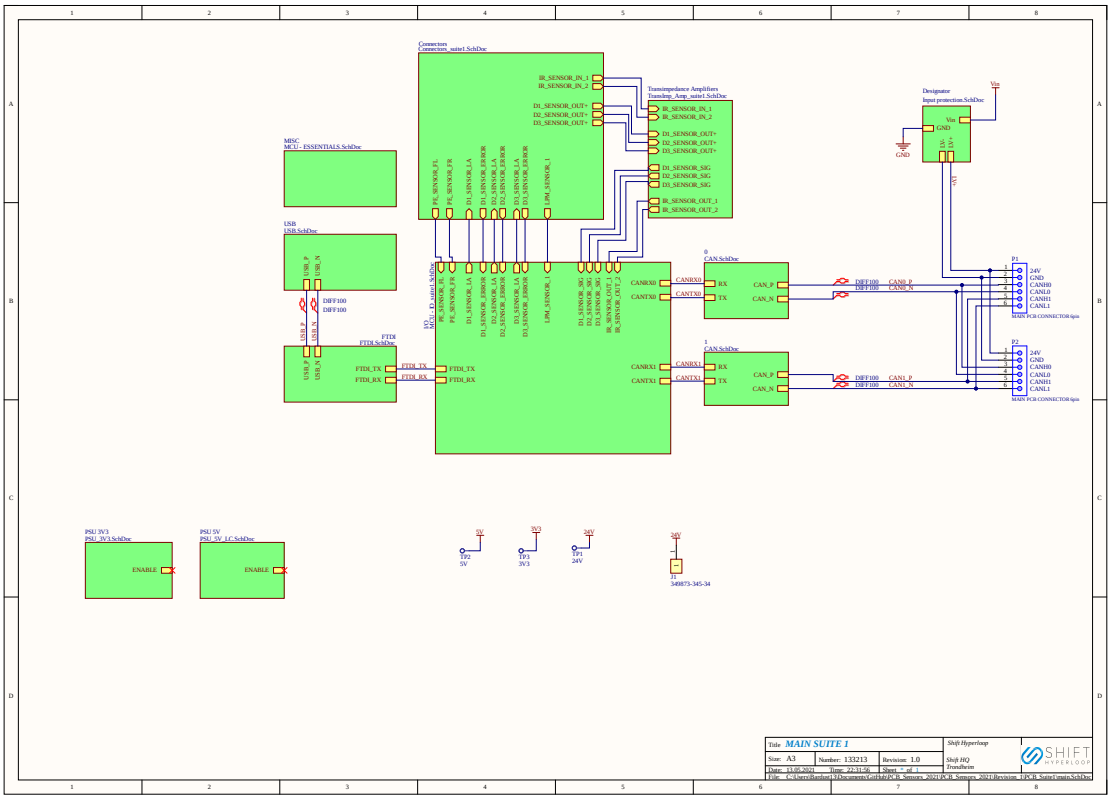
Appendix

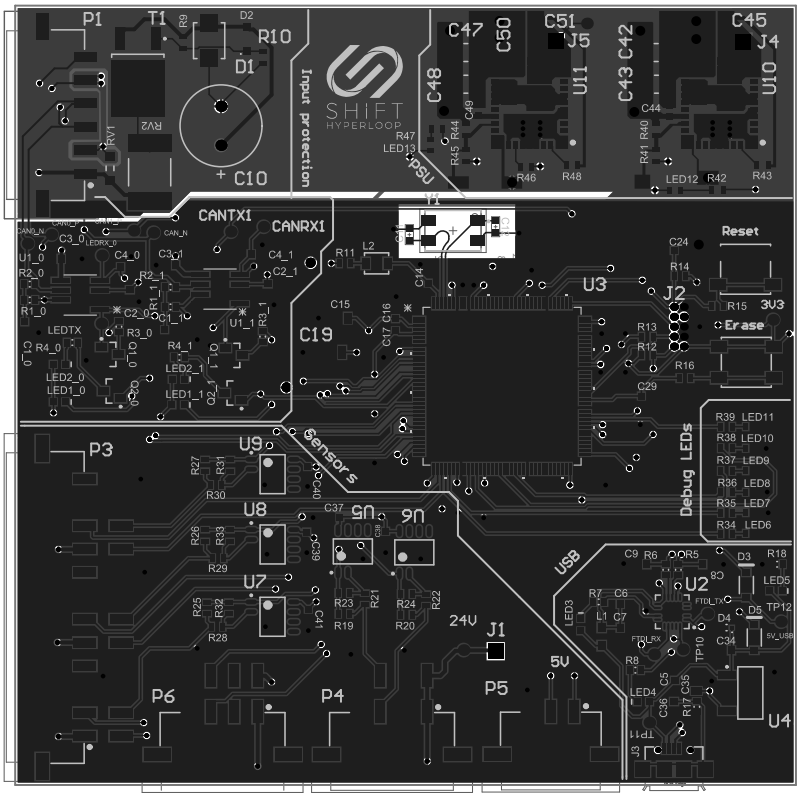
A Altium Suite 1 - Revision 1



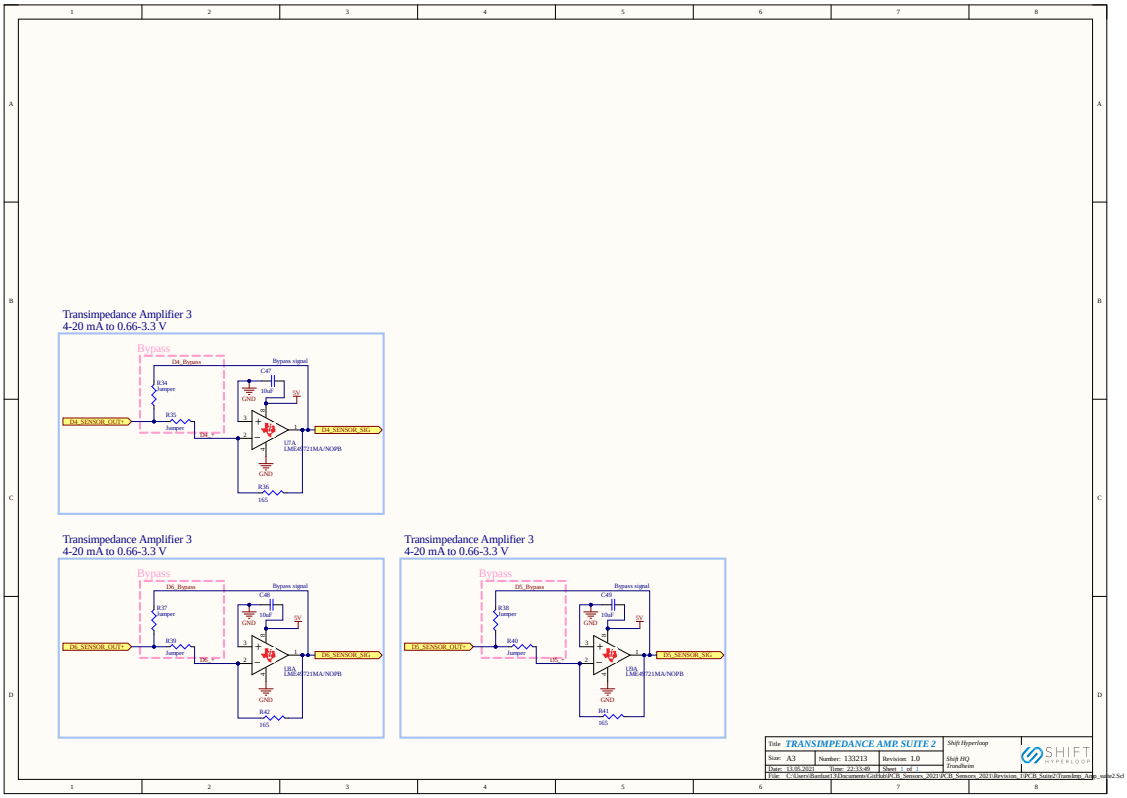


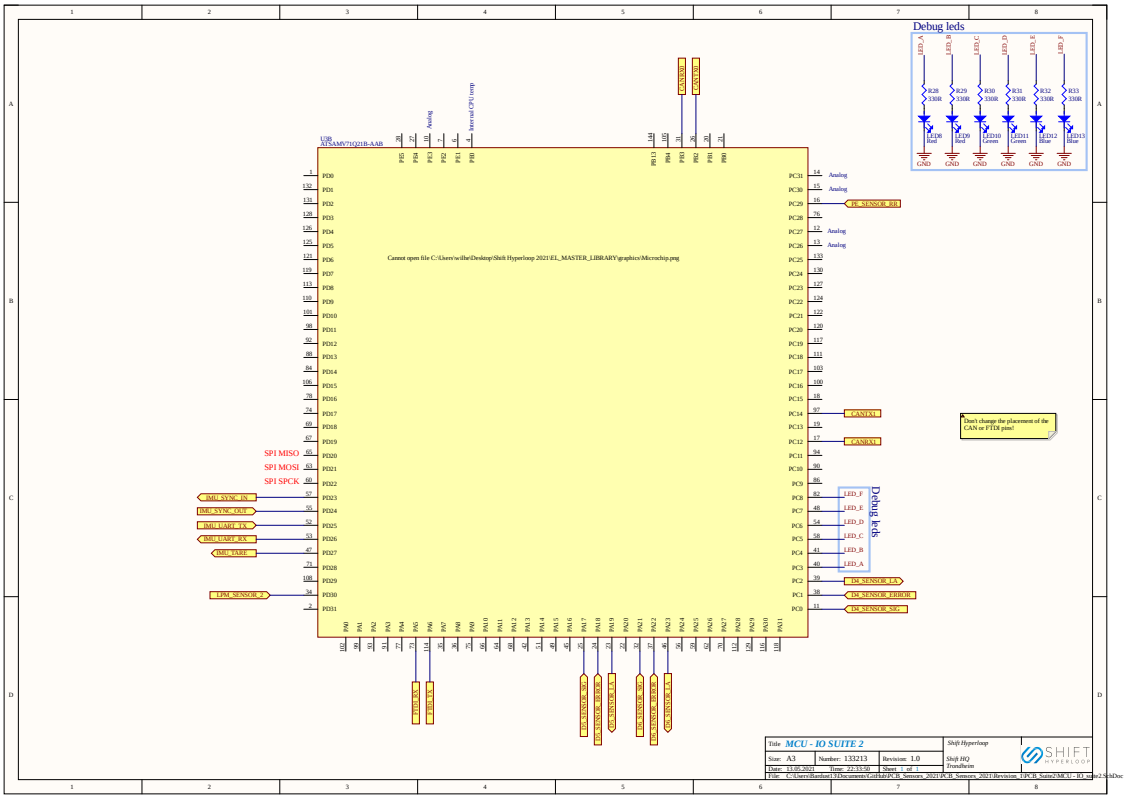

Distance sensor
PE, X-PRO XP10
IR Sensor
LPM sensor
Title CONNECTORS SUITE 1
Size A3
Number 133213
Revision 1.0
Shift Hyperloop
Shift HQ
Trondheim

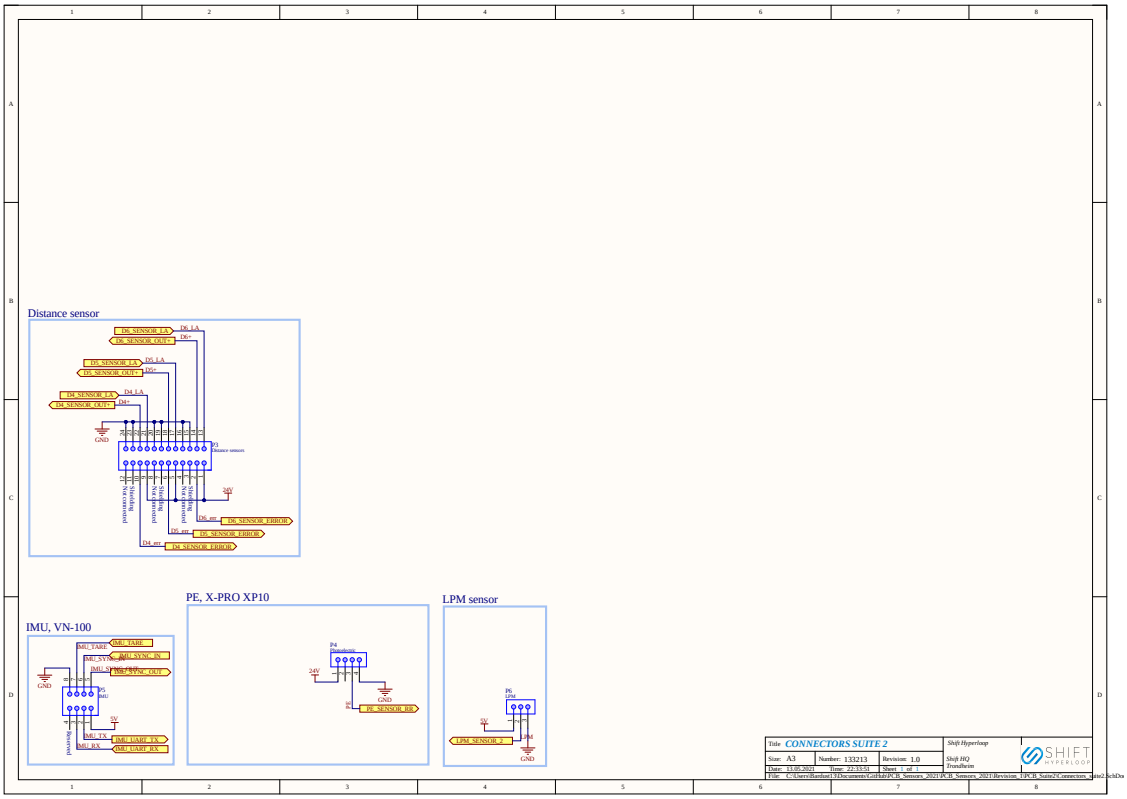

Connectors
Connectors_suite1.SchDoc
Transimpedance Amplifiers
Transimp_Amp_suite1.SchDoc
MISC
MCU - ESSENTIALS.SchDoc
USB
USB.SchDoc
FTDI
FTDI.SchDoc
CAN.SchDoc
Designator
Input protection.SchDoc
PSU 3V3
PSU_3V3.SchDoc
PSU 5V
PSU_5V_LC.SchDoc
Title MAIN SUITE 1
Size A3
Number 133213
Revision 1.0
Shift Hyperloop
Shift HQ
Trondheim

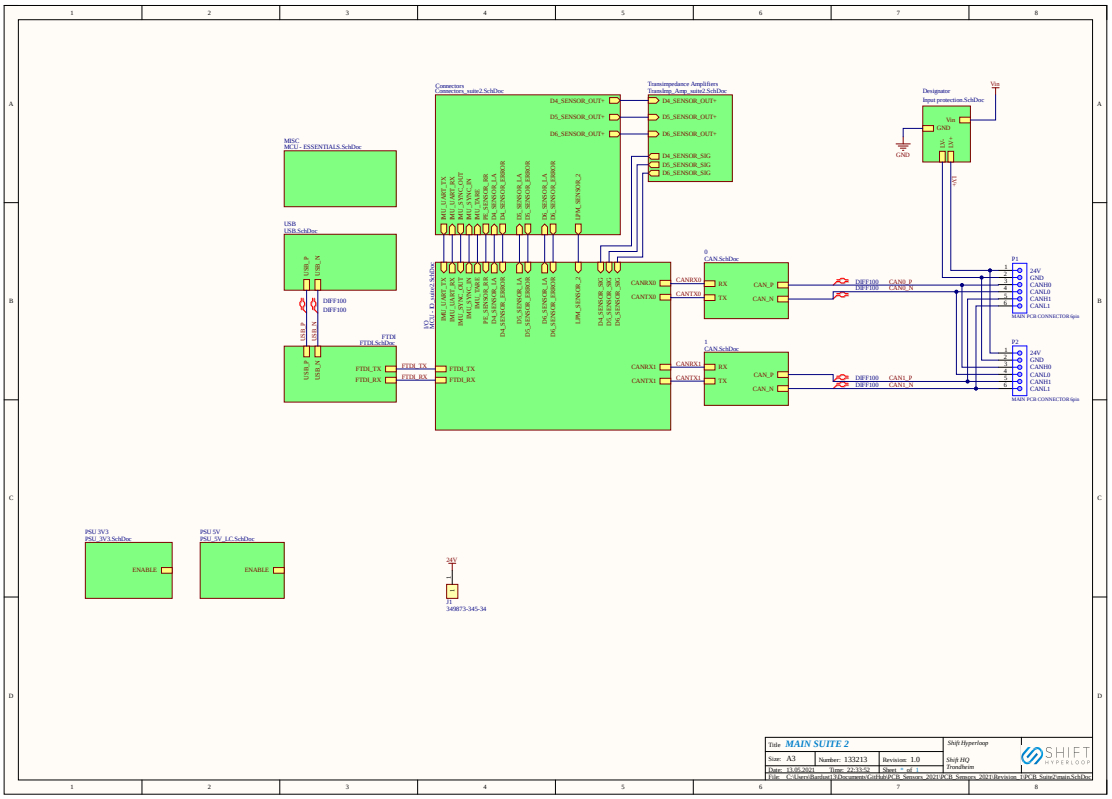

P1
T1
D2
R10
D1
C10
Input protection
SHIFT
HYPERLOOP
C47
C50
C51
J5
U11
C48
C43 C42
C45
J4
U10
psu
CANTX1
CANRX1
U3
J2
Reset
3V3
Erase
C19
C15
LEDTX
LED2_0
LED1_0
LED2_1
LED1_1
Debug LEDs
P3
U9
Sensors
U8
U5
U6
U7
USB
U2
24V
J1
5V
P6
P4
P5
U4

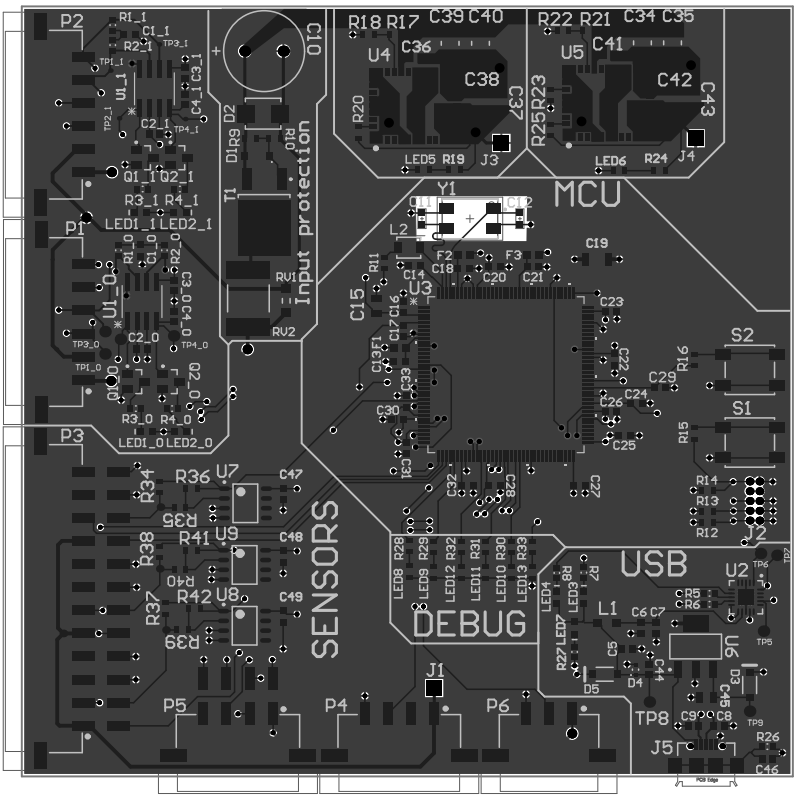
B Altium Suite 2 - Revision 1



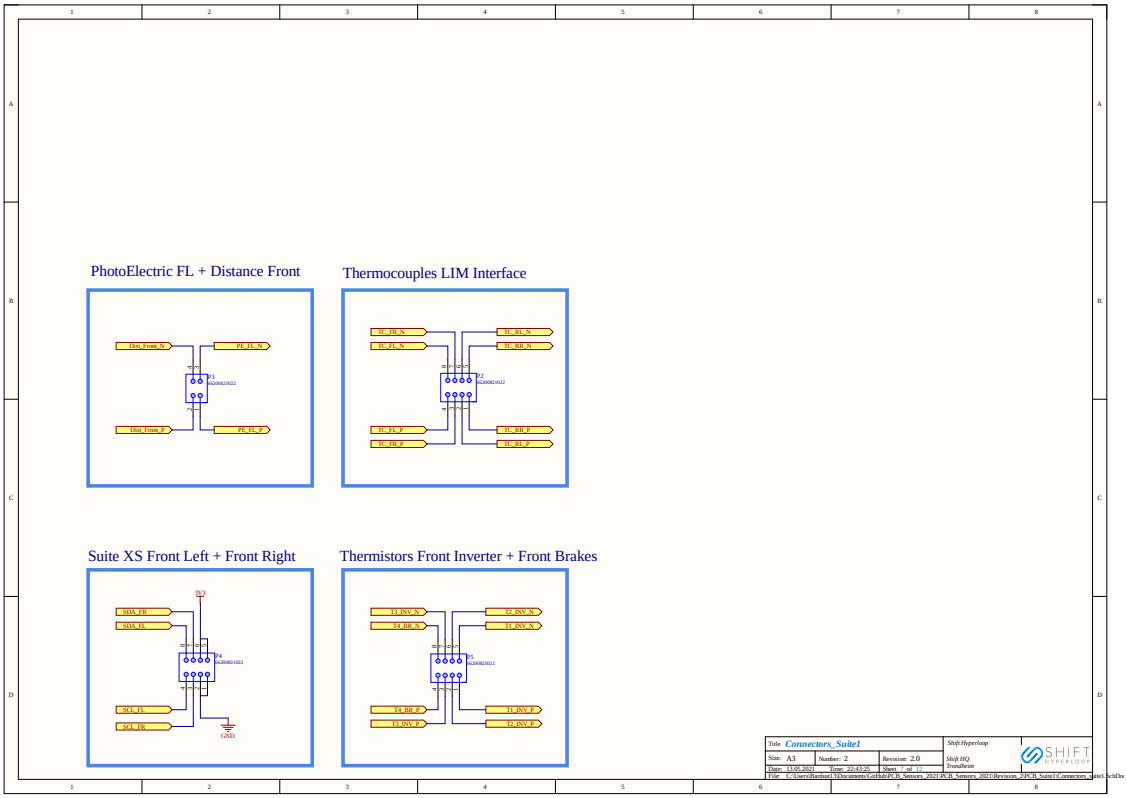


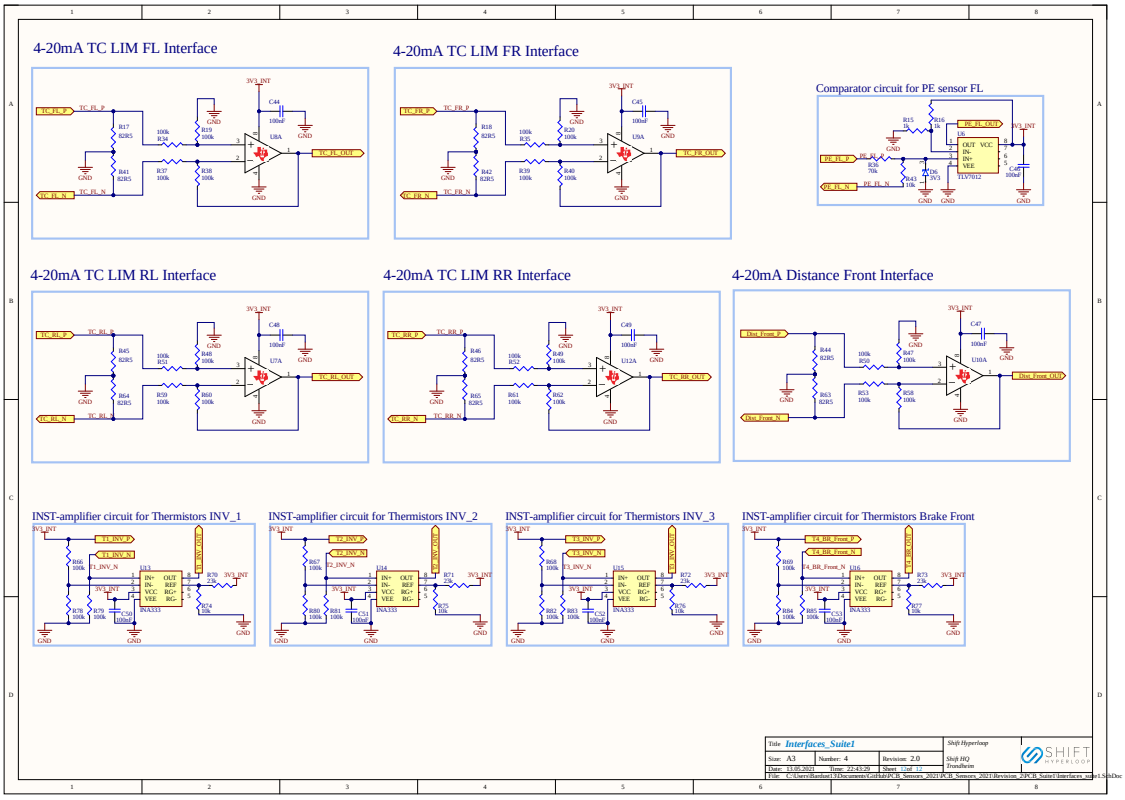

Distance sensor
IMU, VN-100
PE, X-PRO XP10
LPM sensor
CONNECTORS SUITE 2
Shift Hyperloop
Shift HQ
Trondheim

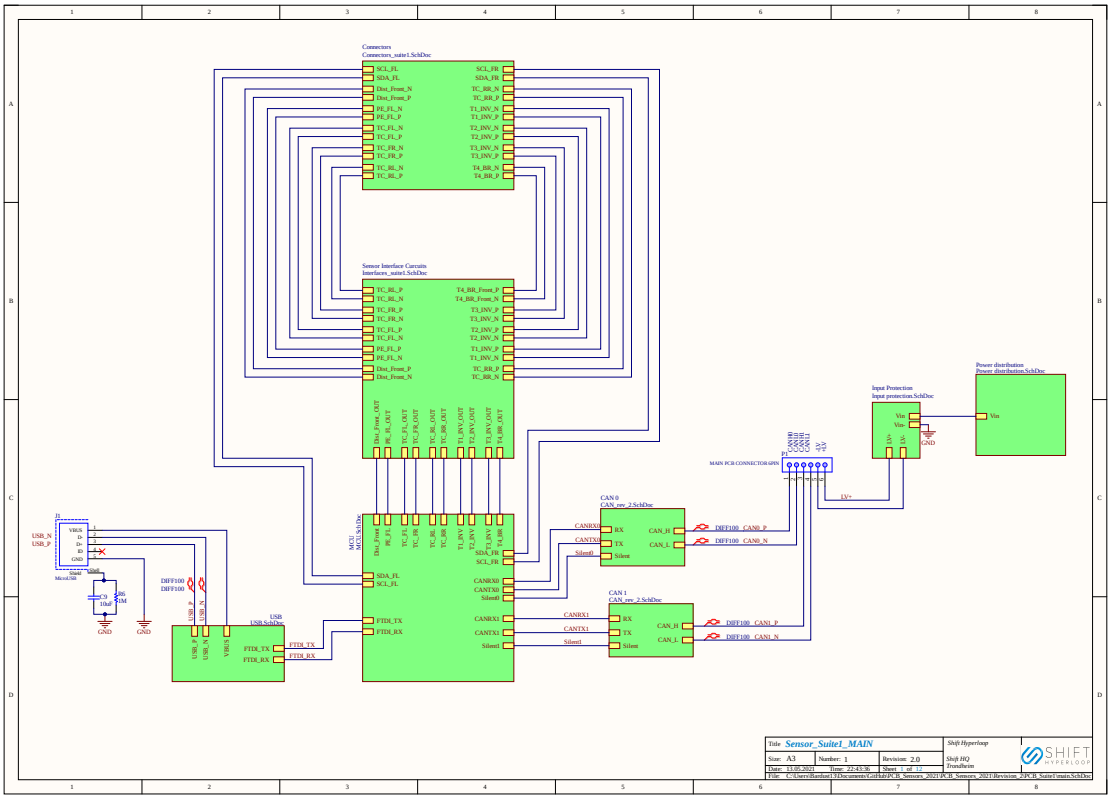

MAIN SUITE 2
Shift Hyperloop
Shift HQ
Trondheim

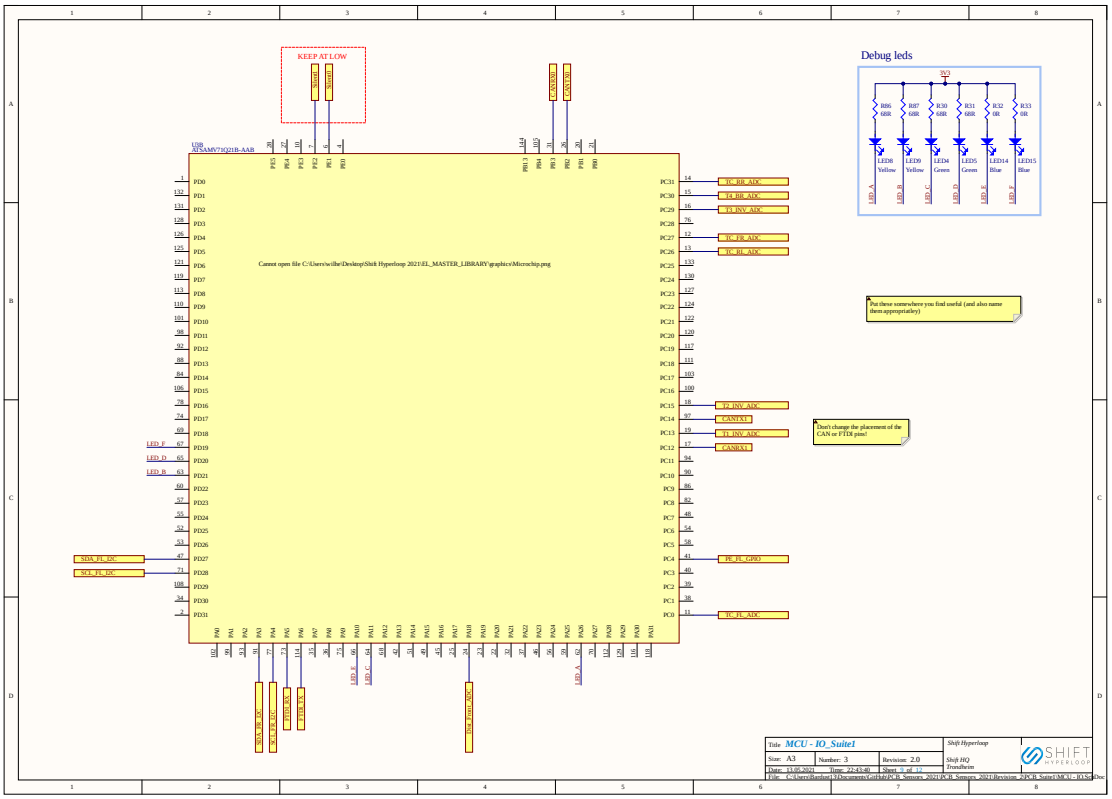

MCU
SENSORS
DEBUG
USB
input protection
P1
P2
P3
P4
P5
P6
J1
J2
J3
J4
J5
S1
S2
U1_0
U1_1
U2
U3
U4
U5
U6
U7
U8
U9
Y1
L1
L2
TP8

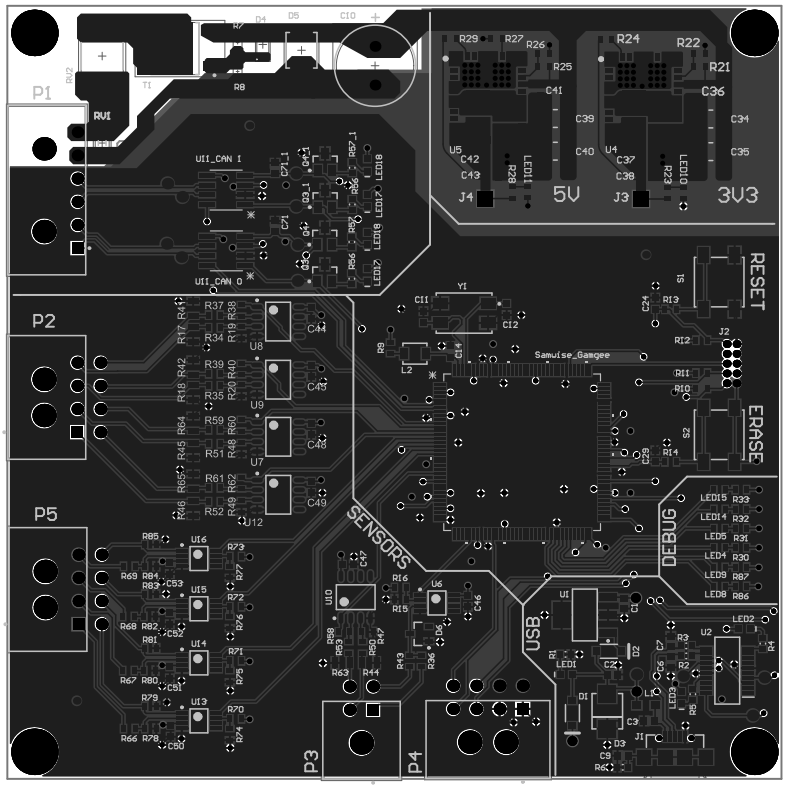
C Altium Suite 1 - Revision 2



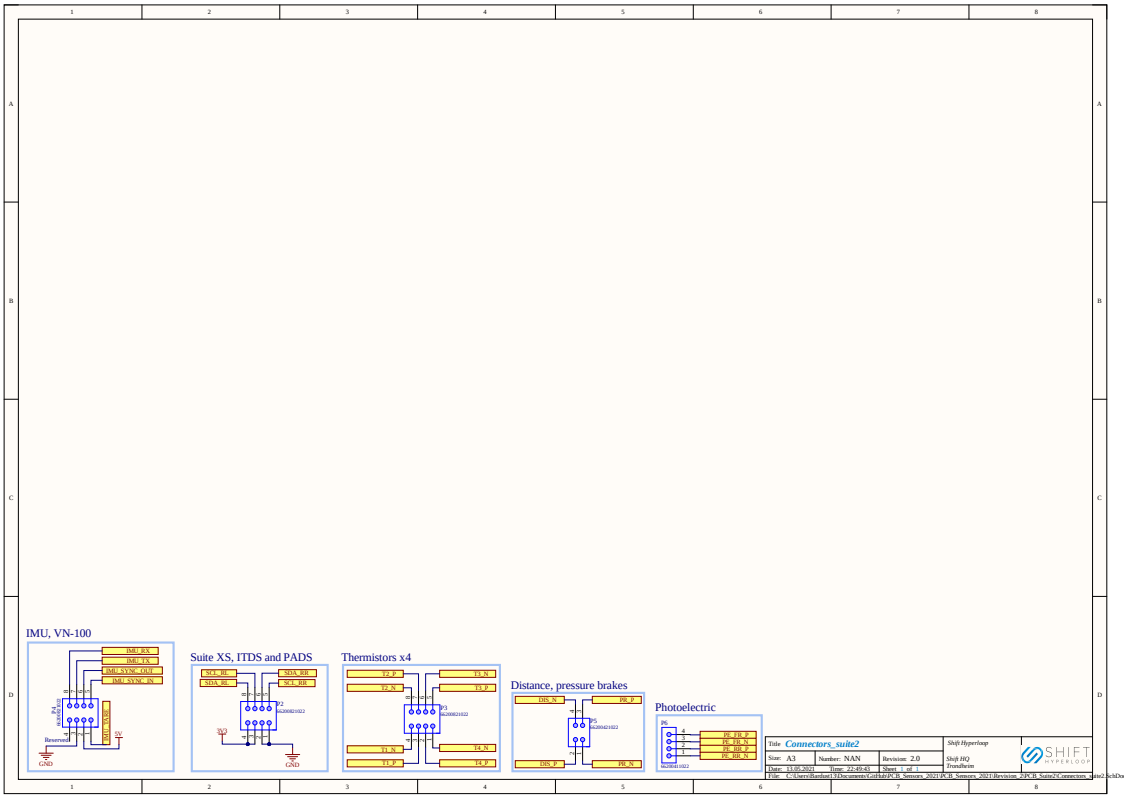


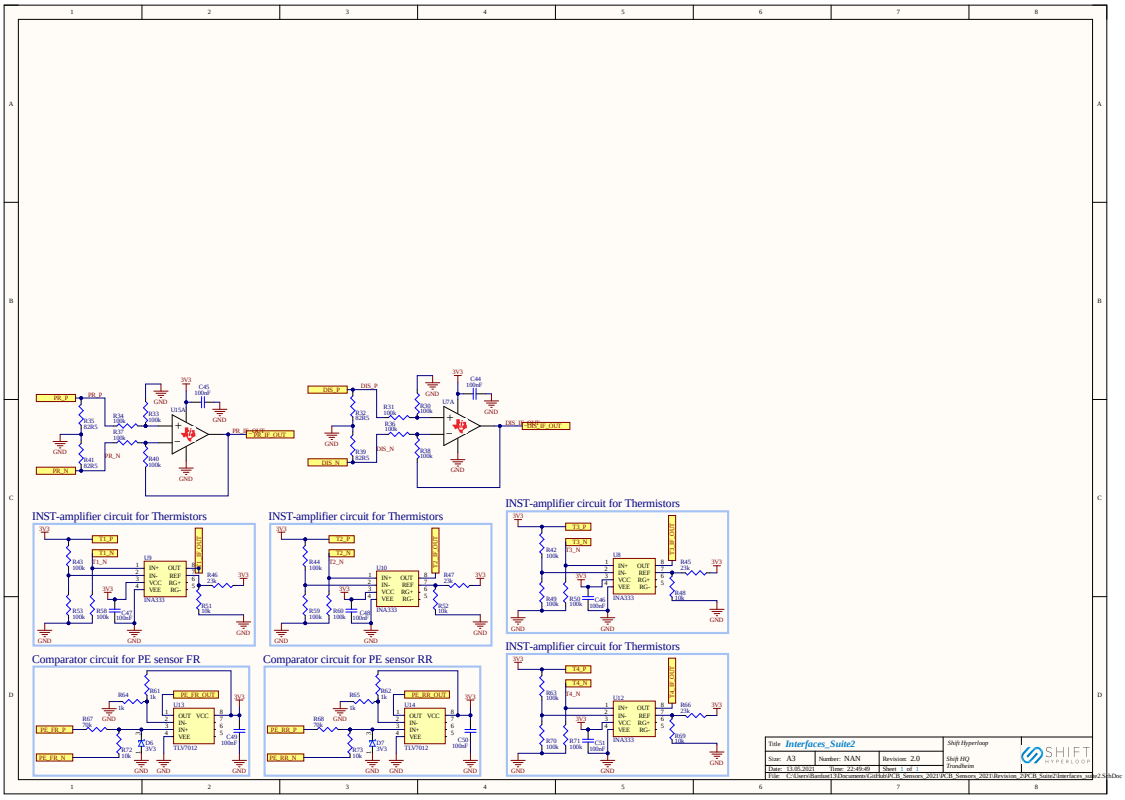

Connectors
Connectors_suite1.SchDoc
Sensor Interface Curcuits
Interfaces_suite1.SchDoc
MCU
MCU.SchDoc
USB
USB.SchDoc
CAN 0
CAN_rev_2.SchDoc
CAN 1
CAN_rev_2.SchDoc
Input Protection
Input protection.SchDoc
Power distribution
Power distribution.SchDoc
MAIN PCB CONNECTOR 6PIN
Title Sensor_Suite1_MAIN
Shift Hyperloop
Shift HQ
Trondheim
Size: A3
Number: 1
Revision: 2.0
Date: 13.05.2021
Time: 22:43:36
Sheet 1 of 12

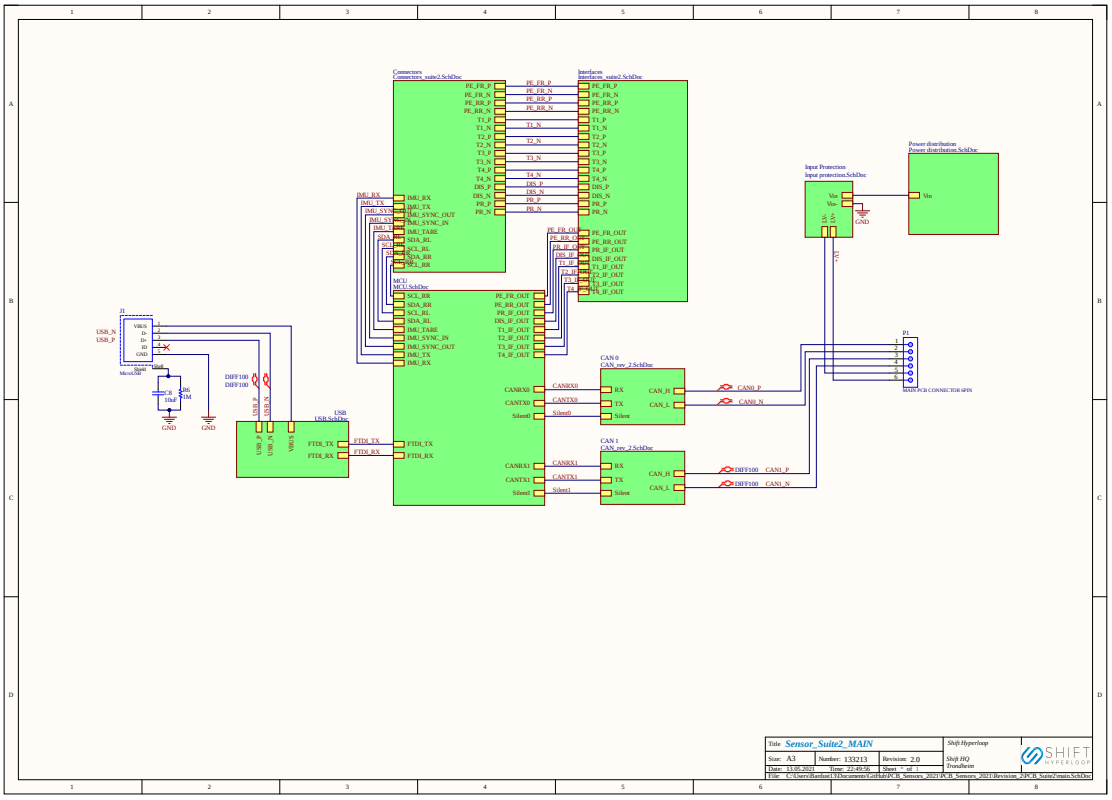

KEEP AT LOW
Debug leds
Cannot open file C:\Users\wilhe\Desktop\Shift Hyperloop 2021\EL_MASTER_LIBRARY\graphics\Microchip.png
Put these somewhere you find usefull (and also name them appropriatley)
Don't change the placement of the CAN or FTDI pins!
Title MCU - IO_Suite1
Shift Hyperloop
Shift HQ
Trondheim
Size: A3
Number: 3
Revision: 2.0
Date: 13.05.2021
Time: 22:43:40
Sheet 3 of 12

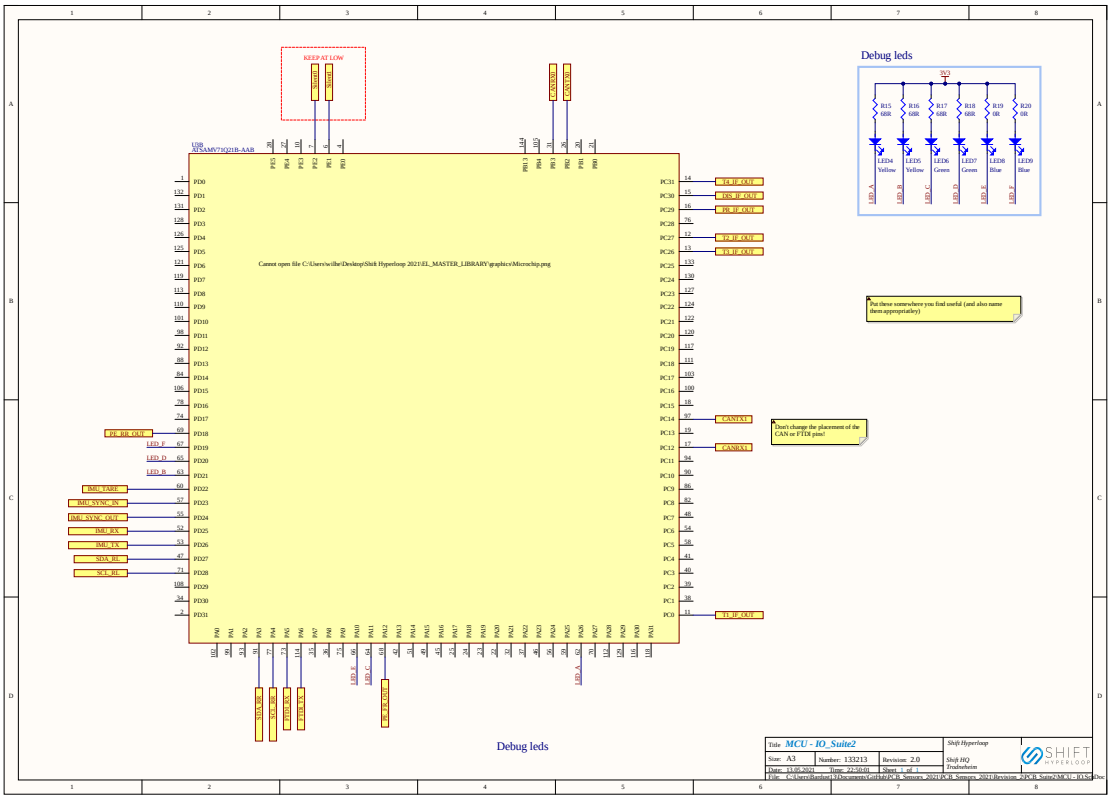

P1
P2
P3
P4
P5
5V
3V3
RESET
ERASE
DEBUG
USB
SENSORS
Samuise_Gamgee

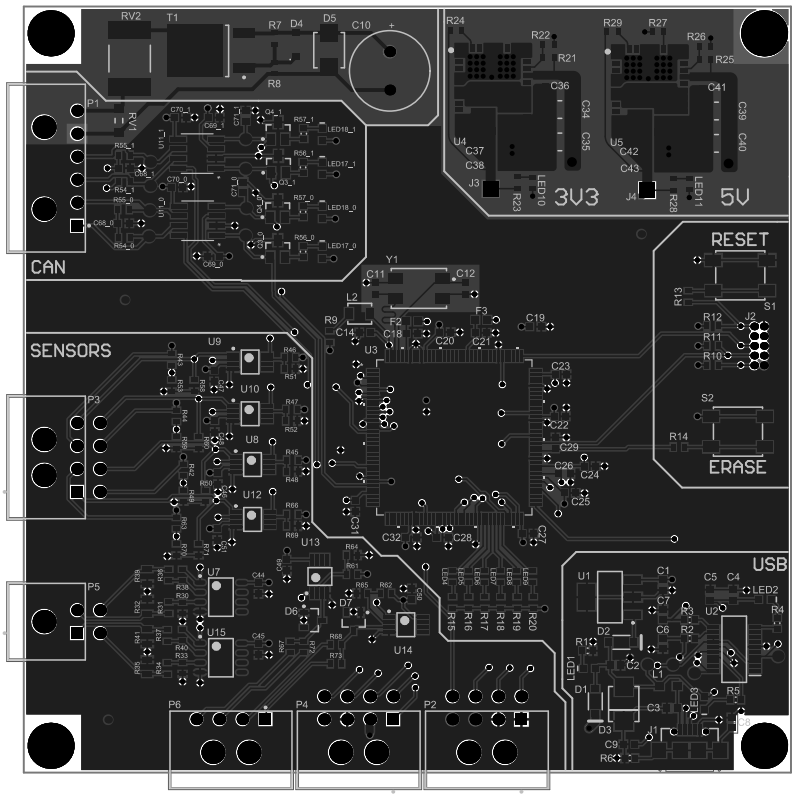
D Altium Suite 2 - Revision 2



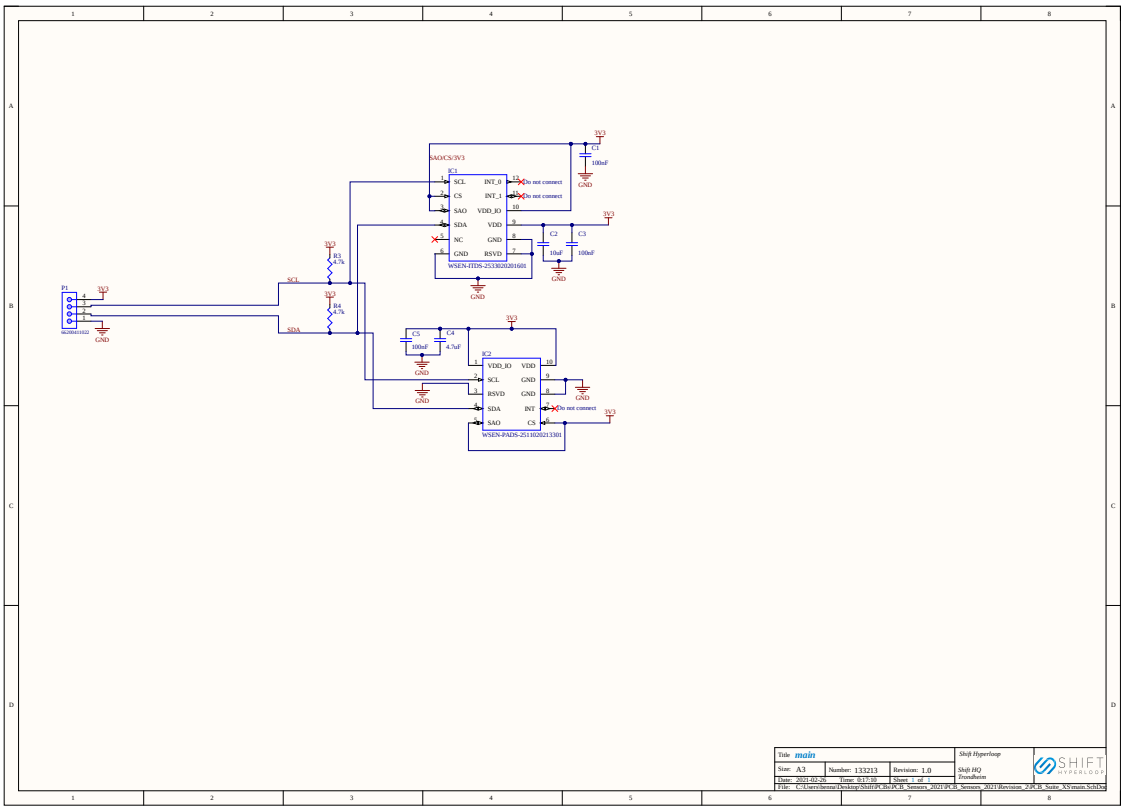


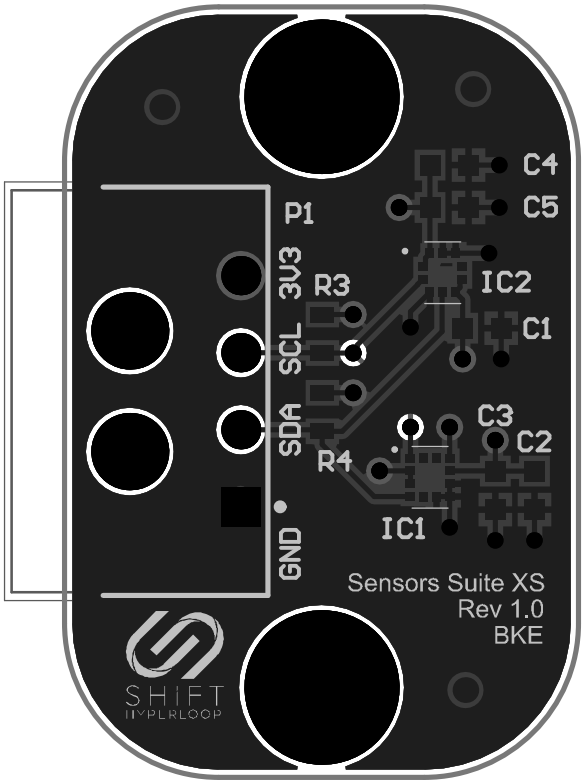

Sensor_Suite2_MAIN
Shift Hyperloop
Shift HQ
Trondheim
Size: A3
Number: 133213
Revision: 2.0
Date: 13.05.2021
Time: 22:49:56
Sheet 1 of 1
File: C:\Users\Bardast1\Documents\GitHub\PCB_Sensors_2021\PCB_Sensors_2021\Revision_2\PCB_Suite2\main.SchDoc


Debug leds
Cannot open file C:\Users\wilhe\Desktop\Shift Hyperloop 2021\EL_MASTER_LIBRARY\graphics\Microchip.png
Debug leds
MCU - IO_Suite2
Shift Hyperloop
Shift HQ
Trondheim
Size: A3
Number: 133213
Revision: 2.0
Date: 13.05.2021
Time: 22:50:03
Sheet 1 of 1

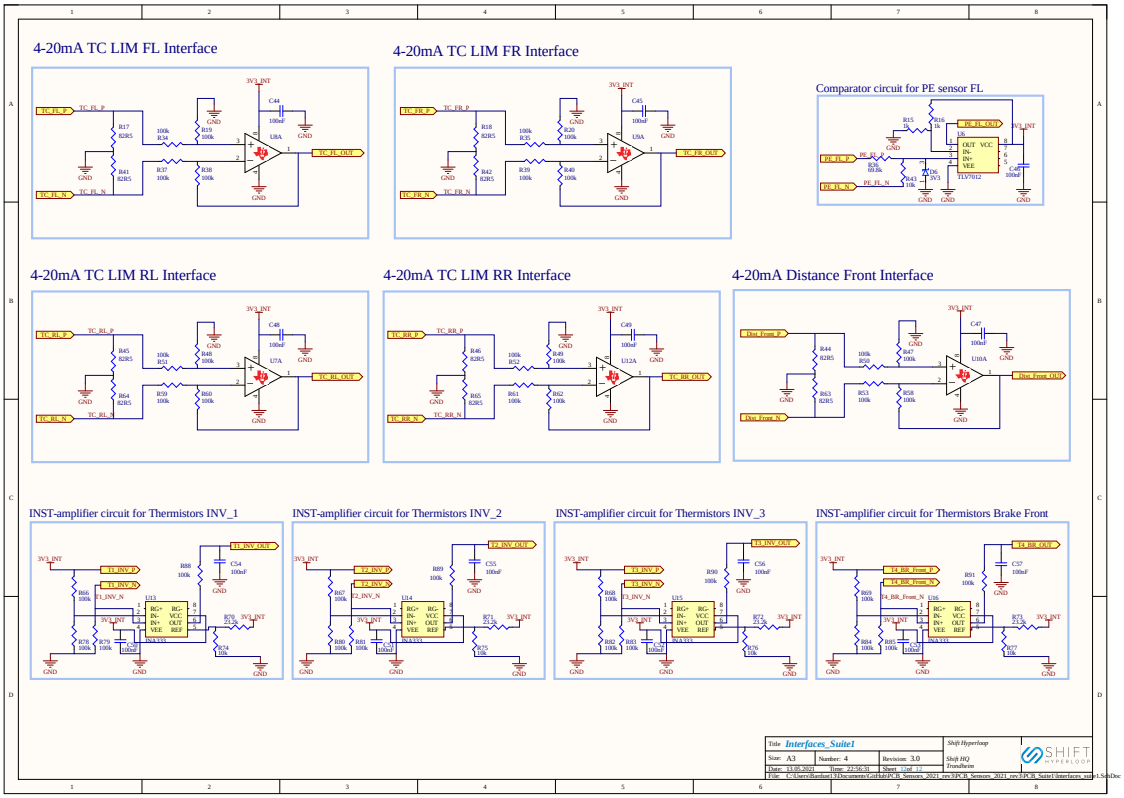

CAN
SENSORS
RESET
ERASE
USB
3V3
5V

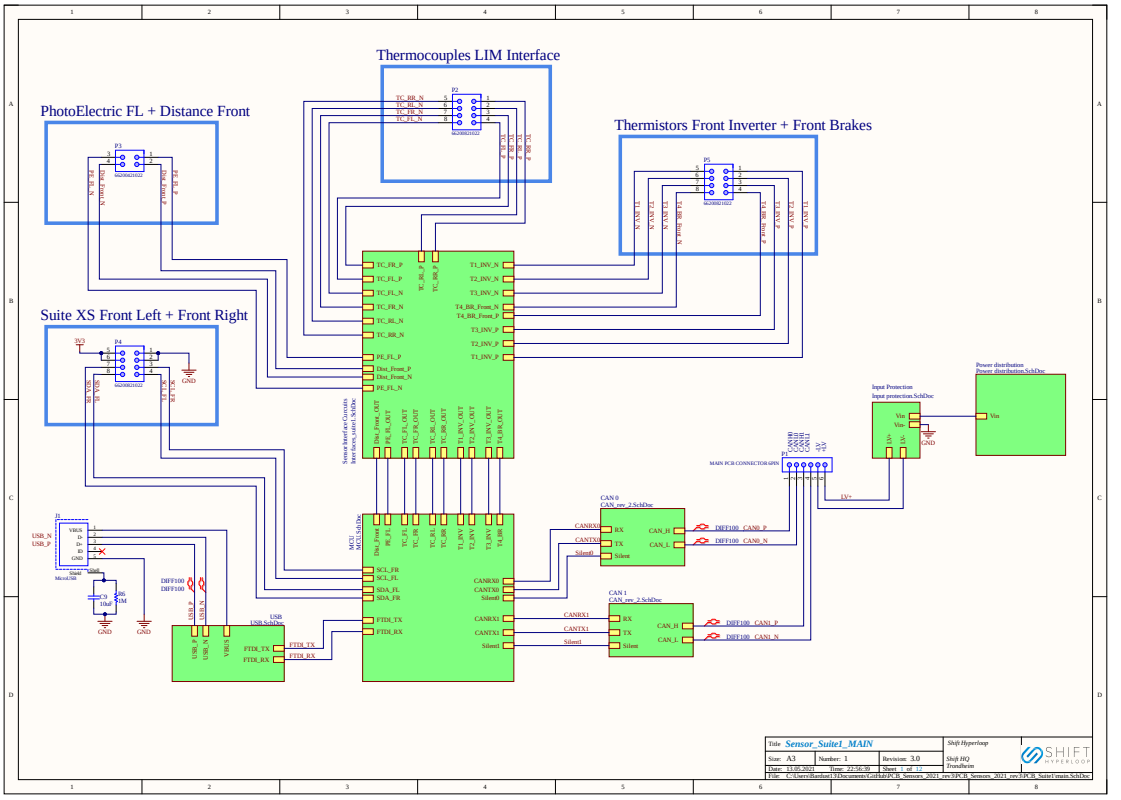
E Altium Suite XS - Revision 1

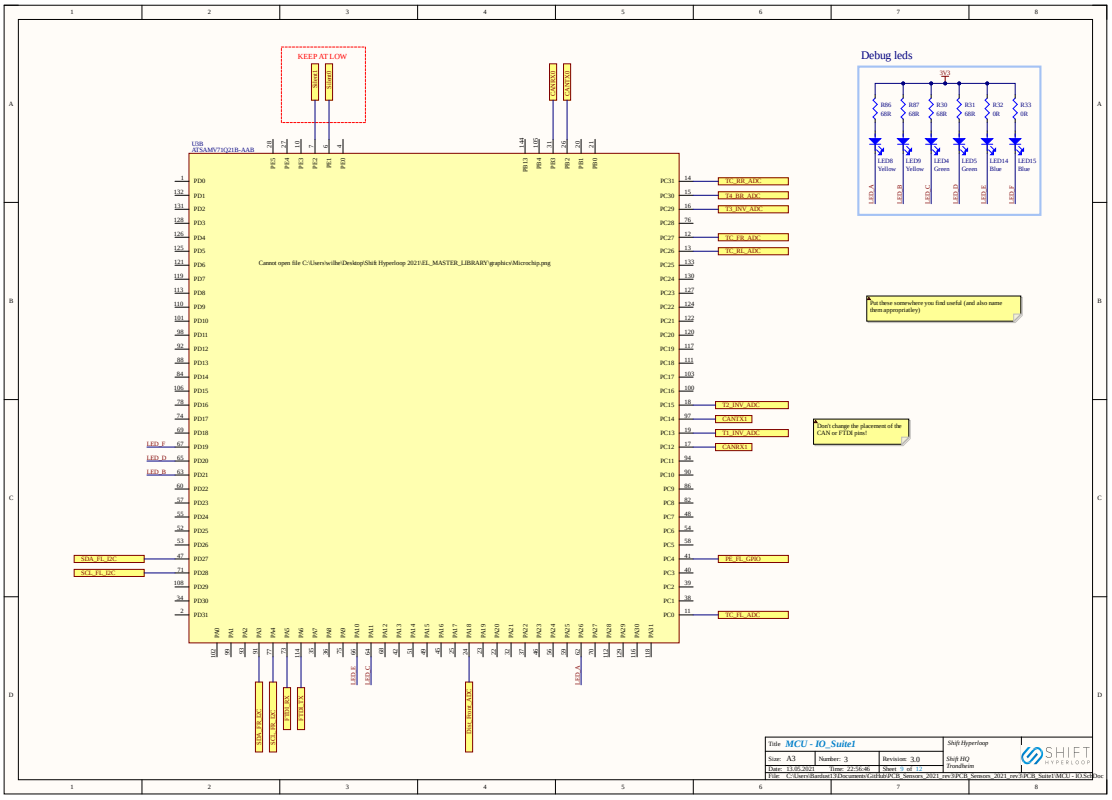


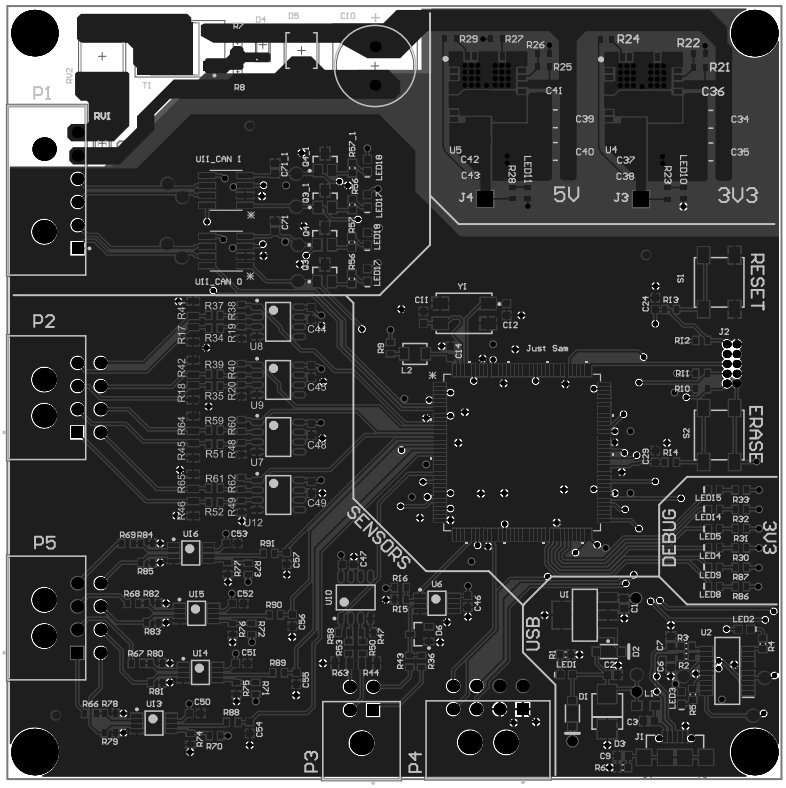


F Altium Suite 1 - Revision 3

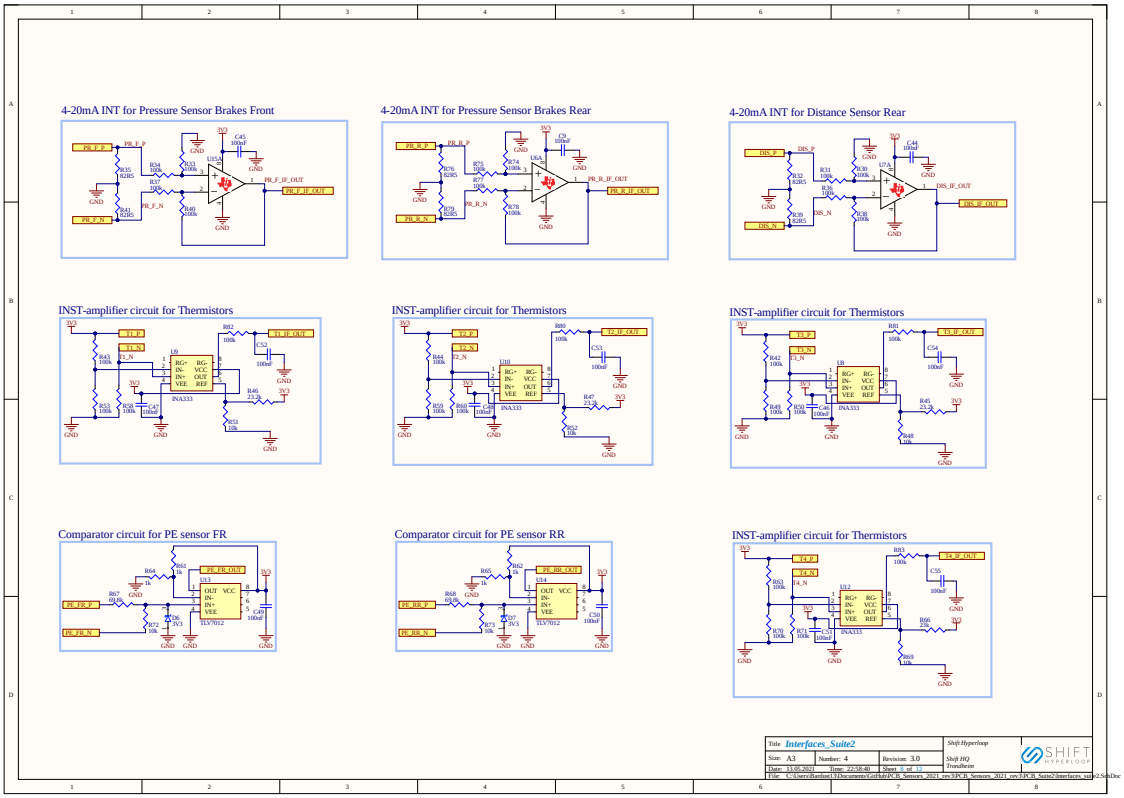


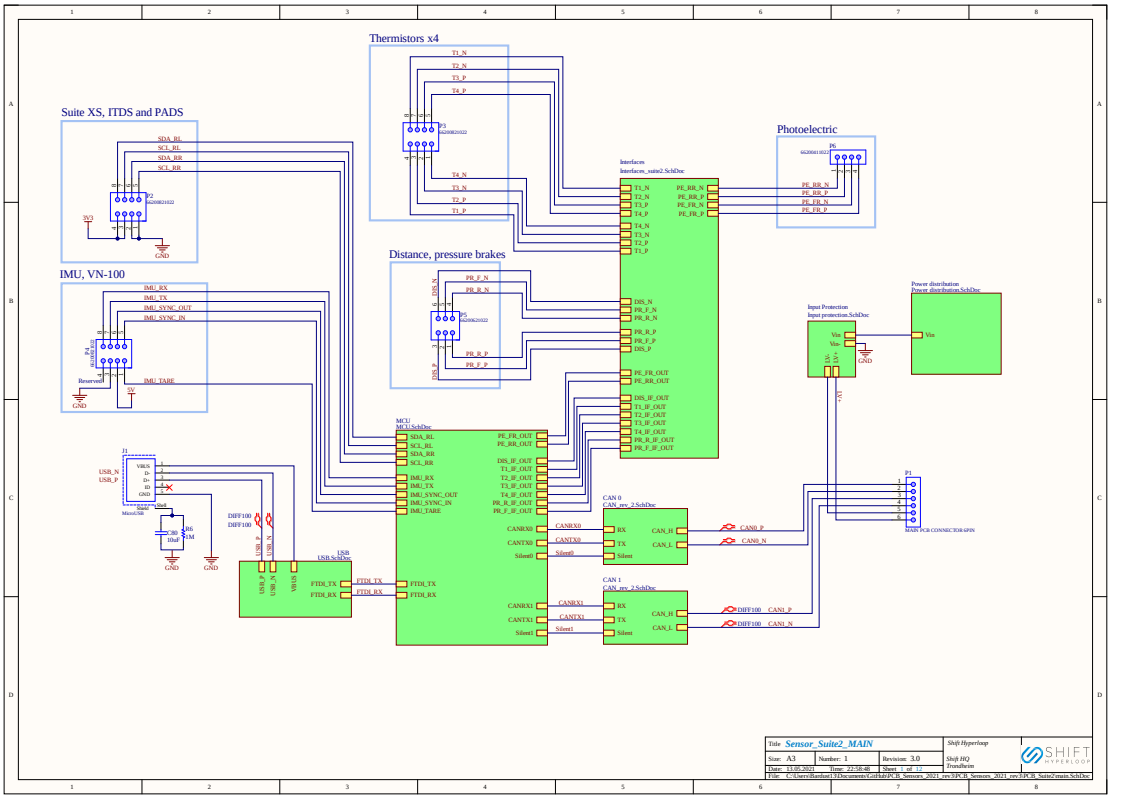


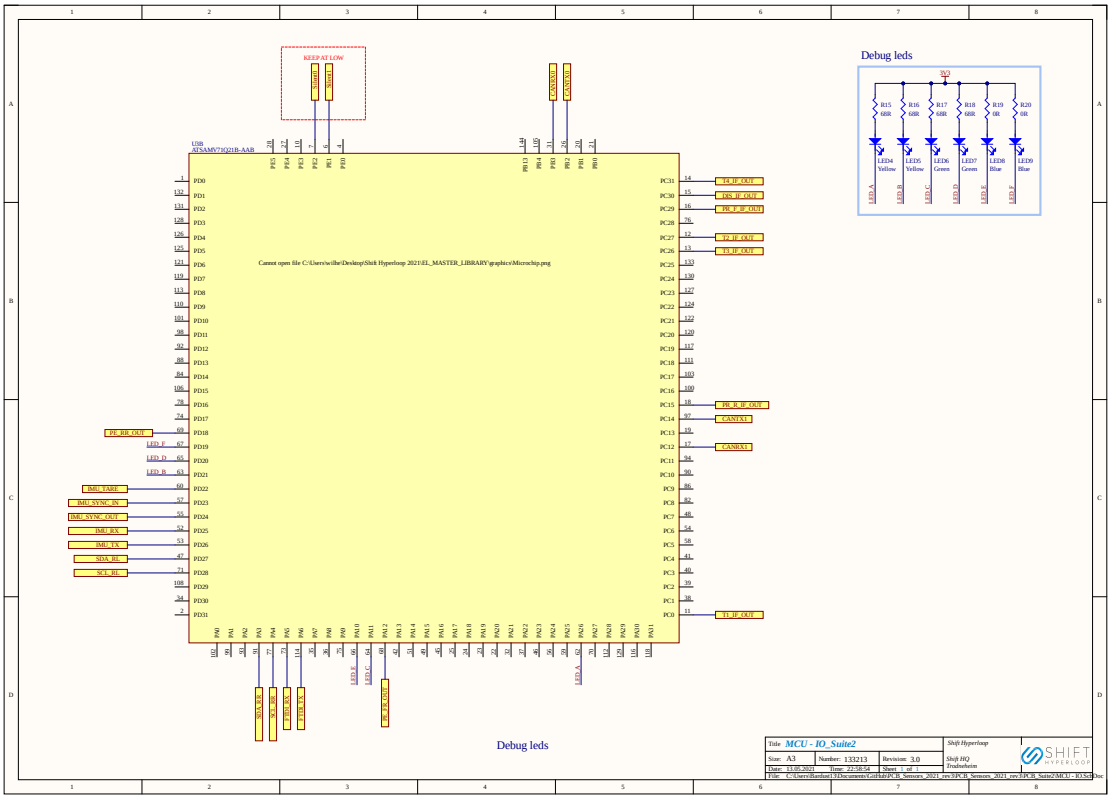

KEEP AT LOW
Debug leds
Put these somewhere you find useful (and also name them appropriately)
Don't change the placement of the CAN or FTDI pins!
MCU - IO_Suite1
Shift Hyperloop
Shift HQ
Trondheim

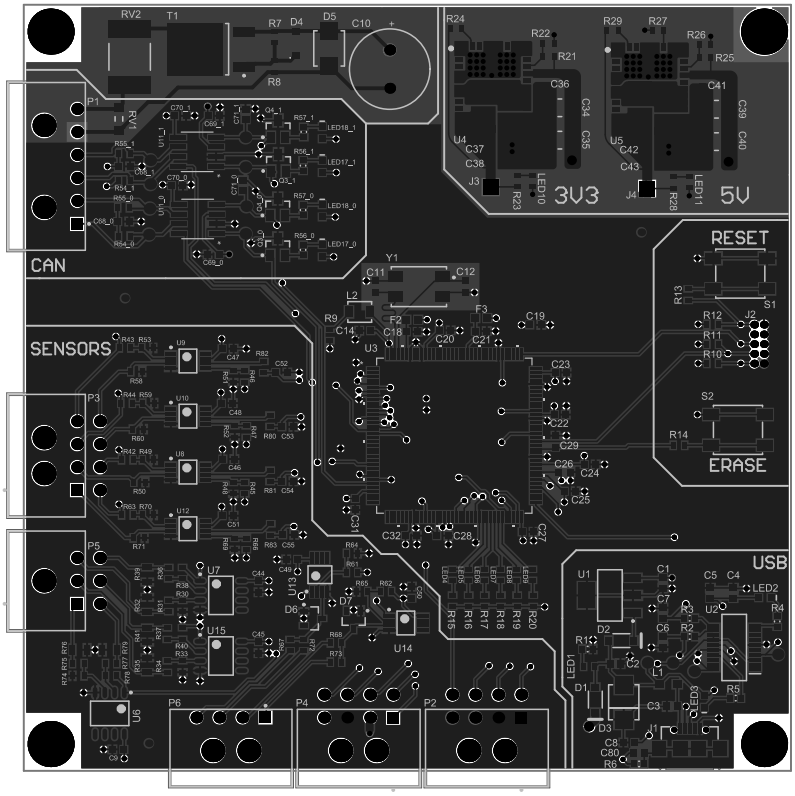

P1
P2
P3
P4
P5
RESET
ERASE
SENSORS
DEBUG
USB
5V
3V3
Just Sam

G Altium Suite 2 - Revision 3






KEEP AT LOW
Debug leds
Debug leds
Cannot open file C:\Users\willw\Desktop\Shift Hyperloop 2021\EL_MASTER_LIBRARY\graphics\Microchip.png
Title MCU - IO_Suite2
Shift Hyperloop
Shift HQ
Tradiselsein
Size: A3
Number: 133213
Revision: 3.0
Date: 13.05.2021
Time: 22:58:54
Sheet 1 of 1
File: C:\Users\Bardan\Documents\CutSat\PCB_Sensors_2021_rev3\PCB_Sensors_2021_rev3\PCB_Suite2\MCU - IO.SchDoc


CAN
SENSORS
RESET
ERASE
USB
3V3
5V

H Test plan

Sensor type	Purpose	Specifications	How will it be tested?
Optical distance sensors: Pepperl+Fuchs OMT300-R200-IEP-IO-V1	Measure distance to track in Y-plane	Supply voltage: 24 V Current draw: < 25 mA at 24 V no-load (+ 4-20 mA output) Range: 100-300 mm Resolution: 0,1 mm Recovery time: 2 ms	1. Initial sensor function testing with breadboard and simple interface circuit. Oscilloscope and multimeter. 2. Test of sensor on rev2 PCB with complete interface circuit, ADC and simple test code. Oscilloscope and serial monitor. 3. Full system test with CAN-BUS transmission of data packets.
Photoelectric sensors: TRI-TRONICS Smarteye X-PRO XP10 R5	Detecting navigation markers along the track. Secondary speed calculation (in combination with IMU)	Supply voltage: 24 V Current draw: >45 mA Switching freq.: 50 kHz Repeatability: 5 us Range: 3 ft.	1. Initial sensor function testing with breadboard and simple interface circuit. Oscilloscope and multimeter. 2. Test of sensor on rev2 PCB with complete interface circuit and simple test code. Oscilloscope and serial monitor. 3. Full system test with CAN-BUS transmission of data packets.
IMU: VECTORNAV VN-100 Rugged	Pod acceleration and pitch/roll.	Supply voltage: 5 V Current draw: 40 mA Acc bandwidth: 260 Hz Angular res.: 0,001 °	1. Initial testing with VECTORNAV software and GUI. 2. Complete test with UART to microcontroller communication and test code. 3. Full system test with CAN-BUS transmission of data packets.
Temperature transmitter for thermocouples: RS PRO 1500V insulated (192-6538)	Interface between thermocouple and sensor PCB	Supply voltage: 24 V Current draw: < 20 mA Input: thermocouple Output: 4-20 mA Temp. input range: -200 - 850 °C Cold junction compensation Galvanically isolated input to output 1500 V AC	1. Initial transmitter function testing with thermocouple, breadboard and simple interface circuit. Oscilloscope and multimeter. 2. Testing transmitter on rev2 PCB with complete interface circuit, ADC and simple test code. Oscilloscope and serial monitor. 3. Full system test with CAN-BUS transmission of data packets.
Thermocouples: RS PRO Type T (847-9687)	Temperature measurement in the LIMs (cast into the epoxy)	Thermocouple type: T Temperature range: -75 - 250 °C Probe diameter: 1/0,3 mm Tip: exposed welded junction Insulation: Teflon PFA	1. Same as for the temperature transmitter. 2. Test with different temperatures (cold, normal, hot). 3. Test with ice water for calibration at 0 degrees.
Pressure sensors (hydraulic fluid): Hydac HDA 4446-A-250-000	Pressure in brake system. Front and rear separated.	Supply voltage: 24 V Current draw: < 20 mA Range: 250 bar Overload / burst: 500 / 1000 bar Accuracy: 0,5 % FS Rise time: < 1 ms Vibration resistance: < 20 g (DIN EN 60068-2-6 at 10 ... 500 Hz)	1. Initial sensor function testing with breadboard and simple interface circuit. Oscilloscope and multimeter. 2. Test of sensor on rev2 PCB with complete interface circuit, ADC and simple test code. Oscilloscope and serial monitor. 3. Full system test with CAN-BUS transmission of data packets.
Accelerometers: Würth Elektronik WSEN-ITDS 3-axis	Suspension performance	Supply voltage: 3,3 V Resolution: 14 bit Range: +- 16 g Bandwidth: < 1600 Hz Interface: I^2C	1. Initial testing of fully soldered sensor PCB (Sensor Suite XS) connected to Sensor Suite 1 or 2 for testing of I^2C interface. 2. Full system test with CAN-BUS transmission of data packets.
Absolute pressure sensor (air): Würth Elektronik WSEN-PDS	Ambient air pressure Ambient air temperature	Supply voltage: 3,3 V Resolution (P): 24 bit Resolution (T): 16 bit Range (P): 26 - 126 kPa Range (T): -40 - 85 °C Abs. accuracy (P): ± 100 Pa Abs. accuracy (T): ± 1,5 °C Bandwidth: < 200 Hz Interface: I2C	1. Initial testing of fully soldered sensor PCB (Sensor Suite XS) connected to Sensor Suite 1 or 2 for testing of I^2C interface. 2. Full system test with CAN-BUS transmission of data packets.

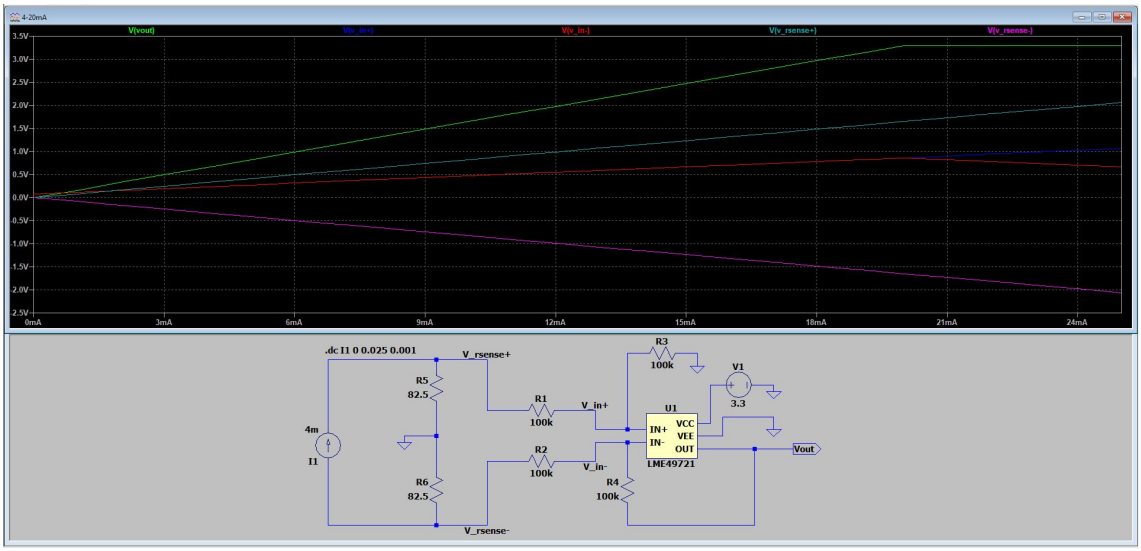
Requirement / What need to be tested	**Priority**	**Info**	**How it will be tested?**	**What is the criteria for it to pass?**	**Comment / expected time frame**	**Equipment?**
Short circuits 1	High	We test for short circuits before soldering anything on the PCB.	Use a multimeter with a buzzer. Connect one node to PCB ground, and place the other node on circuit nodes around the PCB.	▪ No short circuits	90-120 minutes Good knowledge of your PCB is required here. Have the schematics on Altium next to you while doing this to identify critical testing points.	▪ Multimeter with a buzzer
Short circuits 2	High	After soldering passive components (resistors, capacitors, LEDs, inductors), we test for short circuits again	Use a multimeter with a buzzer. Connect one node to PCB ground, and place the other node on circuit nodes around the PCB. Test the same nodes as in the previous point, and be spesifically thorough around the recently soldered components.	▪ No short circuits	90-120 minutes NOTE: there might be practical reasons to solder on some active components before passive components. Good knowledge of your PCB is required here. Have the schematics on Altium next to you while doing this.	▪ Multimeter with a buzzer
Short circuits 3	High	After soldering active components, we test for short circuits again	Use a multimeter with a buzzer. Connect one node to PCB ground, and place the other node on circuit nodes around the PCB. Test the same nodes as in the previous points, and especially around the recently soldered components. Also check if the solder from the first components is still in place if you used the soldering oven from Omega Verksted.	▪ No short circuits	90-120 minutes. Good knowledge of your PCB is required here. Have the schematics on Altium next to you while doing this.	▪ Multimeter with a buzzer
3V3 PSU low current	High	Needs to be tested: ▪ Output voltage level ▪ Output peak to peak ripple ▪ Output RMS ▪ No overheating (State indication needs seperate testing for their HC PSU).	Connect the signal generator to the PSU input with between 30 and 50 volts DC. Change this voltage up and down inside the range during testing to ensure that the PSU can handle different input voltages. All of the PSU testing should be done while the PSU is delivering operating currents. This can be done by making a simple external resistor-circuit. Do the math to find resistors that draw more than the PCB operating current (1A is more than sufficient) when connected to 3V3 volts. Connect the circuit to the PSU output with probes. Now we do the measurements. Use an oscilloscope to characterize voltage output from the PSU. Use cursors to measure peak levels. Use an in-built oscilloscope tool to measure RMS (if there is one). Use a thermal camera to measure temperatures. Run the PSU until the temperatures stabilize (unless	• 3V3 output voltage and RMS. • Atsam tolerates 20mV peak to peak ripple (datasheet page 1795). • Maximum temp is 105 degrees celcius according to data sheet. We probably want less than this.	60-90 minutes NOTE: do several tests. Don't trust one single measurement!	Required equipment: • Oscilloscope • Signal generator • Thermal camera

			it is about to melt...) and read temperatures. Notes: ▪ If you want to stress test the PSU, absolute max current is 2.5A ▪ State Indication needs separate testing for their high current PSU			
5V PSU low current	High	Needs to be tested: • Output voltage level • Output peak to peak ripple • Output RMS • Can handle operating currents without overheating Also consider stress testing with higher currents than operating currents. Max current is 2.5A but there is no need to go this high. (State indication needs seperate testing for their 5V HC PSU).	Connect the signal generator to the PSU input with between 30 and 50 volts DC. Change this voltage up and down inside the range during testing to ensure that the PSU can handle different input voltages. All of the PSU testing should be done while the PSU is delivering operating currents. This can be done by making a simple external resistor-circuit. Do the math to find resistors that draw more than the PCB operating current (1A is more than sufficient) when connected to 3V3 volts. Connect the circuit to the PSU output with probes. Now we do the measurements. Use an oscilloscope to characterize voltage output from the PSU. Use cursors to measure peak levels. Use an in-built oscilloscope tool to measure RMS (if there is one). Use a thermal camera to measure temperatures. Run the PSU until the temperatures stabilizes (unless it is about to melt...) and read temperatures. Notes: • If you want to stress test the PSU, absolute max current is 2.5A • State Indication needs separate testing for their high current PSU	• 5V output voltage and RMS. • Atsam tolerates maximum 20mV peak to peak ripple (datasheet page 1795) • Maximum temp of the PSU is 105 degrees celcius. We probably want less than this.	60-90 minutes NOTE: do several tests. Don't trust one single measurement!	Required equipment: • Oscilloscope • Signal generator • Thermal camera (Yamen has one)
USB 5V	Medium	Needs to be tested: • Output voltage level • Output peak to peak ripple • Output RMS	Testing should be done with operating currents, so use a simple external resistor-circuit like above as load. NOTE: depending on your USB port, your max current is roughly: ▪ USB 1.0 and 2.0: **0.5A** ▪ USB 3.0: **0.9A** ▪ USB 3.0 "charging downstream and dedicated charging ports": **1.5A** ▪ USB 3.1 and 3.2: **1.5A** Use these numbers to make sure that your external testing circuit	• 5V output voltage and RMS. • Atsam tolerates maximum 20mV peak to peak ripple (datasheet page 1795)	60-90 minutes NOTE: do several tests. Don't trust one single measurement!	Required equipment: ▪ Oscilloscope ▪ Computer

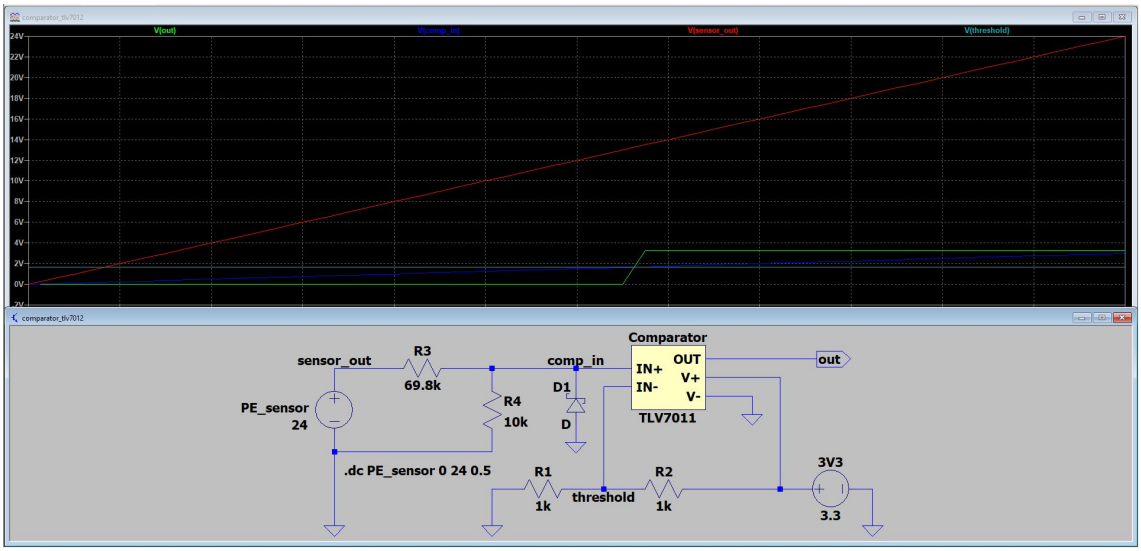
			draws less than the max current with good margin! Now we do the measurements. Use an oscilloscope to characterize the voltage at **one of your 5V test pads.** This is where the external resistor-circuit should be probed on as well. Use cursors to measure peak levels. Use an in-built oscilloscope tool to measure RMS (if there is one).			
USB 3V3	Medium	Needs to be tested: • Output voltage level • Output peak to peak ripple • Output RMS	Testing should be done with operating currents, so use a simple external resistor-circuit like above as load. NOTE: depending on your USB port, your max current is roughly: • USB 1.0 and 2.0: **0.5A** • USB 3.0: **0.9A** • USB 3.0 "charging downstream and dedicated charging ports": **1.5A** • USB 3.1 and 3.2: **1.5A** Read more about this here. Use these numbers to make sure that your external testing circuit draws less than the max current with good margin! Now we do the measurements. Use an oscilloscope to characterize the voltage at **one of your 3V3 test pads.** This is where the external resistor-circuit should be probed on as well. Use cursors to measure peak levels. Use an in-built oscilloscope tool to measure RMS (if there is one).	• 3V3 output voltage and RMS. • Atsam tolerates maximum 20mV peak to peak ripple (datasheet page 1795)	60-90 minutes NOTE: do several tests. Don't trust one single measurement!	Required equipment: • Oscilloscope • Computer
Oscillator/crystal	High	We test that the oscillator provides correct frequency	Connect an oscilloscope to the two pads XIN and XOUT next to the oscillator.	• 12MHz • Tolerance margin +- 50ppm (parts per million), which means 50*12= +- 720Hz tolerance	20 minutes Note: if your PCB has a via on the XIN/XOUT trace, this might cause troubles according to the data sheet.	• Oscilloscope
FTDI/flashing code	High	We test that we are able to flash code to the Atsam	Write a simple program in Atmel Studio that for example turns a debug LED on (for visuals) and flash it to the Atsam. NOTE: I'm not sure spesifically how to flash code, so someone else should probably write this section hehe	▪ Code needs to be flashed	30 minutes If your PCB doesn't have debug LEDs, you can use a GPIO and try to set it to high/low. Setting pins to high/low requires reading the datasheet to find the correct memory adresses.	▪ (Multimeter if you don't have debug LEDs) ▪ Atmel Eyes?
CAN	High	▪ Test that we are able to send and recieve messages using	Use test message IDs between two CAN-compatible PCBs to send and recieve messages with different datatypes.	▪ Every message needs to be recieved and sent without errors	15-60 minutes The embedded group might need to have som standardized test	▪ Two PCBs ▪ CAN-connection

		message IDs ▪ Test latency ▪ Datarate?	Test latency using a ping-pong program. PCB 1 sends a message, when PCB 2 reads it, it instantly sends a message back. PCB 1 measures the latency.	▪ Check latency standards to see what we tolerate!	code/bench that every PCB can use. This will save us a lot of time.	
GPIO header (if you have one)	Low	Needs testing: ▪ Pins match header number ▪ Correct voltages, no short circuits	Write code to set Atsam pins to alternating high/low and see that the GPIO header voltage matches their respective pin. (This can probably also be done by using a signal generator and connecting it to pads leading to the GPIO header. Saves a lot of time if it works, but practical reasons might stop you (no pads, probe too big, etc.). NOTE: if you do this, you need to make sure that you are using the correct pads to actually test that pins matches the header).	▪ Voltages have to be correct	45-60 Might take some time writing the code for setting all the pins	▪ Multimeter/oscilloscope ▪ (Signal generator)
Overheating	High	After doing all the other testing, we check for general overheating.	Run the PCB under operating conditions and see how the temperatures are. Wait for the temperatures to stabilize with operating conditions. Operate in normal room temperature.	▪ Not sure what the maximum temperature we can allow is, but definitely less than 100.	30 min	▪ Thermal camera

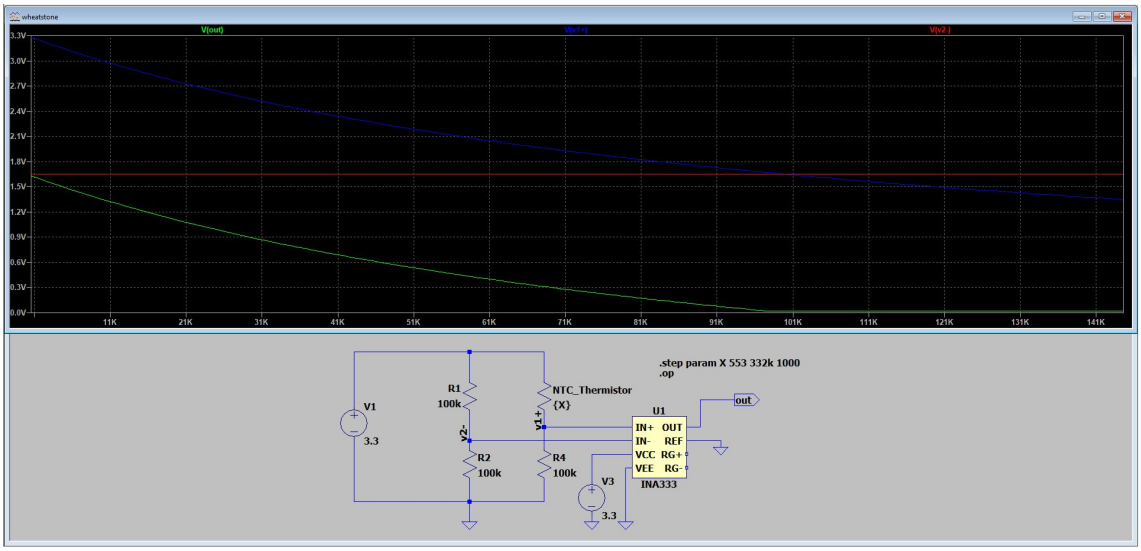
I LTspice - simulations



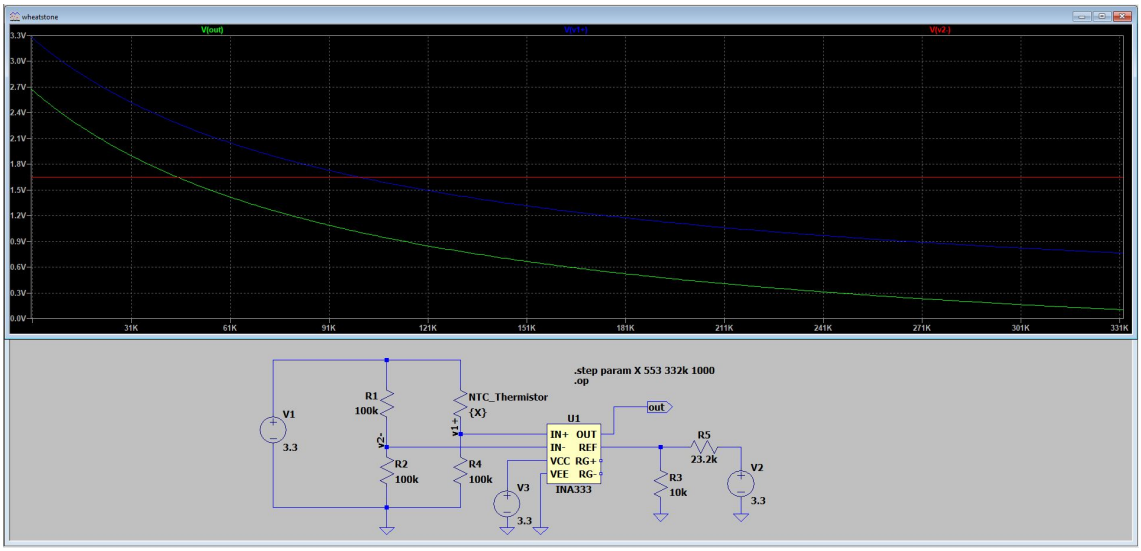
Simulation of current to voltage conversion circuit.



Simulation of comparator circuit.



Simulation of Wheatstone bridge without reference voltage (0V).



Simulation of Wheatstone bridge with adjusted reference voltage.

J FMECA

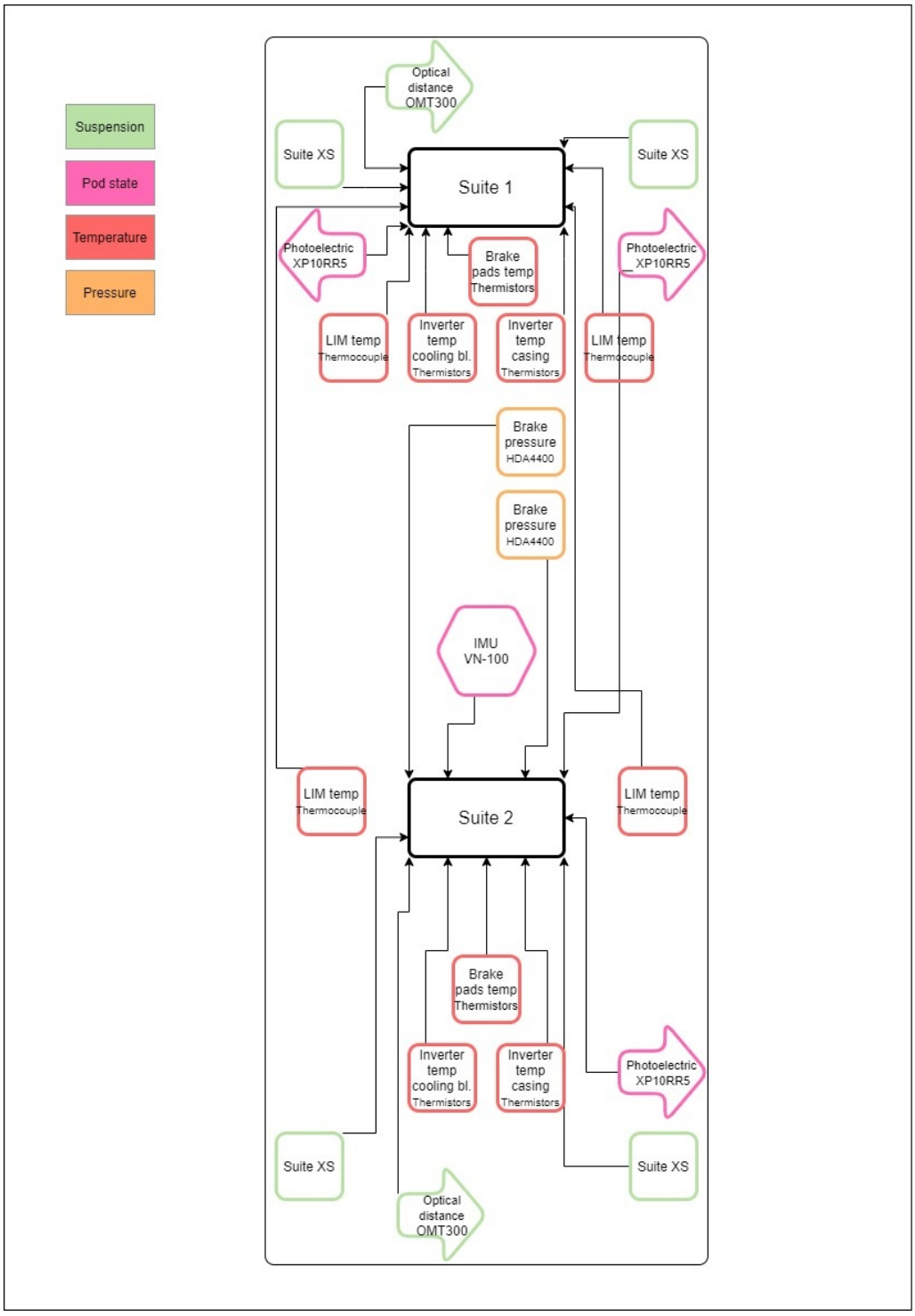
Sensors - FMECA

(Ligger inne sammen med alle andre systemer også LINK)

Description of unit				Description of failure			Effect of failure		Evaluation				
Name of subsystem/ component	Dependency to other systems	Function	Operational mode	Failure mode	Failure cause or mechanism	Detection of failure	On the subsystem	On the system function	Detection rate *1 (highest) - 10 (lowest)*	Failure rate *1 (unlikely) - 10 (frequent)*	Severity rating *1 (minor) - 10 (catastrophic)*	RPN	Risk reducing measures
Photoelectric sensors	Pod State / VCU	Pod state, register navigational markers along the track	Running	Sensor does't register a navigation marker.	Electrical malfunction / loss of power	No values detected /received	Cannot provide positional data	Wrong speed calcuation Unknown position	5	3	4	60	Comprehensive testing. Make sure no markers are missed at high speeds during testing. Make sure only navigation markers are registered (nothing form surrounding area)
				Sensor doesn't register a navigation marker	Speed > 300 km/h	No values detected /received	Positional data is not reliable	Wrong speed calcuation Unknown position	5	2	4	40	Comprehensive testing. Make sure no markers are missed at high speeds during testing. Make sure only navigation markers are registered (nothing form surrounding area)
				Wire malfunction	Physical damage to connected wires	Erroneous sensor readings or no readings at all	Unreliable or non-existing positional data	Wrong speed calcuation Unknown position	2	5	4	40	Take care when installing wires. Keep away from moving parts and fasten properly. Visually inspect before each run.
VectorNAV IMU	Pod State / VCU	Acceleration in X/Y /Z, pod state	Running	Electrical malfunction	Bad soldering, short circuits, PSU stops providing power	No values detected /received	Unreliable or non-existing data, wrong speed calculation	Wrong speed calcuation	2	2	4	16	Coprehensive testing.
				Wire malfunction	Physical damage to connected wires	Erroneous sensor readings or no readings at all	Non-existing data, wrong speed calculation	Wrong speed calculation	2	5	4	40	Take care when installing wires. Keep away from moving parts and fasten properly. Visually inspect before each run.
Thermocouples	Motor (LIM)	Temperature	Running	Overheating	Motor temperature higher than anticipated	Reaches max temperature (> xxx degrees) (before giving weird readings)	Unreliable sensor data	Uncertain temperature in motor	1	1	3	3	Coprehensive testing.
				Electrical failiure	Transmitter is damaged	No sensor reading or visually detectable	No sensor data	Unknown temperature in motor	1	2	2	4	Visually inspect transmitters before each run. Take care not to install in exposed area.
				Wire malfunction	Physical damage to connected wires	Erroneous sensor readings or no readings at all	No sensor data	Non-existing or uncertain temperature in motor	2	5	2	20	Take care when installing wires. Keep away from moving parts and fasten properly. Visually inspect before each run.
Thermistors in cooling block	Inverter	Temperature	Running	Overheating	Cooling block temperature higher than anticiplated	Reaches max temperature (> 250 degrees)	Unreliable sensor data	Uncertain temperaure in cooling block	1	1	1	1	Coprehensive testing.
				Electrical failure	Bad soldering, short circuits,	Much lower or higher resistance than expected	Unreliable sensor data	Uncertain temperature in cooling block	2	2	1	4	Inspect during installation and do control measurements
				Wire malfunction	Physical damage to connected wires	Erroneous sensor readings or no readings at all	Unreliable or no sensor data	Non-existing or uncertain sensor data	2	5	1	10	Take care when installing wires. Keep away from moving parts and fasten properly. Visually inspect before each run.
Pressure sensor in hydraulics fluid	Brakes	Leakage	Running	Wire malfunction	Physical damage to connected wires	Erroneous sensor readings or no readings at all	Non-existing sensor data	Non-existing sensor data, unknown if brakes are leaking	2	5	2	20	Take care when installing wires. Keep away from moving parts and fasten properly. Visually inspect before each run.
				Gives wrong readings	Damage to sensor membrane	Wrong readings which does not make sense	No useable data	Unknown brake-leakage status	4	3	4	48	Chose a sufficiently robust sensor. Take care when installing it, not to mishandle it.
Brake pad thermistors	Brakes	Measure temperature of brake pad	Running	Gives wrong readings	Thermistor damaged either cause of temperature or physical impact	Werid / wrong sensor readings	Non-existing or uncertain data	Unknown temperature status of brake pads	2	6	1	12	Choose a thermistor which will handle potentially high temperatures, and be careful when placing it.
				Wire malfunction	Physical damage to connected wires	Erroneous sensor readings or no readings at all	Non-existing or uncertain data	Unknown temperature status of brake pads	2	5	1	10	Take care when installing wires. Keep away from moving parts and fasten properly. Visually inspect before each run.
PCB PSU 3V3	Every sensor	Supply Sensor Suite PCB, CAN tranceivers and 3V3 sensors with power	Running	Stops supplying power	Overload / fuse blows	Loss of sensor readings / Suite stops responding on CAN	Affected Suite stops working	Pod enters emergency brake mode	1	2	3	6	Calculate power consumption as accurately as possible, and scale supply and fuses thereafter
PCB PSU 5V	CAN-BUS / Vector NAV IMU	Power the local CAN tranceivers, supply VectorNAV IMU	Running	Stops supplying power	Overload / fuse blows	Loss of VectorNAV readings / Suite stops responding on CAN	Affected Suite loses CAN and VectorNAV reading if Suite 2	Pod enters emergency brake mode	1	2	3	6	Calculate power consumption as accurately as possible, and scale supply and fuses thereafter
CAN-bus	Sensor Suite	Communication with other PCBs	Running	Connection loss	Wires come lose	No alive ping message	Loss of communication	Pod enters emergency brake mode	2	5	3	30	Take care when installing wires.

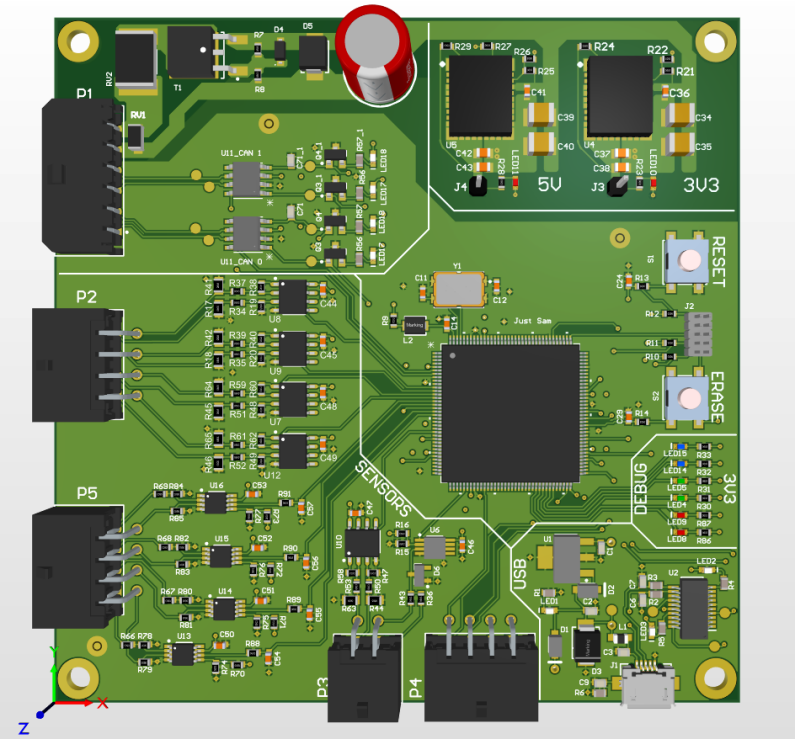
				Connection issues	Internally damaged tranceiver	Loss of data / no alive ping	Loss of communication	Pod enters emergency brake mode	2	2	3	12	Take care to handle ICs with care, and always wear ESD wristband when handling
ATSAM SAMV71			Running	Not working	Static discarge	Suite testing	No sensor data	Pod enters emergency brake mode	1	2	3	6	Take care to handle ATSAM with care, and always wear ESD wristband when handling

K Sensor placement

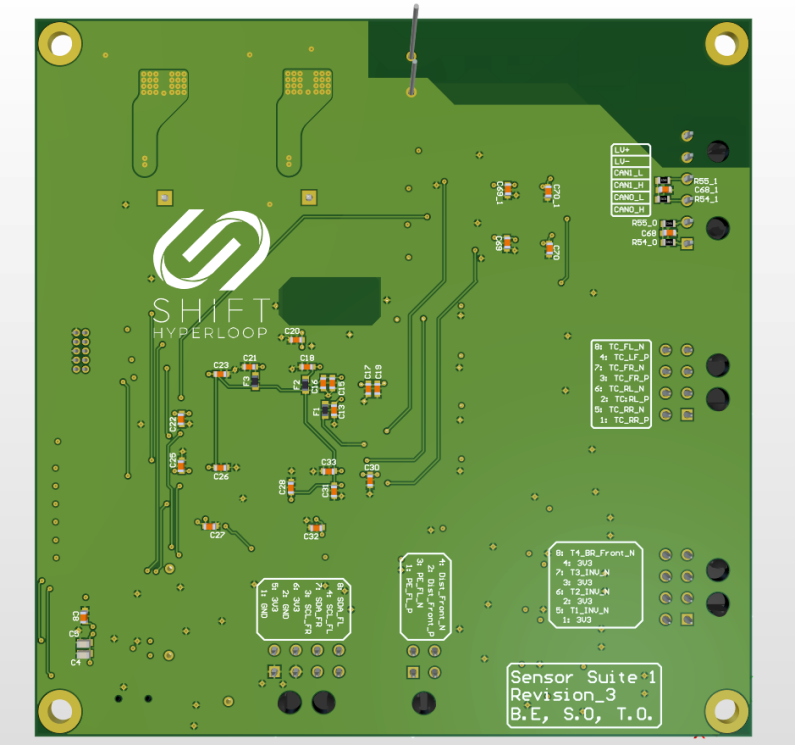


Overview and rough sensor placement. Up represents front of pod

L 3D models of sensor Suite 1, revision 3

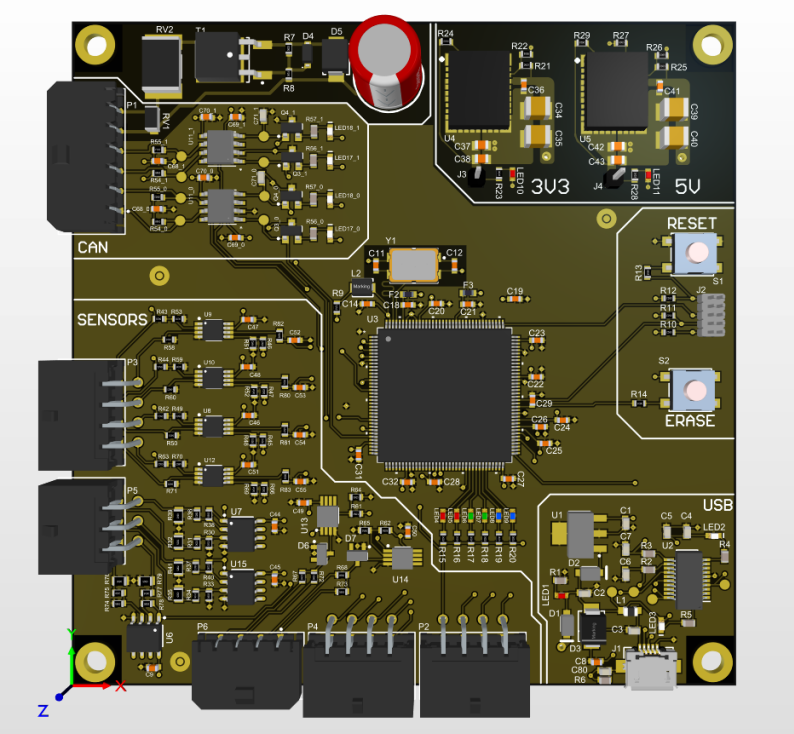


3D model, sensor Suite 1 revision 3, top view

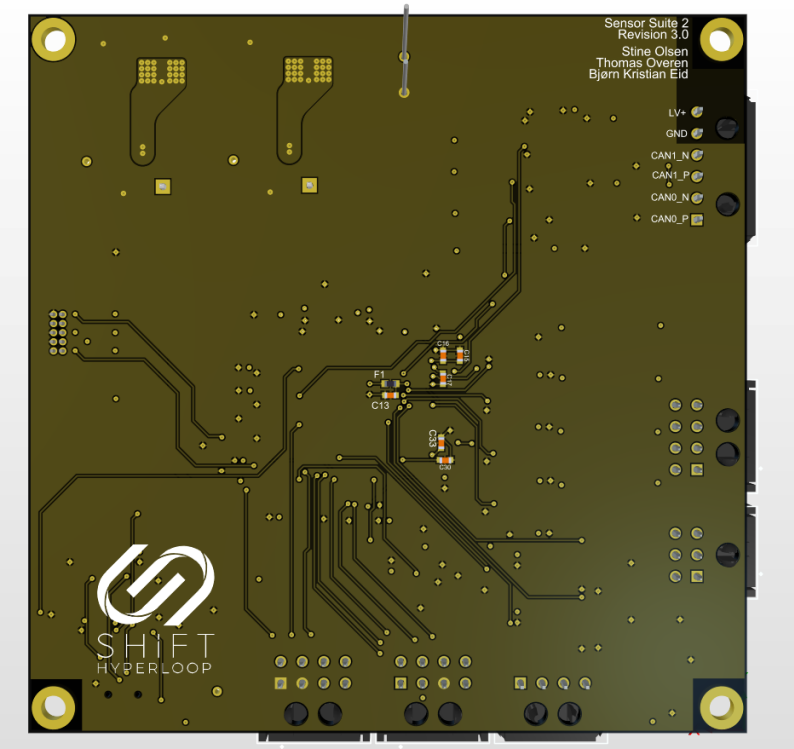


3D model, sensor Suite 1 revision 3, bottom view

M 3D models of sensor Suite 2, revision 3



3D model, sensor Suite 2 revision 3, top view



3D model, sensor Suite 2 revision 3, bottom view

N Code - main.c

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 1

```
1  /*
2   * Created by:  Thomas Overen
3   *              Stine Olsen
4   *              Bjørn Kristian Eid
5   *
6   * Date: 13/05/2021
7   * Time: 20:59
8   *
9   * \main.c
10  *
11  * \brief Main executable code
12  */
13
14 /*  SOURCES:
15     NR.1: https://github.com/fdcl-gwu/arduino-vn100/blob/master/read-vn100/read-vn100.ino
16     NR.2: https://startingelectronics.org/software/atmel/asf-arm-tutorial/adc/
17 */
18
19 #include <asf.h>
20 #include <string.h>
21 #include <math.h>
22 #include "shift_can.h"
23 #include "timer.h"
24 #include "watchdog.h"
25 #include "general.h"
26 #include "stdio.h"
27
28 /*
29     DECLARATION OF FUNCTIONS
30     START
31 */
32
33 static void configure_USB_uart(void) {
34     //USB options
35     static usart_serial_options_t uart_USB_options = {
36         .baudrate   = CONF_UART_BAUDRATE,
37         #ifdef CONF_UART_CHAR_LENGTH
38         .charlength = CONF_UART_CHAR_LENGTH,
39         #endif
40         .paritytype = CONF_UART_PARITY,
41         #ifdef CONF_UART_STOP_BITS
42         .stopbits   = CONF_UART_STOP_BITS,
43         #endif
44     };
45
46     //initialize USB USART, UART1
47     sysclk_enable_peripheral_clock(USB_UART_ID);
48     stdio_serial_init(USB_UART, &uart_USB_options);
49 }
50
51 static void configure_IMU_uart(void) {
52     //IMU options
53     static usart_serial_options_t uart_IMU_options = {
54         .baudrate   = CONF_UART_BAUDRATE,
55         #ifdef CONF_UART_CHAR_LENGTH
56         .charlength = CONF_UART_CHAR_LENGTH,
57         #endif
58         .paritytype = CONF_UART_PARITY,
59         #ifdef CONF_UART_STOP_BITS
60         .stopbits   = CONF_UART_STOP_BITS,
61         #endif
62     };
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 2

```
63
64      //initialize IMU UART, UART2
65      sysclk_enable_peripheral_clock(IMU_UART_ID);
66      usart_serial_init(IMU_UART, &uart_IMU_options);
67  }
68
69  void AFEC1_start (int ch_);
70  void AFEC0_start (int ch_);
71  int AFEC1_value (int ch_);
72  int AFEC0_value (int ch_);
73
74  void TWIHS0_start (void);
75  void TWIHS2_start (void);
76  void TWIHS0_ITDS_write (uint8_t reg_adrs, uint8_t data_tx);
77  void TWIHS0_PADS_write (uint8_t reg_adrs, uint8_t data_tx);
78  void TWIHS2_ITDS_write (uint8_t reg_adrs, uint8_t data_tx);
79  void TWIHS2_PADS_write (uint8_t reg_adrs, uint8_t data_tx);
80  int8_t TWIHS0_ITDS_read (uint8_t reg_adrs);
81  int8_t TWIHS0_PADS_read (uint8_t reg_adrs);
82  int8_t TWIHS2_ITDS_read (uint8_t reg_adrs);
83  int8_t TWIHS2_PADS_read (uint8_t reg_adrs);
84
85  void UART2_IMU_read(void);
86  void check_sync_byte(void);
87  unsigned short calculate_imu_crc(Byte data[], unsigned int length);
88
89  double ADC_math (float a, float b, uint16_t y);
90  float THERM_math (uint16_t y);
91
92  /*
93      DECLARATION OF FUNCTIONS
94      END
95  */
96
97  /*
98      GLOBAL VARIABLES
99      START
100 */
101 //Based on source NR.1
102 //Attitude data
103 union {float f; Byte b[4];} yaw;
104 union {float f; Byte b[4];} pitch;
105 union {float f; Byte b[4];} roll;
106
107 //Angular rates
108 union {float f; Byte b[4];} W_x;
109 union {float f; Byte b[4];} W_y;
110 union {float f; Byte b[4];} W_z;
111
112 //Acceleration
113 union {float f; Byte b[4];} a_x;
114 union {float f; Byte b[4];} a_y;
115 union {float f; Byte b[4];} a_z;
116
117 //Delta time
118 union {float f; Byte b[4];} dt;
119
120 //Delta velocity
121 union {float f; Byte b[4];} dv_x;
122 union {float f; Byte b[4];} dv_y;
123 union {float f; Byte b[4];} dv_z;
124
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 3

```
125 //Checksum
126 union {unsigned short s; Byte b[2];} checksum;
127
128 Bool imu_sync_detected = false;
129 Byte IMU[100];
130
131 /*
132     GLOBAL VARIABLES
133     END
134 */
135
136 int main (void)
137 {
138
139     sysclk_init();
140     board_init();
141
142     configure_IMU_uart();
143     configure_USB_uart();
144
145     /*
146         PHOTOELECTRIC CODE, SUITE 2
147         START
148     */
149
150     Bool PE_RR, PE_FR;
151     int PE_RR_cnt = 0, PE_FR_cnt = 0;
152     int MARK_DIST = 35; //Distance between markers, in meters
153     uint16_t PE_DIST_esti;
154     Bool PE_RR_last = false;
155     Bool PE_FR_last = false;
156
157     while(1) {
158         //-------------------------
159         PE_FR = ioport_get_pin_level(PE_FR_PIN);
160         if((PE_FR == true) & (PE_FR_last == false)) {
161             //TIMER START
162             PE_FR_last = PE_FR;
163             PE_FR_cnt++;
164             PE_DIST_esti = MARK_DIST * PE_FR_cnt;
165             printf("Front count: %d \n\r", PE_FR_cnt);
166         }
167         else if((PE_FR == false) & (PE_FR_last == true)) {
168             PE_FR_last = false;
169         }
170         //-------------------------
171         PE_RR = ioport_get_pin_level(PE_RR_PIN);
172         if((PE_RR == true) & (PE_RR_last == false)) {
173             //TIMER STOP
174             PE_RR_last = PE_RR;
175             PE_RR_cnt++;
176             printf("Rear count: %d \n\r", PE_RR_cnt);
177
178             if(PE_FR_cnt != PE_RR_cnt) {
179                 //SEND ERROR MESSAGE, POSITION HAS BECOME UNKNOWN
180                 error_struct_t error_struct;
181                 error_struct.error_code = 0xB1;
182                 printf("ERROR, COUNT IS WRONG");
183             }
184
185         }
186         else if((PE_RR == false) & (PE_RR_last == true)) {
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 4

```
187             PE_RR_last = false;
188         }
189         //-------------------------
190 
191     }
192 
193     /*
194         PHOTOELECTRIC CODE, SUITE 2
195         END
196     */
197 
198     /*
199         ADC CODE, SUITE 1
200         START
201     */
202 
203     //Suite 1 and 2 share the following:
204     //T1_adc, T2_adc, T3_adc, T4_adc
205     //DIST_adc
206     //These will therefore only be written once even though they may use different ADC channels
207     //Some other ADC channels may overlap because of different placements as well
208 
209     uint16_t TC1_adc, TC2_adc, TC3_adc, TC4_adc;
210     AFEC1_start(AFEC_CHANNEL_9);
211     AFEC1_start(AFEC_CHANNEL_8);
212     AFEC1_start(AFEC_CHANNEL_7);
213     AFEC1_start(AFEC_CHANNEL_6);
214     TC1_adc         = AFEC1_value(AFEC_CHANNEL_9);
215     TC2_adc         = AFEC1_value(AFEC_CHANNEL_8);
216     TC3_adc         = AFEC1_value(AFEC_CHANNEL_7);
217     TC4_adc         = AFEC1_value(AFEC_CHANNEL_6);
218 
219     double TC_LIM_RR_temp, TC_LIM_FR_temp, TC_LIM_RL_temp, TC_LIM_FL_temp;
220     //                          a        b         y
221     TC_LIM_FL_temp  = ADC_math(10.8,    1395,   TC1_adc);
222     TC_LIM_RL_temp  = ADC_math(10.8,    1395,   TC2_adc);
223     TC_LIM_FR_temp  = ADC_math(10.8,    1395,   TC3_adc);
224     TC_LIM_RR_temp  = ADC_math(10.8,    1395,   TC4_adc);
225 
226     /*
227         ADC CODE, SUITE 1
228         END
229     */
230 
231     /*
232         ADC CODE, SUITE 2
233         START
234     */
235 
236     uint16_t T1_adc, T2_adc, T3_adc, T4_adc;
237     uint16_t DIST_adc;
238     uint16_t PR_BRKS_F_adc, PR_BRKS_R_adc;
239     AFEC1_start(AFEC_CHANNEL_9);
240     AFEC1_start(AFEC_CHANNEL_8);
241     AFEC1_start(AFEC_CHANNEL_7);
242     AFEC1_start(AFEC_CHANNEL_6);
243     AFEC1_start(AFEC_CHANNEL_5);
244     AFEC1_start(AFEC_CHANNEL_4);
245     AFEC1_start(AFEC_CHANNEL_2);
246     T1_adc          = AFEC1_value(AFEC_CHANNEL_9);
247     T2_adc          = AFEC1_value(AFEC_CHANNEL_8);
248     T3_adc          = AFEC1_value(AFEC_CHANNEL_7);
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 5

```
249     T4_adc          = AFEC1_value(AFEC_CHANNEL_6);
250     DIST_adc        = AFEC1_value(AFEC_CHANNEL_5);
251     PR_BRKS_F_adc   = AFEC1_value(AFEC_CHANNEL_4);
252     PR_BRKS_R_adc   = AFEC1_value(AFEC_CHANNEL_2);
253
254     //ADC MATHS
255     //Example:
256     // y = ax + b, DIST_value = a*DIST_mm + b
257     double      DIST_R_mm;
258     double      PR_BRKS_F_bar, PR_BRKS_R_bar;
259     float       T1_BRKS_R_temp, T2_INV_R_temp, T3_INV_R_temp, T4_INV_R_temp;
260     //                          a        b         y
261     DIST_R_mm       = ADC_math(16.38,   -819,   DIST_adc);
262     PR_BRKS_F_bar   = ADC_math( 8.19,    819,   PR_BRKS_F_adc);
263     PR_BRKS_R_bar   = ADC_math( 8.19,    819,   PR_BRKS_R_adc);
264     T1_BRKS_R_temp  = THERM_math(T1_adc);
265     T2_INV_R_temp   = THERM_math(T2_adc);
266     T3_INV_R_temp   = THERM_math(T3_adc);
267     T4_INV_R_temp   = THERM_math(T4_adc);
268
269     /*
270         ADC CODE, SUITE 2
271         END
272     */
273
274     /*
275         TWIHS CODE, SUITE 1 AND 2
276         START
277     */
278     //All register IDs gotten from the relevant user manuals for each sensor.
279
280     int8_t      TEMP_RR, XL_RR, XH_RR, YL_RR, YH_RR, ZL_RR, ZH_RR;
281     int8_t      TEMP_RL, XL_RL, XH_RL, YL_RL, YH_RL, ZL_RL, ZH_RL;
282     int8_t      PRES_H_RR, PRES_L_RR, PRES_XL_RR;
283     int8_t      PRES_H_RL, PRES_L_RL, PRES_XL_RL;
284     int32_t     PRES_RR, PRES_RL;
285     int16_t     X_RR, Y_RR, Z_RR;
286     int16_t     X_RL, Y_RL, Z_RL;
287     float       acc_sens  = 1.952;
288     float       pres_sens = 0.02441406;
289     double      X_RR_acc, Y_RR_acc, Z_RR_acc;
290     double      X_RL_acc, Y_RL_acc, Z_RL_acc;
291     double      PRES_RR_Pa, PRES_RL_Pa;
292     TWIHS0_start();
293     TWIHS2_start();
294
295     //Write to CTRL_1, 200Hz, normal mode
296     //                  where   ,   what
297     TWIHS0_ITDS_write(REG_CTRL_1, 0b01100010);
298     TWIHS2_ITDS_write(REG_CTRL_1, 0b01100010);
299     //Write to CTRL_6, no filter, +-16g scale, low noise enabled
300     TWIHS0_ITDS_write(REG_CTRL_6, 0b00110100);
301     TWIHS2_ITDS_write(REG_CTRL_6, 0b00110100);
302
303     //Temperature Rear-Right
304     TEMP_RR     = TWIHS0_ITDS_read(REG_T_OUT);      //25C is base, then -1 is 24C and +1 is 26C
305     TEMP_RR     = 0b00011001 + TEMP_RR;             //Real temp value in degrees celsius
306
307     //X,Y,Z acceleration Rear-Right
308     XL_RR       = TWIHS0_ITDS_read(REG_X_OUT_L);    //0bLLLLLL00
309     XH_RR       = TWIHS0_ITDS_read(REG_X_OUT_H);    //0bHHHHHHHH
310     X_RR        = (XH_RR << 8) | XL_RR;             //0bHHHHHHHHLLLLLL00
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 6

```
311     X_RR        = X_RR >> 2;                        //0b00HHHHHHHHLLLLLL (14-bit value Two's
          complement)
312     X_RR_acc    = X_RR * acc_sens;                  //Value in m/s^2
313     YL_RR       = TWIHS0_ITDS_read(REG_Y_OUT_L);
314     YH_RR       = TWIHS0_ITDS_read(REG_Y_OUT_H);
315     Y_RR        = (YH_RR << 8) | YL_RR;
316     Y_RR        = Y_RR >> 2;
317     Y_RR_acc    = Y_RR * acc_sens;
318     ZL_RR       = TWIHS0_ITDS_read(REG_Z_OUT_L);
319     ZH_RR       = TWIHS0_ITDS_read(REG_Z_OUT_H);
320     Z_RR        = (ZH_RR << 8) | ZL_RR;
321     Z_RR        = Z_RR >> 2;
322     Z_RR_acc    = Z_RR * acc_sens;
323
324     //Temperature Rear-Left
325     TEMP_RL     = TWIHS2_ITDS_read(REG_T_OUT);      //25C is base, then -1 is 24C and +1 is 26C
326     TEMP_RL     = 0b00011001 + TEMP_RL;             //Real temp value in degrees celsius
327     //X,Y,Z acceleration Rear-Left
328     XL_RL       = TWIHS2_ITDS_read(REG_X_OUT_L);    //0bLLLLLL00
329     XH_RL       = TWIHS2_ITDS_read(REG_X_OUT_H);    //0bHHHHHHHH
330     X_RL        = (XH_RL << 8) | XL_RL;             //0bHHHHHHHHLLLLLL00
331     X_RL        = X_RL >> 2;                        //0b00HHHHHHHHLLLLLL (14-bit value Two's
          complement)
332     X_RL_acc    = X_RL * acc_sens;                  //Value in m/s^2
333     YL_RL       = TWIHS2_ITDS_read(REG_Y_OUT_L);
334     YH_RL       = TWIHS2_ITDS_read(REG_Y_OUT_H);
335     Y_RL        = (YH_RL << 8) | YL_RL;
336     Y_RL        = Y_RL >> 2;
337     Y_RL_acc    = Y_RL * acc_sens;
338     ZL_RL       = TWIHS2_ITDS_read(REG_Z_OUT_L);
339     ZH_RL       = TWIHS2_ITDS_read(REG_Z_OUT_H);
340     Z_RL        = (ZH_RL << 8) | ZL_RL;
341     Z_RL        = Z_RL >> 2;
342     Z_RL_acc    = Z_RL * acc_sens;
343
344     //Pressure Rear-Right
345     PRES_H_RR   = TWIHS0_PADS_read(REG_P_OUT_H);
346     PRES_L_RR   = TWIHS0_PADS_read(REG_P_OUT_L);
347     PRES_XL_RR  = TWIHS0_PADS_read(REG_P_OUT_XL);
348     PRES_RR     = (PRES_H_RR << 16) | (PRES_L_RR << 8) | PRES_XL_RR;
349     PRES_RR_Pa  = PRES_RR * pres_sens;      //Pascal
350
351     //Pressure Rear-Left
352     PRES_H_RL   = TWIHS2_PADS_read(REG_P_OUT_H);
353     PRES_L_RL   = TWIHS2_PADS_read(REG_P_OUT_L);
354     PRES_XL_RL  = TWIHS2_PADS_read(REG_P_OUT_XL);
355     PRES_RL     = (PRES_H_RL << 16) | (PRES_L_RL << 8) | PRES_XL_RL;
356     PRES_RL_Pa  = PRES_RL * pres_sens;      //Pascal
357
358     /*
359         TWIHS CODE, SUITE 1 AND 2
360         END
361     */
362
363     /*
364         UART IMU CODE LOOP
365         START
366     */
367     //Based on source NR.1
368     while (1) {
369         imu_sync_detected = false;
370
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 7

```
371         if (usart_serial_is_rx_ready(IMU_UART) > 0) check_sync_byte();
372
373         if (imu_sync_detected) UART2_IMU_read();
374     }
375
376     /*
377         UART IMU CODE LOOP
378         END
379     */
380
381 }
382
383 /*
384     FUNCTIONS
385     START
386 */
387
388 //Based on source NR.2, START
389 //Configure and enable AFEC1 with ASF driver module
390 void AFEC1_start (int ch_) {
391
392     struct afec_config afec_conf;
393     struct afec_ch_config afec_ch_conf;
394     afec_enable(AFEC1);
395     afec_get_config_defaults(&afec_conf);
396     afec_init(AFEC1, &afec_conf);
397     afec_set_trigger(AFEC1, AFEC_TRIG_SW);
398     afec_ch_get_config_defaults(&afec_ch_conf);
399     afec_ch_conf.gain = AFEC_GAINVALUE_0;
400     afec_ch_set_config(AFEC1, ch_, &afec_ch_conf);
401     afec_channel_set_analog_offset(AFEC1, ch_, 0x200);
402     afec_channel_enable(AFEC1, ch_);
403 }
404
405 //Configure and enable AFEC0 with ASF driver module
406 void AFEC0_start (int ch_) {
407
408     struct afec_config afec_conf;
409     struct afec_ch_config afec_ch_conf;
410     afec_enable(AFEC0);
411     afec_get_config_defaults(&afec_conf);
412     afec_init(AFEC0, &afec_conf);
413     afec_set_trigger(AFEC0, AFEC_TRIG_SW);
414     afec_ch_get_config_defaults(&afec_ch_conf);
415     afec_ch_conf.gain = AFEC_GAINVALUE_0;
416     afec_ch_set_config(AFEC0, ch_, &afec_ch_conf);
417     afec_channel_set_analog_offset(AFEC0, ch_, 0x200);
418     afec_channel_enable(AFEC0, ch_);
419 }
420
421 //Starting conversion and retrieves the value of AFEC1
422 int AFEC1_value (int ch_) {
423
424     uint16_t afec_val;
425     afec_start_software_conversion(AFEC1);
426     while (!(afec_get_interrupt_status(AFEC1) & (1 << ch_)));
427     afec_val = afec_channel_get_value(AFEC1, ch_);
428     return afec_val;
429 }
430
431 //Starting conversion and retrieves the value of AFEC0
432 int AFEC0_value (int ch_) {
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 8

```
433
434     uint16_t afec_val;
435     afec_start_software_conversion(AFEC0);
436     while (!(afec_get_interrupt_status(AFEC0) & (1 << ch_)));
437     afec_val = afec_channel_get_value(AFEC0, ch_);
438     return afec_val;
439 }
440 //Based on source NR.2, END
441
442 //All TWIHS code is based on example code from Microchip Technology
443 //Configure and enable TWIHS0 in Master mode
444 void TWIHS0_start (void) {
445     twihs_options_t opt;
446
447     pmc_enable_periph_clk(ID_TWIHS0);
448
449     opt.master_clk  = sysclk_get_peripheral_hz();
450     opt.speed       = TWIHS_CLK;
451
452     twihs_enable_master_mode(TWIHS0);
453     if (twihs_master_init(TWIHS0, &opt) != TWIHS_SUCCESS) {
454         //set LED_A and _B HIGH if initialization failed
455         ioport_set_pin_level(LED_A, true);
456         ioport_set_pin_level(LED_B, true);
457         while (1) {
458             // Capture error
459         }
460     }
461 }
462
463 //Configure and enable TWIHS2 in Master mode
464 void TWIHS2_start (void) {
465     twihs_options_t opt;
466
467     pmc_enable_periph_clk(ID_TWIHS2);
468
469     opt.master_clk  = sysclk_get_peripheral_hz();
470     opt.speed       = TWIHS_CLK;
471
472     twihs_enable_master_mode(TWIHS2);
473     if (twihs_master_init(TWIHS2, &opt) != TWIHS_SUCCESS) {
474         //set LED_A and _B HIGH if initialization failed
475         ioport_set_pin_level(LED_A, true);
476         ioport_set_pin_level(LED_B, true);
477         while (1) {
478             // Capture error
479         }
480     }
481 }
482
483 void TWIHS0_ITDS_write (uint8_t reg_adrs, uint8_t data_tx) {
484     twihs_packet_t packet_tx;
485
486     packet_tx.chip          = ITDS_ADRS;
487     packet_tx.addr[0]       = reg_adrs;
488     packet_tx.addr_length   = 1;
489     packet_tx.buffer        = &data_tx;
490     packet_tx.length        = 1;
491
492     if (twihs_master_write(TWIHS0, &packet_tx) != TWIHS_SUCCESS) {
493         ioport_set_pin_level(LED_A, true); //set LED_A HIGH if packet write failed
494         while (1) {
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 9

```
495             // Capture error
496         }
497     }
498
499 }
500
501 void TWIHS0_PADS_write (uint8_t reg_adrs, uint8_t data_tx) {
502     twihs_packet_t packet_tx;
503
504     packet_tx.chip              = PADS_ADRS;
505     packet_tx.addr[0]           = reg_adrs;
506     packet_tx.addr_length       = 1;
507     packet_tx.buffer            = &data_tx;
508     packet_tx.length            = 1;
509
510     if (twihs_master_write(TWIHS0, &packet_tx) != TWIHS_SUCCESS) {
511         ioport_set_pin_level(LED_A, true); //set LED_A HIGH if packet write failed
512         while (1) {
513             // Capture error
514         }
515     }
516 }
517
518 void TWIHS2_ITDS_write (uint8_t reg_adrs, uint8_t data_tx) {
519     twihs_packet_t packet_tx;
520
521     packet_tx.chip          = ITDS_ADRS;
522     packet_tx.addr[0]       = reg_adrs;
523     packet_tx.addr_length   = 1;
524     packet_tx.buffer        = &data_tx;
525     packet_tx.length        = 1;
526
527     if (twihs_master_write(TWIHS2, &packet_tx) != TWIHS_SUCCESS) {
528         ioport_set_pin_level(LED_A, true); //set LED_A HIGH if packet write failed
529         while (1) {
530             // Capture error
531         }
532     }
533 }
534
535 void TWIHS2_PADS_write (uint8_t reg_adrs, uint8_t data_tx) {
536     twihs_packet_t packet_tx;
537
538     packet_tx.chip              = PADS_ADRS;
539     packet_tx.addr[0]           = reg_adrs;
540     packet_tx.addr_length       = 1;
541     packet_tx.buffer            = &data_tx;
542     packet_tx.length            = 1;
543
544     if (twihs_master_write(TWIHS2, &packet_tx) != TWIHS_SUCCESS) {
545         ioport_set_pin_level(LED_A, true); //set LED_A HIGH if packet write failed
546         while (1) {
547             // Capture error
548         }
549     }
550 }
551
552 int8_t TWIHS0_ITDS_read (uint8_t reg_adrs) {
553     twihs_packet_t packet_rx;
554     static int8_t data_rx;
555
556     packet_rx.chip          = ITDS_ADRS;
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 10

```
557     packet_rx.addr[0]       = reg_adrs;
558     packet_rx.addr_length   = 1;
559     packet_rx.buffer        = &data_rx;
560     packet_rx.length        = 1;
561
562     if (twihs_master_read(TWIHS0, &packet_rx) != TWIHS_SUCCESS) {
563         ioport_set_pin_level(LED_F, true); //set LED_F HIGH if packet read failed
564         while (1) {
565             // Capture error
566         }
567     }
568
569     return data_rx;
570 }
571
572 int8_t TWIHS0_PADS_read (uint8_t reg_adrs) {
573     twihs_packet_t packet_rx;
574     static int8_t data_rx;
575
576     packet_rx.chip          = PADS_ADRS;
577     packet_rx.addr[0]       = reg_adrs;
578     packet_rx.addr_length   = 1;
579     packet_rx.buffer        = &data_rx;
580     packet_rx.length        = 1;
581
582     if (twihs_master_read(TWIHS0, &packet_rx) != TWIHS_SUCCESS) {
583         ioport_set_pin_level(LED_F, true); //set LED_F HIGH if packet read failed
584         while (1) {
585             // Capture error
586         }
587     }
588
589     return data_rx;
590 }
591
592 int8_t TWIHS2_ITDS_read (uint8_t reg_adrs) {
593     twihs_packet_t packet_rx;
594     static int8_t data_rx;
595
596     packet_rx.chip          = ITDS_ADRS;
597     packet_rx.addr[0]       = reg_adrs;
598     packet_rx.addr_length   = 1;
599     packet_rx.buffer        = &data_rx;
600     packet_rx.length        = 1;
601
602     if (twihs_master_read(TWIHS2, &packet_rx) != TWIHS_SUCCESS) {
603         ioport_set_pin_level(LED_F, true); //set LED_F HIGH if packet read failed
604         while (1) {
605             // Capture error
606         }
607     }
608
609     return data_rx;
610 }
611
612 int8_t TWIHS2_PADS_read (uint8_t reg_adrs) {
613     twihs_packet_t packet_rx;
614     static int8_t data_rx;
615
616     packet_rx.chip          = PADS_ADRS;
617     packet_rx.addr[0]       = reg_adrs;
618     packet_rx.addr_length   = 1;
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 11

```
619     packet_rx.buffer        = &data_rx;
620     packet_rx.length        = 1;
621
622     if (twihs_master_read(TWIHS2, &packet_rx) != TWIHS_SUCCESS) {
623         ioport_set_pin_level(LED_F, true); //set LED_F HIGH if packet read failed
624         while (1) {
625             // Capture error
626         }
627     }
628
629     return data_rx;
630 }
631
632 //Based on source NR.1, START
633 void UART2_IMU_read (void) {
634     usart_serial_read_packet(IMU_UART, IMU, 69);
635     checksum.b[0] = IMU[68];
636     checksum.b[1] = IMU[67];
637
638     if (calculate_imu_crc(IMU, 67) == checksum.s) {
639         for (int i = 0; i < 4; i++) {
640             yaw.b[i]    = IMU[3 + i];
641             pitch.b[i]  = IMU[7 + i];
642             roll.b[i]   = IMU[11 + i];
643             W_x.b[i]    = IMU[15 + i];
644             W_y.b[i]    = IMU[19 + i];
645             W_z.b[i]    = IMU[23 + i];
646             a_x.b[i]    = IMU[27 + i];
647             a_y.b[i]    = IMU[31 + i];
648             a_z.b[i]    = IMU[35 + i];
649             dt.b[i]     = IMU[39 + i];
650             //Skipping DeltaTheta, 12 bytes
651             //                43
652             //                47
653             //                51
654             dv_x.b[i]   = IMU[55 + i];
655             dv_y.b[i]   = IMU[59 + i];
656             dv_z.b[i]   = IMU[63 + i];
657         }
658         //WRITE TO SERIAL MONITOR FOR DEBUGGING
659     }
660 }
661
662 void check_sync_byte (void) {
663     for (int i = 0; i < 6; i++) {
664         usart_serial_read_packet(IMU_UART, IMU, 1);
665         if (IMU[0] == 0xFA) {
666             imu_sync_detected = true;
667             break;
668         }
669     }
670 }
671
672 unsigned short calculate_imu_crc(Byte data[], unsigned int length) {
673     unsigned int i;
674     unsigned short crc = 0;
675
676     for(i=0; i < length; i++) {
677         crc = (Byte)(crc >> 8) | (crc << 8);
678         crc ^= data[i];
679         crc ^= (Byte)(crc & 0xFF) >> 4;
680         crc ^= crc << 12;
```

...cuments\Shift_Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\main.c 12

```
681             crc ^= (crc & 0x00FF) << 5;
682         }
683         return crc;
684     }
685     //Based on source NR.1, END
686
687     double ADC_math (float a, float b, uint16_t y) {
688         // y = a*x + b,
689         double x;
690
691         x = (y - b) / a;
692         return x;
693     }
694
695     float THERM_math (uint16_t y) {
696         float kelvin = 273.15;
697         float beta = 4092;
698         float T0 = 298.15;
699         float THERM_temp_K;
700         float THERM_temp_C;
701         float divi_one, divi_two, divi_Rt;
702         float log_Rt;
703         float Rt;
704
705         Rt = (330000 / (((y * 3.3) / 4095) + 0.66)) - 100000;
706
707         divi_one = 1 / T0;
708         divi_two = 1 / beta;
709         divi_Rt = Rt / 100000;
710
711         log_Rt = log(divi_Rt);
712
713         THERM_temp_K = 1 / (divi_one + divi_two * log_Rt);
714         THERM_temp_C = THERM_temp_K - kelvin;
715
716         return THERM_temp_C;
717     }
718
719     /*
720         FUNCTIONS
721         END
722     */
```

O Code - init.c

...ensors1_2021\EMB_Sensors1_2021\src\ASF\common\boards\user_board\init.c 1

```
1  /**
2   * \Init.c
3   *
4   * \brief User board initialization template
5   *
6   */
7
8  #include <asf.h>
9  #include <board.h>
10 #include <conf_board.h>
11
12 void board_init(void)
13 {
14     WDT->WDT_MR = WDT_MR_WDDIS; //Disables watchdog
15     ioport_init();  //Call before using IOPORT services
16
17     //TWIHS pins
18     //Set pin mode to TWIHS data line
19     ioport_set_pin_mode(PIO_PA3_IDX, PIO_PA3A_TWD0);
20     //disables IOPORT pin, so it can be used for TWIHS0
21     ioport_disable_pin(PIO_PA3_IDX);
22     //Set pin mode to TWIHS clock line
23     ioport_set_pin_mode(PIO_PA4_IDX, PIO_PA4A_TWCK0);
24     //disables IOPORT pin, so it can be used for TWIHS0
25     ioport_disable_pin(PIO_PA4_IDX);
26     //Same, but for TWIHS2
27     ioport_set_pin_mode(PIO_PD27_IDX, PIO_PD27C_TWD2);
28     ioport_disable_pin(PIO_PD27_IDX);
29     ioport_set_pin_mode(PIO_PD28_IDX, PIO_PD28C_TWCK2);
30     ioport_disable_pin(PIO_PD28_IDX);
31
32     //CANBUS Silent pins, KEEP AT LOW
33     ioport_set_pin_dir(SILENT0, IOPORT_DIR_OUTPUT);         //set direction to output
34     ioport_set_pin_level(SILENT0, IOPORT_PIN_LEVEL_LOW);    //set pin to low
35     ioport_set_pin_dir(SILENT1, IOPORT_DIR_OUTPUT);
36     ioport_set_pin_level(SILENT1, IOPORT_PIN_LEVEL_LOW);
37
38     //CANBUS PINS
39     ioport_set_pin_mode(PIO_PB2_IDX, PIO_PB2A_CANTX0);
40     ioport_disable_pin(PIO_PB2_IDX);
41     ioport_set_pin_mode(PIO_PB3_IDX, PIO_PB3A_CANRX0);
42     ioport_disable_pin(PIO_PB3_IDX);
43     ioport_set_pin_mode(PIO_PC14_IDX, PIO_PC14C_CANTX1);
44     ioport_disable_pin(PIO_PC14_IDX);
45     ioport_set_pin_mode(PIO_PC12_IDX, PIO_PC12C_CANRX1);
46     ioport_disable_pin(PIO_PC12_IDX);
47
48     //DEBUG LEDS
49     ioport_set_pin_dir(LED_A, IOPORT_DIR_OUTPUT);
50     ioport_set_pin_level(LED_A, IOPORT_PIN_LEVEL_HIGH);
51     ioport_set_pin_dir(LED_B, IOPORT_DIR_OUTPUT);
52     ioport_set_pin_level(LED_B, IOPORT_PIN_LEVEL_HIGH);
53     ioport_set_pin_dir(LED_C, IOPORT_DIR_OUTPUT);
54     ioport_set_pin_level(LED_C, IOPORT_PIN_LEVEL_HIGH);
55     ioport_set_pin_dir(LED_D, IOPORT_DIR_OUTPUT);
56     ioport_set_pin_level(LED_D, IOPORT_PIN_LEVEL_HIGH);
57     ioport_set_pin_dir(LED_E, IOPORT_DIR_OUTPUT);
58     ioport_set_pin_level(LED_E, IOPORT_PIN_LEVEL_HIGH);
59     ioport_set_pin_dir(LED_F, IOPORT_DIR_OUTPUT);
60     ioport_set_pin_level(LED_F, IOPORT_PIN_LEVEL_HIGH);
61
62     //IMU UART PINS
```

...ensors1_2021\EMB_Sensors1_2021\src\ASF\common\boards\user_board\init.c 2

```
63      ioport_set_pin_mode(PIO_PD25_IDX, IOPORT_MODE_MUX_C);
64      ioport_disable_pin(PIO_PD25_IDX);
65      ioport_set_pin_mode(PIO_PD26_IDX, IOPORT_MODE_MUX_C);
66      ioport_disable_pin(PIO_PD26_IDX);
67
68      //FTDI UART PINS
69      ioport_set_pin_mode(PIO_PA5_IDX, IOPORT_MODE_MUX_C);
70      ioport_disable_pin(PIO_PA5_IDX);
71      ioport_set_pin_mode(PIO_PA6_IDX, IOPORT_MODE_MUX_C);
72      ioport_disable_pin(PIO_PA6_IDX);
73
74      //AFEC PINS
75      ioport_set_pin_mode(THERM1, PIO_PC0X1_AFE1_AD9);
76      ioport_disable_pin(THERM1);
77      ioport_set_pin_mode(THERM2, PIO_PC27X1_AFE1_AD8);
78      ioport_disable_pin(THERM2);
79      ioport_set_pin_mode(THERM3, PIO_PC26X1_AFE1_AD7);
80      ioport_disable_pin(THERM3);
81      ioport_set_pin_mode(THERM4, PIO_PC31X1_AFE1_AD6);
82      ioport_disable_pin(THERM4);
83      ioport_set_pin_mode(DIST, PIO_PC30X1_AFE1_AD5);
84      ioport_disable_pin(DIST);
85      ioport_set_pin_mode(PR_BRKS_F, PIO_PC29X1_AFE1_AD4);
86      ioport_disable_pin(PR_BRKS_F);
87      ioport_set_pin_mode(PR_BRKS_R, PIO_PC15X1_AFE1_AD2);
88      ioport_disable_pin(PR_BRKS_R);
89
90
91      //DIGITAL PINS / PHOTOELECTRIC
92      ioport_set_pin_dir(PIO_PD18_IDX, IOPORT_DIR_INPUT); //PD18
93      ioport_set_pin_dir(PIO_PA12_IDX, IOPORT_DIR_INPUT); //PA12
94
95  }
96
```

P Code - conf_board.h

..._Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\config\conf_board.h 1

```
1  /*
2   * \conf_board.h
3   *
4   * \brief User board configuration template
5   */
6
7  #ifndef CONF_BOARD_H
8  #define CONF_BOARD_H
9
10 //UART
11 #define IMU_UART                    (Usart *)UART2
12 #define IMU_UART_ID                 ID_UART2
13 #define USB_UART                    (Usart *)UART1
14 #define USB_UART_ID                 ID_UART1
15 #define CONF_UART_CHAR_LENGTH       US_MR_CHRL_8_BIT
16 #define CONF_UART_STOP_BITS         US_MR_NBSTOP_1_BIT
17
18 //Clock resonators
19 #define BOARD_FREQ_SLCK_XTAL        (32768UL)
20 #define BOARD_FREQ_SLCK_BYPASS      (32768UL)
21 #define BOARD_FREQ_MAINCK_XTAL      (12000000UL)
22 #define BOARD_FREQ_MAINCK_BYPASS    (12000000UL)
23 #define BOARD_MCK                   CHIP_FREQ_CPU_MAX
24 #define BOARD_OSC_STARTUP_US        15625
25
26 //DEBUG LEDS
27 #define LED_A                       PIO_PA26_IDX
28 #define LED_B                       PIO_PD21_IDX
29 #define LED_C                       PIO_PA11_IDX
30 #define LED_D                       PIO_PD20_IDX
31 #define LED_E                       PIO_PA10_IDX
32 #define LED_F                       PIO_PD19_IDX
33
34 //CANBUS Silent pins
35 #define SILENT0                     PIO_PE2_IDX
36 #define SILENT1                     PIO_PE1_IDX
37
38 //AFEC pins
39 #define THERM1                      PIO_PC0_IDX
40 #define THERM2                      PIO_PC27_IDX
41 #define THERM3                      PIO_PC26_IDX
42 #define THERM4                      PIO_PC31_IDX
43 #define DIST                        PIO_PC30_IDX
44 #define PR_BRKS_F                   PIO_PC29_IDX
45 #define PR_BRKS_R                   PIO_PC15_IDX
46
47 //DIGITAL pins
48 #define PE_RR_PIN                   PIO_PD18_IDX
49 #define PE_FR_PIN                   PIO_PA12_IDX
50
51 //TWIHS defines and register defines
52 #define TWIHS_CLK                   90000
53 #define ITDS_ADRS                   0x19
54     #define REG_T_OUT                   0x26
55     #define REG_X_OUT_L                 0x28
56     #define REG_X_OUT_H                 0x29
57     #define REG_Y_OUT_L                 0x2A
58     #define REG_Y_OUT_H                 0x2B
59     #define REG_Z_OUT_L                 0x2C
60     #define REG_Z_OUT_H                 0x2D
61     #define REG_CTRL_6                  0x25
62     #define REG_CTRL_1                  0x20
```

```
..._Hyperloop\EMB_Sensors1_2021\EMB_Sensors1_2021\src\config\conf_board.h 2
63 #define PADS_ADRS                   0x5D
64     #define REG_P_OUT_XL                0x28
65     #define REG_P_OUT_L                 0x29
66     #define REG_P_OUT_H                 0x2A
67
68 #endif // CONF_BOARD_H
69
```