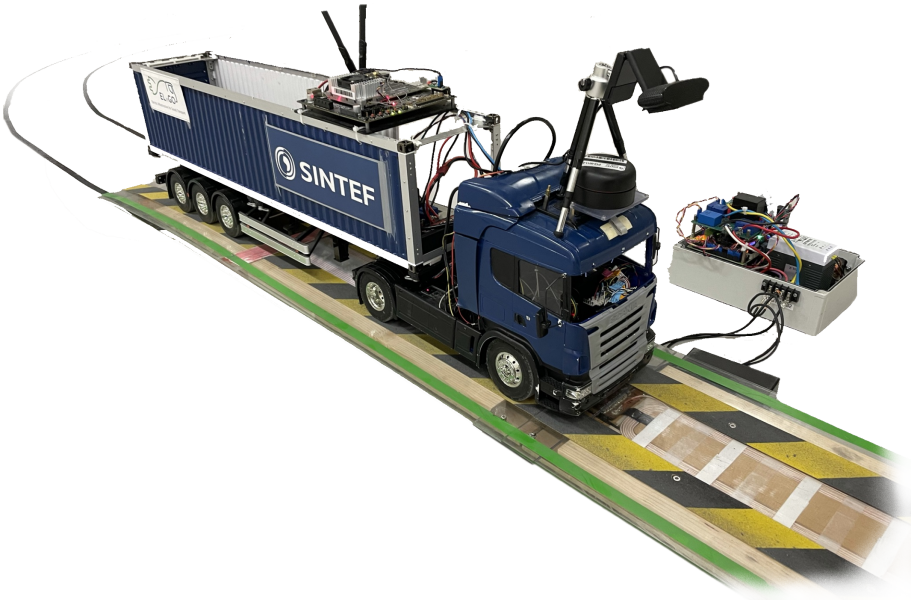
Master's thesis

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

Mats Jørgensen

Computer vision based path tracking for a small-scale electric truck model with dynamic or static wireless charging

Master's thesis in Cybernetics and Robotics
Supervisor: Jon Are Suul
June 2021



NTNU
Norwegian University of
Science and Technology

Mats Jørgensen

Computer vision based path tracking for a small-scale electric truck model with dynamic or static wireless charging

Master's thesis in Cybernetics and Robotics
Supervisor: Jon Are Suul
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Fossil-fueled heavy freight transport generates significant emissions of NO_x and greenhouse gases such as CO_2. In Norway, heavy freight transport is expected to increase by 65 percent within 2050, but Norway aims to reduce greenhouse gas emissions by 80 to 95 percent within 2050. Electrifying heavy freight transport will significantly reduce NO_x and greenhouse gas emissions. Electric trucks usually feature Li-ion battery packs with a 100-300 kWh capacity, and charging such a massive battery pack in a reasonably short time is challenging. For electrifying heavy freight transport, alternative charging opportunities for the batteries are essential for a feasible solution [1].

Wireless power transfer from a road is an alternative to the conventional plug-in-based charging stations operating today for charging battery-electric vehicles. Recent technology advances have made such a solution practically feasible with inductive charging technology. There are many advantages for wireless charging applications, such as no need for a mechanical plug. This makes it more mechanically robust, and more accessible for automated charging with no manual intervention. The alignment of coils in wireless inductive charging is crucial for an efficient charging cycle. This is where the combination of autonomous driving and wireless charging seems like a perfect match. With the use of a variety of sensors, autonomous driving systems could keep a precise position of an electric vehicle down to the centimeter [2] [3].

SINTEF has used a 1:14 scale battery-electric truck model with an associated dynamic wireless inductive charging system for demonstrating this technology on a smaller scale. Previous students at NTNU have developed autonomous driving functions for the truck model and Bluetooth communication with the charging station. There is still room for improvements, and this study has furthered the work to reach a more stable and robust autonomous computer vision based path tracking to achieve a more efficient charging cycle. A feedback signal was received by connecting to the internal potentiometer at the servo operating the steering angle. Then a closed-loop PID controller for the steering was implemented. In addition, a new camera has been mounted at the top of the cab at the truck model and merged with an existing camera to reach a wider detection area. The field of view has been expanded from 30 - 100 cm to 0 - 100 cm in front of the truck model. Sharp corners could previously be overlooked, and these improvements gave more precise steering.

A mockup track was made for testing, and the position of the truck model has been logged with both an IMU and LiDAR to provide a visual display of enhanced autonomous driving. A new function has been implemented to activate and deactivate the road-side coils at an exact position. Various tests have been performed to find an optimal charging cycle with activation and deactivation of the road-side coils in different positions. A null-point is used as the starting point, which is the position where the vehicle-side coil starts to receive power from the road-side coils. Results from the tests showed that the null-point activation reached the highest transferred energy at 66 percent efficiency, but a 5 cm shorter activation of the null-point gave an overall higher efficiency at 67 percent. There has also been implemented an opportunity charging alternative to test static wireless charging where one of the rectangular road-side coils was exchanged with a square road-side coil. With a static solution, the highest efficiency was reached at 80 percent efficiency. However, the increased efficiency comes at the cost of zero distance traveled while charging.

Sammendrag

Fossildrevet tungtransport genererer betydelige utslipp av NO_x og klimagasser som CO_2. I Norge forventes tungtransport å øke med 65 prosent innen 2050, men Norge har som mål å redusere klimagassutslippene med 80 til 95 prosent innen 2050. Elektrifisering av tungtransport vil redusere NO_x og klimagassutslipp betydelig. Elektriske lastebiler har vanligvis Li-ion batteripakker med en kapasitet på 100-300 kWh, og det er utfordrende å lade en så massiv batteripakke på rimelig kort tid. For å elektrifisere tungtransport er alternative lademuligheter for batteriene avgjørende for en gjennomførbar løsning [1].

Trådløs effektoverføring fra en vei er et alternativ til de konvensjonelle plugg-baserte ladestasjonene som fungerer i dag for lading av batteri-elektriske biler. Nyere teknologiske fremskritt har gjort en slik løsning praktisk mulig med induktiv ladeteknologi. Det er mange fordeler for applikasjoner for trådløs lading, slik som at det ikke er behov for et mekanisk støpsel, noe som gjør den mer mekanisk robust og mer tilgjengelig for automatisk lading uten manuelle inngrep. En presis overlapping av spoler i trådløs induktiv lading er avgjørende for en effektiv ladesyklus. Det er her kombinasjonen av autonom kjøring og trådløs lading virker som en perfekt kombinasjon. Ved bruk av en rekke sensorer kan autonome kjøretøysystemer holde en presis posisjon for et elektrisk kjøretøy med centimeterpresisjon [2] [3].

SINTEF har brukt en 1:14 skalert elektrisk lastebilmodell med et tilhørende trådløst induktivt ladesystem for å demonstrere denne teknologien i mindre skala. Tidligere studenter ved NTNU har utviklet autonome kjøretøysfunksjoner for lastebilmodellen og Bluetooth-kommunikasjon med ladestasjonen. Det er fremdeles rom for forbedringer, og denne studien har bidratt til arbeidet med å nå en mer stabil og robust autonom datasynsbasert banesporing for å oppnå en mer effektiv ladesyklus. Et tilbakemeldingssignal ble mottatt ved å koble til det interne potensiometeret ved servoen som betjener styringen. Deretter ble en lukket-sløyfe PID kontroller implementert. I tillegg er det montert et nytt kamera øverst på førerhuset på lastebilmodellen som er kombinert sammen med et eksisterende kamera for å oppnå et større deteksjonsområde. Synsfeltet er utvidet fra 30 - 100 cm til 0 - 100 cm foran lastebilmodellen. Skarpe hjørner kunne tidligere overses, og disse forbedringene ga en mer presis styring.

Det er laget en bane for testing, som inkluderer ladestasjonen for lastebilmodellen. Lastebilmodellens posisjon er logget med både en IMU og LiDAR for å gi en visuell visning av forbedret autonom kjøring. En ny funksjon er implementert for å aktivere og deaktivere ladestasjonsspolene i en nøyaktig posisjon. Det er utført forskjellige tester for å finne en optimal ladesyklus med aktivering og deaktivering av spolene ved ladestasjonen i forskjellige posisjoner. Nullpunktet brukes som startpunkt, som er den nøyaktige posisjonen der spolene begynner å overføre og motta effekt. Resultatet av testene viser at en nullpunktaktivering nådde den høyeste overførte energien ved 66 prosent effektivitet, men en 5 cm kortere aktivering av nullpunktet ga en samlet høyere effektivitet på 67 prosent. En mulighets-lading system er implementert for å teste statisk trådløs lading der en av de rektangulære vegspolene er byttet med en kvadratisk spole. Med en statisk løsning ble den høyeste effektiviteten nådd, ved 80 prosent effektivitet. Dette utgjør i gjengjeld ingen tilbakelagt avstand under lading.

Preface

This master's thesis is written as the work of the 30 credits subject TTK4900 - Engineering Cybernetics at NTNU (Norwegian University of Science and Technology) in the spring of 2021. The study program is Cybernetics and Robotics, and is located in Trondheim. The thesis aims to integrate further autonomous driving and wireless inductive charging from a charging station for a small-scale electric truck model. The truck model has been used in two previous master's theses, where other autonomous functions have been implemented. The purpose is to develop a more stable and robust autonomous driving for the truck model and to find an optimal charging cycle. Thus, one can investigate the effect of different positions for activation and deactivating the coils at the charging station and how it affects the efficiency of the charging cycle.

The hardware is provided from SINTEF, which includes a 1:14 scale battery-electric truck model with a wireless inductive charging system. The truck model is mounted with an embedded computer Nvidia Jetson TX2, a microcontroller Teensy 3.2, cameras, LiDAR, a hall effect sensor, IMU, and a joystick for operating. The main new implementations are a new camera merged with an existing camera for a computer vision based path tracking, PID controllers, opportunity charging solution, LiDAR logging, and precision activation of the charging station coils. The software is running on an embedded Linux platform, with significant use of library and tools from ROS (Robot Operating System) for system interface and the library OpenCV for real-time computer vision. The development is done in Python and C, with some minor use of C++.

I would like to thank my supervisor, Associate Professor Jon Are Suul at the Department of Engineering Cybernetics, NTNU, and Research Scientist Dr. Giuseppe Guidi at SINTEF Energy Research, for their support in this study.

Contents

Figures

Tables

Nomenclature

Symbols

ω	Resonant frequency
k	Normalized magnetic coupling coefficient
Q	Quality factor
V_x	Base voltage
P_N	Base power
I_N	Base current
Z_N	Base impedance
I_N	Base current
r_x	Normalized resistance
i_x	Normalized current
v_x	Normalized voltage
C	Capacitance
R	Resistance
M	Mutual coupling coefficient
L	Inductance
x_c	Scaling factor
u	Controller output
K_p	Proportional gain
K_i	Integral gain
K_d	Derivative gain
e	Regulation deviation
δ	Steering angle
R	Radius
l	Axle length
t	t-test
σ	Standard deviation
α	Significance level
η_E	Energy transfer efficiency
η_{OE}	Overall energy transfer efficiency

Acronyms and Abbreviations

ADC	Analog to Digital Converter
AI	Artificial Intelligence
ALKS	Automated Lane Keeping Systems
CAN	Controller Area Network
DDS	Data Distribution Service
EMF	Electromagnetic Field
ESC	Electronic Speed Controller
GPS	Global positioning system
HF	High Frequency
HSV	Hue, Saturation, Lightness
I/O	Input/Output
IGBT	Insulated Gate Bipolar Transistors
IMU	Inertial Measuring Unit
IoT	Internet of Things
IPT	Inductive Power Transmission
LiDAR	Light Detection And Ranging
MOSFET	Metal-Oxide Semiconductors Field Effect Transistor
MQTT	Message Queuing Telemetry Transport
NAF	Norwegian Automobile Federation
NTNU	Norwegian University of Science and Technology
OpenCV	Open Source Computer Vision Library
P2P	Peer-to-Peer
PID	Proportional Integral Derivative
PLL	Phase Locked Loop
PWM	Pulse-Width Modulation
Radar	Radio detection and ranging
RFCOMM	Radio frequency communication
RGB	Red, Green, Blue
ROS	Robot Operating System
SAE	Society of Automotive Engineers
SLAM	Simultaneous Localization And Mapping
SSH	Secure shell
TCP/IP	Transmission Control Protocol/Internet Protocol
WPT	Wireless Power Transfer
ZN	Ziegler-Nichols

Chapter 1

Introduction

This thesis aims to contribute to the integration of autonomous driving and wireless inductive charging systems for a small-scale electric truck model. A dynamic inductive power transfer system to a moving electric vehicle has been developed at SINTEF Energy Research. Previous students at NTNU (Norwegian University of Science and Technology) have implemented autonomous driving functions for the truck model. However, there is still room for improvements in order for the systems to be stable and fully reliable. This study will further the truck model with hardware and software upgrades to demonstrate stable autonomous driving and a fully functional wireless inductive charging on a moving electric vehicle. This chapter will describe the background and motivation for this study, followed by a description of the truck model and finally a presentation of the problems this study aims to solve.

1.1 Background and motivation

Battery-electric vehicles, from tank-to-wheel, are emission-free. They only have upstream emissions from manufacturing and electricity generation, and analyses from a complete CO_2 cycle has concluded that the average electric vehicle has less emission than the average fossil-fuel vehicle [4]. Heavy freight transport generates significant NO_x and greenhouse gas emissions but is essential for business to thrive. In Norway, heavy freight transport is expected to increase by 65 percent within 2050 [1]. Electrifying heavy freight transport will significantly reduce NO_x and greenhouse gas emissions. Almost a third of Norwegian greenhouse gas emissions originate from the transport sector, and over half of these emissions are from road traffic. Norway has obligated itself to the Paris agreement, an international agreement to limit climate change. With this commitment, Norway aims to reduce greenhouse gas emissions by 40 percent within 2030, and by 2050 emissions will be reduced with 80 to 95 percent. This is an ambition to become climate neutral [5].

Despite the progress that has been made within the area of charging electric vehicles, a part of the general public still considers battery-electric vehicles as unsuitable for long-distance driving. This is because of the time needed for recharging the batteries and with limited onboard energy storage capability [6]. According to numbers from NAF (Norwegian Automobile Federation), Norway is missing 1100 fast-chargers for electric vehicles (numbers from March 2021), where the number of electric vehicles is increasing significantly, and the number of fast-chargers is not keeping up in the same pace. To date, there are about 2000 fast-chargers in Norway, but by the end of year 2021, NAF estimates that there should be about 3000 fast-chargers. This causes daily queues and waiting for a fast-charger for the average electric vehicle owner. According to the Norwegian government, there will be about one million electric vehicles in 2025, and in 2030 this will increase to two million electric vehicles (as of March 2021, there are 340 000 electric vehicles) [7][8]. To put the numbers in perspective, in 2016, there were 12.7 electric vehicles per fast-charger. In 2020 there were 19 electric vehicles per fast-charger according to Statistics Norway [9]. One solution to this problem

could be wireless power transfer to the vehicle from the road, for propulsion and/or battery charging.

In recent years, a variety of battery-electric trucks have been developed, and many others are currently under planning. They all have featured Li-ion batteries as the primary source for the truck's propulsion, where the size of the battery depends on the driving range and payload of the truck. The batteries are limited by their weight, volume, and cost. It is expected that the batteries will have a capacity of 100-300 kWh, which must be able to deliver 250-500 kW of driving power. Constructing infrastructure to charge such massive battery packages in a reasonable short time will be challenging, thus, investigating different charging methods is essential. Recent technology advances have made dynamic wireless power transfer from the road practically feasible solution with wireless inductive charging technology [2].

Wireless charging is becoming standard in many applications, and there are many advantages to this. It does not need a physical charging plug, making it more robust for mechanical damage. Further, the casing could be sealed off and be more resistant to dust and water contamination. This eliminates the direct electric contact, thus, keeping elements such as snow, ice, other fouling, corrosion, and the potential for leakage currents away. Significant research and development efforts have been directed at wireless inductive power transfer systems in the last few years. However, most of the attention has been directed at replacing conventional plug-based chargers with a power level of a few kilowatts [3].

Autonomous technology has many advantages, where it, for instance, allows vehicles to drive very precisely. For an efficient wireless inductive charging system, the alignment of the transmitting and receiving coil is crucial. Autonomous vehicles and wireless charging systems seem like a perfect match, where autonomous driving systems could keep a precise position of an electric vehicle down to the centimeter. A dynamic inductive power transfer to a moving autonomous electric vehicle will allow the vehicle to have a long-term automated operation, thus making it fully self-sustained. This allows for a long-term zero-emission operation of an electric vehicle. Consequently, this will streamline the process with a self-driving vehicle without the need for any manual interaction. In the future prospect, these technologies could have a considerable impact, where they can be utilized for a variety of applications. That is, these could create fully autonomous, self-supplied independent systems for vehicles as discussed, and e.g. drones with an inductive charging pad, ferries with auto-docking and contact-less charging, etc.

1.2 The small-scale electric truck model used in this study

The small-scale truck model used in this study is shown in figure 1.1. It is a Tamiya 1:14 scale replica of a Scania R470 Highline model, with a three-speed transmission. The truck model is operated by two servos and an ESC (Electronic Speed Controller). The first servo is employed for the gear selection, the second servo steers the two front wheels, and the ESC controls the speed. The truck is mounted with an Nvidia Jetson TX2 embedded computer to operate the autonomous functions. A Teensy 3.2 microcontroller is mounted inside the truck's cab for control of the servos, ESC, and sensors. A hall effect sensor is used to feedback the speed and distance traveled, and an IMU (Inertial Measuring Unit) sensor is used for odometry.

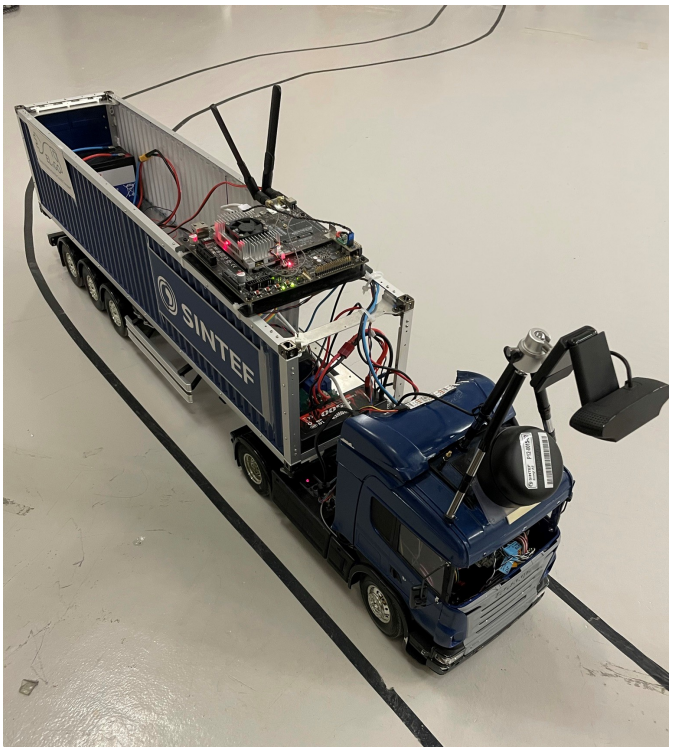


Figure 1.1: Truck overview

A new camera is installed on a tripod on top of the cab in this study, a second camera is located inside the cab, and a LiDAR (Light Detection And Ranging) sensor located on top of the cab. These are the primary sensors in the self-driving functions. Inside the trailer, there are two batteries: one for powering the truck's propulsion and one for powering the electronics. The vehicle-side of the wireless charging system is also located in the trailer, with an associated square coil mounted under the truck model. The hardware and software of the truck model are described in detail in chapter 3. A part of the mockup track for testing the truck model can be seen in figure 1.1, with a 25 cm lane width, which corresponds to a 1:14 scale of a Norwegian standard road of 3.5 m width.

1.3 Objectives to solve

Previous students at NTNU have developed various autonomous functions for the truck model, such as SLAM (Simultaneous Localization And Mapping) based path tracking, deep learning steering controller, and computer vision steering controller for path tracking. Bluetooth communication with the road-side charging station has also been implemented. The truck model does not yet have a consistent charging capability, which could lead to inefficient charging cycles. This is a consequence of the lack of precision in the autonomous driving functions while operating and the camera detection area has a blind spot in front of the truck model. An open-loop controller is utilized for the steering, which could lead to inaccurate steering. There is also a need for precise activation and deactivation of the road-side coils at the charging station for an efficient charging cycle, because the road-side coils have an energy loss at energized idling time. This study intends to further the previous students' work, create a more robust and reliable autonomous driving function, and find optimal wireless inductive charging for the truck model. The upgrades will be done by implementing a new camera for a computer vision based path tracking to eliminate the blind spot in front of the truck model, and receiving feedback from the steering angle to design a closed-loop controller for the steering. Other functions and methods will be investigated to improve the truck model, such as precision activation and deactivation of the road-side coils, LiDAR logging of the truck's position, enhanced obstacle detection, and a wireless opportunity charging expansion. These upgrades could give the truck model

a more stable operation and more precision while driving. In addition, with improved self-driving and precise road-side coil activation and deactivation, tests to investigate when it is most efficient to activate and deactivate the road-side coils could be studied, and the effects of an opportunity charging. The most important result will be included in a scientific paper for the IECON conference, and a draft of the paper is available in appendix A.

1.4 Thesis overview

In this section, an overview of the chapters in this thesis is presented with a short description.

Chapter 1 - Introduction
The first chapter describes the background and motivation behind this study and introduces the technologies that could improve the small-scale truck model. Additionally, there will be a presentation of the truck model and the objectives this study aims to solve.

Chapter 2 - Autonomous vehicles and wireless power transfer systems
The second chapter contains a deeper insight into the main technologies used in this study, namely autonomous vehicles, and wireless transfer systems. Firstly, there will be given a short historical introduction and a description of these systems and how they are utilized today. Lastly, a combination of these systems is examined.

Chapter 3 - System structure
The third chapter presents the truck model's hardware and software structure. The charging system is described with a system for dynamic and static wireless charging. This thesis builds on the work from previous students, and it is described what this study has contributed to. The libraries used to create the software system are explored before protocols for a full-scale solution are investigated.

Chapter 4 - Truck-model upgrades and implementation
The fourth chapter describes in detail the new upgrades and how they have been implemented in the system. The new improvements are the feedback from the servo and its new closed-loop PI controller. Further improvements are the new camera implemented and the modeling for a new computer vision based path tracking. Additionally, a charging cycle with precise activation/deactivation of the road-side coils is implemented, and a new opportunity charging solution is described. The LiDAR implementations have been extended, with a new logging function for the position and enhanced object detection.

Chapter 5 - Experimental testing of autonomous driving and the charging cycle
The fifth chapter presents and examines the test results of the new upgrades and implementations. These results are first logged with the IMU for the autonomous driving test before the new LiDAR logging function. Next, extensive testing of the optimal charging cycle is presented. The main goal is activation and deactivation of the road-side coils to find the most efficient charging cycle and how this impacts the transferred energy. Lastly, the opportunity charging solution is tested, with two approaches for charging. These approaches are quasistatic and static charging.

Chapter 6 - Discussion
The sixth chapter is a discussion of the results and methods of the upgrades and implementations in the previous chapters. New solutions are discussed.

Chapter 7 - Conclusion and future work
The seventh and final chapter is the conclusion of this study, with the new upgrades and implementations and the experimental tests. It presents how this study could be furthered with new solutions.

Chapter 2

Autonomous vehicles and wireless power transfer systems

A description of the two main technologies used in this project, namely autonomous vehicles and wireless charging systems, is provided in this chapter for a deeper insight into this study. There will be given a brief historical introduction, a description of their designs, and how and where these systems are in use. Furthermore, a combination of these systems will also be explored.

2.1 Autonomous vehicles

For the last 100 years, autonomous vehicles, also known as self-driving vehicles, have been researched. One of the first demonstrations of an autonomous vehicle was a radio-controlled driver-less car in the 1920s. Since then, car manufacturers and universities have made a variety of efforts to pioneer autonomous vehicles. A few decades after the 1920s, road-powered autonomous vehicles from embedded electronic devices in the roadway were reviewed in the UK and parts of the US, but re-designing the streets to include electronic railings was considered too expensive at the time. Since then, the focus was mainly shifted to autonomous cars that could drive on the existing streets. In the 1980s, Ernst Dickmanns and a team at Bundeswehr University Munich, Germany, successfully drove autonomously on an empty highway, over a distance of 20 km with a top speed of 96km/h. In 1989 the vehicle could recognize obstacles, and in the 1990s, it performed an autonomous lane change. Since then, even with several successful demonstrations throughout the years, fully autonomous vehicles did not seem possible to become a reality until recent years [10].

An autonomous vehicle needs robust situation systems that will allow the vehicle to understand its surroundings. This includes detection of the road, other vehicles, cyclists, pedestrians, animals, other unforeseen obstacles, etc. With these systems, the vehicle must assess and calculate a corresponding response and find the best possible action for the vehicle. Still, the most advanced perception methods were insufficient until recent breakthroughs in Deep Learning (DL), a subfield of Artificial Intelligence (AI). This allowed large data sets with high computing power to develop a system of superhuman performance to assess situational awareness properly. In 2012 the first breakthrough in DL within computer vision was made. Since then, facial recognition and performing image segmentation in various environments have been achieved. In camera images, convolutional neural networks can detect and track objects of interest and accomplish much better performance than traditional computer vision strategies. This is due to how the DL system can learn from large datasets and discover beyond them [6].

Autonomous vehicles rely on a variety of sensors for the perception of the environment they operate in, allowing the vehicles to drive with centimeter precision. The key sensors are cameras, LiDAR, and radar (radio detection and ranging). Other sensors in use could be ultrasonic sensors, IMU, and

GPS (Global positioning system). Self-driving systems often use a combination of these sensors to control the speed, brake, and steering of the vehicle, where they all have different applications with advantages and disadvantages. The camera is essential for self-driving vehicles and is the leading choice for car manufacturers. Cameras cover the visual objects, such as lanes, streetlights, distances, etc., but the cameras could be less robust against different light conditions and have high computational costs. The LiDAR transmits infrared laser pulses and measures the reflection time. From this information, a 3D map could be constructed to model the environment, and it can identify moving objects. The LiDAR is an expensive sensor and is not the most reliable sensor in bad weather conditions and at reflective surfaces. The radar uses radio waves and measures their reflection time to map the surroundings, and unlike the LiDAR, the radar is not affected by weather conditions. A radar can be used for object detection, speed and distance measurement, etc. The radar has less angular accuracy, which indicates how small of an angle the sensor can measure, and it generates less data than LiDAR. Ultrasonic sensors are useful in close range detection by emitting ultrasonic waves and transfers the reflected sound to an electric signal, which could be used as an aid when parking. The GPS could be used for the location of the vehicle, and IMU for the vehicle odometry [11]. A simplified block diagram of an autonomous vehicle system is illustrated in 2.1.

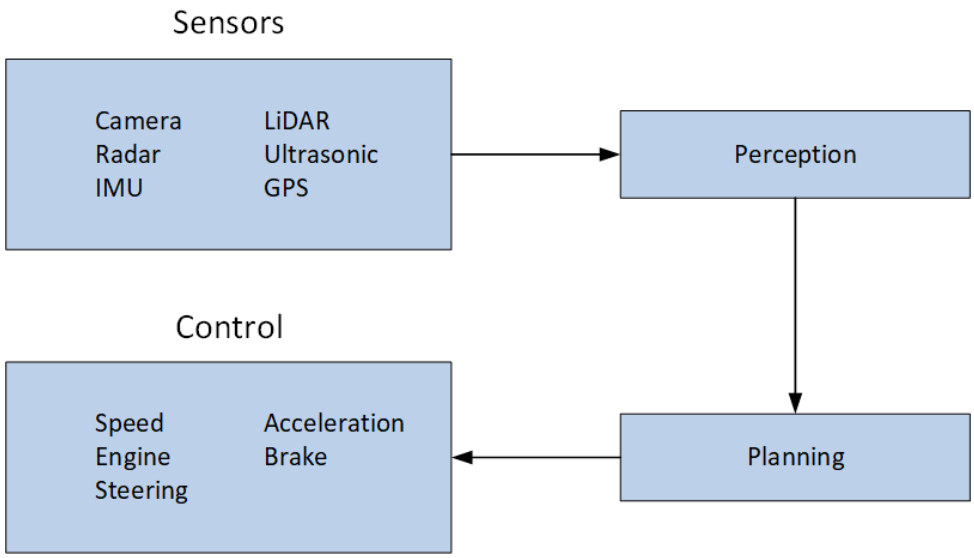


Figure 2.1: Block diagram of an autonomous system [11]

2.1.1 Levels of driving automation

There are six different levels of driving automation defined in the SAE (Society of Automotive Engineers) International standard J3016, which are based on the levels of functionality of the autonomous technology, and is shown in figure 2.2. The levels are a step-wise progression, which categorizes if a person or an automated driving system performs the dynamic driving tasks. The dynamic driving task involves the operational (steering, braking, accelerating, monitoring the roadway and vehicle) and the tactical (responding to events, change lanes, turning, signals, etc.) aspects. At level 0 a human driver operates all functions, at level 1 the driver has limited help, and at level 2 the system can perform the critical functions. At level 3 a human driver still has to monitor the system as a fallback solution if the systems fail, and at level 4 the system have a high automation and could operate by it-self at roads designed for this type of autonomy. At level 5 the system can handle all situations under all conditions. In the levels 0-1-2 the environment is monitored by a human driver, and in the levels 3-4-5 it is monitored by an automated driving system. Although there are many vehicles today that can practically drive by themselves, the law states that it is always the driver who has the full responsibility of the driving in the case of an accident [12].

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 2.2: SAE autonomous driving levels. Ill: [12]

2.1.2 The laws and challenges of autonomous vehicles

In 2017, a law on testing autonomous vehicles was introduced in Norway. The purpose was to uncover the effects that self-driving vehicles have on the traffic safety, efficiency in traffic flow, mobility, and environment. In principle, in the law, there must be a person in the driver's seat who can be kept liable under the road traffic act for accidents. However, for testing purposes, exceptions can be granted, and there is no responsible person in the self-driving vehicle. If a person affects the driving system under the self-driving state, the person is still considered to be the responsible driver. The law states that there must be a designated person responsible for the trial of self-driving vehicles. In some cases, both the driver and the designated person could be held accountable in the case of an accident [13]. For instance, the US, one of the world-leading countries for technology and innovation for autonomous vehicles, has laws for testing, but the state statutes are not identical. The states have different requirements for testing and operation, and many states allow for manufacturers to test autonomous vehicles at level 3 and 4 autonomy [14].

In 2020, 60 countries reached the UN (United Nations) Regulations on Automated Lane Keeping Systems (ALKS) that will allow for level 3 automated vehicles in certain traffic environments. The regulations are expected to take effect in 2021, and the countries must still adapt their regulations on autonomous cars. A level 3 autonomous car still implies that the driver must be aware and prepared to take over the control quickly when the system requires it. It is still not legal to use your mobile or other electronic devices during driving. Today, there are many cars with automation level 2, where the technical term is a driver support system where the driver has full responsibility for controlling the vehicle. There is a big difference from the automation level 2 to level 3, where the car manufacturer has to take responsibility for the self-driving and could not shift the responsibility onto the driver [15].

The UN Regulations of ALKS only apply in the first instance to passenger cars with a maximum of eight seats and speed up to 60 km/h. It has to use a road where pedestrians and cyclists are prohibited from intervening, and there must be a physical separation that divides the traffic in the opposite direction. There are some minimum safety requirements that must be in place:

- Emergency maneuvers in case of an imminent collision.
- Transition demand, when the system requires the driver to take control.
- Minimum risk maneuvers when the driver does not respond to the transition demands, for example, being able to stop at the road shoulder with the warning lights on.
- Data Storage System for Automated Driving (DSSAD), which can record when the ALKS is activated and store the data of driver input, failures, transitions demands etc.
- ALKS have to be compliant with cybersecurity and software updates.

The technology for a level 3 autonomous vehicle is still vulnerable, and it is to this date unclear who has the full responsibility in an accident. Because of the regulations today, there may be situations where the people involved, such as the person driving, the car manufacturer, or the road owner, will try to shift the responsibility to each other. One could imagine the debate if an accident involves an autonomous vehicle with serious personal injury, or in the worst case, death. With the guilt claims operated today, it would be difficult to hold a person who was not even at the accident scene responsible [16].

Among the many challenges of an autonomous vehicle, cybersecurity is central. It is crucial that adequate measures are taken to prevent the systems in the vehicles from being hacked, where outsiders can take control of the vehicle or gain unauthorized access to information. From the Act of testing autonomous vehicles in Norway, it is the designated person responsible for the testing, together with the technology developer, who is responsible for cybersecurity. The law states that it is assumed that there is invested resources in developing secure systems, which are well equipped against computer attacks. In 2019, a prestigious annual hacking event took place, the Pwn2Own event. One of the challenges was to hack a Tesla Model 3, and two persons only used a few minutes to discover a weakness in Tesla's infotainment system, and was able to get inside one of the car's computers [17]. This demonstrates that one of the market leaders in self-driving cars could have security flaws, although the security weakness was fixed by Tesla after it was reported. Since autonomous vehicles are required to get online, access data traffic, download new patches, etc., there will always be a risk for computer attacks [18].

The algorithms used in autonomous vehicles are also facing many ethical dilemmas. For instance, how the vehicle should act if it is facing an unavoidable accident. The ethical dilemmas include whether it should protect the occupant of the vehicle at all cost, or if it should sacrifice the occupant in the vehicle to minimize the loss of life. There is also the question of whether the vehicle should choose randomly from these extremes. However, these types of ethical dilemmas are beyond the scope of this study, but are a challenge of autonomous vehicles, which could lead to life and death situation [14]. Difficult weather conditions are one of the problems autonomous vehicles could encounter, such as heavy rain, snow, ice, storms, fog, etc. The vehicle sensors could be blocked, and snow or ice can block lanes and road signs. These technical challenges have been researched, and one possible solution for the blocked lanes has been found by the car manufacturer Ford. They used the LiDAR to detect landmarks above the road, and then the vehicle could use these landmarks to identify and switch to pre-stored high-resolution maps onboard the vehicle to drive. One flaw of this solution is that the vehicle needs to have pre-stored maps, which could in some places not yet exist. This technique is not unique to Ford, but it was the first car company to publicly show it could use the maps to navigate in a snow-covered road within a centimeter at any given moment [19].

2.1.3 Autonomous navigation techniques

Computer vision enables a computer to process and identify objects from digital images or videos that a human would do. It aims to duplicate the effects of how a person intervenes by understanding and perceiving an image. Many modern computer vision uses AI in DL and neural networks to detect and label objects [20]. DL can process unlabeled raw data and use these data to analyze the context of a scene. It focuses on essential objects and ranges them in hierarchy levels, from small to large objects and at low to high resolution. DL requires a large amount of high-quality data to reach a high accuracy, and it is insensitive to variations in the environment. A neural network could be explained as multiple algorithms that endeavor to recognize underlying relationships in large datasets, which consist of processes that mimic a human brain. An AI neural network can learn complex interactions between features by considering and processing sample observations, minimizing the observed error to improve accuracy or better handle a task [21]. An illustration of object detection with a convolutional neural network like YOLO (You Only Look Once) is shown in figure 2.3.

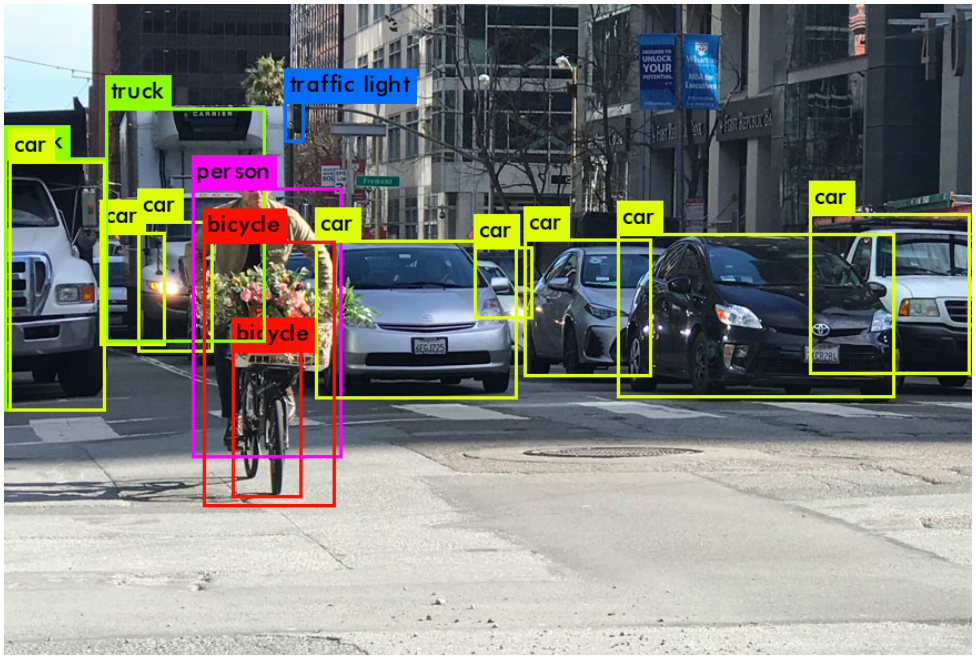


Figure 2.3: Object detection with neural network YOLOv2. Ill: [22]

A SLAM method could be used in computer vision for mapping the environment. The method must have the ability to build a reliable map, and locate itself in the map, without any human interactions. SLAM is a complex operation, where localization is needed for mapping, but a map is needed for the localization. Both of these factors are unknown. There are various methods for solving this problem, where most methods rely on odometry, LiDAR readings, and digital camera images. If camera images are used, it is called VSLAM (Visual Simultaneous localization and mapping) [23].

2.1.4 Autonomous vehicles operating today

Many major car companies are in the process of testing autonomous vehicles, and one of the leading companies in self-driving technology is Waymo, a sister company of Google. Their car is a typical self-driving car, operating at level 4 autonomy where no one is sitting behind the steering wheel. It uses a high-resolution camera and LiDAR for the autonomous functions. The cars have millions of km of driving data for the training data, and still, it was not enough. To compensate and extend the cars' training, the cars also train based on simulated data. By February of 2020, the self-driving cars of Waymo have driven 32 million km, primarily in its testing market in Arizona, US, and not had a single fatal accident. Still, it is too soon to draw some conclusion if these self-driving vehicle is safer

than a human driver, as in the US, where it is tested, a fatal accident accrues about every 160 million km driven. In July 2015, after the first million miles (1.61 million km) were driven, it was involved in 14 minor accidents. In all cases, there was no fault with the self-driving functions. Either the other driver was responsible, or the Waymo car was driven manually. The Waymo cars operating today are sedans, and they are used for personal transports. In March 2020, the Waymo Via truck, a class 8 heavy truck, was launched. The purpose of the trucks is to transport goods in the shipping routes southwest in the US [24] [25].

Several different companies are testing autonomous trucks, such as Tesla, Uber, Volvo, and TuSimple. One advantage of an autonomous truck is its large size and height, which gives an improved field of view for sensors and room for more power to the computers. TuSimple's project is being tested in Arizona and Texas, US, where a truck is transporting goods for shipping companies. The trucks are being run with level 3 supervised autonomy, where a human driver is ready to overtake the control of the truck if needed. By 2021, TuSimple's plan is to let the trucks drive by themselves without any human driver on board. An average LiDAR sensor, used by most autonomous vehicles, has a practical range of detecting a 360 degree range of 200 meters. TuSimple has calculated that it is not sufficient for a fully loaded truck, traveling up to 120 km/h, to rely on a 200 meters range to make its decisions. Instead, their primary sensors are multiple HD cameras, which are capable of detecting up to 1000 meters ahead. According to TuSimple, 1000 meters is twice as far as an experienced truck driver can see ahead while driving. They point out that the efficiencies of the autonomous system keep the truck to break less often than with a human truck driver, leading to an improvement of fuel economy of 10 percent and less tire wear. By 2024 they plan to achieve a level 4 autonomy [26].

Tesla is one of the leading car manufacturers of self-driving technology. Their self-driving functionality relies on multiple cameras, ultrasonic sensors, and a radar. Their sensors give up to 250 meters range in front of the vehicle, and a forward-facing radar provides ambient data. The radar allows the vehicle to see through heavy rain, fog, dust, and even the car in front. Although Tesla's autopilot enables the vehicle to operate autonomously, with file changes, optimization of the route, navigation by itself, etc., and have the hardware to drive itself under almost all conditions, the vehicle is still classified as a level 2 autonomy. Tesla state that autopilot requires active driver supervision, which does not make the Tesla autonomous, but has the ability to drive on its own in the future [27].

Uber has decided to close down their self-driving truck program and only focus on self-driving cars. Their reason was to focus on one vehicle platform rather than two groups working side by side. They will still keep a relationship with truck manufacturers and may return after Uber has developed the self-driving system's foundation. In 2018, Uber was involved in a fatal accident, where a pedestrian was killed by a self-driving car. This was the first recorded case of a pedestrian fatality where a self-driving car was involved. The car involved was not operating at level 4 or 5 autonomy, and there was a human driver inside the car monitoring the vehicle [28].

2.2 Wireless power transfer

Wireless transmission of electricity has been investigated for centuries, from the pioneer Faradays experiments of electromagnetic induction and energy transmission in 1832, Hertzs radio frequency communication in 1895, and Nicola Teslas wireless electric energy transmission at long ranges in 1904. N. Tesla made a patent for wireless transfers by using inductors, where he identified two essential parameters for the transmission. These parameters indicated that increased frequency improved the power transfer capability, and by connecting capacitors to the coils it created a resonant system that enhanced the effectiveness of the transmission. Since then there has been developed various wireless power transfer (WPT) systems over the years, and figure 2.4 shows the main technologies of the wireless power transfer [29]. In this study, the resonant inductive power transmission is used.

A resonant inductive power transfer utilizes magnetic field induction between two coils for power transfer, where the efficiency depends on the distance between the coils, the frequency, the current excitation, and the coils geometry [30].

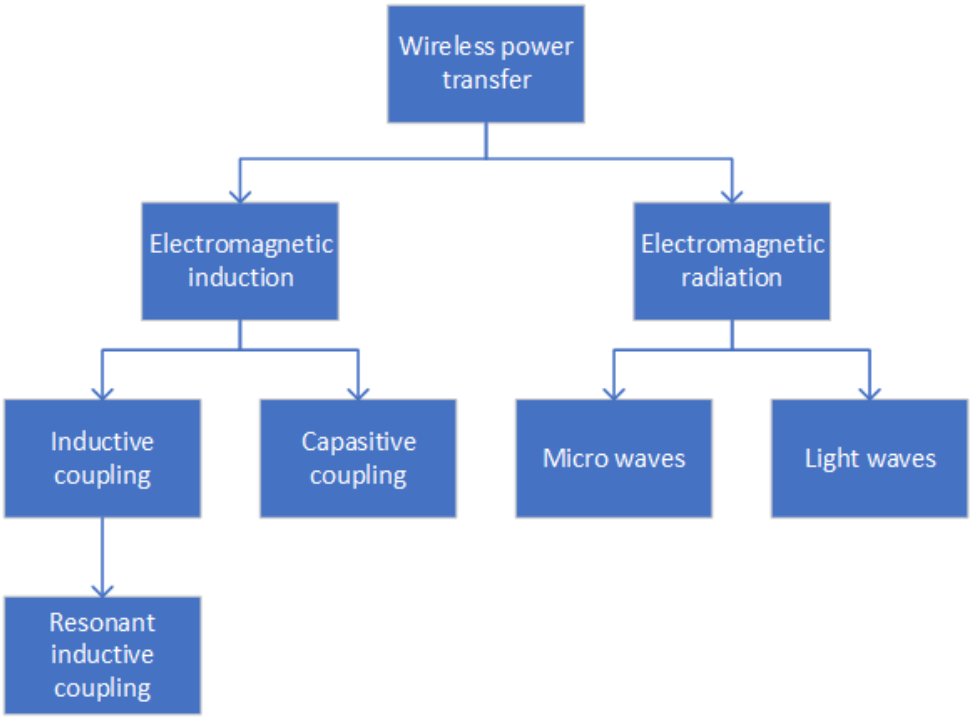


Figure 2.4: Wireless power transfer systems

The first application of an IPT system to an electric vehicle was performed in 1943 by a Soviet engineer, Georgiy Babat. With an electron-tube oscillator, at an air gap of 20 cm, hundreds of amperes were provided with a frequency at 50 kHz for a direct supply of a 2 kW motor. The system had an efficiency of 4 percent. The first working IPT system to a moving vehicle was achieved in the 1980s in California. With an air gap between 5 to 10 cm, up to 2000 A was provided at a frequency of 400 Hz, and gave a power of 200 kW. This system had an efficiency of 60 percent. Since the 1990s, a massive interest for IPT systems started, and there has been proposed and developed a variety of systems for static and dynamic charging. This is due to the improved performances with hundreds of transmitted amperes with a frequency of tens of kilohertz [29].

The main coil in the road could be circular or rectangular, where they both have advantages and disadvantages. The circular coils are more compact, have a lower electromagnetic field (EMF), and a lower weight. However, they are limited by their power transfer capacity and are less lateral tolerant, indicating that they are more sensitive for distance alignment of the power transfer. The rectangular coils have a higher lateral tolerance and coupling factor but have a larger EMF and higher weight [31].

A dynamic IPT system will enable an electric road system to charge a battery-electric vehicle while it is moving. An advantage of this system is that the required capacity of the electric batteries on board can be reduced, and thus the costs are reduced. This is because the batteries are the most expensive component in an electric vehicle. The battery capacity determines the range of the vehicle, the charging time, the battery weight, and the second-hand value as well. With heavy freight transport vehicles, the payload could be affected significantly because of the weight reduction of the batteries [32].

However, there are some challenges related to IPT systems. There are risks of electromagnetic safety, where strong electromagnetic fields could cause harm to the biological environment. To limit the electromagnetic field, high ferrite or aluminum plane could be used for shielding. Also, a misalignment of the coils could cause high currents and have a negative impact on the magnetic field emissions. The SAE J2954 standard has defined the maximum allowed misalignment of the coils in a WPT as 0.075 m in the direction of travel, and 0.1 m in the traverse direction [33]. There is always a risk with cybersecurity, where alteration of data could cause hardware damage, denial of service, etc.

2.2.1 Resonant inductive power transmission system

A block diagram of a truck IPT system is illustrated in figure 2.5. The wireless power transfer consists of the inductive coupling between the transmitter coil under the ground and the receiver coil on the moving vehicle. The power transfer from the transmitter coil is powered by converting the grid, via power electronics converters AC/DC to DC/AC, through a compensation network. The DC/AC converter creates a high frequency (HF) field and HF current, allowing it to couple with the receiver coil. The compensation network consists of a capacitor, resistor, and coil to improve the system efficiency. The receiver coil converts the oscillating magnetic flux field created by the transmitter coil to an HF AC. Through the compensation network on the vehicle, the HF AC is rectified by the AC/DC converter, which allows the battery to be charged. The coils are added with ferromagnetic, conductive, and auxiliary materials to improve the magnetic flux distribution and reduce any harmful leakage fluxes [29].

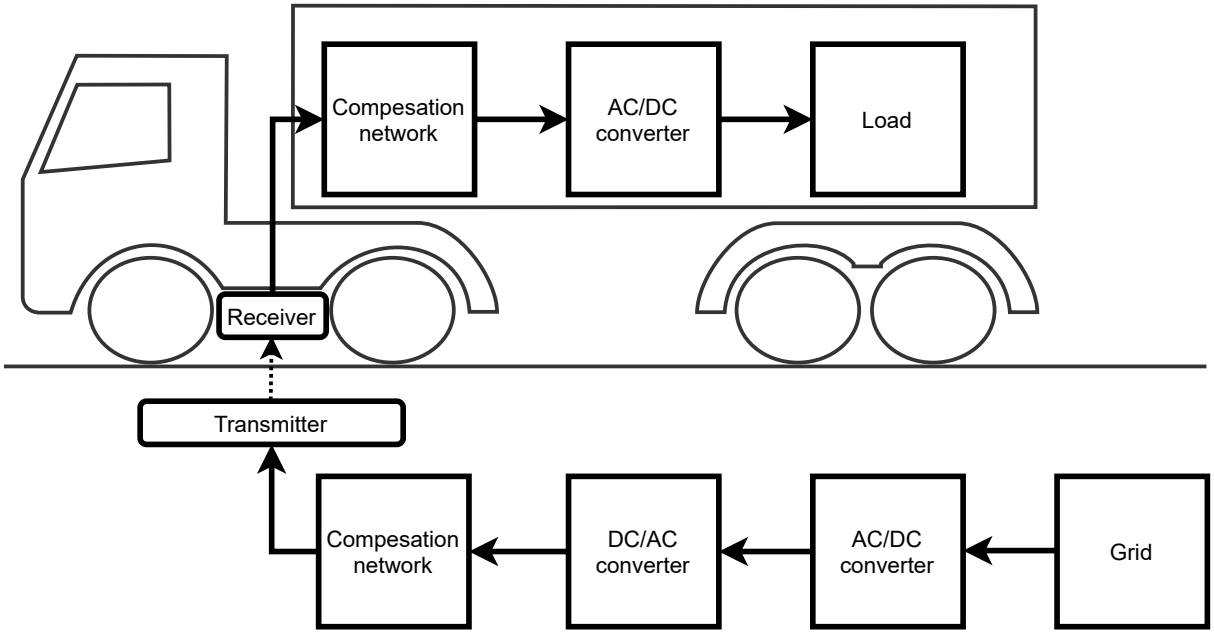


Figure 2.5: Wireless charging system

2.2.2 System topology

The physical principles of determining the functioning of IPT systems are, in general, the same as for a conventional transformer. The difference is the air gap between the receiving and transmitting coils in the IPT system, which gives a much lower coupling factor than the conventional transformer. A large air gap between the coils in a dynamic IPT system will cause a relatively low magnetic coupling of the coils. When comparing this to the conventional transformer, this leads to a low magnetic inductance and a high leakage inductance. As a result, the inductive power transfer will consume a large amount of reactive power. The source or load-side must then supply the reactive power requested to support the power transfer over the air gap. When it is supplied by a power converter,

it is required that the current rating and resulting cost of the power converters would increase correspondingly. These capacitors are usually used to supply the reactive power in the compensation network, which is designed to obtain a specific resonance frequency, according to the equivalent inductance of the receiving and transmitting coils [34] [35].

There are several different methods to design the compensation network, where the most basic and simplest method is in series or in parallel with the coils on each side. Selecting the most suitable method depends on the application of the system it is designed for as well as the power electronic converter topology [34]. A Series-Series (SS) compensated network is used at the charging system in this study, as this ensures a minimum number of passive components. This leads to minimum conduction losses and avoids unnecessary costs [35]. The topology of a SS compensated resonant topology with H-bridges for the IPT system is illustrated in figure 2.6 [35].

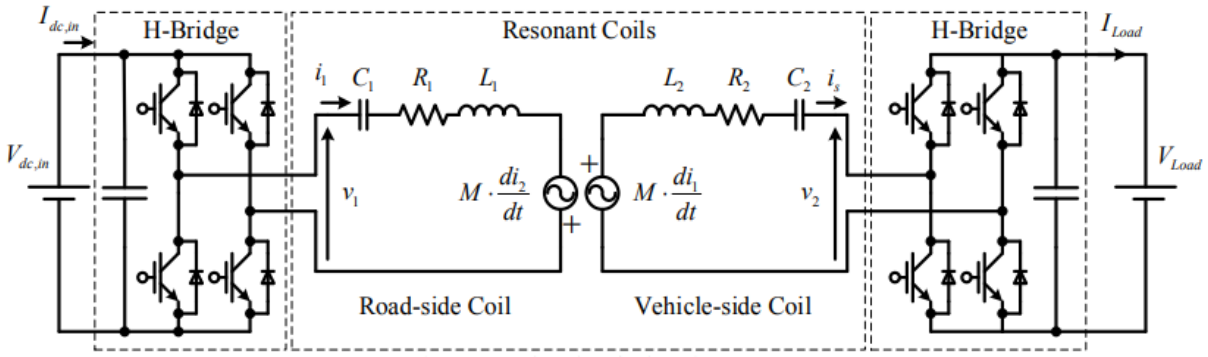


Figure 2.6: Topology of a SS compensated network. Ill: [35]

The H-bridge allows for better controllability, where the transistor gates are operated by Pulse-Width Modulation (PWM) signals. The vehicle-side H-bridge could be exchanged with a simpler application, a passive diode rectifier, but this would mean a loss of active control of the receiving side. When operating with high power, as in a full-scale solution, the H-bridge is usually designed with Si-based Insulated Gate Bipolar Transistors (IGBTs). The IGBTs are preferred for low frequency and high voltage. For a low power application as the project in this study, Metal-Oxide Semiconductors Field Effect Transistors (MOSFETs) are used. The MOSFETs have a fast switching rate, making them suitable in an application with a high switching frequency, where the operating frequency of this project is 85 kHz [36]. The operating frequency is the same as the maximum frequency set in the standard SAE J2954 for WPT for electric vehicles [37].

The resonant coils could be modeled as two coupled voltage sources. The voltage sources are proportional to the mutual inductance M, and the rate of current change in the responding coil [38]. When tuned, the LC filtering on both sides in the compensating network has a high band-pass effect on the currents. An advantage of this compensating network is that the resonant frequency is not affected by the loading conditions, and is not sensitive to the variations in the couplings between the coils. Operating close to the resonant frequency implies that the voltage and current on each side will be in phase. This will lead to close to zero-current switching of the semiconductors, thus limiting the conversions losses of the system [6].

When the system in figure 2.6 is ideally tuned, the unique resonant frequency ω_0 is defined as [35]:

$$\omega_0 = \omega_{0,1} = \frac{1}{\sqrt{L_1 C_1}} = \omega_{0,2} = \frac{1}{\sqrt{L_2 C_2}} \tag{2.1}$$

Furthermore, by normalizing the system, the mutual inductance M can be substituted with a non-dimensional magnetic coupling coefficient k and a quality factor Q, which are independent of scaling [35]:

$$k = \frac{M}{\sqrt{L_1 L_2}} \tag{2.2}$$

$$Q = \sqrt{\frac{\omega L_1}{R_1}\frac{\omega L_2}{R_2}} \tag{2.3}$$

With the voltage applied to the resonant coils [35]:

$$v_x = \frac{V_x}{V_{N,x}}\ ;\ V_{N,x} = V_{dc,x} \cdot \frac{4}{\pi}\ ;\ with\ x = 1,2 \tag{2.4}$$

And following base values for power, current, and impedance [35]:

$$P_N = \frac{V_{N,1} V_{N,2}}{\omega_0 M_{max}}\ ;\ I_{N,x} = \frac{2P_N}{V_{N,x}}\ ;\ Z_N = \frac{V_{N,x}}{I_{N,x}}\ ;\ with\ x = 1,2 \tag{2.5}$$

And resistance [35]:

$$r_{12} = \sqrt{\frac{R_1}{Z_{N,1}}\frac{R_2}{Z_{N,2}}}\ ;\ r_x = \sqrt{\frac{R_x}{Z_{N,x}}}\ ;\ with\ x = 1,2 \tag{2.6}$$

Subsequently, the normalized currents in resonance are expressed as [35]:

$$i_1 = \frac{\frac{v_1}{r_1} + \frac{v_2}{r_{12}} \cdot kQ}{1 + (kQ)^2};\ i_2 = \frac{\frac{v_1}{r_{12}} \cdot kQ - \frac{v_2}{r_2}}{1 + (kQ)^2} \tag{2.7}$$

The equation in (2.7) highlights the varying effect the coupling coefficient k (2.2) have on the current levels in i_1 and i_2. The current levels are normalized by their rated values at the maximum overlap between the coils, at the road-side coil and vehicle-side coil, respectively [38]. Under assumptions of small losses ($kQ >> 1$), both currents tend to be 1.0 pu (per-unit) when rated power is transferred at rated coupling [35]. Figure 2.7 shows the sensitivity of the coupling factor k when $v_2 = 0$ and $v_1 = 1\ pu$, thus the maximum short-circuit currents can be evaluated (Q is assumed constant). The red and blue curves are the ideal tuned cases, and the purple and green curves are slightly detuned cases.

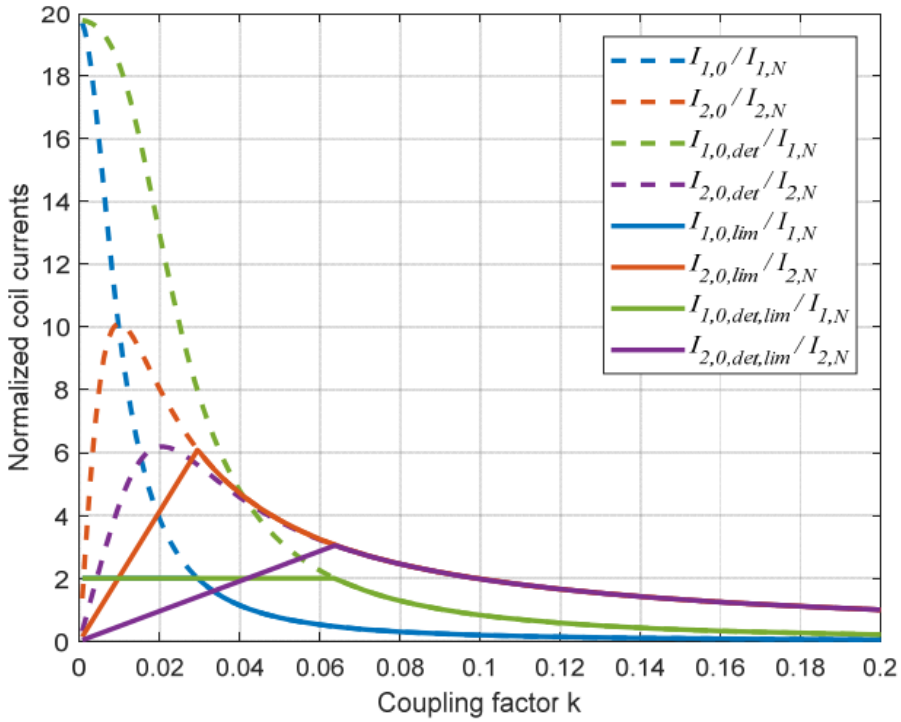


Figure 2.7: Normalized coil currents as a function of coupling factor k for the vehicle-side coil. Ill: [35]

With full voltage at the road-side coil, and short-circuited at the vehicle-side coil, the currents will grow several times higher than the rated current, when the coupling factor k has a small value. The solid red and blue curves show the currents when the road-side coil is limited by 2.0 pu in the power converter. Still, in the worst-case, the I_2 currents in the vehicle-side coil is raised to 6.0 pu [35]. The worst-case could be drastically reduced by detuning the resonance frequencies of the coils according to a factor x_c: [35]

$$x_c = \frac{C_1 L_1}{C_2 L_2} > 1 \tag{2.8}$$

The purple and green curves are a solution where $x_c = 1.05$, where the worst-case has been reduced to less than 3.0 pu, and could even be lower with more detuning. The solid purple and green curves show that the current limitations could be active for a broader range of coupling conditions [35]. This indicates that WPT is very sensitive to the misalignment of the receiving and the transmitting coils, and there are different ways to improve the lateral tolerance. Two potential methods are parameter adjustment for the optimum state of the WPT system, and optimizing the coil structure to smooth the decay of the coil coupling [39]. Both of these methods are assumed optimal and are beyond the scope of this study. However, the physical alignment of the coils is the crucial part of this study, where the coupling factor k (2.2) will be attempted to reach the max coupling factor for an effective and reliable system. The transmitting coils can also be tested where they should be triggered, to keep the power losses due to idling time as close to zero as possible and receive a high efficiency.

In a study [35], a Finite Element method analysis of the coupling factor k has been achieved, this is illustrated in figure 2.8. The figure shows the coupling factor of a single road-side coil with a vehicle-side coil at perfect alignment. The size of the road-side coil is 1.0 pu, and in this case, a road-side coil of 570 x 100 mm and a vehicle-side coil of 100 x 100 mm. The maximum coupling was achieved at 0.12 pu and 0.88 pu of the road-side coil with a coupling factor of 0.18, but it has a coupling factor of 0.16 at the middle of the coil.

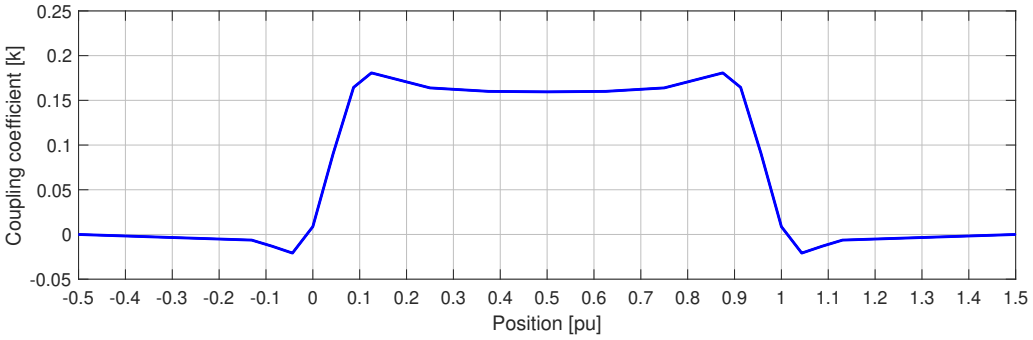


Figure 2.8: Coupling characteristics of perfect alignment, as a function of position [35]

2.2.3 Wireless power transfer systems operating today

There are a variety of demonstrations and developments from companies, research organizations, and governmental institutions of dynamic and static IPT systems operating today. A few of these will be presented. Companies, such as Bombardier in Germany, has developed IPT systems in Europe. The main application is stationary opportunity charging for buses and trams. In at least four cities in Germany and Sweden, buses are operating in-route with static IPT systems. They have also demonstrated this technology with dynamic charging on an 800 m long test track in Germany. The operating frequency is 20 kHz in a three-phase system to obtain a high power density without exceeding the EMF guidelines. For the vehicles, it is rated for transferring at 200 kW. At minimal coupling distance, the trams could reach a power transfer of $\sim$ 250 kW [40].

Since 2009, the Korea Advanced Institute of Science and Technology (KAIST) has developed various generations of IPT systems for on-line electric vehicle systems. The generations have demonstrated different designs and solutions for the road-side installation. At least six buses are operating at generation 3+G with 15 kW pick-up power over an air gap of 20 cm at 83 percent efficiency. At the same generation, at least three light rail trans have a pick-up power of 15 kW with an air gap of 12 cm at 74 percent efficiency. In their fourth generation, with an air gap of 20 cm, they have reached 25 kW of pick-up power and an efficiency of 80 percent, operating at 20 kHz. A full review of the generations is available in reports, such as [40]. The Oak Ridge National Laboratory IPT systems for a road-powered electric vehicle (RPEV), operating at 20 kHz, reached a power transfer of 2.2 kW with an efficiency of 74 percent. This was completed with a circular transmitter and receiver coils, and the 72 V lead-acid battery limited the power transfer [31].

In the last decade, various cities in the world are in the process of testing and implementing battery-electric buses for public transportation. In Madrid, a solution was found suitable for everyday use with wireless opportunity charging. This was done by charging the buses at a wireless power transfer depot overnight and with static inductive power transfer systems while the buses operated in traffic. This opportunity charging system consists of two terminals for charging at the bus route, where 5-10 minutes charging of a small battery of 60 kWh could sustain an entire day of operation [41][42].

An EU project, FABRIC, is under development, but limited technical information is currently available. They target different types of vehicles, such as passenger cars, light weight duty vehicles, heavy vehicles and buses. The testing sites are in Sweden, Italy and France. In Sweden, this is a joint operation with Volvo, with a conductive electric road. Rails in the ground works like an upside pantograph. It has a 475 m track, where 275 m are the electrified part working at 750 V DC at speed up to 100 km/h. In Italy, both dynamic and static inductive charging is being tested. A 260 m inductive test track is available to power between one and three vehicles at 20 kW power transfer to each vehicle. In France, three different tracks are available for testing. The tracks utilize dynamic inductive charging,

where up to two vehicles can be charging simultaneously. Various power levels have been tested, with a greater power flow of 20 kW at the highest speed [43].

2.3 Combination of autonomous and WPT systems

As presented in the previous section, it is crucial to keep the lateral offset of the vehicle-side and road-side coil to a minimum for an efficient charging cycle. This is where the autonomous control systems and wireless charging seem like a perfect match, where it allows for an easily and fully automated charging of the batteries in the electric vehicle [6]. If a vehicle is driven manually, it would be challenging to keep a perfect alignment of the coils, where each centimeter could impact the coupling factor. A solution for this problem is the autonomous driving functions, where the vehicle can be kept at a precise position, with the help of the sensors, down to the centimeter.

As described in the previous section, extensive research and development of WPT systems have been done in the last two decades. However, these solutions are mainly being commercialized for electric vehicles that are intended as driver-operated vehicles. This technology could be utilized for autonomous operated electric vehicles, and could make them cover their own energy demand making them fully self-sustained vehicles. This implies that the vehicle must have enough energy to drive to the nearest available wireless charging point, or could use its idling time to charge the batteries. Such a solution could be utilized for autonomous vehicles for private use, in the transport sector, and in the public transport system [6]. This type of solution could also be applied for other means of transport, such as marine applications. There are several vessels that are being tested with inductive wireless power transfer, where the vessels use the docking time for charging the batteries. Automated operations could better the utilization process with docking time for charging the batteries [3].

Chapter 3

System structure

The main object of this project is the 1:14 scale electric truck model from SINTEF, where it has been used for trials and demonstrations of inductive power transfer, but only controlled manually by an R/C radio transmitter controller. In 2019, a student at his master's thesis [44], developed three autonomous driving systems. In 2020, a new student at a master's thesis [38] optimized the transfer efficiency for the charging system, by implementing a Bluetooth connection with the charging station and an autonomous driving system. This chapter will describe the hardware and software used in this study, and the new components implemented. The truck at the charging station is shown in figure 3.1.

Figure 3.1: Truck model

3.1 Hardware of the truck model

A description of the hardware components installed at the truck model follows in this section, and how they are used to provide the autonomous driving systems of the truck model will also be provided.

3.1.1 Embedded computing device

An embedded system is a dedicated computing system that often serves specific tasks within a real-time system, consisting of processors, memory, storage, and I/O (Input/Output) peripheral interfaces. The main computing unit in this project is the embedded AI computer, Nvidia Jetson TX2, which is a fast, power-efficient computing device with a power consumption of about 7.5 W, where 15 W is the maximum. This makes it very efficient in a battery-powered application such as this truck model system. It is pre-flashed with a Linux development environment, and contains the software for the operation of the autonomous driving system. The technical specifications of the Nvidia Jetson TX2 are [45]:

- 256-core NVIDIA Pascal Architecture GPU
- Dual-core NVIDIA Denver 2 64-bit CPU and Quad-Core ARM A57 Complex
- 8 GB L128 bit DDR4 Memory
- 32 GB eMMC 5.1 Flash Storage
- 802.11ac Wi-Fi and Bluetooth-enabled Devices
- 10/100/1000BASE-T Ethernet

The Nvidia Jetson TX2 features two built-in CAN (Controller Area Network) controllers, and a Texas Instruments SN65HVD230D CAN transceiver is connected to the controller. The transceiver is the interface between the physical bus and the CAN protocol controller. This CAN bus is utilized for communication with the wireless power transfer on the vehicle-side of the system [44].

Figure 3.2: Jetson TX2 with CAN transceiver

3.1.2 Microcontroller development board

A microcontroller is often implemented in an embedded system to serve specific operations. It is an integrated circuit, which consists of a processor, memory, and I/O peripherals. The PJRC Teensy 3.2 is integrated into this project, which is a small and powerful microcontroller. It is used to connect and interface the electronic components on the truck model. The components connected are the servos, the electronic speed controller (ESC), the hall effect sensor, and the IMU. The technical specifications of the Teensy 3.2 microcontroller are [46]:

- ARM Cortex-M4 at 72 MHz
- 256K Flash, 64K RAM, 2K EEPROM
- USB device 12 Mbit/sec
- Logic level 3.3 V

It is compatible with the Arduino platform, which makes it operable with the ROS (Robot Operating System) platform for direct control and interaction via a serial interface.

3.1.3 Camera

A camera achieves a visual recognition of an environment. For the software analysis, the camera uses visual data from the optics in the lens. This could be lane and object detection, distance measurement, road signs, etc. A camera is good at capturing texture and color, but could be sensitive to weather conditions, such as low light conditions. The truck model is already mounted with a Logitech C922 Pro camera, a budget rolling-shutter camera preferred for a slow-moving vehicle. The camera has a maximum resolution of 1080 x 720 p, a 60 Hz refresh rate, and a 78 degree field of view. If a camera is to be used on a fast-moving vehicle, a global shutter is preferred but is usually more expensive. The global shutter updates the image as a whole, while the rolling shutter updates the image line-by-line [44]. In this study, the rolling-shutter camera is sufficient due to the vehicle's low speed, where the truck model's test speed varies between 0 m/s and 1 m/s. A new Logitech C922 Pro camera is mounted in this study to achieve an extended view of the lanes. The images from the new camera are merged with the images from the camera already mounted. This implies that the merged image from the cameras can detect the previous blind spot in front of the vehicle, and gain a more robust autonomous driving functionality.

3.1.4 LiDAR

The LiDAR is a sensor used to determine the range/variable distance of an object with an emitted light-pulse, by measuring the time-of-flight of the reflected light. LiDAR is becoming one of the standard sensors for autonomous vehicles, where mapping the environment is essential. In this study, a low-cost 2D LiDAR - Slamtec RPLIDAR A2M8 is used, mainly for odometry and obstacle detection. A 2D LiDAR is sufficient because the vehicle will only be moved around in a horizontal direction during the testing. It has a ranging distance of 0.15 - 12 m, a scan rate of 5-15 Hz, and an angular resolution of 0.45 - 1.35 degrees [44].

3.1.5 IMU

An IMU uses a combination of gyroscopes, accelerometers, and magnetometers to measure orientation, linear velocity, and angular rate in 3D space. In this project, a Bosch BNO080 IMU is used. 9 DoF (Degrees of Freedom) is measured (gyroscope, accelerometer, and magnetometer) and are is to determine the odometry of the truck model, where the position of the truck model is logged [44].

3.1.6 Hall effect sensor

A hall effect sensor, type 3144, has been implemented in a pre-study for this study for speed and distance traveled measurement [47]. It is mounted right above an adapter from the propulsion motor to the driveshaft. The sensor is triggered by magnets, where there are four magnets glued evenly around the adapter. This is to reach a higher resolution of the speed measurement at one rotation with four magnets, rather than one rotation with one magnet. The adapter is 10 mm in diameter, and it has two deep notches on each side. Thus, four magnets provides the maximum resolution one could archive when they are evenly distributed around the circumference. An illustration is showed in figure 3.3 with a picture of the adapter where the magnets are glued. One rotation of the driveshaft corresponds to 100 mm of movement for the truck model, and with four magnets, this implies a reading from the sensor for each 25 mm driven.

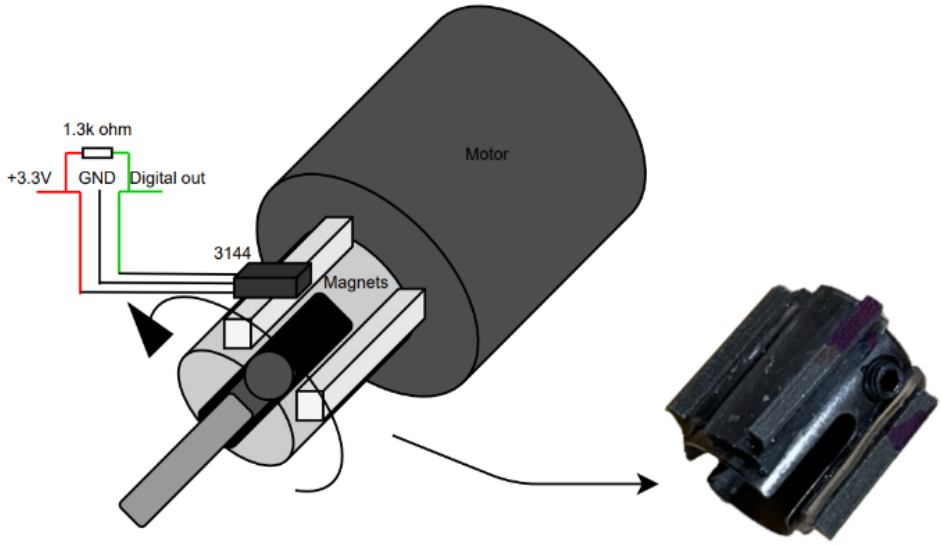


Figure 3.3: Hall effect sensor

The digital out is connected to the microcontroller, with a 1.3 kΩ pull-up resistor. The digital signal goes low when a magnet is close and is registered with interrupts at the microcontroller. The sensor is used both for speed and distance driven measurement.

3.1.7 Batteries

The truck model is powered by two batteries, one for the propulsion and one for the control electronics. For the propulsion, the truck has a 2-cell 7.4V LiPo 5500mAh battery, and this is the battery charged when the truck model moves over the road-side coils. A 4-cell 12.7V LiFePO4 7500mAh battery is mounted for the control electronics, which has to be charged manually. The control electronics battery does not have a voltage sensor, as the propulsion battery has when it is connected to the Xilinx Zynq board at the charging system. Thus, a small voltmeter with a LED display has been mounted in this study, as showed in figure 3.4, for a better overview of the power electronics battery. If the battery is fully discharged, the cells could take damage.

Figure 3.4: Voltmeter

3.1.8 Joystick

A Logitech F710 joystick is used to operate the truck model and is shown in figure 3.5. The joystick can switch between autonomous and manual driving modes. At autonomous driving mode, the speed could either be set manually with two buttons, or it can follow the code to adapt the speed according to a track. In manual mode, one could control the truck model with the two sticks, and the arrows to change gears. One button can activate the truck model, and another button is used as a dead-switch lock to power off the truck model. A full user manual for the joystick is available in appendix C.

Figure 3.5: Logitech joystick

3.2 Hardware of the charging system

The charging system was designed and developed by SINTEF Energy Research, with a scaled 1:14 ratio specification. The charging system consists of a road-side part and a vehicle-side part. The road-side control system part is mounted in a box and is located next to the road-side coils for their operation. The vehicle-side control system part is located inside the trailer of the truck model to control the vehicle-side coil. At the control system parts, two Xilinx Zynq development boards, one at each side, controls the charging system's current flow. The boards contain Field Programmable Gate Array (FPGA) chips, which are utilized for generating the High-Frequency (HF) PWM signals to control the switching of the MOSFETs [38]. The development boards consist of various I/O signal parameters, and they are used to observe the states of the charging system in this study. The charging system consists either of a dynamic charging system, or the alternative opportunity charging system implemented in this study.

3.2.1 Dynamic charging system

The coils of the dynamic charging system are based on an assumed full-scale dynamic charging system. These specifications were based on general expected requirements of a full-scale size of a heavy vehicle and its power consumption. The specifications are presented in table 3.1 [6]:

Table 3.1: Assumed specifications [6]

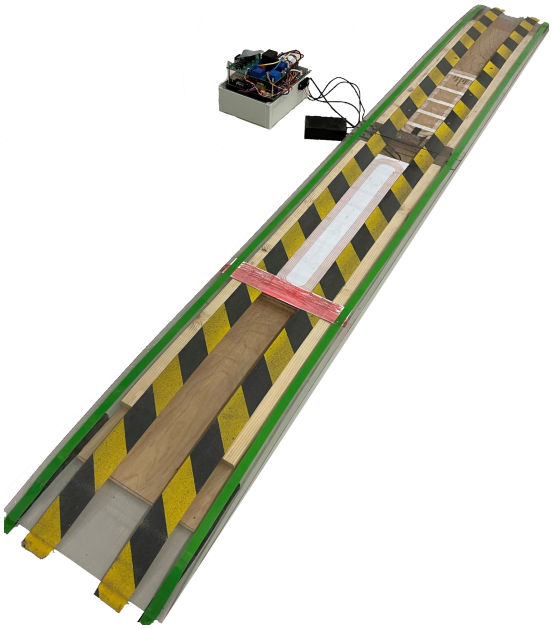
Full-scale system	
Nominal power, P_0	200 kW
On-board coil max planar size	1.4 m by 1.4 m
Road coil max width	1.4 m
Road coil length	5.0 - 15.0 m
Coil-to-coil min clearance	0.3 m

The dynamic charging system consists of two different road-side coils, mounted in a 3D-printed housing with plexiglas and grip-tape to make the surface drivable. The coils have different designs to illustrate the flexibility of the design. This shows the interoperability of the design, but they have to be tuned to the same resonant frequency. The first coil was designed with focus on high efficiency, with more copper for the winding and high-grade ferrites. The second coil was designed for low cost option, which resulted in lower power efficiency and a lower quality factor Q [6]. The quality factor was introduced in equation (2.3). The specifications of the small-scale dynamic system is presented in table 3.2 [6]:

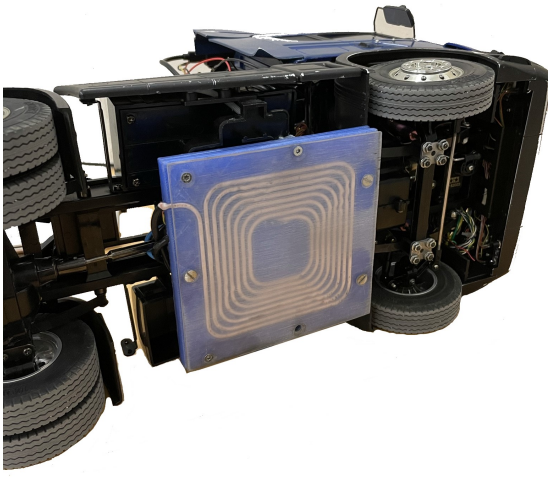
Table 3.2: Specifications of present system [6]

Small-scale dynamic system	
Nominal power, P_0	75 W
Nominal I/O voltages, $V_{dc,in}, V_{dc,out}$	12.0 V, 7.4 V
Nominal operating frequency	75 kHz
Vehicle-side coil	
Planar dimensions	100 mm by 100 mm
Self-inductance (above road-side coil), L_2	7.9 μH
Quality factor Q (at 75 kHz)	163
Road-side coil 1	
Planar dimensions	570 mm by 100 mm
Equivalent length in full-scale	8.0 m
Self-inductance (with no pick-up), L_1	37.0 μH
Quality factor Q (at 75 kHz)	293
Road-side coil 2	
Planar dimensions	440 mm by 100 mm
Equivalent length in full-scale	6.2 m
Self-inductance (with no pick-up), L_1	31.0 μH
Quality factor Q (at 75 kHz)	143
Coupling conditions	
Air-gap distance	22 mm
Coupling factor, k (centered, max coupling)	0.16, 0.18

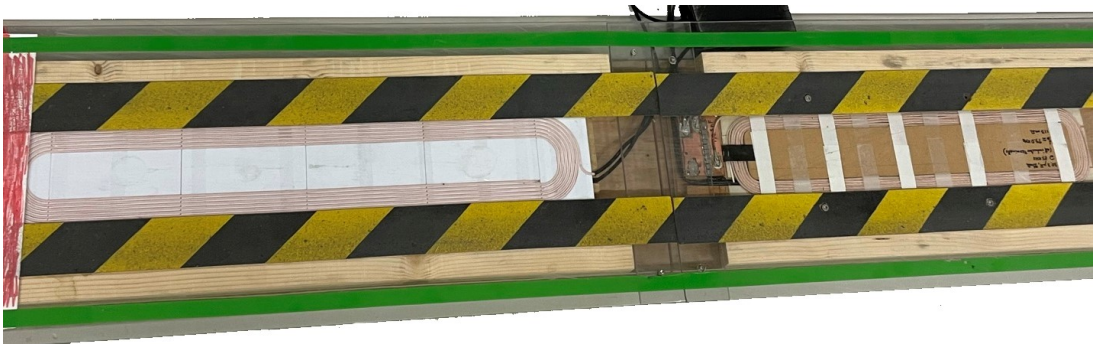
The charging system is shown in figure 3.6, with the charging station in 3.6a, the truck models coil in 3.6b, and a closer look at the coils at the charging station in 3.6c. In 3.6c the left coil is the high-efficiency road-side coil 1, and the right coil is the low cost road-side coil 2.



(a) Charging station



(b) Charging coil truck



(c) Charging coils

Figure 3.6: Charging system

3.2.2 Opportunity charging system

The road-side coil 2 was exchanged with a square road-side coil for the tests with a static opportunity charging system. The vehicle-side coil is the same as used within the dynamic charging system, but the road-side charging system now consists of a road-side coil with the same specification as the vehicle-side coil. The square road-side coil is designed with a high-quality litz wire and more copper than the rectangular road-side coils at the dynamic charging system. The specifications of this system are listed in table 3.3.

Table 3.3: Small-scale static charging

Small-scale system square coil	
Nominal power, P_0	50 W
Nominal I/O voltages, $V_{dc,in}, V_{dc,out}$	10.0 V, 7.4 V
Nominal operating frequency	75 kHz
Vehicle-side coil	
Planar dimensions	100 mm by 100 mm
Self-inductance (above road-side coil), L_2	7.9 μH
Quality factor Q (at 75 kHz)	163
Road-side coil	
Planar dimensions	100 mm by 100 mm
Equivalent length in full-scale	1.4 m
Self-inductance (with no pick-up), L_1	7.9 μH
Quality factor Q (at 75 kHz)	163

The square road-side coil is shown in figure 3.7, where it has replaced the rectangular road-side coil 2.

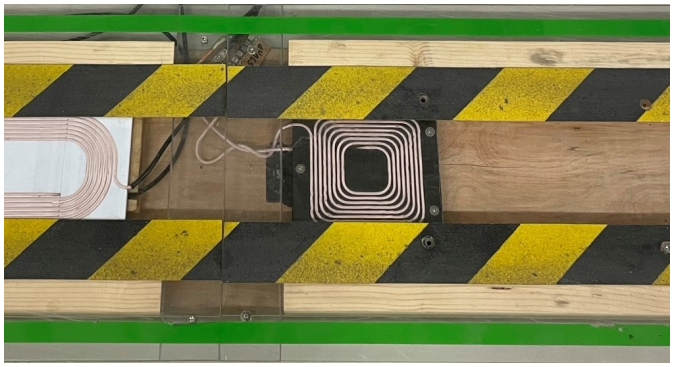


Figure 3.7: Square road-side coil

3.3 Hardware overview

The hardware structure is shown in figure 3.8. The Nvidia Jetson TX2 connects the cameras, LiDAR, joystick, and the microcontroller Teensy 3.2, where they are powered and communicate via USB. The joystick communicate via a 2.4 GHz radio frequency from an included USB devise, and is powered with AA batteries. The wireless charging system communicates with the Jetson TX2 with a Bluetooth RFCOMM (radio frequency communication) protocol with the road-side of the charging system, and a CAN bus for the vehicle-side. The laptop is connected through WiFi, with a Secure shell (SSH) protocol for activating/deactivating the truck model and monitoring the states. The Teensy 3.2 microcontroller connects the hall effect sensor, which is interfaced with interrupts. At the microcontroller, PWM is used to operate the servos and ESC. The steering servo is interfaced with an analog feedback signal, implemented in this study, and is further described in 4.1. Furthermore, the IMU is interfaced with the microcontroller with the I2C (Inter-Integrated Circuit) serial communication protocol. The LED indicator strip is activated with GPIO (general-purpose input/output).

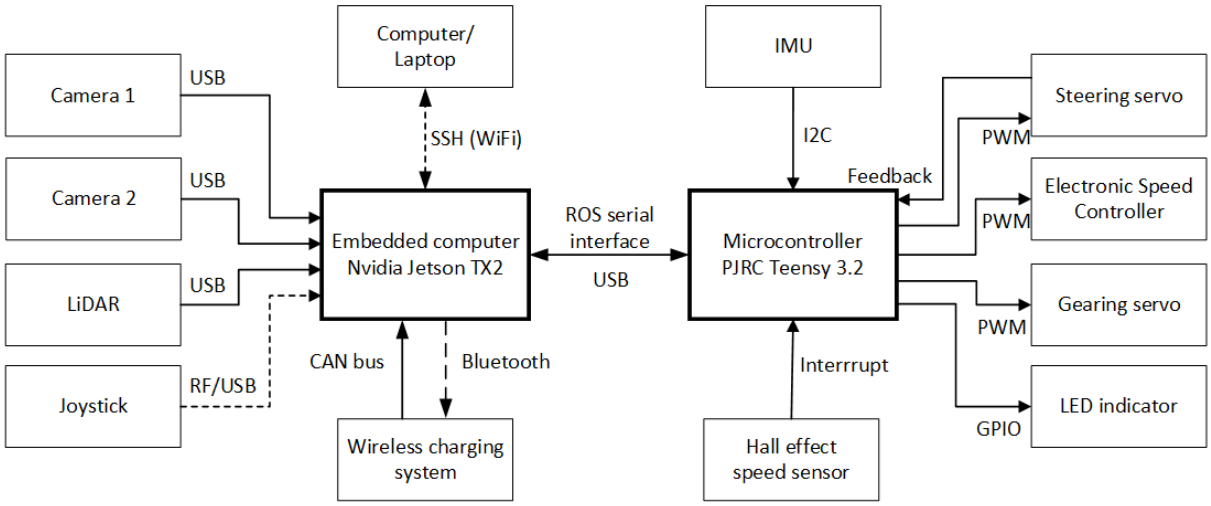


Figure 3.8: Hardware overview

3.4 Software

The foundation of the software in this study builds on the work done by the students at the master's thesis from 2019 [44] and 2020 [38]. This section describes how the software has been implemented and furthered in this project. A description of the main libraries used will be provided, as well as protocols for a real-size solution are explored.

3.4.1 Robot Operating System

The Robot Operating System (ROS) is a system that provides libraries and tools to create robot applications. In this study, the ROS Kinetic is used for communication within the system. It is an open-source and is meta-operating, which means that the system could run on top of the Linux operating system, such as Ubuntu, and allow nodes to communicate with each other. The ROS system is a peer-to-peer (P2P) network and provides low-level device control, hardware abstraction, implementation of commonly-used functionality, package management, and message-passing between processes. The P2P is a way to organize the resources in a network, which interconnects the peers (nodes) and shares resources between them, unlike a client-server model, where the clients have to request the resources from a centralized server [48].

The main ROS elements used in this project are the nodes, topics, messages, and a master. In ROS, the nodes are a process that performs computations, and the nodes are combined in a graph. A graph is a data structure that combines the nodes with the help of edges. This means that the nodes communicate with each other using streaming topics as their edges. The nodes can subscribe and publish multiple topics for the inter-communications with other nodes. Topics can be described as messages passing between the nodes, and there can be numerous publishers and subscribers of a topic. An illustration of this process is shown in figure 3.9. The messages can have a variety of formats, such as int, float, sensor, etc. An advantage of the ROS framework is that it is easy to implement in many well-known programming languages, such as Python, C++, C, which are used in this project. A ROS master is responsible for the name registration and lookup in the graph, much like a DNS (Domain Name Server) [48].

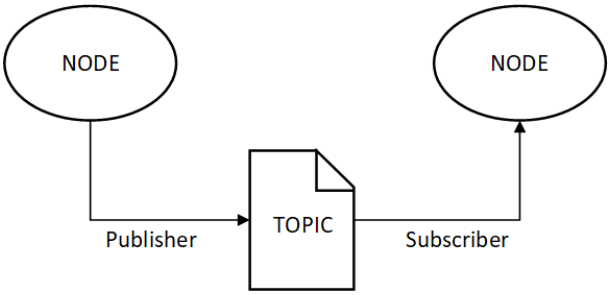


Figure 3.9: ROS basic concept [48]

3.4.2 Computer vision library

OpenCV (Open Source Computer Vision Library) is an open-source computer vision library, and contains hundreds of computer vision algorithms for real-time computer vision. In this project, it is mainly used for image manipulation, such as converting colors, merging, warping, adaptive threshold, etc. These functions will be described when they are used in this study. The library has support for Python, C++, Java, and Matlab, making it suitable in this project where Python is used with the computer vision [49].

3.4.3 Software overview

The foundation of the software used in this project is inherited from previous work. It was decided not to rewrite the entire software code but rather expand, alter and rewrite parts of the code where it was found necessary to further this study. This made it possible not to spend time writing new code that was already implemented and working, but rather to spend time further developing and optimizing parts of the code that gave a more satisfactory result. The software overview is shown in figure 3.10, where the ROS nodes are in the blue boxes, the support scripts are in green boxes, the road-side charging station is in the yellow box, and laptop is in the orange box. The three central nodes are marked with wider boxes, namely the Ackermann drive, Car cmd, and Teensy microcontroller node. The backslash indicates a ROS topic, the plus sign a function, and the minus sign marks a wireless communication.

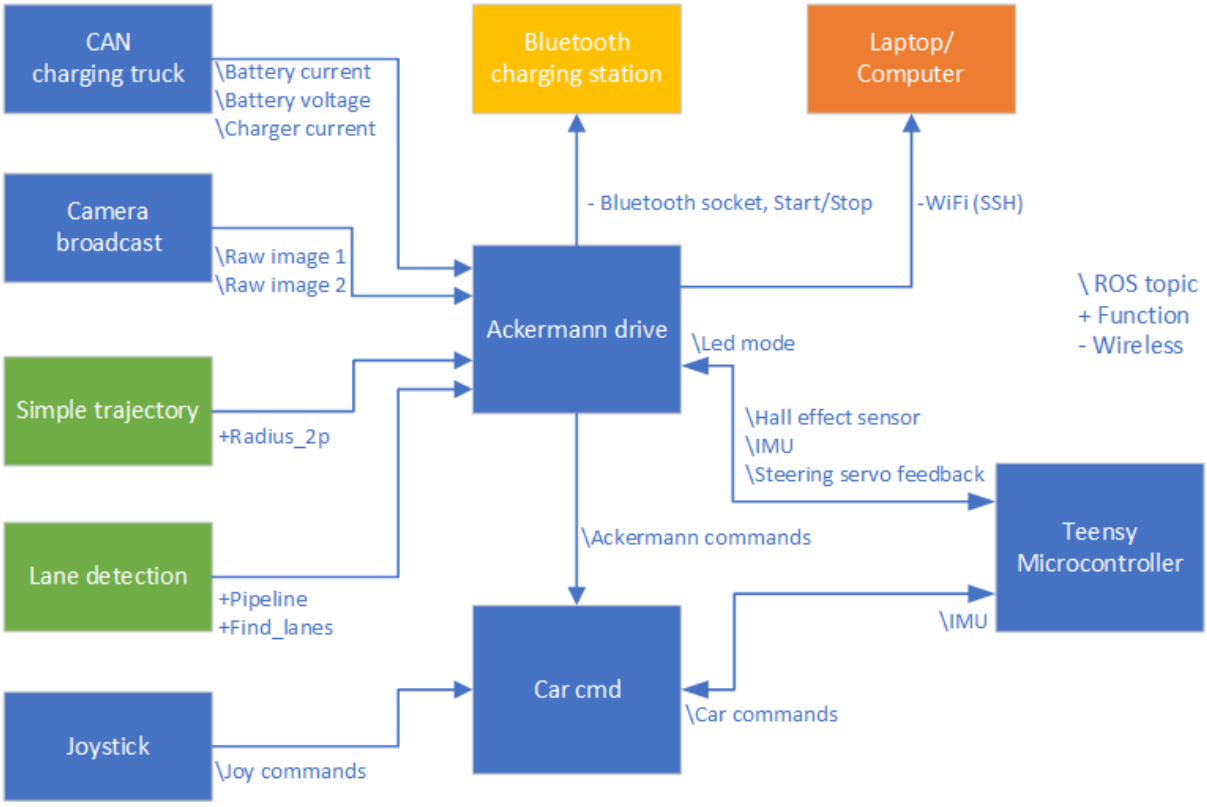


Figure 3.10: Software overview

The following is a short description of the ROS nodes and support scripts. It is an overall description, where the specific objects implemented in this study are described in more detail later in the thesis:

- **CAN charging truck**
  The CAN charging truck node is where the CAN bus interfaces to ROS, from the Nvidia Jetson TX2 to the Xilinx Zynq board on the truck model. The ROS node publishes the topics; battery current, battery voltage, and charging current. This implies that the energy usage of the truck model and charging current can be monitored and plotted in real-time. In Python, the python-can library and SocketCan interface modules are used for the support to enable the communication [44]. The CAN bus is a message-based protocol that allows devices to communicate without the need for a host computer. A block diagram of the CAN system in this project is shown in figure 3.11. This node has not been altered and is inherited from previous work [44].

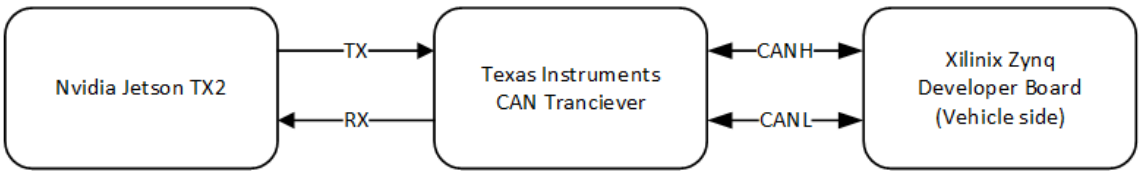


Figure 3.11: CAN block diagram

- **Camera broadcast**
  The Camera broadcast node is where the raw image from the two cameras is captured and published at a rate of 30 frames per second. The images are captured with the OpenCV library and are scaled down to a resolution of 640 x 320 p. This implies that the images have a less computational cost when manipulating the images for the path tracking, but they still gave

the desired result in this project. This node has been extended to be able to capture the new images from the newly installed camera and publish the raw images as a topic.

- **Simple trajectory**
  Simple trajectory is a support script with the functions used in the path tracking methods. The new Radius2p function, written in this study and is described in detail in section 4.3.2, is used for the current path tracking strategy. Different functions were tested for path tracking, written in this study and others inherited from previous work [38], but those are currently not in use. The node as available in appendix E.4.

- **Lane detection**
  Lane detection is another support script with functions for image processing, where different functions merge and process the images. Some of these function searches for the lanes with computer vision, depending on which part of the track they are in use, and they are named Pipeline(2-5). These functions are rewritten from previous work, where they have been altered to receive two images instead of one image. New functions are written to warp the images, etc. The find_lanes function extracts the lanes from the processed image, and locates the points for the path tracking. This function is altered and extended to comprehend the merged image and the curvature when only one lane is detected. These processes are described in detail in section 4.3.1. The node is available in appendix E.3.

- **Joystick**
  The Joystick node is the node for publishing the joystick commands. This is an open-source ROS driver for Linux joysticks, and is inherited from previous work [44].

- **Bluetooth charging station**
  The Bluetooth charging station is the communication with the road-side Xilinx Zynq development board on the charging station. To establish the Bluetooth communication, the RFCOMM protocol is used. The RFCOMM is a simple transport protocol, which supports up to 60 simultaneous connections between two Bluetooth devices. The Bluetooth link is a communication segment with a direct connection from one device to another, creating a reliable data stream, similar to the TCP (Transmission Control Protocol) [50]. There are currently three connections used in this project; two to activate each road-side coil at the charging station, and one to deactivate the road-side coils. The implementation of Bluetooth communication was one of the main works of a master's thesis in 2020 [38], and is used as-is for its purpose in this project.

- **Ackermann drive**
  The Ackermann drive is the main node in this study and is central for controlling the functions of the truck model. This node is inherited from previous work, but has been completely rewritten and extended in this study. This node handles the communication with the charging station, where a new function for a precise activation/deactivation of the road-side coil is implemented. It controls the raw camera images, with continuous image processing, etc. The odometry for logging the truck model positioning is processed here, where the position of the truck model is logged with the IMU and the LiDAR. A function in the node logs at the same time the power from the ROS topics from the CAN charging truck node. One of the objectives for the node is to control the speed and the steering angle for the truck model. A PID (Proportional Integral Derivative) controller is used to control the speed, which is implemented in a pre-study for this study [47], and a new PID controller is utilized in this study for the steering angle. The computer vision based path tracking controller is located here, and processes the ROS topic Ackermann commands to operate the truck model. The new static opportunity charging algorithm is implemented in the node, where a variable can be set to either operate with dynamic or static wireless charging. For a visual overview of the truck model states, the node publishes selected values, which can be viewed at the laptop via the WiFi (SSH) connection.

The LED strip on top of the truck model can be set to bright or blinking, and is used to view which settings the truck model operates with. The node is available in appendix E.1.

- **Car cmd**
  The Car cmd node is used to control the truck model. The different settings can be set with the joystick, such as a driving modes for autonomous or manual control of the truck model. The node publishes the topics to the Teensy microcontroller, which controls the steering servo, the electronic speed controller, and the gearing servo. The topics are scaled to a Twist message format, which means that the message is either a linear vector for the ESC or an angular vector for the servo, making them readable for the microcontroller. The node also handles the obstacle detection from the LiDAR. The node has been rewritten for these functions. The node is available in appendix E.2.

- **Teensy microcontroller**
  The Teensy microcontroller node receives the Car command topics from the Car cmd node, which are scaled to PWM signals to control the steering servo, the electronic speed controller, and the gearing servo. The node sets the mode for a LED strip at the top of the cab from the Led mode topic with a logic output signal. The microcontroller receives signals from the hall effect sensor and steering servo and publishes them as topics. The hall effect signals are used as speed feedback of the truck model. The signals from the steering servo are used as steering angle feedback, and are written as a part of this study. The IMU is interfaced with the I2C protocol, a multi-user serial data bus. The node publishes the odometry values from the IMU as a topic. The node is available in appendix D.

- **Laptop/computer**
  The laptop is used to monitor the values in real-time of the truck model and start up/shut down of the truck model. A text editor is used to alter and update the code within the Nvidia Jetson TX2. This is done using the SSH (Secure Shell) protocol, a protocol for secure login from one computer to another computer. It connects to the computer with the use of a client-server architecture via WiFi. The data is transferred as encrypted, providing integrity and confidentiality of the data if they are transferred over an unsecured network [51]. This function is updated in this project, where it shows: driving mode, battery voltage, speed, distance traveled, and the reference speed. The protocol is accessed through the command-line in Linux at the host laptop. This is shown in figure 3.12. In the figure, the states indicate state 0, which is the regular track. The reference speed was decreased from 0.50 m/s to 0.40 m/s, where one can see the speed reached its steady-state. The distance is logged for each 25 mm.

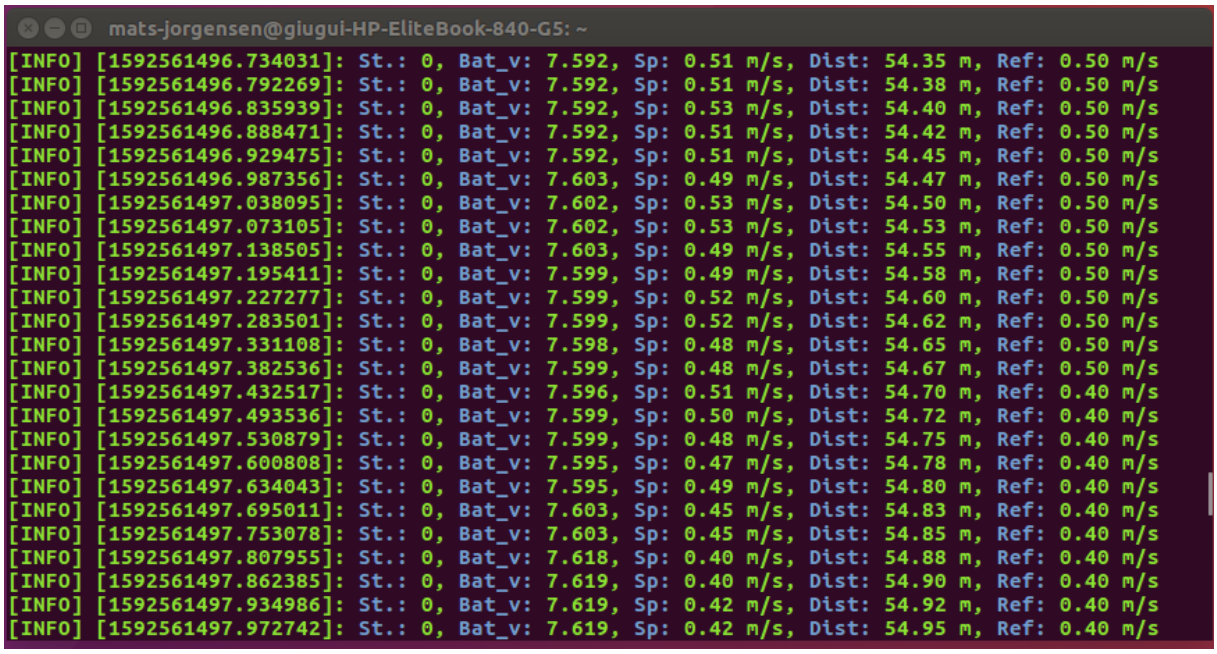


Figure 3.12: Command line monitoring

3.4.4 Software for the charging system

The software for the charging system is provided by SINTEF, where there is independent control of the road-side part and the vehicle-side part. The topology of the IPT system is shown in figure 2.6 in the previous chapter. At the road-side, the current is controlled and regulated with a PI controller at the Xilinx Zynq development board at the charging station. The controller provides a voltage reference for the phase-shift modulation to the H-bridge rectifier. Since there is an actively controlled H-bridge at the vehicle-side, the system must be synchronized to the same resonant current. A Phase Locked Loop (PLL) is utilized to synchronize the resonant current for the H-bridge modulation. The same principle is used with a PI controller to provide a voltage reference for the phase-shift modulation of the H-bridge at the Xilinx Zynq development board at the vehicle-side. This allows for control of the power received by the load at the truck model. The vehicle-side coil is short-circuited at idle operation, and the H-bridge delivers zero voltage at this state. When there is a non-negligible coupling with an energized road-side coil, the current will start to flow. The load current control is activated when the PLL is synchronized to the phase angle and frequency of the resonant current. If there is a low coupling or the road-side converter is de-energized, the PLL will lose synchronization, and the load current control will be deactivated [35].

3.4.5 Protocols for a real-life solution

Protocols for a real-scale are investigated, where the Internet of Things (IoT) 4.0 protocols Message Queuing Telemetry Transport (MQTT) and Data Distribution Service (DDS) could be interesting to explore. Both protocols use a publish/subscribe pattern with nodes and topics for message passing, much like the ROS system described in section 3.4.1. A simplified principle of IoT is that computing devices, sensors, software, etc., have the ability to exchange data with other devices and systems over the internet, without any human interaction.

The MQTT is a lightweight publish/subscribe message transport designed, and it is ideal for connecting devices with low bandwidth and small code footprint. It usually runs over the TCP/IP (Transmission Control Protocol/Internet Protocol) layer. MQTT protocol is already frequently used in the automotive industry and telecommunications, offshore industry, manufacturing, etc. MQTT broad-

cast messages from a device to a cloud and from the cloud to the device. It is also able to operate with millions of IoT devices. Furthermore, it makes it easy to encrypt the messages using Transport Layer Security (TLS) and modern authentication protocols. MQTT communicates through a message broker on a server, where all communication is routed through the centralized broker. A broker is a software running on a computer or in the cloud [52].

DDS is a middleware protocol, where middleware is the software layer between applications and the operating system. This indicates that it provides services for applications beyond those that are available from an operating system. It connects components in a system together, creating low-latency data connectivity, reliability, and scalable architecture to adapt it to IoT. It also includes security mechanisms for providing authentication, confidentiality, access control, and integrity. DDS uses a P2P architecture, and the data is not required to be brokered by a cloud or a server. This is a decentralized architecture, which establishes communications between the peers (nodes). The data only runs to a data center, when the data center requires this [53].

While both the MQTT and DDS protocols provide communications for the IoT, the significant difference between them is the deployment topology. DDS is to prefer when not all the data is centralized, where it, for example, would be inefficient to route all sensor data through a data center. MQTT is preferred when, for example, remote asset monitoring is utilized. In this project, the DDS protocol is likely to be the most optimal solution. This is because the system is already built as a P2P network with the ROS system, and will then be the most efficient system.

To draw parallels with examples where the protocols are used today, the DDS is used in Volkswagen's smart cars. The driver assistance and safety systems in these cars use DDS to connect LiDARs, radars, and cameras. This is used in functions such as lane detection, collision avoidance, keeping the vehicle inside the lanes, and to detect doziness of a driver [54]. The car manufacturer BMW uses MQTT for a car-sharing service, where the MQTT protocol allows for remote control of the cars, such as in the lock/unlock function of the vehicle and in the collecting data [55].

Chapter 4

Truck model upgrades and implementations

This chapter will describe the upgrades of the truck model. That is, both the hardware installation and how it has been implemented in the system. These upgrades are the steering angle feedback and its PI controller, the computer vision based path tracking, the charging cycle activation and deactivation, the opportunity charging, and the LiDAR implementations.

4.1 Servo

In the previous work, the steering angle of the truck model was controlled by an open-loop controller. An open-loop controller implies that it does not have feedback on the current steering position, and this could lead to inaccurate steering. To improve this, a closed-loop controller is desired, and different sensors were investigated for achieving a closed-loop. Unfortunately, beneath the truck model, there was not enough space to install a new sensor. Additionally, the truck model is mounted with dampers, which means that there were too many moving parts under the truck model, which further made it more challenging to fit a new sensor. Instead, there was investigated an alternative to measure the steering angle position. This was achieved by measuring the position of the axle from the steering servo's internal potentiometer, and to use the potentiometer output variable to calculate the steering angle position.

The servo consists of a rotary potentiometer that output a various voltage related to the angle, a gearbox to the transmission, and a DC motor connected to a control circuit on a circuit board. The internal potentiometer has three pins, where the centre pin 2 is the output variable, as illustrated in figure 4.1.

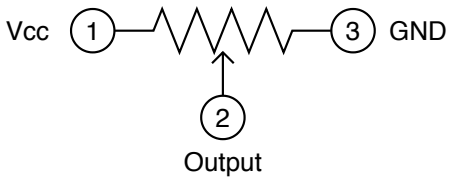
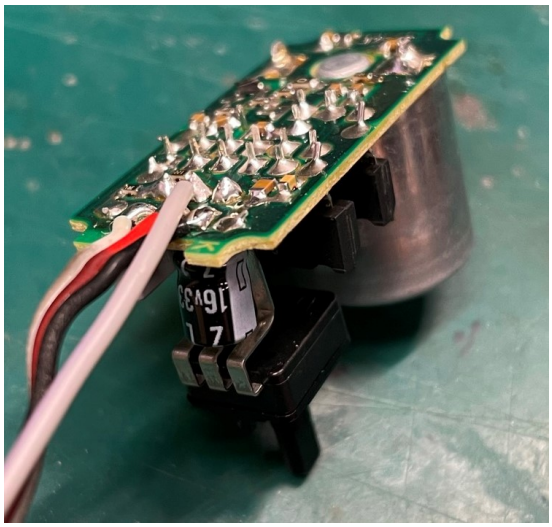
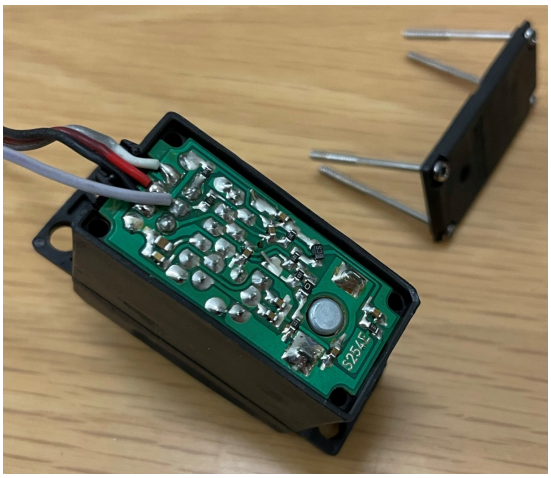


Figure 4.1: Potentiometer

This pin had to be located inside the servo, and this was done by dismantling the servo. See figure 4.2. The pin of the output variable was located, and a new wire was soldered into the circuit board to receive the voltage signal of the output variable. This is the grey wire in the figure. An advantage of this modification is that it makes it possible to receive the feedback on the servo angle position practically for free.



(a) Servo soldering



(b) Servo fitted back

Figure 4.2: Soldering the feedback wire on the servo

The output variable varied between 0.870V in the left position and 1.803V in the right position when measured to the ground. The output variable was connected to the microcontroller Teensy 3.2 on an ADC (Analog to Digital Converter) pin. By default, the ADC pins on the microcontroller have a 10 bit ADC resolution, with a reference voltage of 3.3 V. A 10 bit resolution meant that it could convert a digital signal to a range r of 0 - 1023. The step range was calculated as:

$$\frac{3.3\ V}{1024\ r} = 0.00322\ V/r \tag{4.1}$$

Then 3.22 mV is divided for each step in the range. For the signal from the potentiometer, this gave a voltage range of:

$$\frac{1.803\ V - 0.870\ V}{0.00322\ V/r} = 289\ r \tag{4.2}$$

This leads to a resolution of 289 steps for the region of the operable steering angle. To improve the resolution, the bit was scaled to a 12 bit ADC on the microcontroller, which has a signal range of 0 - 4095. Using the same procedure with 12 bits:

$$\frac{3.3\ V}{4096\ r} = 0.0008\ V/r \tag{4.3}$$

$$\frac{1.803\ V - 0.870\ V}{0.0008\ V/r} = 1166\ r \tag{4.4}$$

From this, the final resolution was 1166 steps for a total $\sim$ 90 degrees at the steering servo. When measuring the output variable signal, there was too much noise in the signal. Hence, a filter had to be made to compensate for this, which did not result in a considerable time delay. After testing with different types of filtering, the best-found solution for this case was a type of band-pass filter. In the Teensy Microcontroller node, a time delay of 2 ms was implemented to read the output variable. In an interval of 20 values, at least half of them gave a reliable signal. The values were then inserted and sorted in an array of 20 values, where the top five and bottom five values were discarded. The mean of the 10 middle values was calculated, and the mean was published as a ROS topic as the final output value. This gave a more stable and reliable output signal which could be used in the closed-loop controller. A time delay of 2 ms for each reading of the signal from the potentiometer meant a new output value was published each 40 ms or, in other words, 25 times per second. This was an acceptable time delay for the process. A flowchart of the process is illustrated in figure 4.3

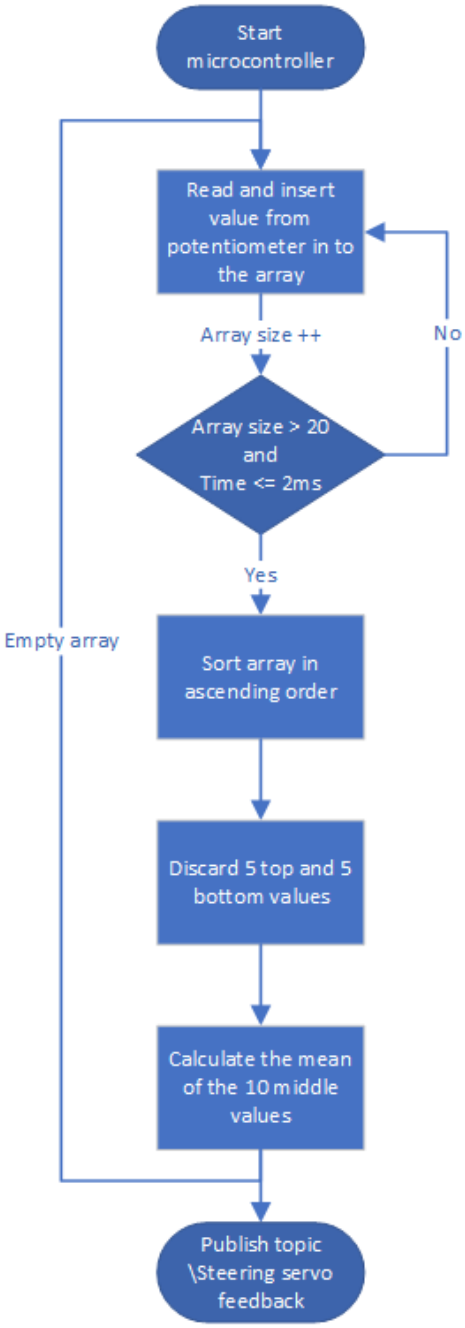


Figure 4.3: Flow chart servo

4.2 PID controller

In process control, the PID controller is widely used, and approximately 90 percent of the closed-loop operations in industrial automation use the PID controller due to its robustness and simplicity [56]. The feedback controller consists of three terms: proportional P, integral I, and derivative D. The P term gives an output proportional to the regulation deviation, with a given factor K_p. The factor is the proportional gain and is used to increase or decrease the response time of the process value to get the process value to reach the desired value (set-point). However, the P term alone is insufficient for the process value to stabilize on its set-point, also called the steady-state. This is because the term requires biases to create a corrective response. Then the I term is introduced to make the process value reach and stabilize to the steady-state condition. The I term accumulates (integrates) the regu-

lation deviation over time, multiplied with a given factor K_i. The factor accelerates the process value movement towards the set-point, and the term eliminates the steady-state error. By the time the error is canceled, the term will cease to grow. The D term dampens the regulation deviations' rapid changes by estimating future trends based on the current rate of change, multiplied with a given factor K_d. The factor is used to predict the system behavior and reduces the settling time, which will increase the system response [56].

These three terms are combined to produce the continuous-time PID controller with the control output $u(t)$ [56]:

$$u(t) = \underbrace{K_p e(t)}_{\text{P}} + \underbrace{K_i \int_0^t e(t')dt'}_{\text{I}} + \underbrace{K_d \frac{de(t)}{dt}}_{\text{D}} \tag{4.5}$$

In the continuous-time PID equation, $e(t)$ is the regulation deviation, namely the error between the set-point and the process value. The factors K_p, K_i and K_d are the coefficients of the P, I and D terms, respectively. These are non-negative gains used to tune the controller to the desired process response. A block diagram of the PID controller in a feedback loop is illustrated in figure 4.4, where $r(t)$ is the set-point, $e(t)$ is the error, $u(t)$ is the controller output, $y(t)$ is the process variable, and $p\_v(t)$ is the measured process value [57]:

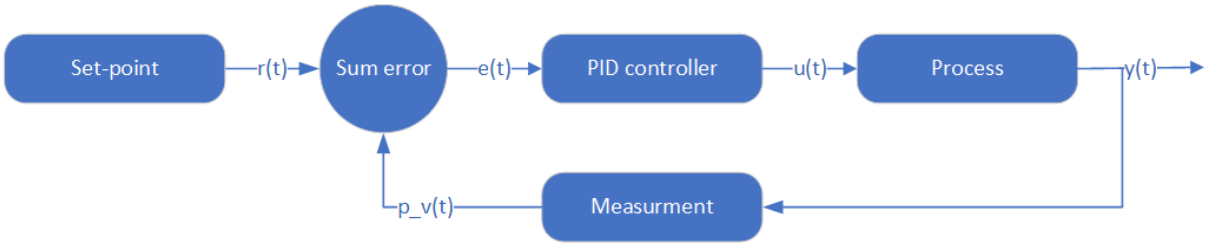


Figure 4.4: PID block diagram

4.2.1 Implementing a discrete-time PID controller

When designing a digital PID controller, the controller needs to be discretized for digital implementation. Discretization is the process of replacing continuous variables with a finite set of points. The method of discretizing will create a certain amount of error, where the error depends on how large the intervals the discrete signals are divided in. There are many ways of discretizing a PID controller, and two common methods are investigated. The first method is replacing the integral with a summation of the errors multiplied by the sampling rate, as shown in equation (4.6). For the derivate, Euler's first-order Backward differentiation approximation is used, as shown in equation (4.7). With these approximations, the first discrete PID controller can be calculated from the continuous-time PID controller in equation (4.5) [58] [59].

$$\int_0^{t_k} e(t')dt' \approx \sum_{n=1}^{k} e(t_n)\Delta t \tag{4.6}$$

$$\frac{de(t_k)}{dt} \approx \frac{e(t_k) - e(t_{k-1})}{\Delta t} \tag{4.7}$$

This gives the first discrete-time PID controller:

$$u(t_k) = K_p e(t_k) + K_i \sum_{n=1}^{k} e(t_n)\Delta t + K_d \frac{e(t_k) - e(t_{k-1})}{\Delta t} \tag{4.8}$$

The second method is more advanced, with differentiation on both sides of the continuous-time PID controller equation (4.5), which gives:

$$\frac{du(t)}{dt} = K_p \frac{de(t)}{dt} + K_i e(t) + K_d \frac{d^2e(t)}{dt^2} \tag{4.9}$$

For the derivate, the Euler's first (4.7) and second order backward differentiation approximation is used, where the second backward differentiation is defined as [58]:

$$\frac{d^2e(t_k)}{dt^2} \approx \frac{\frac{e(t_k)-e(t_{k-1})}{\Delta t} - \frac{e(t_{k-1})-e(t_{k-2})}{\Delta t}}{\Delta t} \tag{4.10}$$

Applying these approximations to the equation (4.9) gives:

$$\frac{u(t_k)-u(t_{k-1})}{\Delta t} = K_p \frac{e(t_k)-e(t_{k-1})}{\Delta t} + K_i e(t_k) + K_d \frac{\frac{e(t_k)-e(t_{k-1})}{\Delta t} - \frac{e(t_{k-1})-e(t_{k-2})}{\Delta t}}{\Delta t} \tag{4.11}$$

Solving for $u(t_k)$ gives the second discrete-time PID controller:

$$u(t_k) = u(t_{k-1}) + K_p[e(t_k)-e(t_{k-1})] + K_i e(t_k)\Delta t + K_d \frac{e(t_k)-2e(t_{k-1})+e(t_{k-2})}{\Delta t} \tag{4.12}$$

An advantage of the second discrete-time controller is that there is no need to keep track of the sum.

4.2.2 Anti-windup mechanism

A vehicle cannot drive faster than the maximum speed, and a servo cannot exceed the maximum/minimum position. Therefore, an anti-windup mechanism had to be implemented, which meant that an integral windup had to be prevented. An integral windup could happen if the set-point exceeds the saturation limit, when there is a significant change in the set-point, or a large disturbance is present. This may cause the process variable to reach the maximum or minimum limits, while the error being different from zero. As a result, the integral will accumulate beyond the saturation. The system could then overshoot, and the integral value would continue to increase or decrease steadily, depending on if its maximum or minimum limit is crossed. This leads to a lag in the system. An increasing windup-lag is demonstrated in the left figure in figure 4.5. Here, the PID control value from the calculation increases steadily from a large positive change of the set-point, where an integral windup occurs. This deviates from the actual PID control value of the system, which is the maximum value the system could respond to. The same situation is illustrated in the right figure, but here an integral anti-windup mechanism is implemented. This shows how the anti-windup makes the system respond as desired and how the anti-windup prevents the lag [58] [60].

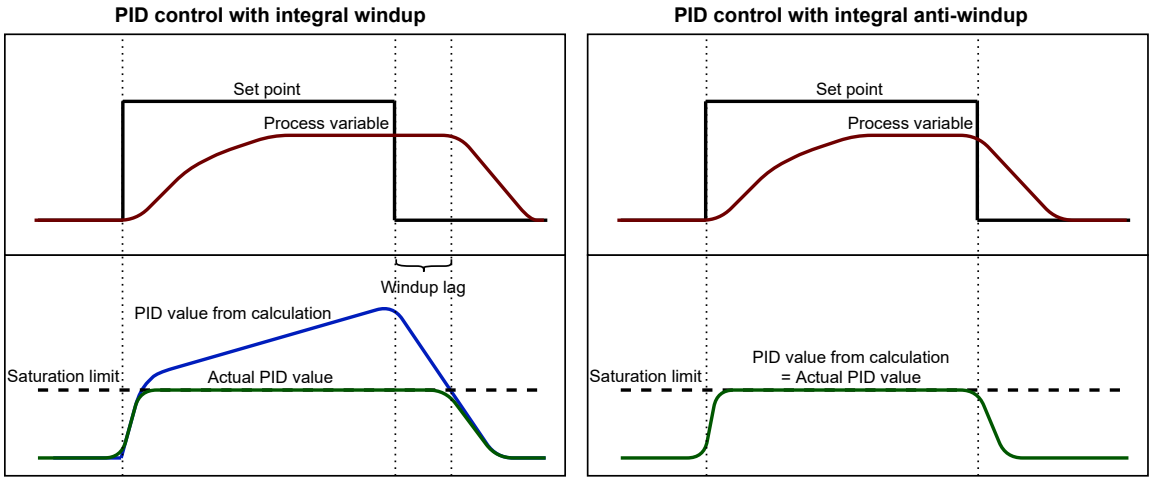


Figure 4.5: Windup lag

There are many different approaches to address the windup issue, and the most common ones are clamping or back-calculation. With clamping, there are various ways to implement it, as for example by limiting the integral term from accumulation when the controller output is saturated. If so, the integral term is prevented from accumulating when it is above or below the pre-determined bounds, where the bounds are the maximum or minimum value the process variable actually can output. It is also possible to disable the integral term when the error is more significant than useful for the integral term. It could be enabled when the process variable is in the controllable region. The back-calculation is a more advanced method. The difference between the saturated and unsaturated output has to be measured and used as feedback to the integral term, multiplied with a new factor K_b. The factor is then the back-calculation gain which needs to be tuned. The new feedback signal is used to unwind the integral accumulation [60] [61].

In this study, the clamping method is used because this method will always work and does not depend on a fine-tuned parameter, which the back-calculation needs. Integral anti-windup is crucial for the steering controller, where the steering angle set-point could exceed the saturation limits. This will prevent any lag in the system and create a robust controller. Integral anti-windup has been implemented for the velocity controller as well, but the truck model will most likely not exceed the speed limit saturation. This is because of the fixed set-points in the automated Ackermann drive node, where the set-points for the speed lays in the middle region of the saturation limits. Since there is still a possibility of setting the set-points manually from the controller, possible misreadings from the hall effect sensor and significant changes in the error cannot be excluded, the anti-windup mechanism is implemented. The chosen method was to limit the integral term $iTerm$ from accumulating when the output was saturated ($maxLimit$ and $minLimit$) with the following pseudo-code:

Algorithm 1 Anti-windup mechanism

if $output > maxLimit$ **then**
  $iTerm \leftarrow iTerm - (output - maxLimit)$
  $output \leftarrow maxLimit$
else if $output < minLimit$ **then**
  $iTerm \leftarrow iTerm + (minLimit - output)$
  $output \leftarrow minLimit$
end if

4.2.3 Tuning the discrete-time PID controllers

The ROS does not support real-time changes, thus making it challenging to tune the PID controller. So, the set-point values, the process variable, and the time were stored at a .txt file at the Ackermann drive node at the Jetson TX2. These files were transferred to the host laptop and plotted in Matlab. For a tuning method, the Ziegler-Nichols (ZN) methods were investigated. The first ZN method and the second ZN method are accepted as standards in control system applications [57]. Both of the methods involve a priori assumption for the system model. An advantage is that the model of the system does not need to be specifically known, as is the case in this study. The controller is tuned based on the step response of the process. The first method was found suitable for tuning the controller, requiring fewer tuning changes than the second method. Two buttons on the joystick controller were then used to increase or decrease the steering angle by 10 degrees at the time. In the first method, two parameters are measured; the time delay L and a time constant T. These parameters are found by drawing a tangent line at the inflection point of the process value after the increase/decrease in set-point [57]. This is illustrated in figure 4.7 .

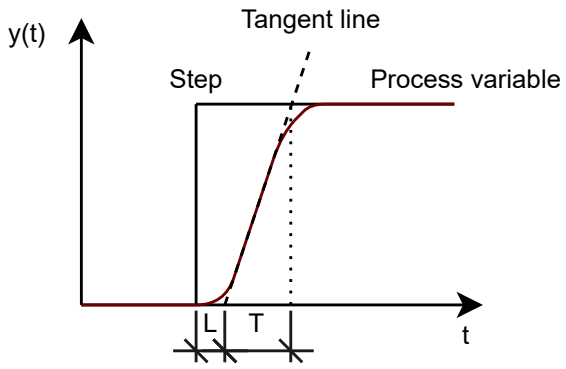


Figure 4.6: First method Ziegler-Nichols

The parameters are used to tune the discrete-time PID controller, based on the equations in table 4.1. A P controller is not sufficient as a controller for this case, because it will have a steady-state error. Thus, a PI or PID controller was considered. Since the process variable still had noise at the operation of the process, the PI controller gave the best result.

Table 4.1: ZN First method parameters

Type	K_p	$T_i = \frac{K_p}{K_i}$	$T_d = \frac{K_i}{K_p}$
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

After repeating these tuning techniques a few times, with the knowledge of the parameter K_p increasing/decreasing the rise time towards the set-point, and K_i accelerates the process variable towards the set-point, optimal parameters were found [57]. With the following conversion in equation (4.13) from T_i to K_i, the PI controller gains were found and presented in table 4.2.

$$K_i = \frac{K_p}{T_i} \tag{4.13}$$

Table 4.2: Tuning parameters

Parameter	Description	Tuned value
T	Time constant	0.043 s
L	Time delay	0.076 s
K_p	Proportional gain	0.51
K_i	Integral gain	2.01

Testing of the tuned controller gains in the first discrete-time PI controller can be seen in figure 4.7, where the controller is from equation (4.8). The equation is derived from the summation of errors and Euler's first order backward differential approximation.

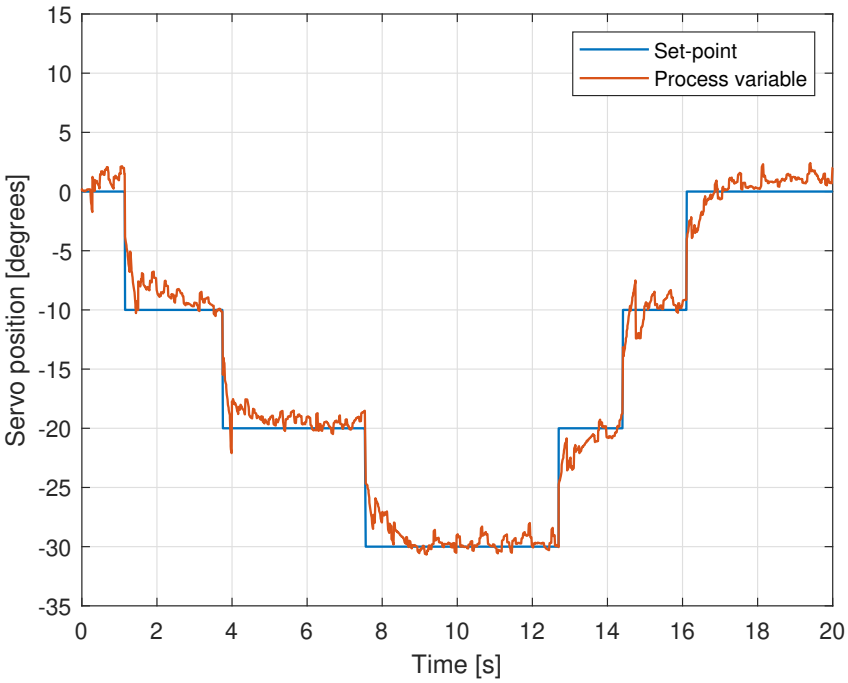


Figure 4.7: First PI controller

Testing the tuned gain values for the second discrete-time PI controller can be seen in figure 4.8. This is the controller from equation (4.12). The equation is derived from Euler's first and second-order backward differentiation approximation.

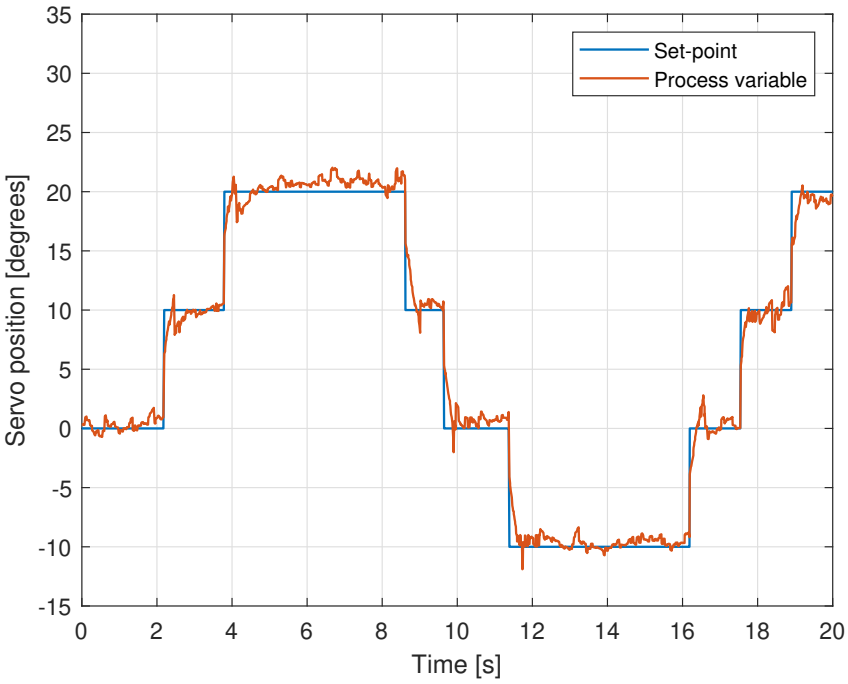


Figure 4.8: Second PI controller

The second discrete-time PI controller in figure 4.8 gave the best result, and is used further in this study to control the steering angle. This PI controller gave critical damping, which is the threshold

between over- and under damping. The oscillations settle at the equilibrium point at a fast rate, and the oscillation oscillates maximum once at the equilibrium. It can be noted that the noise at the process variable could be smoothed out, but this would come at the cost of a larger time delay in the system. However, the tuned PI controller handled the noise and gave an accurate steering for the system.

4.3 Computer vision based path tracking

The computer vision based path tracking is upgraded with a new camera and a new path tracking strategy. This section presents how these functions have been upgraded.

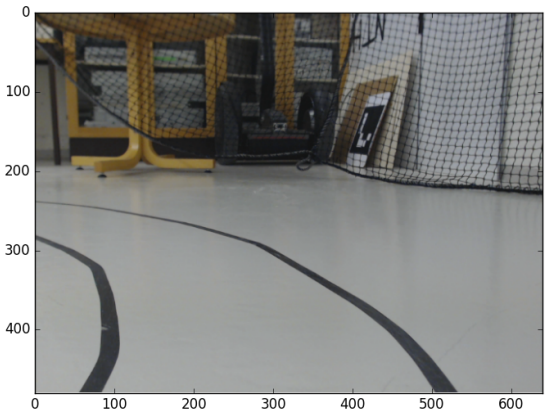
4.3.1 Merging the camera frames

A new camera was mounted to minimize the blind spot in front of the truck model, so that that sharp corners are not cut or overlooked, and to give more precise steering. The already installed camera inside the cab could detect an area of 30-100 cm in front of the truck model. The intentional plan was to mount the new camera in the front bumper as shown in figure 4.9a, but unfortunately, this did not give the intended result. The camera was too close to the road and was not able to detect the lanes. Instead, the camera was mounted to a tripod on top of the cab of the truck model, as shown in figure 4.9b. This gave a much better view of the lanes, and still, the LiDAR sensor could fit between the tripod legs.

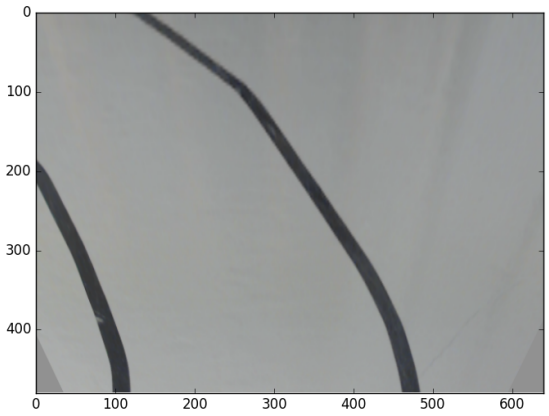
(a) Camera front bumper **(b)** Camera top of cab

Figure 4.9: Testing different camera positions

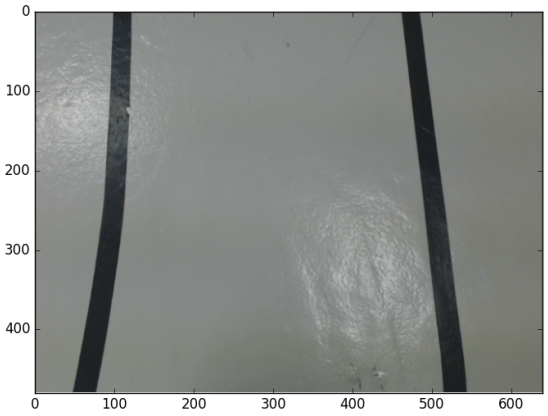
The raw images from the cameras were captured with a function in the OpenCV library at the Camera broadcast node at the Nvidia Jetson TX2, specifically the VideoCapture function. After a raw image is captured, the resolution is scaled down from 1080 x 720p to a 640 x 480p image. With a lower resolution, the image processing needs less computation power, making it more efficient to use in computer vision based path tracking. Although the image has a lower resolution, it did not affect the path tracking. The raw images are shown in figure 4.10a and 4.10c. In the raw image from the new camera mounted at the top of the cab, named Cam 2, it was decided to keep as is to get as much information in front of the vehicle as possible. This meant that the raw images from Cam 1, which was already mounted inside the cab, had to be warped to match the image of Cam 2. The image was warped with functions from the OpenCV library, getPerspectiveTransform and warpPerspective. The perspective transform creates a 3x3 matrix from points in the image to warp these points to a new perspective, and warpPerspective creates the desired image. The result of this process can be seen in figure 4.10b and how it matches the image can be seen in figure 4.10d.



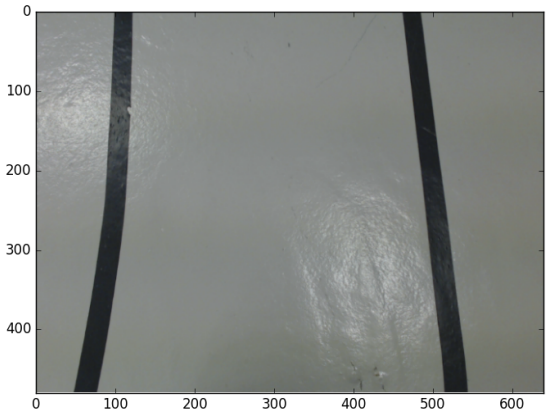
(a) Cam 1



(b) Cam 1 warped final result



(c) Cam 2

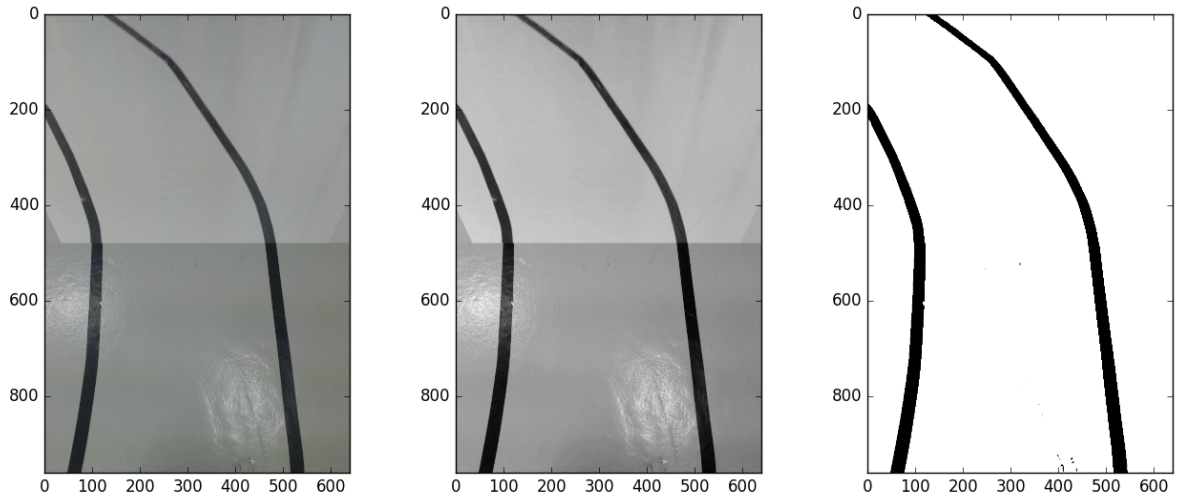


(d) Cam 2 final result

Figure 4.10: Warping image

The images needed to be merged, and for this operation, the NumPy library function Concatenate was used. This combines the two images to a 640 x 960p image, and the result can be seen in figure 4.11a. To process this image for the path tracking, the image was first converted into grayscale. This is shown in figure 4.11b and was done using the OpenCV function cvtColor. This is a straightforward function that converts an image. Next, the grayscale image is converted into a black-and-white image

with the OpenCV function adaptiveThreshold. This is an adaptive function that determines the pixels' information around a small region of the current pixel and sets an adaptive threshold value. If the pixel is beneath the threshold value, it is set to zero to illustrate white. Otherwise, when the pixel is above the threshold value, it is set to the maximum value to illustrate black. The adaptive threshold is more robust to different light conditions and cancels noise in the image. The final image is shown in figure 4.11c, and with this image, center points between the lanes can be found for the path tracking.

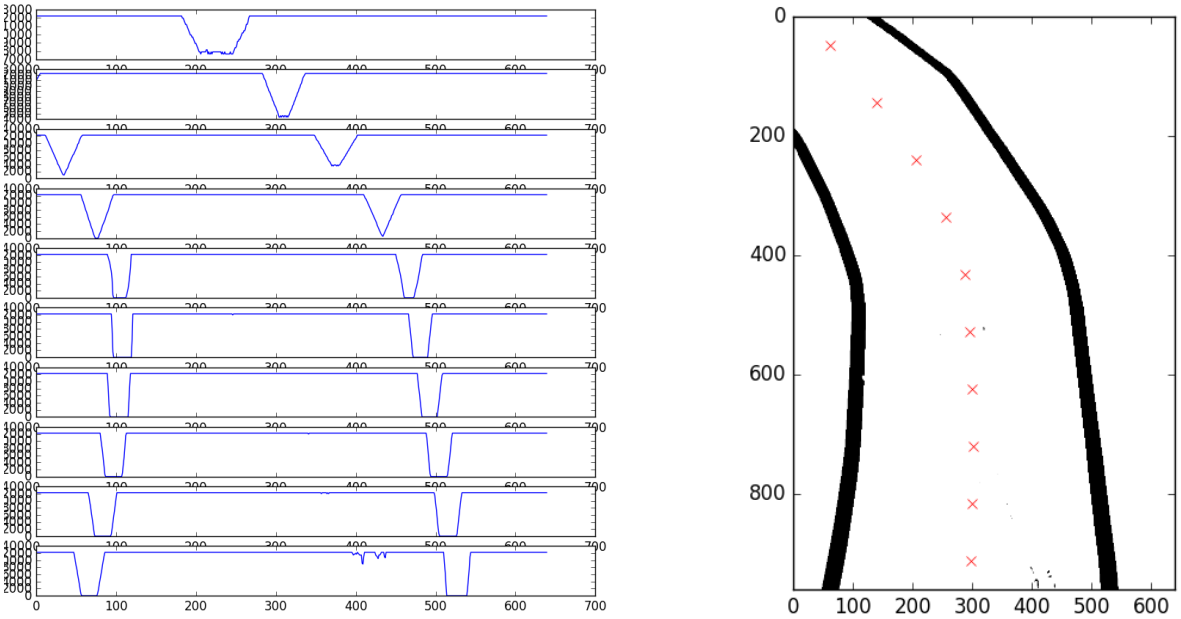


(a) Concatenated image **(b)** Grayscale image **(c)** Black and white image

Figure 4.11: Image processing

The black-and-white image is adapted into a histogram, where the summation of the intensity of the pixels in the image is calculated and plotted as shown in figure 4.12a. Here, the image is divided into 10 slices in the histogram. A threshold value is pre-determined, and now the function Find_lanes in the support script Lane detection loops through each slice, from each side, to detect the lanes. If the value in a slice at the histogram exceeds the threshold value, starting from the left, the left lane point is set. The same applies to the right side, where the right lane point is set.

The center points in the x-direction are calculated by adding the right lane point to the left lane point and dividing the result by two. The point is only valid if the distance between the left and right lane points is over a preset distance of 20 cm. If the distance between the lanes is too short, it indicates that only one lane in the slice is detected. When only one lane is detected, as is the case in the two top slices in 4.12a, the Find_lane function sets the points based on previous points. More specifically, it checks the direction of the previous points according to the circular arc of the previous slices, and the center points in slices with only one lane detected could be determined. Now a detection area of 0 - 100 cm in front of the truck model could be observed. The final image for the computer vision based path tracking is viewed in 4.12b.



(a) Histogram

(b) Path tracking points

Figure 4.12: Image processing for path tracking

4.3.2 Camera vision based path tracking strategy

To implement a path tracking strategy, the truck model had to be modeled. In general, the Ackermann steering model is a good approximation of vehicle motion. In this model, the closest wheel to the turning rotation has a larger turning radius than the outer wheel from the turning radius. It does not account for vehicle stability and mechanical wear, such as transmission slip and limited-slip differential [62]. In the truck model used in this study, this model was found sufficient for its purpose. Since a stiff rod connects the two front wheels and is controlled by one servo, it was adequate to consider a simpler model. This implies that the Ackermann steering model could be reduced to a simplified bicycle model, as shown in figure 4.13. Here, the right and left tires are combined into a "virtual" bicycle.

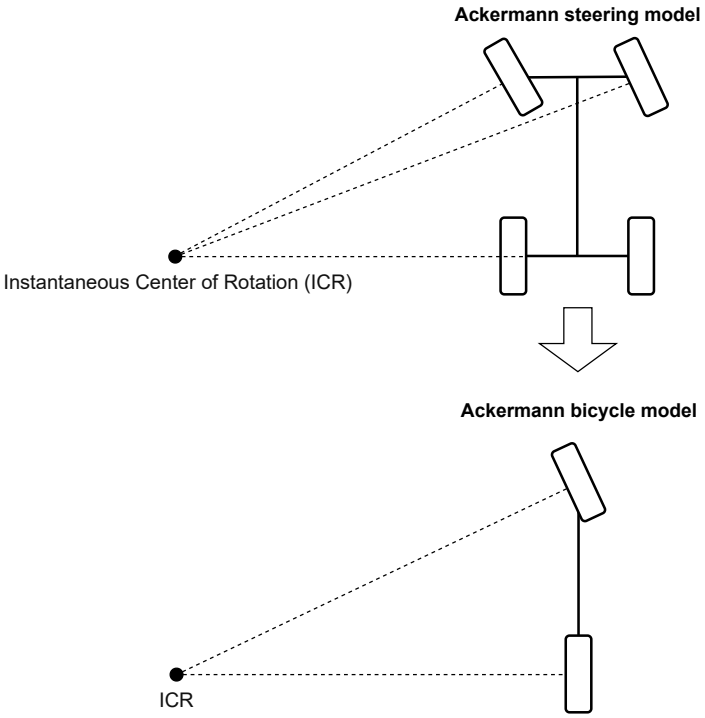


Figure 4.13: Reduced Ackermann steering model

After the new camera was mounted, a new path tracking strategy was implemented. The strategy for controlling the steering angle of the truck model was to use the Ackermann bicycle model and set the steering angle directly. In figure 4.14 the geometry for the bicycle model is shown. The point *B* is an example of one of the points found in the final image 4.12b, and point *A* is a fixed point at the center of the truck models back axle.

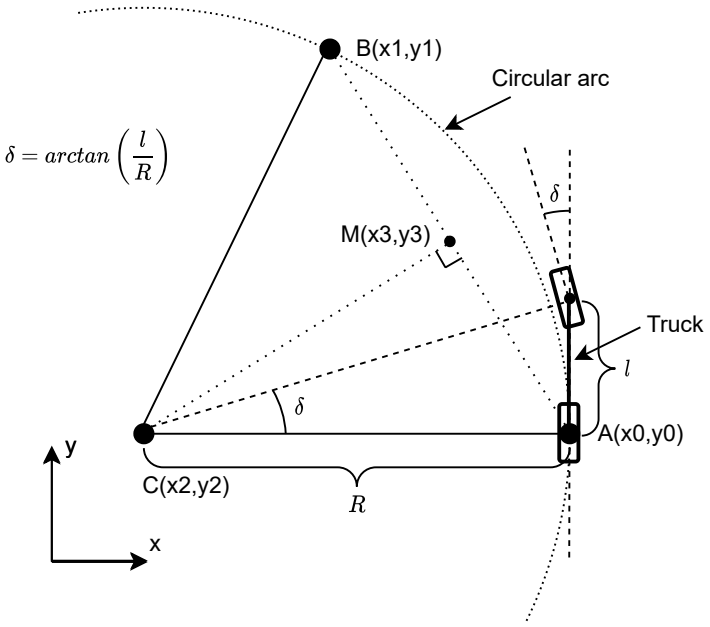


Figure 4.14: Ackermann geometry model

Two points are not sufficient to determine the radius of a specific circle in a circular arc. So, a third point *C* at the center of the circular arc was introduced, as shown in figure 4.14. The point *C* is then the instantaneous center of rotation. l is known from measurement between the axles at the truck

model, and then the main goal is to calculate the radius R to achieve the steering angle δ:

$$\delta = tan^{-1}\left(\frac{l}{R}\right) \tag{4.14}$$

In the three points A, B, and C, the only unknown variable is the $x2$ in point C. $y2$ and $y0$ lies on a perpendicular line, the points in B are obtained from figure 4.12b, and the points in A are used as starting points. Point C is equidistant from both A and B. Therefore, the point C must lie on the perpendicular bisector of the points A and B. A new midpoint M is introduced, and then the perpendicular bisector is defined as the line perpendicular at the midpoint M between the line connecting A and B. The midpoint is calculated as:

$$M(x_3, y_3) = M\left(\frac{x_0 + x_1}{2}, \frac{y_0 + y_1}{2}\right) \tag{4.15}$$

When given two points, a slope m of a line could be calculated with the rise (Δy) over run (Δx) formula:

$$m = \frac{\Delta y}{\Delta x} \tag{4.16}$$

Using the rise over run formula with point A and B, the perpendicular slope m from point C to M is its negative reciprocal from point A to B:

$$m = -\frac{1}{\frac{y_1 - y_0}{x_1 - x_0}} = -\frac{x_1 - x_0}{y_1 - y_0} \tag{4.17}$$

By using the point-slope formula:

$$y - y_p = m(x - x_p) \tag{4.18}$$

The line through C and M with slope m is:

$$y_2 - y_3 = m(x_2 - x_3) \tag{4.19}$$

Solving for $x2$:

$$x_2 = \frac{y_2 + mx_3 - y_3}{m} \tag{4.20}$$

Inserting for m and $x3$:

$$x_2 = \frac{y_2 - \frac{x_1 - x_0}{y_1 - y_0}\frac{x_0 + x_1}{2} - \frac{y_0 + y_1}{2}}{-\frac{x_1 - x_0}{y_1 - y_0}} \tag{4.21}$$

And rearranging:

$$x_2 = \frac{x_0^2 - x_1^2 + (y_0 - y_1)(y_0 + y_1 - 2y_2)}{2(x_0 - x_1)} \tag{4.22}$$

With the previous unknown variable $x2$ the radius R can now be calculated to achieve the steering angle δ from the Ackermann bicycle model:

$$R = x2 - x0 \tag{4.23}$$

Different implementations were tested when implementing the Ackermann bicycle model calculations in the function Radius2p in the support script Simple trajectory. When the truck model was driving at the regular track, the best-found solution was to use the six B points farthest away from the truck model. The average $x2$ of these points was calculated for the radius to set the steering angle delta, which led to smooth and balanced steering. If there were found less than six B points,

which could occur in sharp turns, the average $x2$ of the remaining B points were calculated to determine the radius for the steering angle delta. Calculating the average $x2$ of many B points made the steering more robust if a disturbance led to a misread point at the path tracking function, and it gave a more precise cornering of the truck model. When the truck model entered the charging station, the lanes are a straight path. The best-found solution for a straight path was to use the two B points closest to the truck model and the furthest B point away from the truck as an anchor point. With the same procedure, the average $x2$ of these B points was computed to decide the radius to set the steering angle delta. This change was done since there was stricter demands of the positioning in the tracks for an optimal charging cycle, and then the position directly in front of the truck model was the most important to calculate. This strategy could be utilized since there only were straight lanes to follow with no corners and kept the truck model more aligned in the center of the lanes. A pseudo-code to illustrate the calculation of the steering angle is shown below.

Algorithm 2 Calculate steering angle

```
for B points found do
    Solve for x2
    Compute the radius of the circular arc
end for
Calculate the average radius of the selected B points
Compute the steering angle
return steering angle
```

4.4 Charging cycle

A red mark is taped at the start of the charging station to indicate the charging station for the computer vision. In the support script Lane detection, a function, Pipeline5, at the Ackermann drive node, as described in section 3.4.3, searches for the charging station mark when the truck model is driving. This is done with the OpenCV library function inRange. In this function, objects are detected in the images based on a pixel range value in the HSV (Hue, Saturation, Value) colorspace. Here, a mask is set with a value of the threshold inside the range it searches after, in this case, an HSV colorspace area between light red and dark red. When the mask threshold value is inside the area, the charging station mark is detected. The HSV is a colorspace similar to the RGB (Red, Green, Blue) colorspace, but the hue in HSV models the color type, making it useful in image processing where the task is to segment objects based on a color [49]. An illustration of the truck model detecting the charging station, with the vehicle-side coil and the dynamic road-side coils, is shown in figure 4.15.

For a more stable approach to the charging station, a yellow mark is taped at the regular track before the last corner to the charging station. With the same procedure for detecting the charging station, the Pipeline3 function detects this mark. This indicates that the charging station is near, and the reference speed can be decreased.

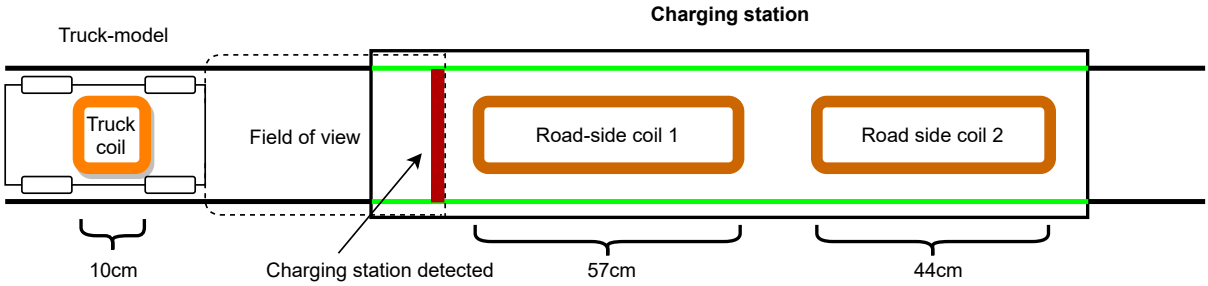


Figure 4.15: Charging station detected

After the charging station mark is detected, a zero position value is set in the Ackermann drive node. Now, the road-side coils could be activated and deactivated at a desired position by calculating the distance traveled from the zero position value to the road-side coils. In figure 4.16 there is introduced null-points of the road-side coils, and where the truck model is positioned for activation and deactivation of the road-side coil 1 at the null-point. This position is determined from the coupling coefficient in figure 2.8, where a non-negligible magnetic coupling coefficient must be present for the vehicle-side coil to receive power. Then, in this case, the start null-point at 32 cm is the exact point where the vehicle-side coil starts to receive power from the road-side coils, and the end null-point at 89 cm is the exact position the power connection between the coils is lost. The same null-point for the truck model position will apply for the road-side coil 2, where the null-points are at 107 cm for activation and 151 cm for deactivation.

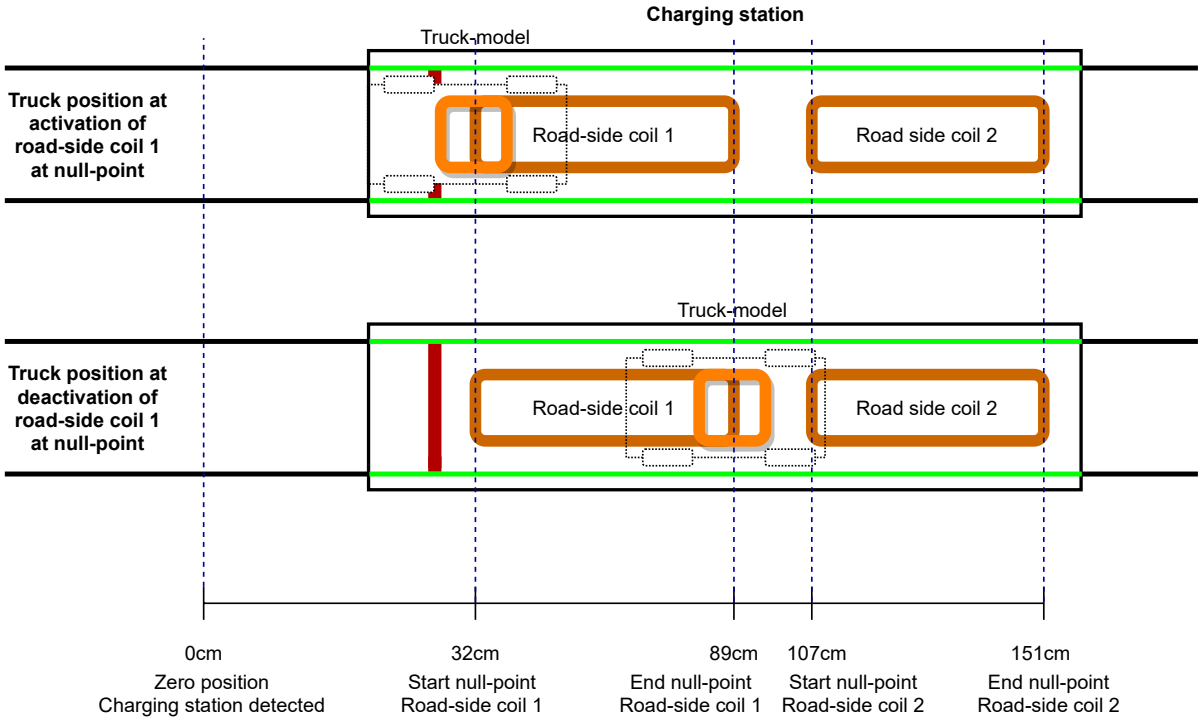


Figure 4.16: Trucks position at null-points at the charging station

A straightforward flowchart of the charging cycle algorithm in the Ackermann drive node is illustrated in 4.17. Both distance measurement and time condition for the charging state between the activation and deactivation of the road-side coils are implemented. The distance measurement is for the dynamic charging, and the time condition could be used if the truck model operates with the static opportunity charging solution. In the pre-study for this study, the green lines were marked at the charging station, and the function Pipeline4 in the support script Lane detection is used for the path tracking at the charging station [47]. Thus, the green lines indicate at the Ackermann drive node that the truck model is driving at the charging station, and the accurate path tracking points are used.

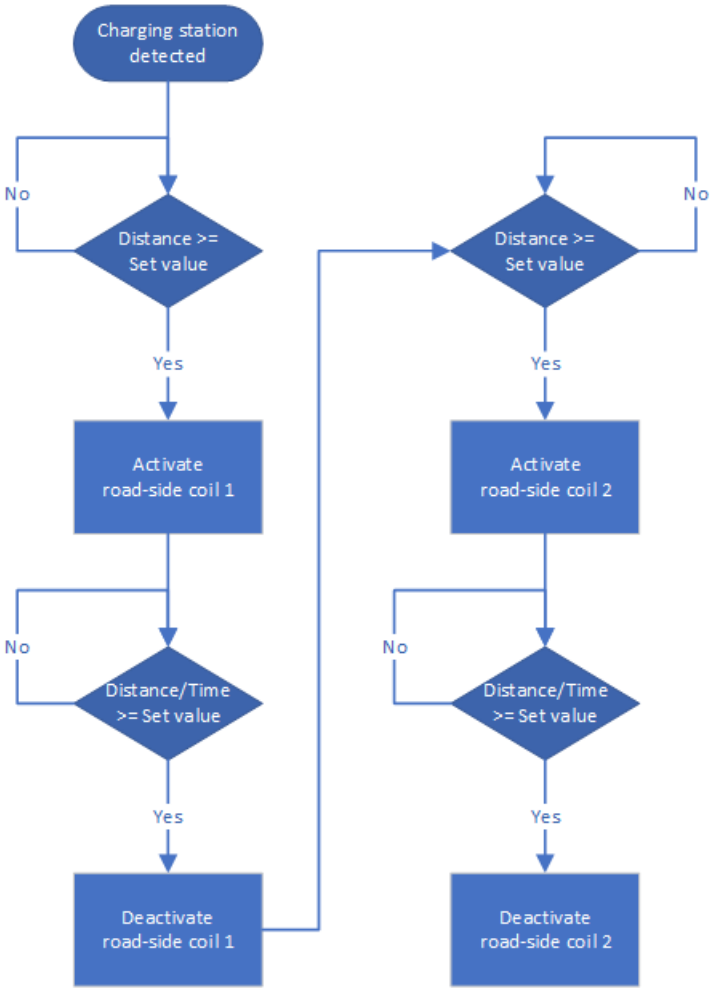


Figure 4.17: Flowchart of the charging cycle

4.5 Opportunity charging

An opportunity charging was implemented in the Ackermann drive node for the truck model, which could occur while the truck model is stationary. A scenario for opportunity charging could, e.g., be when a truck stops at a red light, a depot when it loads/unloads goods, or could be parked for a pit stop. This was to determine how the efficiency of a static charging solution could compare to the dynamic charging solution. The rectangular road-side coil 2 was exchanged with a square road-side coil, with the same specifications as the vehicle-side coil. Now, both the transmitting and receiving coil could overlap each other with no losses due to the different sizes of the coils. Two different approaches were implemented, where the first approach was a quasistatic charging, and the second approach was a static charging. In the quasistatic charging condition, the truck model reduces its

speed as it approaches the square road-side coil. The road-side coil is activated and deactivated at the null-points, which have the same null-point positions in the start and end of the coil, as the rectangular road-side coils described in the previous section. The truck model stops at the position when the coils are aligned, with the criterion that there is established a good coupling between the coils. Here, the truck model can charge for a specific time, or it can wait until the battery reaches a particular battery percentage/voltage. For the static charging condition, the truck model will decrease its speed and stop in the position where the coils are aligned. First, when the truck model is stationary, the road-side coil is activated. The road-side coil can be deactivated after a certain time or when the battery reaches a specific percentage/voltage. After the road-side coil is deactivated, the truck model starts to move. The pseudo-code of the opportunity charging algorithm can be seen below.

Algorithm 3 Opportunity charging

```
1:  if Charging station detected then
2:    if Connection/Position is OK and Battery voltage is low and Opportunity mark is set then
3:      Set speed to zero
4:      Set static mark to activate road-side coil when speed is zero or Activate road-side coil at
        null-point
5:      if Time mark is set then
6:        Reset Time mark
7:        Set Time stamp
8:      end if
9:      if Real time - Time stamp > Pre-set time or Battery voltage/percentage > Pre-set limit then
10:       Reset Opportunity mark
11:       Deactivate road-side coil before moving or Deactivate road-side coil at null-point
12:     end if
13:   else if Road-side coil is approaching then
14:     Set speed low
15:   else
16:     Set speed normal
17:   end if
18: else
19:   Set speed to normal
20:   Set Opportunity mark
21:   Set Time mark
22:   Reset static mark
23: end if
```

The opportunity charging square road-side coil is activated/deactivated accordingly to the flowchart in figure 4.17, where it is connected to one of the road-side coil connections. It is then activated/deactivated depending on whether the quasistatic or the static condition is tested. These two conditions can be detected in the pseudo-code, where the null-point activation/deactivation is quasistatic charging, and activation/deactivation at the stationary is the static condition.

4.6 LiDAR implementations

In this study, the LiDAR is used for two separate functions, where it is utilized for either obstacle detection or logging.

4.6.1 Obstacle detection

The LiDAR can be used for obstacle detection, but since it's a 2D LiDAR, it had to be tilted forwards, facing the road, to detect objects in front of the truck model. This prevents a collision, and the code at the Car cmd node at the Nvidia Jetson TX2 was expanded to detect a wider area in front of the truck model. Now, it can detect 42.5 cm in front of the truck, still with the same -30:+30 degree in front of the centerline of the truck model. This resulted in the truck model stopping in a reasonable position in front of the obstacle. For a fail-safe option, the obstacle detector function was extended, where it has to be reset at the joystick controller. This prevents the truck model from moving before the obstacle is removed from the road. An illustration of the obstacle detection is shown in figure 4.18.

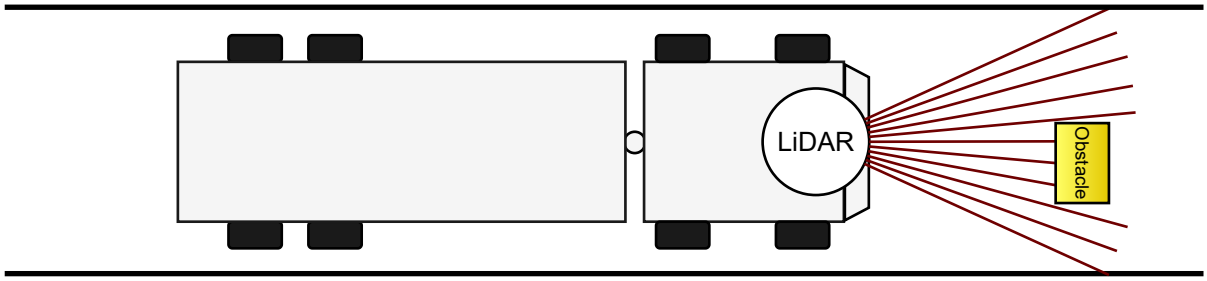


Figure 4.18: 2D illustration of the obstacle detection

The obstacle detection works by sending a laser pulse once every 1 degree, which indicates that every laser scan requires 60 values. If one of the laser pulses has a shorter distance than 42.5 cm, the obstacle detection is activated, and the speed of the truck model is set to zero.

4.6.2 Logging

From the master's thesis in 2019 [44], SLAM methods, such as Hector SLAM, were implemented. Hector SLAM is an odometry-free approach which uses 2D distance readings from the LiDAR. This method could estimate 2D poses and generate maps based on the LiDAR readings. The parameter limits in the SLAM method are set to detect from 0.3 - 12 m, where the map is updated if the truck model has traveled more than 1.5 m or moved more than 1.6 rad. A short description of one iteration of the Hector SLAM method follows: [44]

1. Collect a scan from the LiDAR.
2. Transform the endpoints from the scan to a transformation frame.
3. Discard the endpoints outside the parameter limits.
4. Estimate a 2D pose from the frame.
5. Update the map if the parameter limits are met.

A pose is the position and orientation between a coordinate system, such as a local frame and a world frame [44]. In this study, the ROS Hector Mapping node was used from the Hector SLAM method. The horizontal plane pose position of the truck model, in the x and y position, was extracted with a subscription of a ROS topic from the node. For this function to work correctly, the LiDAR had to be placed horizontally on top of the truck model. In addition, since the testing area was an open monotone space with a mesh net around the track, various object had to be placed around the room and track. These objects were placed such that it did not affect the computer vision based path tracking, but created landmarks so that the LiDAR could obtain a precise mapping. The position values, read as x and y, were stored in a .txt file from the Ackermann drive node at the Nvidia Jetson TX2.

Chapter 5

Experimental testing of autonomous driving and the optimal charging cycle

This chapter presents the results of the tests performed on the truck model on the improved computer vision based path tracking, the different dynamic charging cycle tests, and the opportunity charging tests.

5.1 Autonomous driving

Four consecutive test runs were performed to test the impact of the upgrades for the autonomous driving. This was in regards to the closed-loop PI steering controller with servo feedback and the new path tracking with a two-camera implementation. These test runs are compared to four consecutive runs with the truck model in its previous condition, with an open-loop steering controller and one-camera implementation for the path tracking. There are two methods for mapping and logging the position of the truck model, where the position could be logged with sensor input from the IMU or the LiDAR. Both functions log the values in a .txt file at the Ackermann drive node in Nvidia Jetson TX2 and are transferred to the host computer and plotted with Matlab. A mockup track for testing was created, where black electrical tape was used for the regular track. The charging station consists of green electrical tape for the track. A picture of the mockup track can be seen in figure 5.1. The track's lane-width is 25 cm, corresponding to a 1:14 scale of a Norwegian standard road of 3.5 m. The previous testing with an open-loop steering controller and one-camera implementation in the pre-study showed that the computer vision path tracking worked [47]. Still, this path tracking had some limitations, such as how the truck model could cut or overlook sharp corners, and it could need more precision to keep the truck model in the center of the lanes.

At the truck model there are implemented two driving functions in regards to regulating the speed. The speed could either be set manually at the joystick, or automatic, where the speed is set accordingly to the track from the computer vision based path tracking in the Ackermann drive node. The automatic mode is used in the tests, with a speed of 0.5 m/s at the regular track. When the truck model approaches the last corner before the charging station, a yellow mark has been taped at the track, which the computer vision detects and decreases the speed to 0.4 m/s. This speed is kept at 0.4 m/s at the charging station, and after the truck model has passed the charging station and its following corner, the speed increases to 0.5 m/s. If the speed was increased any higher, it could lead to some inaccurate steering, and these settings gave a consistent level for testing. It is the second PI controller described in section 4.2.3 that is used in the truck model upgraded tests of the truck model.

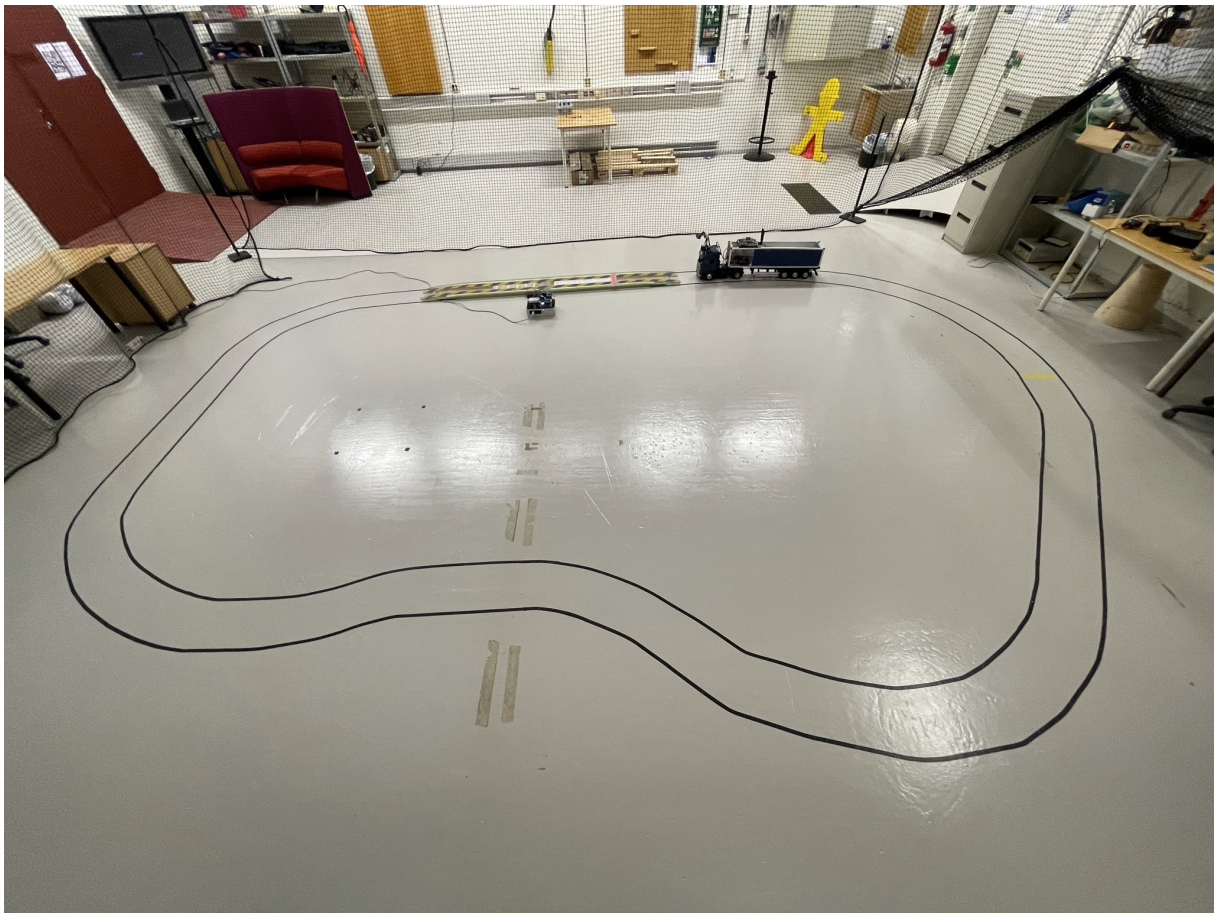
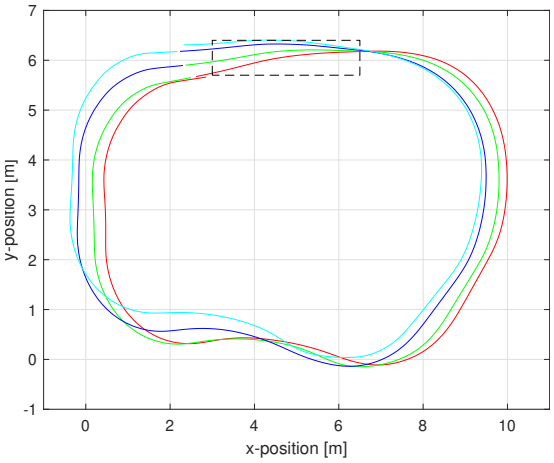
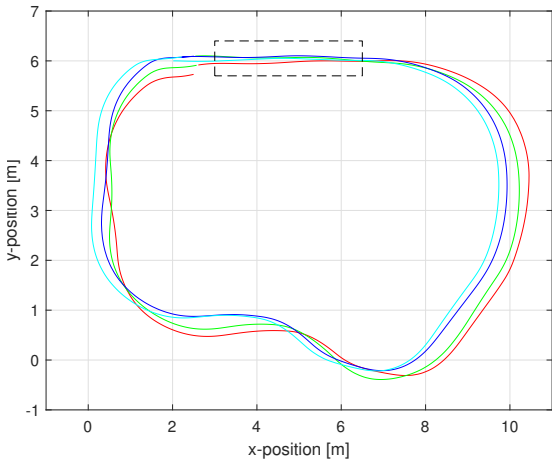


Figure 5.1: Mockup track

First, the truck model's position of the four test runs was logged with the IMU sensor, and the results can be seen in figure 5.2. Comparing the test runs in 5.2a and 5.2b, it is possible to see how the path tracking was improved by implementing the closed-loop controller and two cameras. The problem with cutting and overlooking sharp corners was improved, and two cameras gave a more consistent driving. In the figures, the charging station is marked with black stippled lines to compare the results with the actual photo of the mockup track. At the charging station, it can be detected how the truck model kept a more straight line when it was driving. However, the data from the IMU sensor drift over time, leading to the laps shifting slightly relative to each other, and it is providing an artificial representation of the actual result. Still, it is observable to see the same trend in the laps and how the closed-loop controller and two cameras improved the positioning of the truck model in the tracks.



(a) Path tracking with one camera and open-loop controller



(b) Path tracking with two cameras and PI controller

Figure 5.2: Path tracking test with IMU logging

In the second test, the four test runs position were logged with the LiDAR sensor, and the result can be seen in figure 5.3. The Hector SLAM LiDAR mapping gave a very consistent mapping, where the data had no drift. This made it possible to overlap both of the four test runs for comparison. The four test laps with the open-loop controller and one-camera path tracking are viewed with the colors red, green, yellow, and cyan. This is marked as 1 camera in the figure. The four test laps with the PI controller and two-camera path tracking are shown in the colors magenta, orange, blue and black. This is marked as 2 camera in the figure. From the tests, one can detect with the open-loop controller and one camera path tracking, that there is a deviation of many centimeters where it cuts corners, in comparison to the path tracking with the closed-loop controller and two cameras. There are some noise in the measurements, which made it slightly more challenging to see the effects of the closed-loop controller in the figure. Still, the trend of the laps is almost identical, and it is possible to detect how the improvements of the path tracking gave a better result.

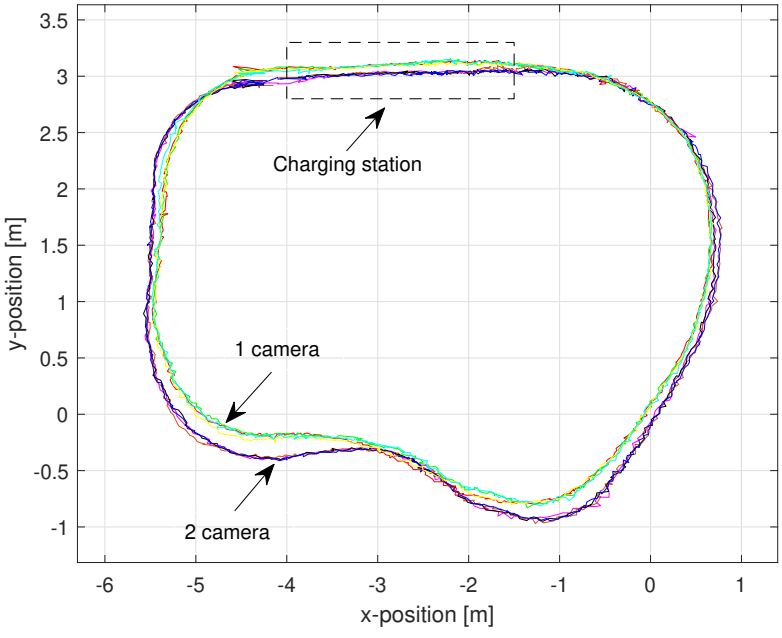


Figure 5.3: Path tracking tests with LiDAR logging

A test of the servo position as a function of time has been logged for one lap in figure 5.4. The logging starts at approximately point (x, y) = (-4.5, 3.0) in figure 5.3. Note that the saturation limit for the steering angle is reached from about 4 to 9 seconds and how the anti-windup for the integral prevents a delay for the process variable. When the servo position is positive, the truck model turns right. A negative servo position value indicates that the truck model turns left.

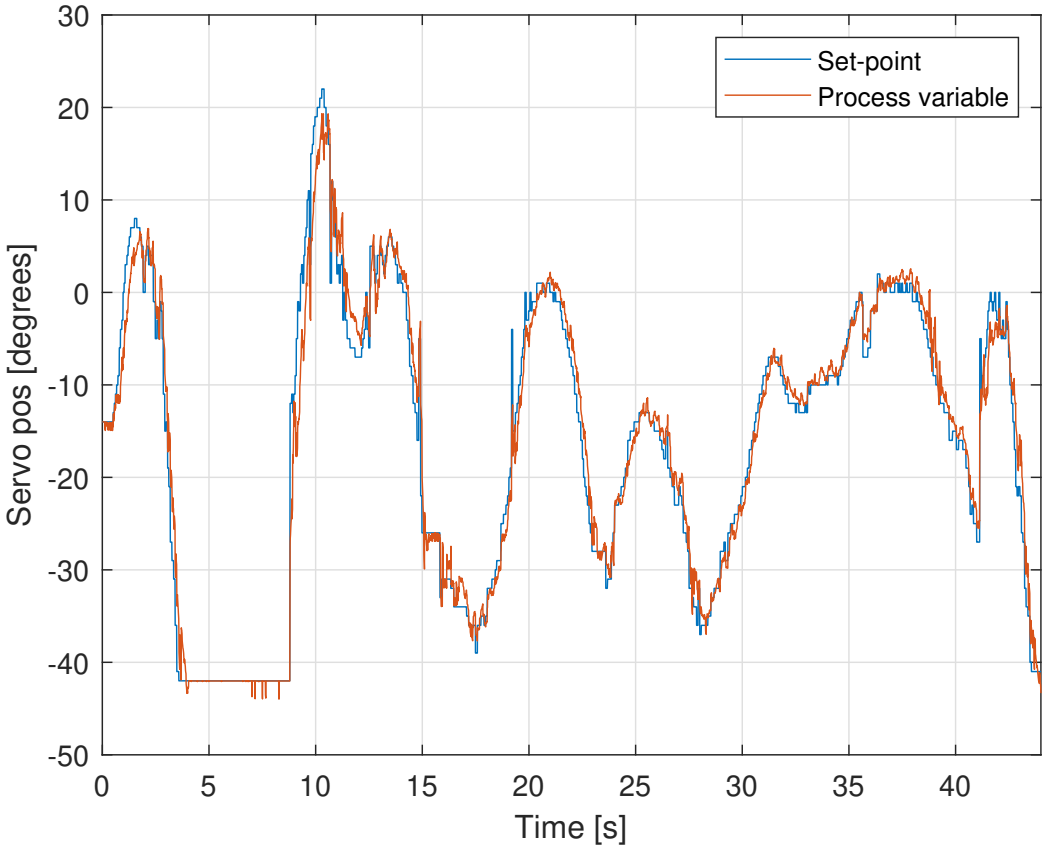


Figure 5.4: Servo position as a function of time in one lap

5.2 Optimizing the dynamic charging cycle

Given that the currents are limited to a safe value, the detuning of the road-side coils in the charging station makes it possible to keep the road-side coils energized at all times due to its low induced currents. Still, keeping the coil energized at idling time will produce unnecessary losses, and it is desired to find the most optimal strategy for energizing the coils. A solution for activation and de-activation of both of the road-side coils at a specific position has been implemented. Five different conditions have been tested to find the optimal solution, where the goal is to find the energy transfer efficiency η_E of each coil, the overall energy transfer efficiency η_{OE}, and which condition could have the maximum transferred energy. The tests will use the null-point condition, as described in section 4.4, as the starting point for the tests. In the first tests, the conditions will activate the road-side coils when the vehicle-side coil is 5 and 10 cm in front of the start null-point, and deactivate 5 and 10 cm, respectively, after the vehicle-side coil has passed the end null-point. Then the null-point condition for activation and deactivation will be tested, before the test conditions will activate the road-side coils when the vehicle-side coil is 5 and 10 cm after the start null-point, and deactivate 5 and 10 cm, respectively, before the vehicle-side coil has passed the end null-point.

The test runs at the charging station are tested with a speed of 0.4 m/s, which in 1:14 scale corresponds to an actual size truck at 5.6 m/s (20 km/h). This was the maximum speed which gave a consistent result, due to the design of the road-side charging station. The road-side charging station consists of a slope before entering the road-side coils. A higher speed than 0.4 m/s could lead to inaccurate steering when entering the charging station, where the alignment of the coils is crucial. The results are shown in energy [J] (1 Joule = 1 Watt · second), where the energy is calculated as the integral over time from the measured load power. Position figures of the charging cycle, which are to scale, are illustrated. The power of the road-side coils is measured directly by a USB connection to the Xilinx Zynq board at the road-side charging station, but the pick-up power at the vehicle-side coil is measured in the Ackermann drive node at the Jetson TX2. In the node, the charging current is multiplied by the battery voltage, where the values are provided from the CAN bus at the Xilinx Zynq board in the vehicle-side of the charging system. This tends to give two different resolutions of measurement. However, both are measured in seconds, which made it possible for a reliable result when adding them together in a plot in Matlab.

5.2.1 Test 10 cm extended activation of the null-point

This first test was at activation/deactivation of the road-side coil when there was one vehicle-side coil width before/after the start/end null-points of the road-side coil, and is illustrated in figure 5.5. When there was an overlap of 1 cm when road-side coil 1 was deactivated and road-side coil 2 was activated, it sometimes caused the charging station to fail to activate/deactivate the coils. Instead, it worked well when they were activated/deactivated at the same time, with respectively 9 cm after the end null-point of road-side coil 1, and 9 cm in front of the start null-point of road-side coil 2, which is presented.

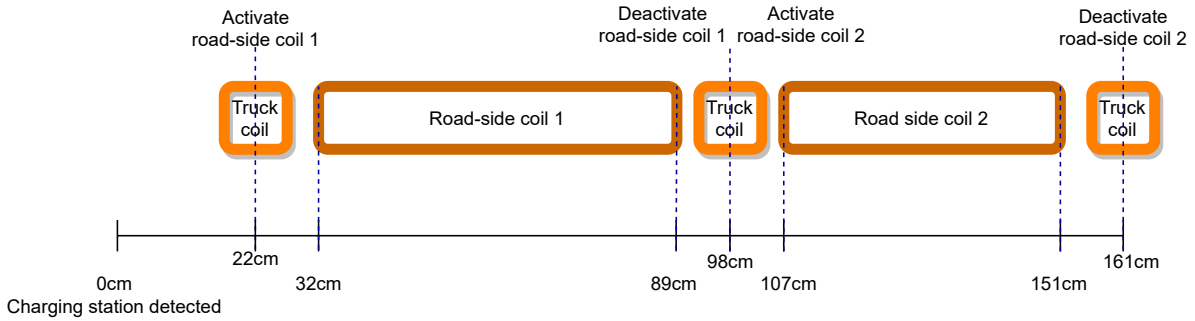


Figure 5.5: 10 cm extended activation

In figure 5.6, the charging cycle of this condition is measured, where the blue line represents the transferred power of road-side coil 1, the red line represents the transferred power of road-side coil 2, and the green line represents the pick-up power from the vehicle-side coil. Note that this color representation is valid for all the next five tests. The coupling coefficient k for the road-side coil 1, in a case of a perfect alignment, is presented in the subplot. In this test, it can be seen that the energy losses at the idling time of the road-side coils occur for about half a second at the start and end of the energizing time of each road-side coil. This is when the vehicle-side coil does not overlap with the road-side coils, resulting in lower efficiency. It can be detected that when the coupling coefficient had a small negative value, the power had a slight increase at road-side coil 1 in the idling time. This was not picked up by the vehicle-side coil because this is interpreted as a negligible coupling. With the early activation of the road-side coils, the coupling coefficient is low when the vehicle-side coil reaches the null-point, as detected from the subplot in the figure. This results in a quick rise of the current at the null-point transition. Thus, the generated power spikes at the start and end null-point of each charging cycle of the road-side coils. The reason for the dips in the power after the power spikes, before the power stabilizes at a slightly higher level, is that the coupling coefficient

reaches its maximum coupling. The road-side coil reaches the maximum coupling at approximately 0.8 seconds and 2 seconds. It was noticed that road-side coil 2 had a power spike at approximately 2.3 seconds due to activation of road-side 1 and deactivation of road-side 2 at the same time, which is not optimal for efficiency.

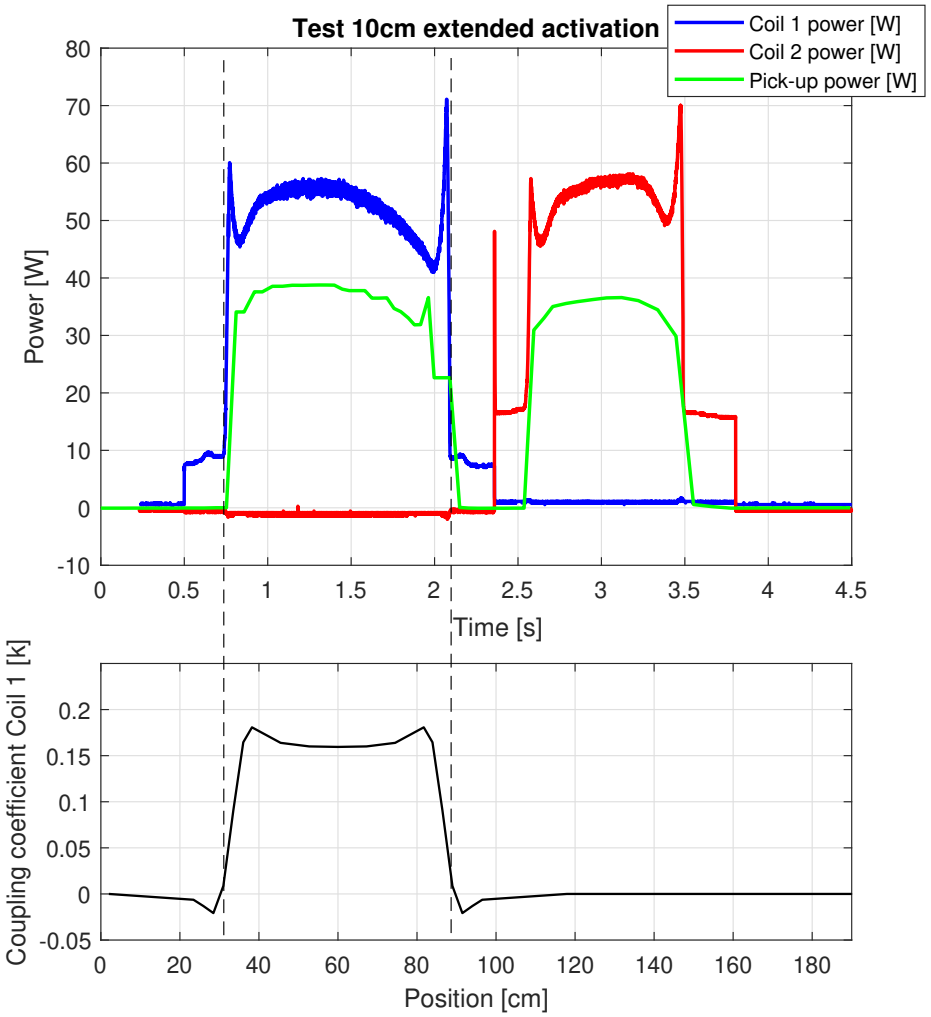


Figure 5.6: Test 10 cm extended activation

This condition gave a low efficiency η_E for each coil, thus, a low overall efficiency η_{OE} of 0.58, as shown in table 5.1. Regardless, the transferred energy picked up by the vehicle-side coil seems acceptable, but a comparison with the next test cases is necessary to conclude.

Table 5.1: Transferred energy and efficiency at 10 cm extended of the null-point

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Overall efficiency
1	76.24	45.64	0.60	0.58
2	57.09	31.51	0.55	

5.2.2 Test 5 cm extended activation of the null-point

The second test condition was with activation/deactivation of the road-side coils where there was a half vehicle-side coil width before/after the start/end null-point of the road-side coils. This is exactly the point where the road-side coil and the vehicle-side coil start to overlap. This is illustrated in figure 5.7.

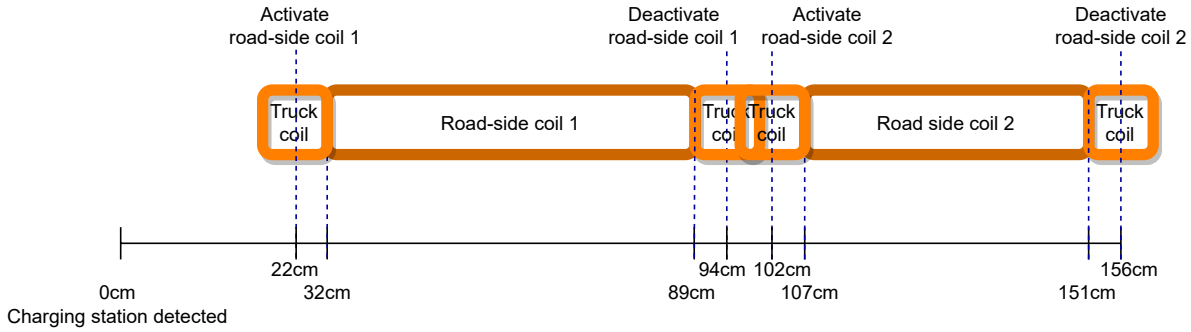


Figure 5.7: 5 cm extended activation

Figure 5.8 shows the charging cycle for this condition, and it can be seen that there are still some energy losses at the road-side coils. This happens at the idling time at the road-side coils for about a quarter of a second at the start and end of the road-side coils' energizing time, resulting in lower efficiency. There is still a low coupling coefficient at the start and end null-points in each of the road-side coils, as in the previous test with 10 cm extended activation of the null-point, which gives the power spikes. This results in energy losses and lower efficiency. The coupling coefficient k conditions for a perfect alignment and will apply for road-side coil 1, as shown in subplot in the previous test case, and is also valid for the rest of the test cases for road-side coil 1. It is noticeable in this test case that the lower transferred energy from the road-side coil at the maximum coupling coefficient at approximately 0.7 seconds and 1.9 seconds gives slightly higher efficiency.

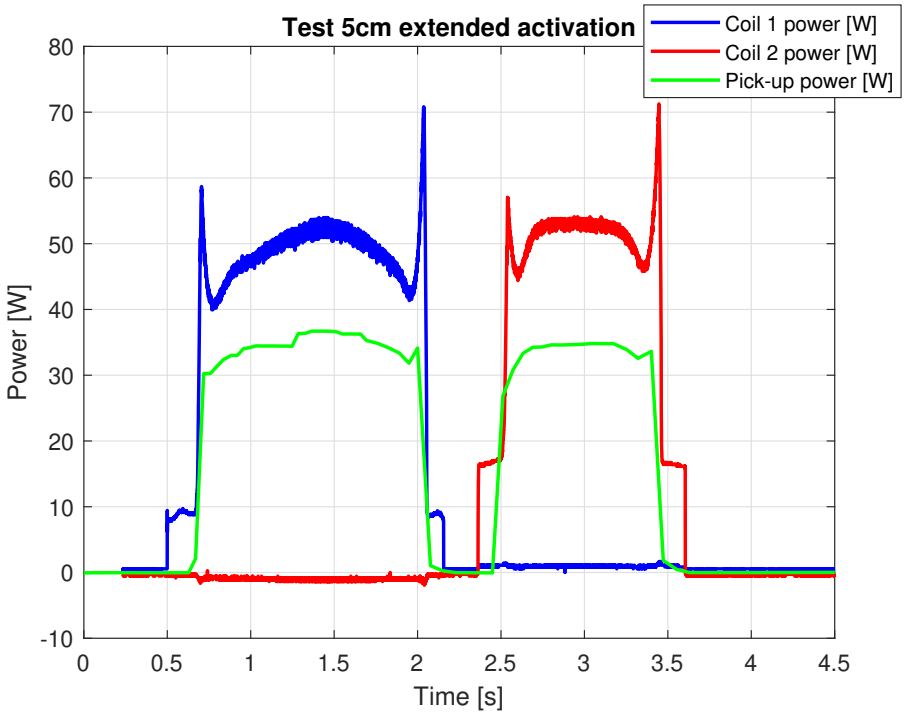


Figure 5.8: Test 5 cm extended activation

From table 5.2, the condition test results indicate that the transferred energy picked up by the vehicle-side coil is at the same level as 10 cm extended activation of the null-point. However, the efficiency of each coil η_E has increased, indicating that the overall efficiency η_{OE} has increased. This is due to the idling time at the start and end of the energizing time of the road-side coils has been halved and the overall efficiency η_{OE} has increased by five percent when compared to the last condition test of 10 cm extended activation of the null-point.

Table 5.2: Transferred energy and efficiency at 5 cm extended of the null-point

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Overall efficiency
1	71.53	46.41	0.65	0.63
2	51.65	31.37	0.61	

5.2.3 Test of null-point activation

The third test was the condition of an activation/deactivation of the road-side coils at the null-point of the road-side coils, and this condition is illustrated in figure 5.9. This is the exact point where the vehicle-side coil starts/stop receiving power from the road-side coils, as discovered with the coupling coefficient k.

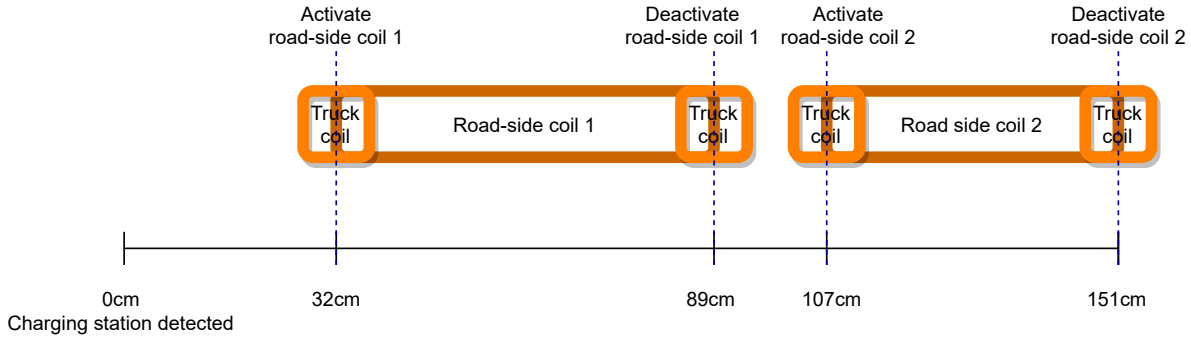


Figure 5.9: Null-point activation

The results of the charging cycle at the null-point activation/deactivation condition is shown in figure 5.10. Since this is the exact activation point were the road-side coils starts to transfer power to the vehicle-side coil, one can see that the idling time before the activation and after the deactivation of the road-side coils are avoided. Still, there is a low coupling at the start and end null-points at the road-side coils, which results in the power spikes due to the high current at a low coupling coefficient.

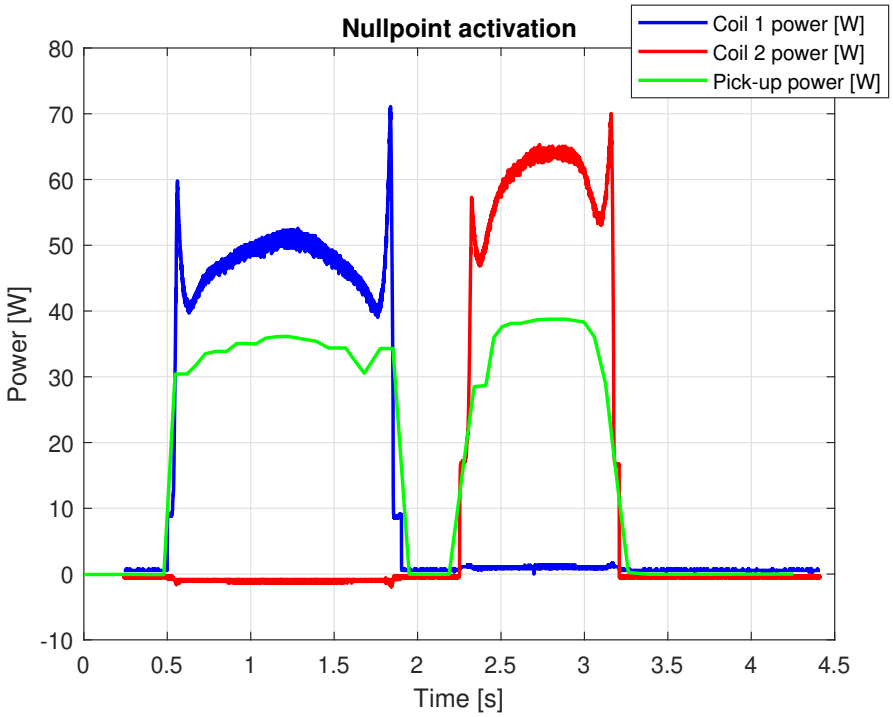


Figure 5.10: Test null-point activation

Table 5.3 shows the results for this condition, and now the efficiency η_E and overall efficiency η_{OE} have increased by a couple of percentage points from the previous test cases. This is due to the elimination of energized idling time before/after the null-points at the road-side coils. Regardless, the transferred energy rates are at approximately the same values as the previous test cases with activation/deactivation extended of the null-point.

Table 5.3: Transferred energy and efficiency at the null-point

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Overall efficiency
1	65.46	44.49	0.68	0.66
2	51.07	32.26	0.63	

5.2.4 Test 5 cm shorter activation of the null-point

The fourth test condition is an activation/deactivation of the road-side coils when the vehicle-side coil and the road-side coil are precisely at the start/end alignment of each other, which is illustrated in figure 5.11. This corresponds to half a vehicle-side coil width shorter than the null-points. The coupling coefficient indicates that this position is the approximate location with the maximum coupling coefficient.

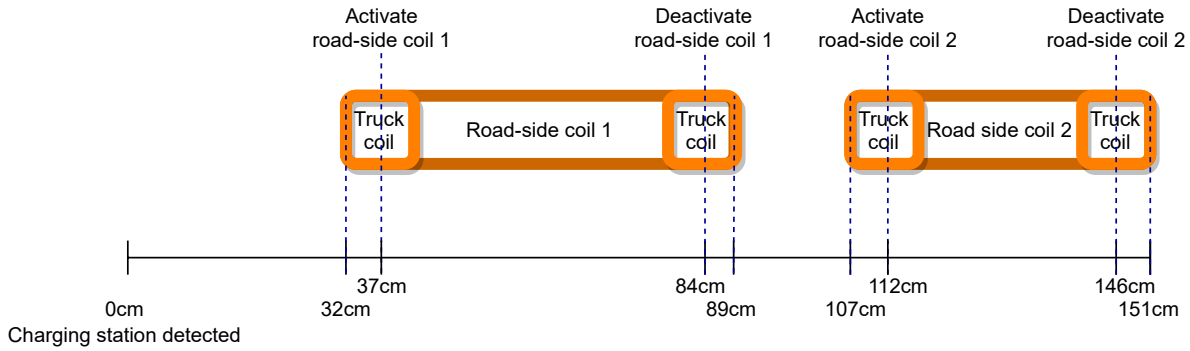


Figure 5.11: 5 cm shorter activation

Figure 5.12 shows this test condition, and the first thing noticed was that the power spikes at the start/end null-points of the road-side coils were almost avoided. Now, the road-side coils transferred power stay almost at the same level when they are energized, with the exceptions at the maximum coupling where the power levels from the road-side coil are at their minimum during the energizing time.

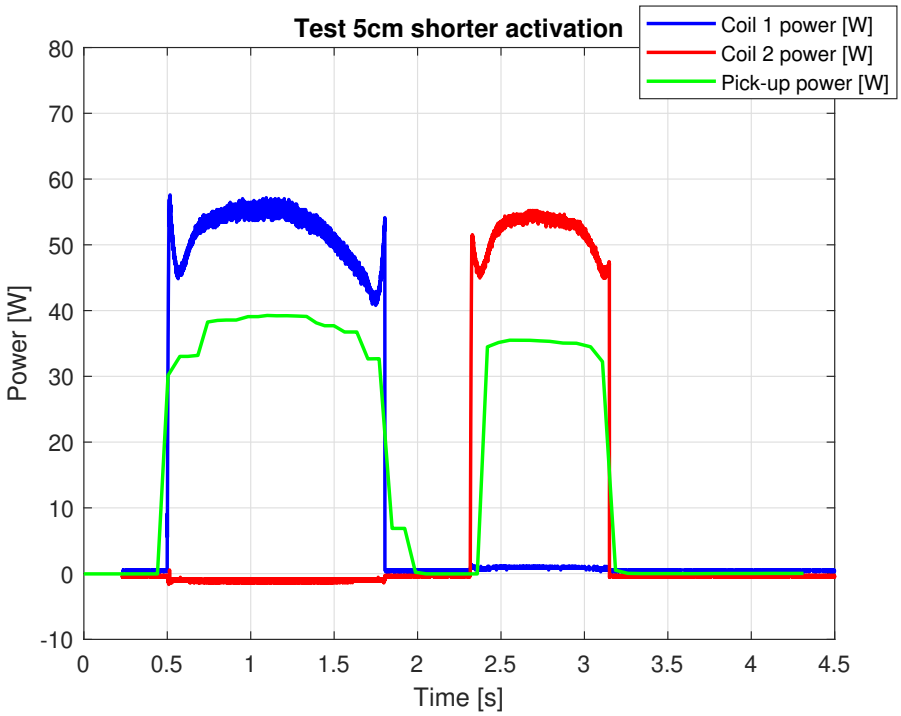


Figure 5.12: Test 5 cm shorter activation

In table 5.4, the results from this test condition is shown. From this, it can be inspected that this condition has a three percent higher efficiency η_E at the road-side coil 1, one percent lower efficiency η_E at the road-side coil 2, but one percent increased overall efficiency η_{OE} from the previous test

case with the null-point activation. The slight increased overall efficiency η_{OE} is a result of the power spikes that are avoided at the start/end null-points at the road-side coils in this test condition. It can be seen that the road-side coil 1 transferred energy picked up by the vehicle-side coil is still about the same level as the previous test conditions, although there is a combined length of 10 cm where transferred power could have occurred. The vehicle-side coil pick-up energy from the road-side coil 2 has decreased, as one could expect from a 10 cm shorter distance where transferred power from a road-side coil could have occurred. An explanation for this could be that the road-side coil 2 is a low-cost coil, with a lower quality factor, or the impact small deviations in the alignment of the road-side coil and vehicle-side coil could have.

Table 5.4: Transferred energy and efficiency at 5 cm shorter of the null-point

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Overall efficiency
1	69.54	49.09	0.71	0.67
2	41.03	25.43	0.62	

5.2.5 Test 10 cm shorter activation of the null-point

The fifth test is the test condition with an activation/deactivation of the road-side coils at one vehicle-side coil width shorter than the null-points of the road-side coils, and is illustrated in figure 5.13. This is the point where the road-side coils have an even coupling factor, as discovered from the coupling coefficient k.

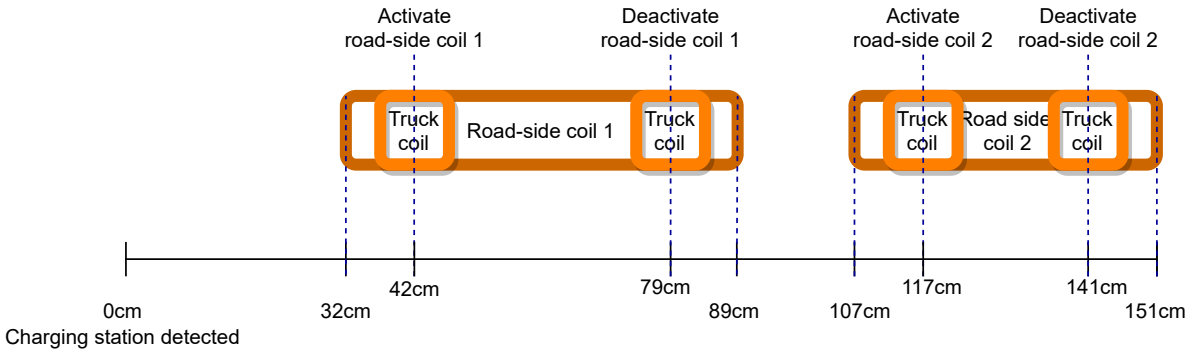


Figure 5.13: 10 cm shorter activation

In figure 5.14 this test condition is shown, and one can see that the power spikes are completely avoided at the start and end at the energizing of the road-side coils. It can be detected how the power levels are almost proportional between the coils due to the same coupling coefficient.

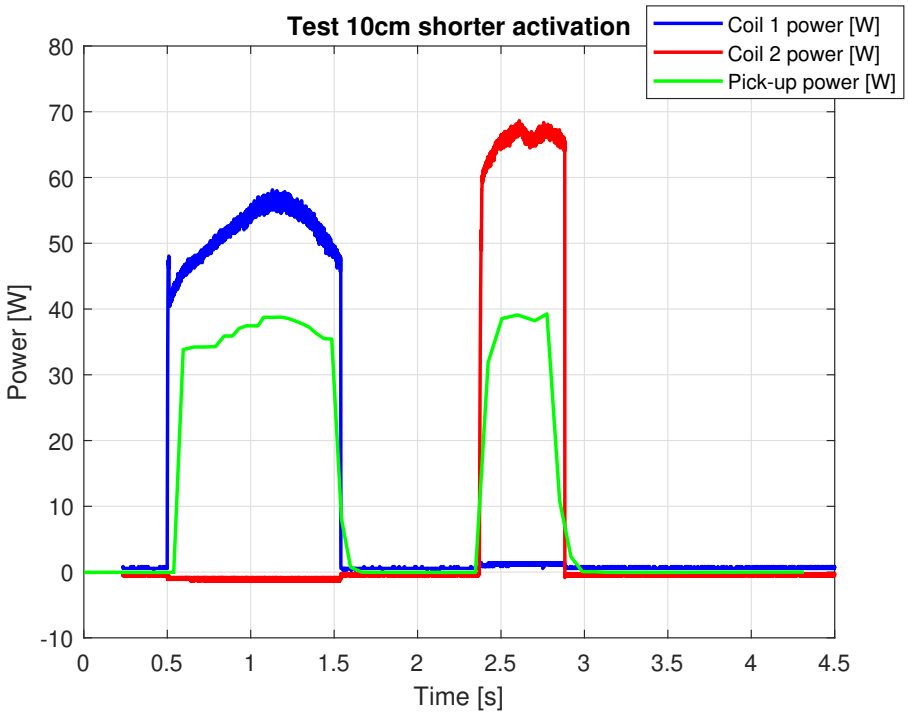


Figure 5.14: Test 10 cm shorter activation

Table 5.5 presents the results from this test condition. It can be inspected that this condition gave the lowest efficiency η_E for the road-side coil 2 and thus, the lowest overall efficiency η_{OE} of all the test cases. Since there is a limited area where the transferred power now occurred, it resulted in the lowest pick-up energy at the vehicle-side coil compared to the other test cases.

Table 5.5: Transferred energy and efficiency at 10 cm shorter of the null-point

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Overall efficiency
1	53.74	33.80	0.60	0.57
2	31.06	15.27	0.51	

5.2.6 Summary of the dynamic charging cycle tests

For comparison, a summary of the transferred energy of the previous dynamic charging cycle tests is presented in table 5.8 and total transferred energy is shown in figure 5.15. The total pick-up energy at the vehicle-side coil was about the same level in the first four test cases, with early activation of the road-side coils, at the null-point activation, and the 5 cm shorter activation. However, there was a more significant energy difference between the road-side coils in the fourth test case with a 5 cm shorter activation than in the previous cases. From the road-side coil 1, the vehicle-side coil picked up the highest transferred energy of all the test cases. The pick-up energy from the road-side coil 2 had decreased, as one could expect from both road-side coils with a shorter energizing area. Then, in total, this case resulted in total transferred energy levels at the same level as the three previous cases with a wider area of transferring power. This could have a connection with a more precise alignment with the road-side coil 1 and the vehicle-side coil in this particular test case, or the different design of the road-side coils. The last test case with a 10 cm shorter activation had the lowest transferred and picked up energy due to the shorter energizing area. From the total transferred energy presented in figure 5.15, one can observe the rapid decreasing values for this test case.

Table 5.6: Summary of the transferred energy of the charging cycle tests

Activation	Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Total road-side energy [J]	Total vehicle-side pick-up energy [J]
10 cm extended	1	76.24	45.64	133.33	77.15
	2	57.09	31.51		
5 cm extended	1	71.53	46.41	123.18	77.78
	2	51.65	31.37		
Null point	1	65.46	44.49	116.53	76.75
	2	51.07	32.26		
5 cm shorter	1	69.54	49.09	110.57	74.52
	2	41.03	25.43		
10 cm shorter	1	53.74	33.80	87.02	49.60
	2	31.06	15.27		

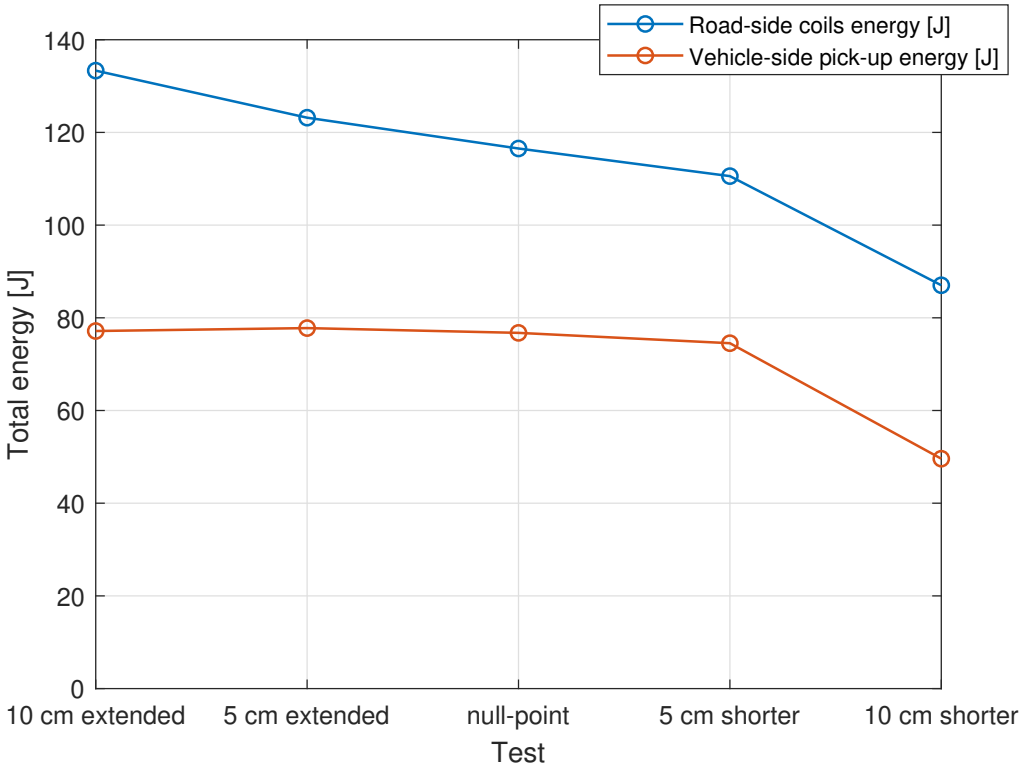


Figure 5.15: Summary total transferred energy

A summary of the efficiency of the previous dynamic charging cycle tests is presented in table 5.7 and a bar graph of the overall efficiency is shown in figure 5.16. In the test case with the 5 cm shorter activation of the null-point, the highest overall efficiency was obtained. Regardless, it was only one percentage higher than the null-point activation. The highest efficiency from road-side coil 1 was acquired at the 5 cm shorter activation of the null-point, and the null-point activation had the highest efficiency for road-side coil 2. As presented in section 3.2 with the different designs of the road-side coils, the road-side coil 1 proved to be more efficient due to its design with more copper for the winding and high-grade ferrites. The road-side coil 1 had an average higher efficiency of 6.4 percent, compared to the road-side coil 2.

Table 5.7: Summary of the efficiency of the charging cycle tests

Activation	Coil	Efficiency	Overall efficiency
10 cm extended	1	0.60	0.58
	2	0.55	
5 cm extended	1	0.65	0.63
	2	0.61	
Null point	1	0.68	0.66
	2	0.63	
5 cm shorter	1	0.71	0.67
	2	0.62	
10 cm shorter	1	0.60	0.57
	2	0.51	

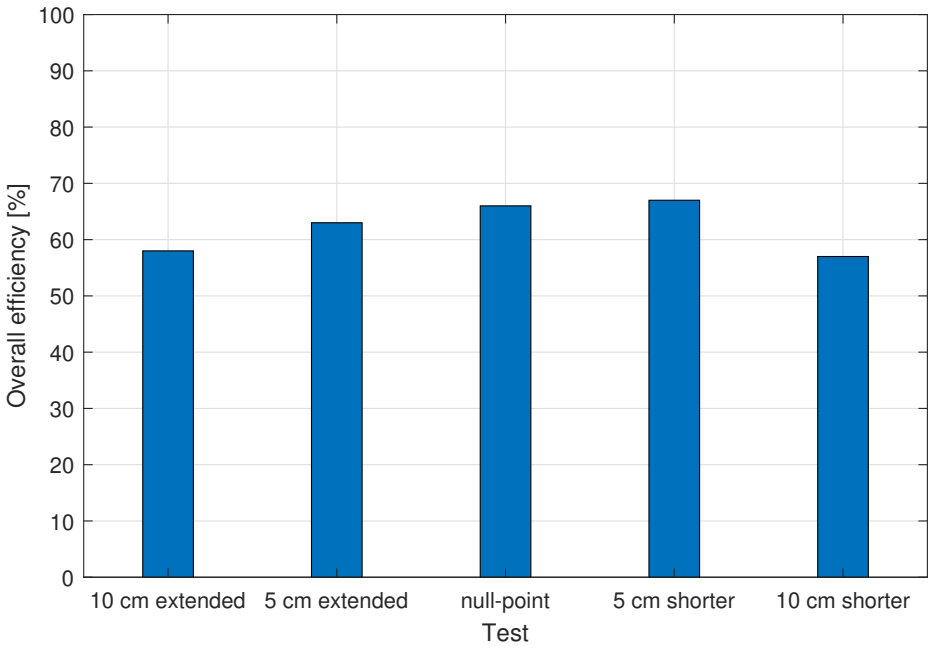


Figure 5.16: Summary total overall efficiency

5.2.7 Reliability test of 5 cm shorter activation of the null-point

The 5 cm shorter activation test of the null-point had a more significant energy variation than the other test cases. A reliability test was then performed to check this condition's consistency, regarding the vehicle-side coil pick-up energy levels from the road-side coils, and whether the previously obtained results were repeatable. The consistency test was performed with five consecutive laps around the track with the same 5 cm shorter activation settings, and the result can be seen in figure 5.17. It used a different color representation than the previous test cases to make it easier to follow the power transfer in one lap. The same color represents one lap in the graph, with the road-side coils power and vehicle-side coil pick-up power. Still, the first part of the graph represents the power of road-side coil 1 and pick-up power from the vehicle-side coil, and the last part of the graph is the power of the road-side coil 2 and pick-up power from the vehicle-side coil.

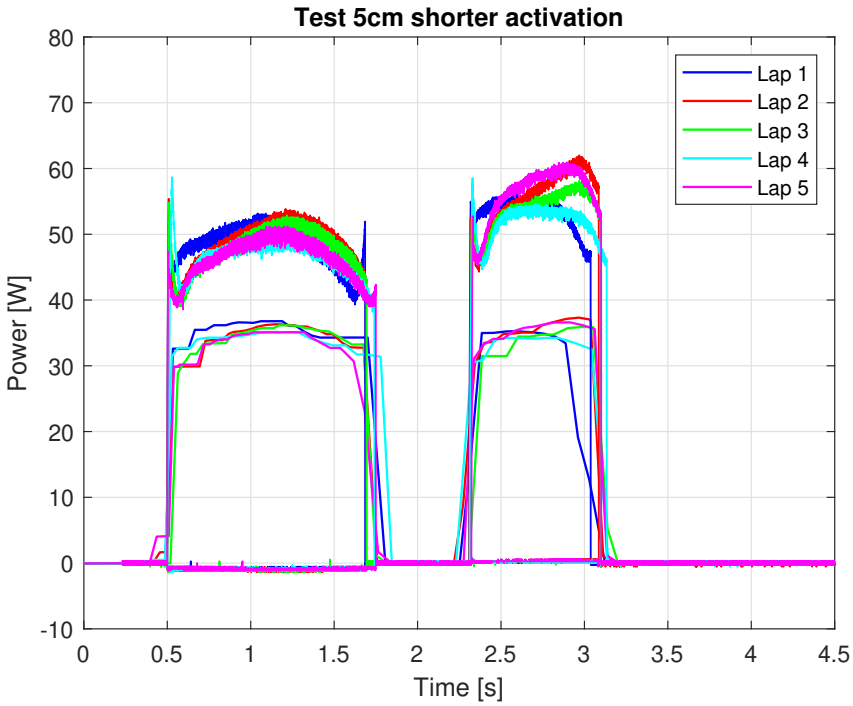


Figure 5.17: Five consecutive laps of 5 cm less

In figure 5.17, one can observe about the same trend for each lap at the charging cycle at the road-side coil 1. The largest variation of the transferred power can be detected at the power values at the end of road-side coil 2. The variation was not detectable from a visual inspection of the test laps, and one can also see how much of a difference even a small misalignment of the coils could impact the transferred power.

Table 5.8: Transferred energy and efficiency of five consecutive laps

Lap	Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Overall efficiency
1	1	57.63	43.76	0.76	0.70
	2	37.17	22.78	0.61	
2	1	58.40	41.03	0.70	0.69
	2	41.25	27.65	0.67	
3	1	57.64	39.51	0.69	0.65
	2	40.23	24.49	0.61	
4	1	58.01	42.27	0.73	0.67
	2	40.60	24.08	0.59	
5	1	58.46	39.70	0.68	0.66
	2	42.05	26.94	0.64	

Table 5.8 presents the charging cycle results from the five reliability test laps. From these results, the average of the reliability test cases is presented in table 5.9.

Table 5.9: Average transferred energy and efficiency of the five consecutive laps

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Total road-side energy [J]	Total vehicle-side pick-up energy [J]	Overall efficiency
1	58.03	41.26	0.71	98.28	66.44	0.67
2	40.26	25.19	0.63			

The results from the single test case are presented in table 5.10, from section 5.2.4, which is the test case that the reliability test is set to compare.

Table 5.10: Transferred energy and efficiency of the single test

Coil	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency	Total Road-side energy [J]	Total vehicle-side pick-up energy [J]	Overall efficiency
1	69.54	49.09	0.71	110.57	74.52	0.67
2	41.03	25.43	0.62			

When comparing the results from tables 5.9 and 5.10 the energy levels are almost identical at the road-side coil 2. The same applies when comparing both efficiency η_E and overall efficiency η_{OE}. However, the author of this study hypothesizes that the vehicle-side coil pick-up energy levels from road-side 1 in the single test was too high for an average test case. A statistical hypothesis test is set up to prove the hypothesis. For this, a one-sample t-test is performed, which is a test that compares the mean of samples to a hypothesized value, and this is a parametric test. A parametric tests assumes a normal distribution of the values [63]. The null hypothesis H_0 is; the vehicle-side coil has the same pick-up energy value on average compared with the test cases with a wider road-side coil energizing area. The alternative hypothesis H_a is; the vehicle-side coil has a lower pick-up energy value on average compared with the test cases with a wider road-side coil energizing area. The test uses μ_0 of 49.09 J as the representative for the null hypothesis value, which is found in table 5.10 at the vehicle-side pick-up energy from road-side coil 1. So, the null hypothesis H_0 and alternative H_a hypothesis can be stated as:

$H_0 : \mu_0 = 49.09\,J$
$H_a : \mu_0 < 49.09\,J$

A level of statistical significance has to be chosen, where a conventional choice of significance level often is $\alpha = 0.05$ (5 percent). The significance level is a term that describes the probability of a result as just a result of chance. A t-test is performed, which could be used with 30 samples or less, and the null hypothesis does not have a standard deviation. The formula for the t-test is:

$$t = \frac{\bar{x} - \mu_0}{\sigma / \sqrt{N}} \tag{5.1}$$

$\bar{x}$ is the mean of the vehicle-side pick up energy from road-side coil 1, and is found in table 5.9. N is the number of samples. The standard deviation σ are the values' average distance x_i from the mean $\bar{x}$ and N samples. The formula for the standard deviation σ is:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{5.2}$$

The standard deviation of the vehicle-side pick-up energy from road-side coil 1 was calculated in Matlab with five samples to $\sigma = 1.79$ percent. This gives a t-test of:

$$t = \frac{41.26 - 49.09}{1.79/\sqrt{5}} = -9.780 \tag{5.3}$$

The hypothesis test is a one-tailed test, which indicates that the alternative hypothesis H_a is only at one side of the null hypothesis H_0. A confidence level has to be looked up in a table [64], with 4 degrees of freedom and a significant level of 0.05. The degrees of freedom are the number of samples, minus one. From the table, the confidence level is 2.132. Since the hypothesis test checks if the alternative hypothesis is lower than the null hypothesis, the one-tailed confidence level is negative. The null hypothesis can be rejected if the t-value is lower than the confidence level [63]. In this hypothesis test, the results are:

-9.780 < -2.132

Since the t-test has a value lower than the confidence level, the null hypothesis H_0 is rejected at significance level α and thus accepts the alternative hypothesis H_a.

However, more tests should be performed for a more reliable standard deviation to achieve a more complementary normal distribution. Still, the reliable test with five test laps was found sufficient for this study, where the goal was to check the consistency of the test condition and the hypothesis. A conclusion could be drawn that the single test was a test case with a very precise alignment of the road-side coil 1 and the vehicle-side coil and the average are lower. The reliability test gave a more expected result with decreasing energy levels at a smaller road-side coil 1 energizing area.

In figure 5.18, the total transferred energy summary figure is updated, where the average of the reliability 5 cm shorter activation test is added. It can be seen how the updated lines from the reliability test gives a more representative result of the test cases. This is because the vehicle-side coil pick-up energy starts to decline in the test cases with a shorter energizing area, and that the total transferred energy from the road-side coils has a more linear decreasing line. Since the efficiency η_E and the overall efficiency η_{OE} had the same values between the reliability and single test case, figure 5.16 still represents the overall efficiency from this updated graph.

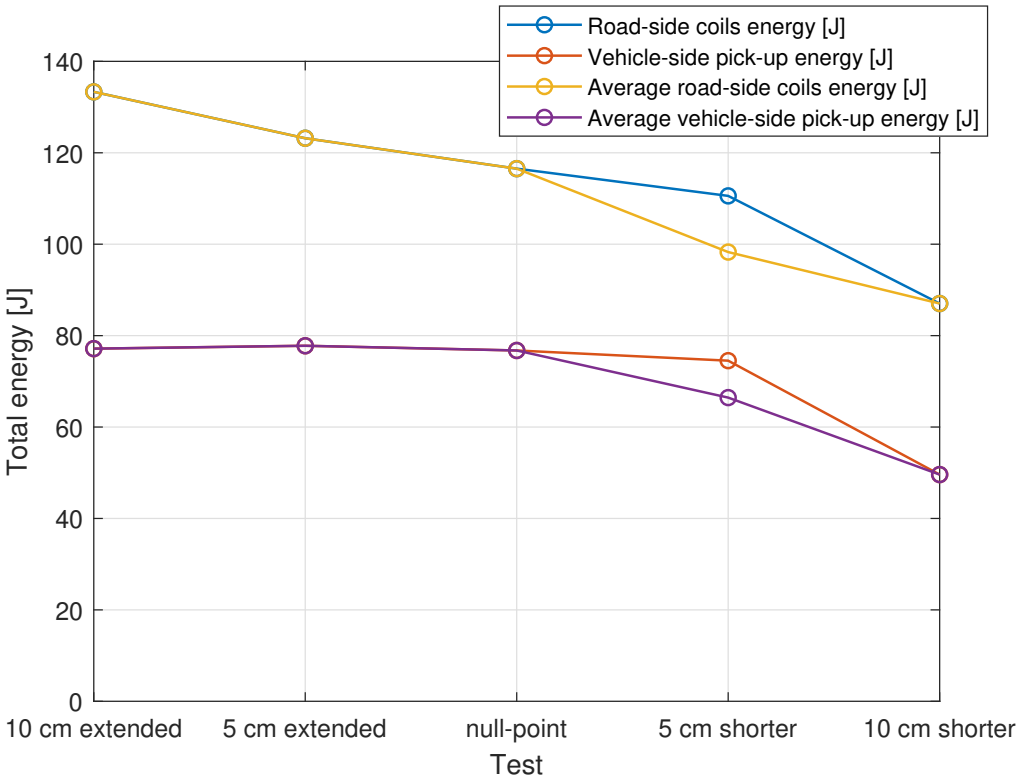


Figure 5.18: Updated summary of total transferred energy

5.3 Test opportunity charging

The rectangular road-coil 2 was exchanged with the square road-side coil, which was a coil with the same specifications as the vehicle-side coil. Now it could be tested how the efficiency of the system responded with a transmitting and receiving coil at the same size, where the losses could be kept at a minimum. The opportunity charging cycle was tested with two approaches. The first approach was a quasistatic charging, where the road-side coil was activated at the start null-point and deactivated at the end null-point. This allowed the truck model to charge as it moved into and out of position to align the road-side and vehicle side coils. The second approach was a static charging, where the road-side and vehicle-side coil was aligned, before the road-side coil was activated. Then the road-side coil was deactivated before the truck model moved. Both of the test sequences were tested with a four second cycle. The square road-side coil was designed for a supply of a maximum of 10 V, which results in a lower power level of transferred power than the rectangular road-side coils. The rectangular road-side coils had a supply of 12 V. However, the square road-side coil was designed with higher quality and more copper than the rectangular coils, which means that a higher efficiency should be achieved.

5.3.1 Quasistatic charging

The truck model speed at the charging cycle was decreased to a low speed of 0.25 m/s when moving over the square road-side coil, and the low speed made it more sufficient to stop and position the truck model at the desired location. A position figure in 5.19 shows the position of the null-point activation and deactivation of the road-side coil. In the figure, the vehicle-side coil is not illustrated due to the lack of space with a short coil. The vehicle-side coil is activated at the center of the dotted lines.

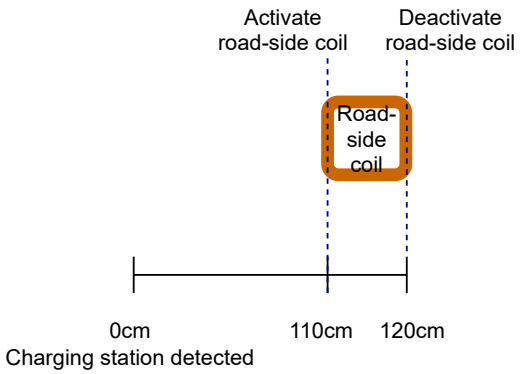


Figure 5.19: Quasistatic activation

The charging cycle test, with the quasistatic charging condition, can be seen in figure 5.20. In the figure, the road-side coil power spikes at the dynamic approach are noticeable, and they occur at a low coupling coefficient. The charging cycle test lasted for 4 seconds, where 3.3 seconds was when the truck model was static at a stationary charging position. This indicates that the truck model used 0.7 seconds from when the road-side coil was activated to move in and out of the static position, which is referred to as the dynamic charging section in this test case.

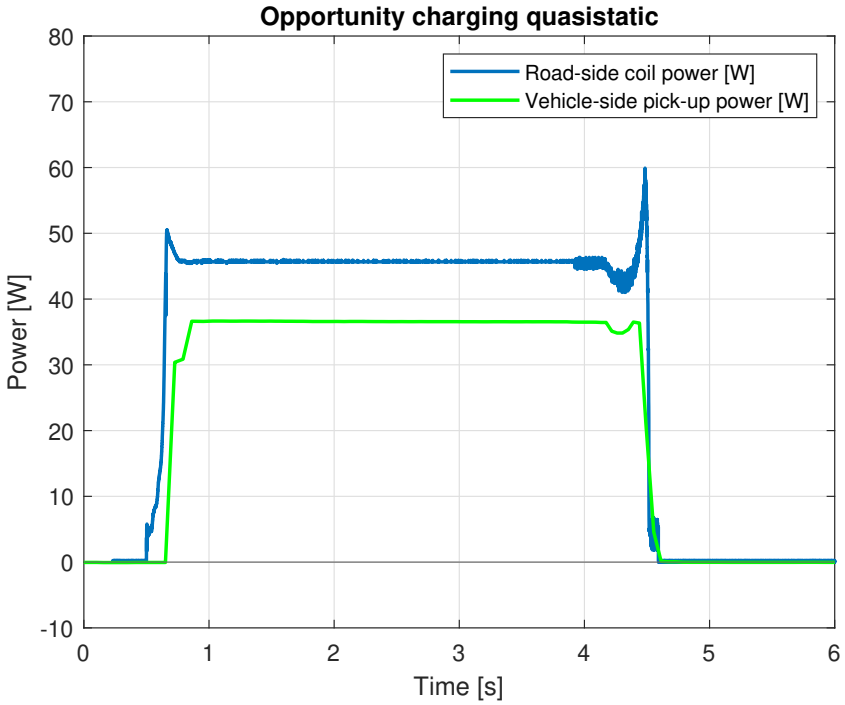


Figure 5.20: Test quasistatic activation

The results are listed in table 5.11, where the results have been divided into three sections. The static section is from 0.85 to 4.15 seconds, and the dynamic section is from 0.5 to 0.85 seconds and 4.15 to 4.5 seconds. The total section is both of the dynamic and static section combined to the total quasistatic section. A significant difference in the efficiency between the dynamic and the static section can be noticed, where the static section had a 22 percent higher efficiency. These sections end up with a 77 percent combined efficiency because most of the charging cycle condition occurred at the static section.

Table 5.11: Transferred energy and efficiency at quasistatic opportunity charging

Section	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency
Static	151.21	121.12	0.80
Dynamic	27.83	16.25	0.58
Total	179.04	137.37	0.77

With the 4 second quasistatic test, where the static section was 3.3 second, the dynamic section increased the total vehicle-side pick-up energy by 11.8 percent from just the static section. This will then vary on the length as a function of time at the static section. Since the static section power levels are continuously linear, a function of the increased efficiency of a dynamic approach is calculated. The instantaneous power of the static section is:

$$\frac{121.12\,J}{3.3\,s} = 36.70\,J/s \tag{5.4}$$

From this, the function of the increased efficiency from the dynamic section to a static section, in a quasistatic solution, could be declared. With 16.25 J as the fixed value of gained dynamic energy at the dynamic section, the function can be stated as:

$$f(t) = \frac{16.25}{36.70 \cdot t + 16.25} \tag{5.5}$$

The function shows how much extra efficiency one could gain at a dynamic approach to the road-side coil, rather than energizing the road-side coil when the vehicle-side coil is at a static position. The function is plotted in figure 5.21 with Matlab, and it is shown in the time interval from 0 to 30 seconds, as could e.g. be the time interval in an assumed stop at a red light.

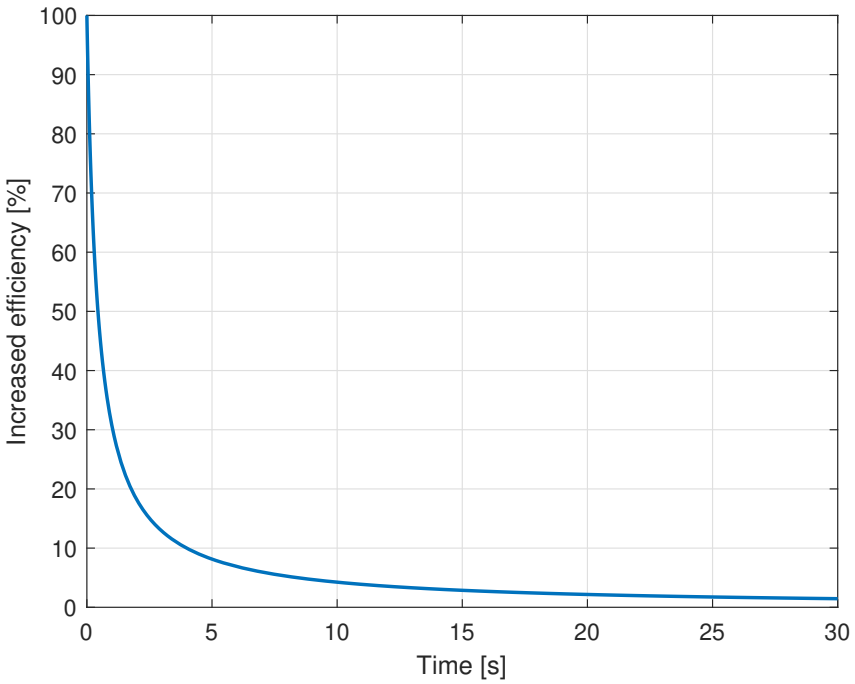


Figure 5.21: Increased total efficiency with a dynamic approach as a function of time of the static stop time

With the t variable as seconds, the increased dynamic effect could be inspected. By, e.g., a 10-second static section, the dynamic approach increased the efficiency of the total pick-up energy from the vehicle-side coil by 4.2 percent. At the maximum of the example, a 30-second static section, the total pick-up energy from the vehicle-side coil was increased by 1.45 percent.

5.3.2 Static charging

At the static opportunity charging test, the vehicle-side coil position itself at the perfect alignment of the square road-side coil, and then the road-side coil was activated for 4 seconds. After the 4 seconds, the road-side coil was deactivated before the truck model moved. The position is illustrated in figure 5.22. This illustration does not show the vehicle-side coil due to the lack of space with a short coil. Still, the vehicle-side coil is activated at the center of the dotted line.

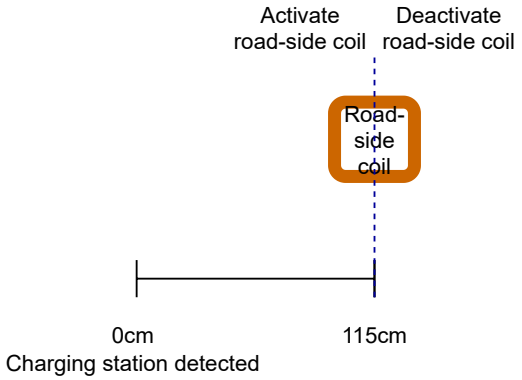


Figure 5.22: Static activation

Figure 5.23 shows the charging cycle for this condition. The figure shows a very consistent charging condition, where both the road-side and vehicle-side power levels are completely flat. From this test, there is only one section, namely the total energy of the static charging cycle.

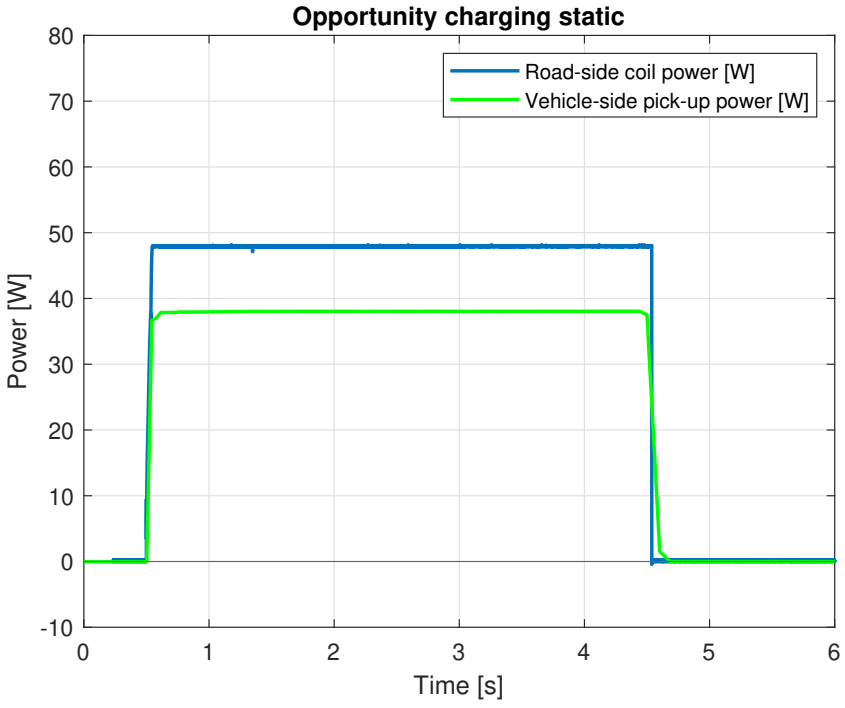


Figure 5.23: Test static activation

In table 5.12, the results are listed for this static condition. Here, one can see an efficiency of 80 percent for the entire charging cycle. This is the same result as in the previous quasistatic test case in the static section. Since this test had an efficiency of 80 percent throughout the entire charging cycle, the vehicle-side pick-up energy was at a higher overall level than the previous test case.

Table 5.12: Transferred energy and efficiency at static opportunity charging

Section	Road-side energy [J]	Vehicle-side pick-up energy [J]	Efficiency
Total	191.98	153.17	0.80

From these opportunity test approaches, it can be summarized that a static approach would give a reasonable higher efficiency than a quasistatic approach to the road-side coil, which has a significantly lower efficiency at the dynamic section. However, if the main goal is to reach the highest pick-up energy, the best solution would be the quasistatic solution, but this comes at the cost of lower efficiency. The highest overall efficiency from the dynamic charging tests was at 67 percent. However, if a static opportunity solution was utilized instead, a 13 percent higher efficiency could be reached. Yet, if the option of dynamic charging is discarded, there is no distance traveled while charging.

5.4 Scientific paper for evaluating the results

A scientific paper for the IECON conference is under planning by the author of this study, Associate Professor Jon Are Suul, and Research Scientist Dr. Giuseppe Guidi. This paper is an evaluation of the most important results from experimental testing of the dynamic wireless charging system for the autonomous truck model in this study. An draft of the paper can be seen in appendix A.

Chapter 6

Discussion

In this chapter, general observations and improvements regarding methods, implementations, results, etc. in the previous chapters are discussed.

6.1 Hardware and software

The small-scale truck model is provided by SINTEF, included all the hardware specs. The truck model is an off-the-shelf toy truck, which has been used for testing and demonstrations, both by SINTEF and previous students at NTNU. This presented a number of challenges due to wear and tear, like backlash and parts breaking. The transmissions from the servos to the gearbox and to the axle of the front wheels are all made of plastic. During testing, both of the transmissions broke in half, which meant that the author had to spend time getting spare parts and maintaining the truck model's mechanical condition. Several wires were worn off, and troubleshooting had to be done to solder the wires in place. Although the backlash in the steering was improved in the pre-study for this study, by mounting a washer to minimize front axle play and tighten the joints connecting the steering system [47], some backlash did accrue. If future testing is to be performed, a more precise and robust driving platform is recommended.

The embedded computer Nvidia Jetson TX2 and the microcontroller PJRC Teensy 3.2 were very power-efficient computers, and fully charged batteries often sustained a full day of testing. Still, it is imperative to monitor the battery voltage so that the voltage does not drop so low that the batteries could take damage. The propulsion battery voltage is measured directly at the Xilinx Zynq board in the trailer and monitored in the command line while operating. It is alerted when the battery voltage is too low and then the truck model stopped. For the control electronics battery, there was no voltage monitoring. Thus, a low-cost voltmeter was installed for visual surveillance. This solution works well when one is aware of the situation and regularly checks the battery voltage level. However, the system could be expanded with direct measurement of the control electronics battery condition, with feedback to the system for a control of the voltage.

The hall effect sensor utilized as feedback for the speed of the truck model is triggered by magnets glued to an adapter at the driveshaft. This is to measure the speed and distance traveled. Although the distance between the magnets is not perfectly aligned and thus creates some ripple at the feedback signal, it does not affect the PI controller's speed regulation. Still, the author of this study does not consider this as a long-term solution, due to the fact that the glue could wear off over time. A permanent solution could be to install a separate wheel-encoder to measure the feedback or even exchange the propulsion DC motor with a DC motor with an embedded wheel-encoder.

For future development and testing of the system, the ARM (Advanced Reduced instruction set computer Machines) version of Ubuntu, which this system runs at, is easy to work with. Although the ROS

could be a hazel to learn for newcomers, its tools and libraries enable a very flexible system, making further development uncomplicated. Also, the OpenCV library is instrumental in an application such as this system.

6.2 Truck model upgrades and implementations

The internal potentiometer in the servo was used as a feedback signal to utilize a closed-loop PI controller. This signal had much noise, which meant that a filter had to be made. The filter caused a slight time delay, but was sufficient for the tests performed in this study. If a higher test speed of the truck model is desired, alternative sensors for measuring the steering position could be reviewed. In this study, an ultrasonic sensor and a slide potentiometer were considered for this purpose, but there was not enough space on the truck model to mount these sensors. If a new driving platform is obtained, these improvements are recommended.

The new camera was mounted on a tripod on the top of the cab to get the desired overview of a blind spot in front of the truck model. However, this caused the camera to be more mechanically exposed for external disturbances when mounted on a tripod with several adjustable joints. A quick fix in this study was to tape all the tripod joints so that that the camera could keep the same position. Nonetheless, minor adjustments had to be performed during the testing period to keep the camera perfectly aligned. In the initial plan for the camera placement, a bracket was made for the position in the front bumper that kept a steady position of the camera. If future testing is to be performed, a camera with a wide-angle could be considered so it could be mounted at the front bumper bracket. Alternatively, a mechanical platform could replace the tripod due to the advantage of there not being any adjustable joints on the mechanical devise.

In the Ackermann bicycle model used to determine the steering angle of the path tracking controller, the trailer geometry is not taken into account. This was considered beyond the scope of this study, as the wheels of the trailer should have the ability to turn if this was to be considered. For the small-scale truck model used in this study, such a solution was not possible. With a new driving platform, or if larger-scale testing is performed, this is recommended for a more comprehensive test of the system.

6.3 Testing of autonomous driving and the charging cycle

The LiDAR was used in this study for obstacle detection and mapping. However, it is a low-cost LiDAR with a limited range. The testing area utilized in this study was an open and monotone space, and when the LiDAR was used for mapping, it could lose track of the truck model's position. This was solved by placing various objects in the room, but the objects had to be placed so that the computer vision based path tracking was not affected, due to the computer vision using colors for path tracking. Nonetheless, these challenges are not a problem if one is aware of the situation, but a better LiDAR sensor could be considered for this purpose.

The LiDAR gave a much more consistent logging of the position of the truck model, where the position logged with the IMU drifted over time. However, if the implemented LiDAR obstacle detection function was to be used, the LiDAR had to be tilted at a sharper angle to detect in front of the truck model. This was because the LiDAR could only detect in a 2D space. This caused the option to either use the LiDAR in a tilted position for obstacle detection or to place it horizontally for logging the position. If these functions are to be used simultaneously, the 2D LiDAR could be exchanged with a 3D LiDAR. A 3D LiDAR is an expensive sensor, thus the impact has to be assessed against the cost.

In the charging cycle graphs in the optimizing test cases of the charging system, the measurements are performed at two different platforms. In these graphs' visual representation figures, the power from the road-side coils and the pick-up power from the vehicle-side coil are manually aligned in Matlab. Still, this does not give an artificial representation of the results, where both platforms measure the power in seconds. The only aspect to consider is that there may be a slight misalignment of the visual result. Regardless, the result presented in the tables is consistent, where the energy is calculated as the integral of the charging cycles.

The coupling coefficient obtained in a FEM study [35] was only analyzed for the road-side coil 1. This could have been done for the road-side coil 2 and the square road-side coil. In this study, the data and the tools to perform such an analysis were not available. The tests for finding the optimal charging cycle were performed once for each test case, and the single liability test was performed with five test laps with the same test conditions. A liability test for each test condition could create a more consistent result, and even with more than five test laps for a liability test, as done in this study. This could create a more comprehensive normal distribution of the test cases, thus, their consistency could be tested with a more precise standard deviation.

Chapter 7

Conclusion and future work

This thesis aimed to further a more stable and robust autonomous driving system for a small-scale electric truck model, and to test for the most optimal wireless inductive charging solution. A wireless inductive charging system can charge the truck model dynamic or static at a charging station. The truck model has a square coil for receiving power. For the dynamic approach, the charging station consists of two rectangular road-side coils for transferring energy to a moving truck model. For the static approach, the charging station has a square road-side coil, with the same specifications as the vehicle-side coil, to transfer power to a stationary truck model. An improved autonomous driving system could lead to a more precise positioning of the truck model when operating. Thus, the truck model could achieve more efficient wireless inductive charging. This chapter is the conclusion of this thesis and future work is proposed.

7.1 Conclusions

The hardware of the truck model has been upgraded. Firstly, feedback of the servo for the steering angle position was implemented. This was done by connecting to the internal potentiometer at the servo. The steering angle feedback made it possible to create a closed-loop controller, and in this study, a discrete PI controller was utilized. At a test track, the controller resulted in more precise steering, and enhanced the truck model positioning to the center of the lane. Furthermore, a new camera has been mounted and implemented, and the images from the new camera were merged with the images from an existing camera inside the truck model's cab. With the merged camera images, the detection area in front of the truck model was expanded from a detection area of 30 - 100 cm to 0 - 100 cm. A computer vision based path tracking controller was proceedingly implemented. The center points in the lane were calculated from the merged image, and a new path tracking controller was put into effect. An Ackermann bicycle model was utilized for the geometry of the controller. This model obtains the controllers steering angle output from the measured geometry of the truck model and the center points of the lanes. The new path tracking controller resulted in improved and more precise steering, where sharp corners could previously be overlooked and cut. The truck model position has been logged with both an IMU and a LiDAR sensor to prove these results. It was concluded that the logging with a LiDAR sensor gave a more consistent result, because the IMU sensor tends to drift over time when it is logging the position. These results showed the upgrades' effects, and it concluded that these improvements gave a more stable and robust autonomous driving.

New distance measurement for the truck model at the charging station has been implemented, where a distance stamp is set when the charging station is detected. This gave the opportunity to activate and deactivate the road-side coils at the charging station at an exact position, and thus, find an optimal charging cycle. A series of tests could then be performed with the new implementations and improved autonomous operation to find an optimal dynamic charging cycle. A null-point was used as a starting point for the tests, which is at the exact position where the vehicle-side coil start to receive

power from an energized road-side coil. Five different tests were performed, with activation before, at, and after the null-point. With activation before, and at the null-point the highest transferred energy was achieved. From the before, and at null-point tests, the null-point activation resulted in the highest efficiency of 66 percent. However, the overall highest efficiency was reached at activation 5 cm after the null point at 67 percent, but at the cost of less transferred energy. The conclusion from these tests is that if maximum transferred energy is desired, null-point activation is the best solution. If the highest efficiency is the goal, then an activation 5 cm after the null-point is the best solution. The two rectangular road-side coils at the charging station have a different design to illustrate the flexibility of the design. The first coil is a high-end coil designed for high efficiency with more copper and high-grade ferrites. The second coil is a low-cost coil, with a budget construction. The conclusion from the different designs was that the high-end coil had an average of 6.4 percent higher efficiency, compared to the low-cost coil.

One of the rectangular road-side coils was exchanged with the square road-side coil to test an opportunity charging solution. Two approaches for the opportunity charging were tested; a quasistatic and a static condition. In the quasistatic test, the road-side coil was activated at its null-point. In the static test, the truck model was stationary before the road-side coil was activated. The static test resulted in the highest efficiency, where 80 percent of the transferred energy from the road-side coil was received at the vehicle-side coil. However, in the quasistatic test, the static section reached the same efficiency as in the static test, with 80 percent efficiency. The overall efficiency was lower because the dynamic section had an efficiency of 57 percent. Due to these aspects, the conclusion is, given that maximum transferred energy is desired, the quasistatic opportunity solution is the best choice. If the highest efficiency is preferred, then the static solution is optimal. The static opportunity charging solution resulted in 13 percent higher efficiency than the most efficient dynamic charging solution. However, this comes at the cost of zero distance traveled while charging.

The software utilized in this study is furthered from previous work from students at NTNU. Rather than rewriting the entire software code, it was expanded, altered, and rewritten where it was found necessary. The code is mainly written in Python in an embedded Linux environment, and the system is based on ROS and OpenCV. Protocols for a real-scale solution have been investigated, specifically the IoT 4.0 protocols MQTT and DDS have been looked into. Both protocols are used in the automotive industry, with a publish/subscriber architecture with nodes and topics, much like the ROS used in this study. The main difference is that the MQTT protocol communication is routed through a broker on a centralized server like a cloud. The DDS protocol is a decentralized architecture, where it establishes communication directly through nodes, and only runs to a data center if needed. Thus, it is concluded that the DDS protocol is the best protocol for a real-scale solution.

7.2 Future work

Although the goals set for the study were completed, with a fully functional system for autonomous driving and wireless inductive power transfer for a small-scale truck model, alternatives and improvements can be explored. A more advanced controller can be implemented, such as LQR (Linear–quadratic regulator) and MPC (Model predictive control). Modeling for such a controller has been investigated and presented in appendix B. Different autonomous navigation techniques, such as machine learning with artificial intelligence in deep learning and neural networks, could also be tested. Such techniques require high computational power. If so, an upgraded embedded computer and a larger battery capacity should then be considered.

As discussed in the previous chapter, hardware upgrades such as the 2D LiDAR could be upgraded with a 3D LiDAR. A wheel-encoder could be installed for the velocity feedback, and other sensors could be utilized for the steering angle feedback. New sensors could also be implemented, such as a

radar for autonomous driving and ultrasonic sensors for obstacle detection or parking options. More extensive testing could be performed, with a reliability test for all the conditions. This could give a more consistent result for an optimal charging cycle. Full-scale testing could be investigated and tested with the DDS protocol for the system architecture. A user manual has been made in appendix C for future work and testing.

Bibliography

[1] Sintef, *Elingo - electrification of heavy freight transport*, 2018. [Online]. Available: https://www.sintef.no/projectweb/elingo/english/.

[2] G. Guidi, *Small-scale model of inductive charging system for long-haul trucks*, 2018. [Online]. Available: https://www.sintef.no/globalassets/project/elingo/18-0733-rapport-8-memo-small-scale-model-til-nett.pdf.

[3] G. Guidi, J. A. Suul, F. Jenset and I. Sørfonn, *Wireless charging for ships high-power inductive charging for battery electric and plug-in hybrid vessels.* 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8025701.

[4] J. Poliscanova, *Electric cars*, 2021. [Online]. Available: https://www.transportenvironment.org/what-we-do/electric-cars.

[5] SSB, *Transport står for 30 prosent av klimautslippene i norge*, 2019. [Online]. Available: https://www.ssb.no/natur-og-miljo/artikler-og-publikasjoner/transport-star-for-30-prosent-av-klimautslippene-i-norge.

[6] G. Guidi, A. M. Lekkas, J. E. Stranden and J. A. Suul, 'Dynamic wireless charging of autonomous vehicles,' 2019. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2675278/Guidi2019dwc_akseptert.pdf?sequence=2&isAllowed=y.

[7] SSB, *Bilparken*, 2021. [Online]. Available: https://www.ssb.no/transport-og-reiseliv/landtransport/statistikk/bilparken.

[8] N. Plikk and O. H. Johansen, *Naf: - stor mangel på elbilladere*, 2021. [Online]. Available: https://www.tek.no/nyheter/nyhet/i/7KBlV8/naf-stor-mangel-paa-elbilladere.

[9] SSB, *Elbiler og allment tilgjengelige ladepunkt*, 2021. [Online]. Available: https://www.ssb.no/transport-og-reiseliv/landtransport/artikler-og-publikasjoner/19-elbiler-per-ladepunkt/tabell-1.elbiler-og-allment-tilgjengelige-ladepunkt.personbiler.hele-landet-og-utvalgte-kommuner.2020-2018-og-2016.

[10] P. Davidson and A. Spinoulas, 'Autonomous vehicles -what could this mean for the future of transport?,' jun. 2015. [Online]. Available: http://transposition.com.au/papers/AutonomousVehicles.pdf.

[11] J. Kocić, N. Jovičić and V. Drndarević, *Sensors and sensor fusion in autonomous vehicles*, 2018. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8612054.

[12] SAE INTERNATIONAL, *Automated driving*, 2014. [Online]. Available: https://web.archive.org/web/20180701034327/https://cdn.oemoffhighway.com/files/base/acbm/ooh/document/2016/03/automated_driving.pdf.

[13] Lovdata, *Lov om utprøving av selvkjørende kjøretøy*, 2017. [Online]. Available: https://lovdata.no/dokument/NL/lov/2017-12-15-112?q=Lov%5C%20om%5C%20utpr%5C%C3%5C%B8ving%5C%20av%5C%20selvkj%5C%C3%5C%B8rende.

[14] P. Tarpley and S. D. Jansma, *Autonomous vehicles: The legal landscape in the us*, 2016. [Online]. Available: https://www.nortonrosefulbright.com/en-af/knowledge/publications/2951f5ce/autonomous-vehicles-the-legal-landscape-in-the-us.

[15] Statens Vegvesen, *Første internasjonale regelverk for automatiserte kjøretøy på plass*, 2020. [Online]. Available: https://www.vegvesen.no/om+statens+vegvesen/presse/nyheter/nasjonalt/forste-internasjonale-regelverk-for-automatiserte-kjoretoy-pa-plass.

[16] T. Arendt and S. O. Klunge, *Første internasjonale regelverk for automatiserte kjøretøy på plass*, 2019. [Online]. Available: https://www.cw.no/artikkel/it-juss/selvkjorende-kjoretoy-hvem-har-ansvaret.

[17] S. Ornes, *How to hack a self-driving car*, 2020. [Online]. Available: https://physicsworld.com/a/how-to-hack-a-self-driving-car/.

[18] Regjeringen, *Lov om utprøving av selvkjørende kjøretøy*, 2017. [Online]. Available: https://www.regjeringen.no/no/dokumenter/prop.-152-l-20162017/id2556972/?ch=3.

[19] Ford Media Center, *Ford conducts industry-first snow tests of autonomous vehicles – further accelerating development program*, 2016. [Online]. Available: https://media.ford.com/content/fordmedia/fna/us/en/news/2016/01/11/ford-conducts-industry-first-snow-tests-of-autonomous-vehicles.htmll.

[20] V. H. Milan Sonka and R. Boyle, *Image Processing, Analysis, and Machine Vision*. 2008, ISBN: 978-1-133-59360-7.

[21] U. G. Manzo, H. Chiroma, N. Aljojo, S. Abubakar, S. I. Popoola and M. A. Al-Garadi, *A survey on deep learning for steering angle prediction in autonomous vehicles*, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/914183?casa_token=zCLPzaRGOW8AAAAA:NljdFAJHfEo4nL0BCQtBI7EM4Da9UyaXtbiQiJInnwYoL0PZYWRJHh2kpEOzjyBme3X5Hy0V1Q.

[22] K. R. Velasco, *Yolo (you only look once)*, 2019. [Online]. Available: https://towardsdatascience.com/yolo-you-only-look-once-17f9280a47b0.

[23] N. Karlsson, E. D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian and M. E. Munich, *The vslam algorithm for robust localization and mapping*, 2005. [Online]. Available: https://www.researchgate.net/publication/221073651_The_vSLAM_Algorithm_for_Robust_Localization_and_Mapping.

[24] S. A. Bagloee, M. Tavana, M. Asadi and T. Oliver, *Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies*, 2016. [Online]. Available: https://link.springer.com/article/10.1007/s40534-016-0117-3.

[25] Waymo, *Technology*, 2021. [Online]. Available: https://waymo.com/tech/.

[26] E. Ackerman, *This year, autonomous trucks will take to the road with no one on board*, 2021. [Online]. Available: https://spectrum.ieee.org/transportation/self-driving/this-year-autonomous-trucks-will-take-to-the-road-with-no-one-on-board.

[27] Teska, *Autopilot*, 2021. [Online]. Available: https://www.tesla.com/no_NO/autopilot.

[28] K. Korosec, *Uber's self-driving trucks division is dead, long live uber self-driving cars*, 2018. [Online]. Available: https://techcrunch.com/2018/07/30/ubers-self-driving-trucks-division-is-dead-long-live-uber-self-driving-cars/?guccounter=1&guce_referrer=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnLw&guce_referrer_sig=AQAAAG-DPhbL_37Wxd8vEm12e%20KsMgqilIYGPvZWROdBBoRDLLRhLynRCKHAQHO_AAOoUYiy7fFnsLTrJZtwsdGeTG3IoFHmFM3_Fqx%20OmStfks-cDbMYudh7R1WqRuxTmi8cIIafmkQz_oI0nAbeOqkIYQst_JqNLip-Yrb6cWomDsC40.

[29] M. Mitolo, F. Freschi and V. Cirimele, *Inductive power transfer for automotive applications: State-of-the-art and future trends*, 2016. [Online]. Available: https://www.researchgate.net/publication/308929434_Inductive_power_transfer_for_automotive_applications_State-of-the-art_and_future_trends/link/59b930e20f7e9bc4ca3d628c/download.

[30] E. Afjei and M. Haerinia, *Resonant inductive coupling as a potential means for wireless power transfer to printed spiral coil*, 2016. [Online]. Available: https://www.researchgate.net/publication/309251050_Resonant_Inductive_Coupling_as_a_Potential_Means_for_Wireless_Power_Transfer_to_Printed_Spiral_Coil.

[31] C. T. Rim, *Wireless power transfer systems for roadway-powered electric vehicles*, 2014. [Online]. Available: https://tec.ieee.org/newsletter/september-october-2014/wireless-power-transfer-systems-for-roadway-powered-electric-vehicles.

[32] F. J. Márquez-Fernánde, J. Bischoff, G. Domingues-Olavarría and M. Alaküla, *Assessment of future ev charging infrastructure scenarios for long-distance transport in sweden*, 2021. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9374483.

[33] SAE, *Wireless power transfer for light-duty plug-in/electric vehicles and alignment methodology*, 2021. [Online]. Available: https://www.sae.org/standards/content/j2954_202010/.

[34] G. Guidi and J. A. Suul, *Technology for dynamic on-road power transfer to electric vehicles*, 2018. [Online]. Available: https://www.sintef.no/globalassets/project/elingo/18-0733-rapport-3-technology-for-dynamic-on-road-6-til-nett.pdf.

[35] G. Guidi and J. A. Suul, 'Transient control of dynamic inductive ev charging and impact on energy efficiency when passing a roadside coil section,' [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2582010/Guidi2018tco.pdf?sequence=1&isAllowed=y.

[36] G. Guidi, *Small-scale model of inductive charging system for long-haul trucks*, 2018. [Online]. Available: https://www.sintef.no/globalassets/project/elingo/18-0733-rapport-8-memo-small-scale-model-til-nett.pdf.

[37] SAE INTERNATIONAL, *Wireless power transfer for light-duty plug-in/electric vehicles and alignment methodology*, 2020. [Online]. Available: https://www.sae.org/standards/content/j2954_202010/.

[38] A. Khalaf, 'Optimization of transfer efficiency of a dynamic wireless charging system for electric vehicles,' M.S. thesis, 2020.

[39] S. Ding, W. Niu and W. Gu, *Lateral misalignment tolerant wireless power transfer with a tumbler mechanism*, 2019. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8808864.

[40] J. A. Suul and G. Guidi, *Technology for dynamic on-road power transfer to electric vehicles*, 2018. [Online]. Available: https://www.sintef.no/globalassets/project/elingo/18-0733-rapport-3-technology-for-dynamic-on-road-6-til-nett.pdf.

[41] I. Technology, *Wireless opportunity charging buses in madrid*, 2020. [Online]. Available: https://ipt-technology.com/case-opportunity-charging-madrid/.

[42] etra, *Inductive charging system*, 2020. [Online]. Available: http://www.grupoetra.com/en/inductive-charging-system/.

[43] Fabric, *Feasibility analysis and development of on-road charging solutions for future electric vehicles*, 2021. [Online]. Available: http://www.fabric-project.eu/www.fabric-project.eu/index07c5.html?option=com_k2&view=itemlist&layout=category&task=category&id=24&Itemid=214.

[44] J. E. Stranden, 'Autonomous driving of a small-scale electric truck model with dynamic wireless charging,' M.S. thesis, 2019.

[45] Nvidia Developer, *Harness ai at the edge with the jetson tx2 developer kit*, 2019. [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2-developer-kit.

[46] PJRC, *Teensy 3.2 3.1*, 2021. [Online]. Available: https://www.pjrc.com/store/teensy32.html.

[47] M. Jørgensen, *Integration of autonomous driving and wireless charging for a small-scale electric truck model*, 2020.

[48] Ros.org, *Ros introduction*, 2021. [Online]. Available: http://wiki.ros.org/ROS/Introduction.

[49] OpenCV, *Introduction*, 2021. [Online]. Available: https://docs.opencv.org/master/d1/dfb/intro.html.

[50] B. DOC, *Rfcomm with ts 07.10*, 2012. [Online]. Available: https://www.bluetooth.com/specifications/specs/rfcomm-1-2/.

[51] SSH.COM, *Ssh protocol*, 2021. [Online]. Available: https://www.ssh.com/ssh/protocol/.

[52] MQTT, *Mqtt: The standard for iot messaging*, 2021. [Online]. Available: https://mqtt.org/.

[53] D. Foundation, *What is dds?* 2021. [Online]. Available: https://www.dds-foundation.org/what-is-dds-3/.

[54] D. Foundation, *Who's using dds?* 2021. [Online]. Available: https://www.dds-foundation.org/who-is-using-dds-2/.

[55] MQTT, *Use cases*, 2021. [Online]. Available: https://mqtt.org/use-cases/#automotive.

[56] H. O. Bansal, R. Sharma and P. R. Shreeraman, 'Pid controller tuning techniques: A review,' 2012. [Online]. Available: https://www.researchgate.net/profile/Hari-Bansal/publication/316990192_PID_Controller_Tuning_Techniques_A_Review/links/591c3972a6fdcc701fd2d92d/PID-Controller-Tuning-Techniques-A-Review.pdf.

[57] B. R. Copeland, 'The design of pid controllers using ziegler nichols tuning,' 2008. [Online]. Available: http://educypedia.karadimov.info/library/Ziegler_Nichols.pdf.

[58] F. Haugen, 'Discretization of simulator, filter, and pid controller,' 2012. [Online]. Available: https://www.mic-journal.no/PDF/ref/Haugen2010.pdf.

[59] Atmel, *Discrete pid controller on tinyavr and megaavr devices*, 2016. [Online]. Available: http://ww1.microchip.com/downloads/en/Appnotes/Atmel-2558-Discrete-PID-Controller-on-tinyAVR-and-megaAVR_ApplicationNote_AVR221.pdf.

[60] D. V. Youbin Peng and R. Hanus, *Anti-windup, bumples, and conditioned transfer techniques for pid controllers*, 1996. [Online]. Available: https://www.researchgate.net/publication/3206463_Antiwindup_Bumpless_and_Conditioned_Transfer_Techniques_for_PID_Controllers.

[61] X.-l. Li, J.-G. Park and H.-B. Shin, *Comparison and evaluation of anti-windup pi controllers*, 2011. [Online]. Available: https://www.researchgate.net/publication/263434823_Comparison_and_Evaluation_of_Anti-Windup_PI_Controllers.

[62] J. Sprinkle, J. M. Eklund, H. Gonzalez, E. Grotli, P. Sanketi and M. Moser, *Recovering models of a four-wheel vehicle using vehicular system data*, 2008. [Online]. Available: https://www.researchgate.net/figure/The-Ackermann-Model-with-a-simplified-bicycle-model-used-for-control-and-predictive_fig1_228914156.

[63] K. S. University, *Spss tutorials: One sample t test*, 2021. [Online]. Available: https://libguides.library.kent.edu/SPSS/OneSampletTest.

[64] B. B. Gerstman, *T table*, 2007. [Online]. Available: https://www.sjsu.edu/faculty/gerstman/StatPrimer/t-table.pdf.

Appendix A

Scientific paper

A scientific paper for the IECON conference is under planning by the author of this study, Mats Jørgensen, Associate Professor Jon Are Suul, and Research Scientist Dr. Giuseppe Guidi. This paper presents the most important results from experimental testing of the dynamic wireless charging system for the autonomous truck model in this study. An draft of the paper can be seen on the next page.

Evaluation of Energy Transfer Efficiency and Path-Tracking Performance for an Autonomous Truck Model with Dynamic Wireless Charging

Mats Jørgensen[1], Giuseppe Guidi[2], Jon Are Suul [1,2]
[1] Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway
[2] SINTEF Energy Research, Trondheim, Norway
E-mail: matjo@stud.ntnu.no, Giuseppe.Guidi@sintef.no, jon.are.suul@ntnu.no

***Abstract*—This paper presents the results from experimental testing of a dynamic wireless charging system for an autonomous truck model in scale 1:14. The truck model is equipped with functionality for autonomous operation and is utilizing two cameras for computer-vision-based path tracking. The model is operated on a marked track including a charging area with two coils for dynamic wireless inductive power transfer. The road-side coils are activated from the truck model when approaching the charging area, and the dc power input at the road-side as well as the power output from the on-board coil after rectification are logged during a charging cycle. The impact on the power transfer and the overall energy transfer efficiency by activating the road side coils at different positions of the truck is then evaluated. The results shows how the total energy transferred to the battery on-board the truck model can be maximised by activating the road-side coils at the point of zero magnetic coupling. Furthermore, the energy transfer efficiency is slightly reduced by utilizing the power transfer capability during the transient in the coupling conditions, while activation before the point of zero magnetic coupling will generate unnecessary idling losses.**

***Index Terms*—Autonomous Vehicles, Camera Vision, Electric Vehicles, Inductive Power Transfer, Dynamic Wireless Charging.**

I. Introduction

Background...[1, 2]

Autonomous Vehicles - current status...

Wireless Charginge - recent developments...

Wireless charging of autonomous vehicles - suitable combintaiton for full autonomy....

II. System

Studied system....

A. Hardware

Overview of the truck model

B. Software

Overview of the software....

III. Results

Present test setup - demonstration of path tracking - wireless charging power transfer and energy efficiency...

IV. Conclusion

This paper

References

[1] G. Guidi, A. M. Lekkas, J. E. Stranden, and J. A. Suul, "Dynamic wireless charging of autonomous vehicles: Small-scale demonstration of inductive power transfer as an enabling technology for self-sufficient energy supply," *IEEE Electrification Magazine*, vol. 8, no. 1, pp. 37–48, 2020.

[2] G. Guidi and J. A. Suul, "Transient control of dynamic inductive ev charging and impact on energy efficiency when passing a roadside coil section," in *2018 IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (Wow)*. IEEE, 2018, pp. 1–7.

Appendix B

Modeling for a LQR or MPC controller

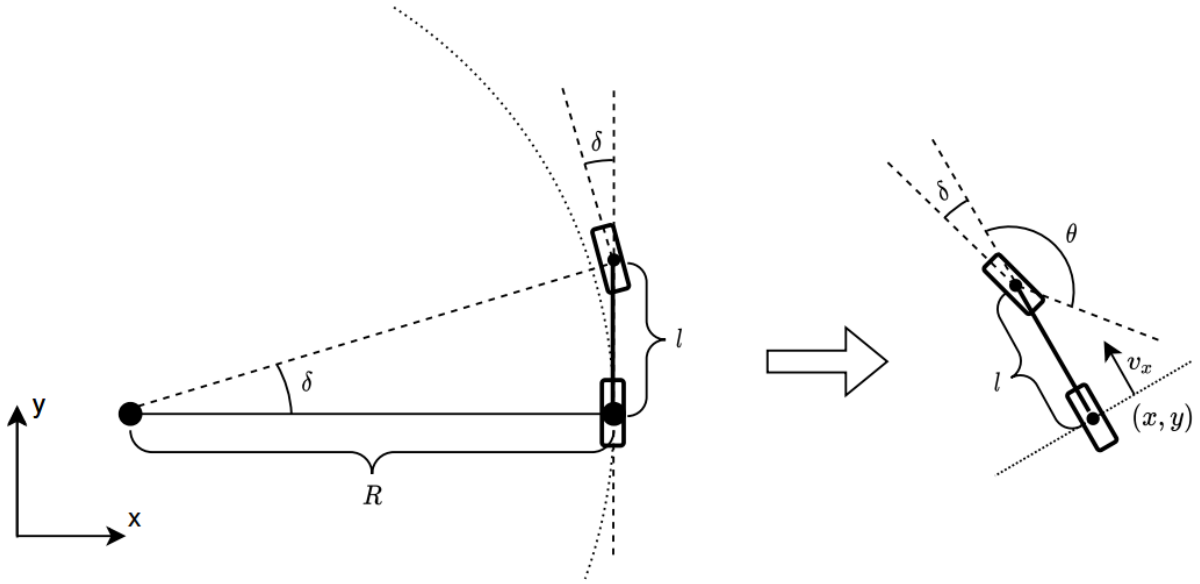


Figure B.1: From geometric to kinematic bicycle model

δ - Steering angle
R - Turning radius
l - Wheel base
v_x - Longitudinal velocity
θ - Heading of the vehicle
(x, y) - Real axle position

A state space for the kinematic bicycle model could be achieved with the following equations.

$$tan(\delta) = \frac{l}{R} \tag{B.1}$$

$$R = \frac{v_x}{\dot{\theta}} \tag{B.2}$$

$$\frac{v_x tan(\delta)}{l} = \frac{v_x}{R} \tag{B.3}$$

$$\dot{x} = v_x cos(\theta) \tag{B.4}$$

$$\dot{y} = v_x sin(\theta) \tag{B.5}$$

$$\dot{\theta} = \frac{tan(\delta)}{l} \tag{B.6}$$

State space:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\delta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \\ \frac{tan(\delta)}{L} \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \dot{\delta} \tag{B.7}$$

REFERENCE

A. Patnaik, M. Patel, V. Mohta, H. Sha, S Agrawal, A. Rathore, R. Malik, D. C. R. Bhattacharyaa, *Design and Implementation of Path Trackers for Ackermann Drive based Vehicles*, 2012.
Available: *https://arxiv.org/pdf/2012.02978.pdf*

Appendix C

User manual

A host computer with a SSH interface is required to start the truck model. For a full visualization with a GUI (Graphic User Interface) the thesis in 2019 have made a user manual for this [44].

C.1 Getting started

1. Connect the two batteries, plug in the power for the Nvidia Jetson TX2, the power for the truck model, the plug from the charging coil power, the ammeter, and the USB plug from the USB hub to the Nvidia Jetson TX2.
2. Press the POWER BTN at the Nvidia Jetson TX2. Se figure C.1. Press the switch on at the truck for energizing the truck model. Press the switch on inside the trailer for energizing the charging system.
3. Connect to the WiFi hotspot $Sintef\_truck$ at the host computer. Password is: $JetsonTX2$. (Usually appears within 1-2 minutes)
4. Open a terminal at the host computer. Type in $ssh\ -X\ nvidia$@10.42.0.1. Password is: $nvidia$
5. Type in $./ros\_startup.sh$ in the terminal to launch the truck model.
6. To edit the scripts, open a new terminal for a text editor. Access the text editor by typing e.g. $Geany$ or another text editor at your own choice.

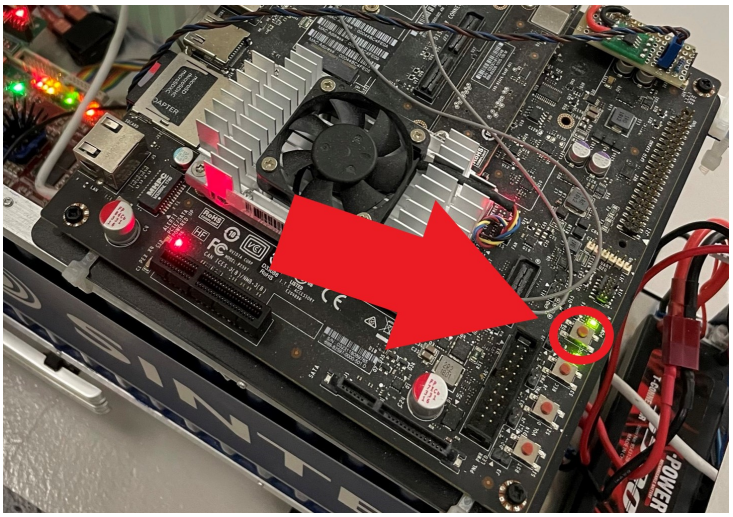


Figure C.1: Nvidia Jetson TX2 Power button

C.2 Joystick controls

An overview of the joystick controls is viewed in figure C.2

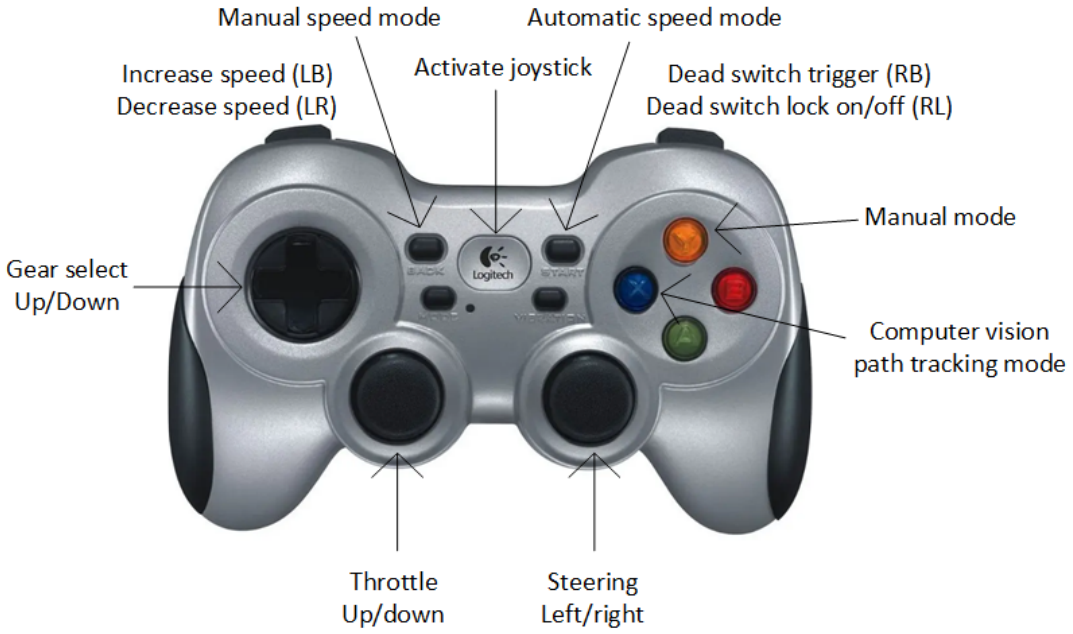


Figure C.2: Logitech joystick controls

- To activate the joystick press the **Activate joystick**. To activate the truck model press the **Dead switch (RB)**. The **Dead switch lock (RL)** locks this switch in a position if needed.
- Press either **Manual model** or **Computer vision path tracking mode** to start the truck model.
- **Manual mode** is controlled by the Throttle, Steering and Gear select.
- **Computer vision path tracking mode** is the autonomous mode. The speed could either be set manually with the **Manual speed mode** and then **Increased speed (LB)/Decreased speed (LR)**. Or it could be set automatically with **Automatic speed mode** and then follows the Ackermann drive node speed variables.
- The red and green button is currently not in use. They could be activated in the Car cmd node for SLAM path following mode and AI/Lane follow mode from the master thesis in 2019. Se reference [44] for this user manual.

C.3 Shutting down

Press **ctrl + c** in the terminal at the host computer to stop the nodes at the Nvidia Jetson TX2. Type *sudo shutdown now* in the terminal to power off the Nvidia Jetson TX2. Then the wires and plugs could be disconnected. **Do not disconnect the wires before the Nvidia Jetson TX2 is powered down, this could cause harm to the system**

Appendix D

C code for microcontroller Teensy 3.2

Code edited by the author is marked with Hall effect and Steering/servo feedback

```
// Car control node for ROS
// Jon Eivind Stranden 2019 @ NTNU
// Part of Master thesis - Self-driving truck with wireless charging (SINTEF
2019)
// Edited by Mats Jørgensen as a part of a Specialization project 2020 and
master thesis 2021

#include <PWMServo.h>
#include <Wire.h>
#include <OctoWS2811.h>
#include "SparkFun_BNO080_Arduino_Library.h"
#include <ros.h>
#include <ros/time.h>
#include <sensor_msgs/Imu.h>
#include <geometry_msgs/Twist.h>
#include <std_msgs/Int16.h>
#include <std_msgs/Float32.h>

/*** IMU ***/
BNO080 IMU;
uint32_t last_time = 0;


/***Hall Effect***/
const byte interruptPin = 6;
volatile byte hall_rising = 0; //flag
volatile unsigned long irqMillis;
unsigned long hallEffectCount = 0;
unsigned long differenceTimeMillis;
unsigned long startMillis;
float distance = 1;
float Speed = 0;
float prevSpeed = 0;
float prevDistance = 0;
float wheelMovement = 25; //in mm
uint32_t last_time_hall = 0;

/*** Servo ***/
PWMServo steering_servo, throttle_servo, gear_servo;  // Servo objects

// Steering servo feedback
int reading[21];
int feedback;
int servo_read = 0;
uint32_t last_time_servo = 0;

// Define PWM pins
int steering_servo_pin = 20;
int throttle_servo_pin = 21;
int gear_servo_pin = 22;

// Variables for storing servo positions
int steering_pos = 90;
int throttle_pos = 90;
int gear_pos = 90;

/*** LED STRIP ***/
// Note: the led strip is connected to pin 2
const int num_leds = 4;
int led_mode = 0;
```

```
// LED colors
#define RED    0xFF0000
#define GREEN  0x00FF00
#define BLUE   0x0000FF
#define YELLOW 0xFFFF00
#define PINK   0xFF1088
#define ORANGE 0xE05800
#define WHITE  0xFFFFFF
#define BLACK  0x000000

DMAMEM int displayMemory[num_leds*6];
int drawingMemory[num_leds*6];
const int config = WS2811_GRB | WS2811_800kHz;
OctoWS2811 leds(num_leds, displayMemory, drawingMemory, config);

// Loop timer variables
float long_loop_timer = millis();
float short_loop_timer = millis();
boolean long_time_passed = false;
boolean short_time_passed = false;


/*** LED sequences ***/
void led_control(int mode){

  // Loop timers
  if((millis() - long_loop_timer) >= 1000.0){
    long_loop_timer = millis();
    long_time_passed = !long_time_passed;
  }
  if((millis() - short_loop_timer) >= 100.0){
    short_loop_timer = millis();
    short_time_passed = !short_time_passed;
  }

  /* Off */
  if(mode==0){
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, BLACK);
    }
    leds.show();
  }

  /* Green */
  else if(mode==1){ // constant
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, GREEN);
    }
    leds.show();
  }
  else if(mode==2 && long_time_passed){ // blink
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, GREEN);
    }
    leds.show();
  }
  else if(mode==2 && !long_time_passed && short_time_passed){
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, BLACK);
    }
    leds.show();
```

```
  }

  /* Blue */
  else if(mode==3){ // constant
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, BLUE);
    }
    leds.show();
  }
  else if(mode==4 && long_time_passed){ // blink
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, BLUE);
    }
    leds.show();
  }
  else if(mode==4 && !long_time_passed && short_time_passed){
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, BLACK);
    }
    leds.show();
  }

  /* Red */
  else if(mode==5){ // constant
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, RED);
    }
    leds.show();
  }
  else if(mode==6 && long_time_passed){ // blink
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, RED);
    }
    leds.show();
  }
  else if(mode==6 && !long_time_passed && short_time_passed){
    for (int i=0; i < leds.numPixels(); i++) {
      leds.setPixel(i, BLACK);
    }
    leds.show();
  }
  long_time_passed = false;
  short_time_passed = false;
}

/*** ROS ***/
ros::NodeHandle nh;
ros::Time last_heartbeat_received;
bool is_jetson_running = true;

/* ROS callbacks */
void controlCallback(const geometry_msgs::Twist& twist_msg){
  // Servo and ESC control
  steering_pos = twist_msg.angular.z;
  throttle_pos = twist_msg.linear.x;
  gear_pos = twist_msg.linear.z;
}
void heartbeatCallback(const std_msgs::Int16& Int16_msg){
  // received heartbeat
  last_heartbeat_received = nh.now();
}
```

```
void ledCallback(const std_msgs::Int16& Int16_msg){
  // set LED mode
  led_mode = Int16_msg.data;
}

// New IMU msg
sensor_msgs::Imu imu_msg;

// New Hall effect msg
std_msgs::Float32 halleffect_msg;
std_msgs::Float32 dist_msg;

// New Steering feedback msg
std_msgs::Int16 servo_fb_msg;

//Publisher
ros::Publisher pub_halleffect("halleffect_data", &halleffect_msg);
ros::Publisher pub_dist("dist_data", &dist_msg);
ros::Publisher pub_servo_fb("servo_fb_data", &servo_fb_msg);
ros::Publisher pub_imu("imu_data", &imu_msg);

//Subscriptions
ros::Subscriber<geometry_msgs::Twist> sub_twist("car_cmd", &controlCallback );
ros::Subscriber<std_msgs::Int16> sub_heartbeat("heartbeat",
&heartbeatCallback );
ros::Subscriber<std_msgs::Int16> sub_led("led_mode", &ledCallback );

// Interrupt request Hall effect
void wheel_IRQ(){
  irqMillis = millis();
  hall_rising = 1;
  hallEffectCount++;
}

// Steering feedback
int getFeedback(int sorting[]){
  int mean;
  int result;
  int dummy;
  boolean sorted;
  sorting[20] = 0;

  sorted = false; //clear flag
  while(sorted != true){
    sorted = true;
    for(int j=0;j<20;j++){ //sort the array
      if(sorting[j] > sorting[j+1]){
        dummy = sorting[j+1];
        sorting[j+1] = sorting[j];
        sorting[j] = dummy;
        sorted = false;
      }
    }
  }
  mean = 0;
  for(int k=5;k<15;k++){//discard the 5 low and 5 high values
    mean += sorting[k];
  }
  result = mean/10;
  return(result);
}
```

```
void setup() {
  leds.begin(); // Start leds
  Wire.begin(); //Start I2C
  IMU.begin(); //Start IMU

  delay(1000);

  Serial.begin(57600); // Start serial comm
  Wire.setClock(400000); //Increase I2C data rate to 400kHz

  analogReadRes(12); //Increase range to 0 - 4095 bits

  //Send data update every 50ms
  IMU.enableRotationVector(50);
  IMU.enableGyro(50);
  IMU.enableLinearAccelerometer(50);

  //Hall Effect
  pinMode(interruptPin, INPUT); //pin 6 for interrupt
  attachInterrupt(digitalPinToInterrupt(interruptPin), wheel_IRQ, FALLING);
// triggers when HIGH goes to LOW

  //Attach servos on PWM pins to servo objects with min/max values
  steering_servo.attach(steering_servo_pin, 1000, 2000);
  throttle_servo.attach(throttle_servo_pin, 1000, 2000);
  gear_servo.attach(gear_servo_pin, 1000, 2000);

  // Init ROS node & topics
  nh.initNode();
  nh.advertise(pub_imu);
  nh.advertise(pub_dist);
  nh.advertise(pub_halleffect);
  nh.advertise(pub_servo_fb);
  nh.subscribe(sub_twist);
  nh.subscribe(sub_heartbeat);
  nh.subscribe(sub_led);

}
void loop(){

  led_control(led_mode); // Set LED mode

  // Check if heartbeat has been received within the last sec, else stop the
truck
  if((nh.now().sec - last_heartbeat_received.sec) <= 1){
    steering_servo.write(steering_pos);
    throttle_servo.write(throttle_pos);
    gear_servo.write(gear_pos);
  }
  else {
    steering_servo.write(90);
    throttle_servo.write(90);
    gear_servo.write(90);
  }
  //Serial.println(Speed);

  // Hall effect
  while(hall_rising == 1){
    differenceTimeMillis = irqMillis - startMillis;
    startMillis = irqMillis;
```

```
    hall_rising = 0;
  }
  if( differenceTimeMillis != 0 ){
    Speed = wheelMovement / differenceTimeMillis; //Speed in m/s
    distance = (wheelMovement * hallEffectCount)/1000; //dist in m

    if (distance != prevDistance){
      halleffect_msg.data = Speed;
      dist_msg.data = distance;
      pub_dist.publish(&dist_msg);
      pub_halleffect.publish(&halleffect_msg);
    }
  }
  prevDistance = distance;


  //Servo feedback
  if (millis() - last_time_servo >= 2 && servo_read < 20){
    reading[servo_read] = analogRead(15);
    servo_read++;
    last_time_servo = millis();
  }
  if (servo_read == 19){
    servo_read = 0;
    feedback = getFeedback(reading);
    servo_fb_msg.data = feedback;
    pub_servo_fb.publish(&servo_fb_msg);
  }

  // Publish IMU data on ROS topic
  if (IMU.dataAvailable() == true && nh.connected() && millis() - last_time
>= 50 ){

    last_time = millis();

    imu_msg.header.stamp = nh.now();

    imu_msg.orientation.x = IMU.getQuatI();
    imu_msg.orientation.y = IMU.getQuatJ();
    imu_msg.orientation.z = IMU.getQuatK();
    imu_msg.orientation.w = IMU.getQuatReal();

    imu_msg.angular_velocity.x = IMU.getGyroX();
    imu_msg.angular_velocity.y = IMU.getGyroY();
    imu_msg.angular_velocity.z = IMU.getGyroZ();

    imu_msg.linear_acceleration.x = IMU.getLinAccelX();
    imu_msg.linear_acceleration.y = IMU.getLinAccelY();
    imu_msg.linear_acceleration.z = IMU.getLinAccelZ();

    //Serial.println(imu_msg.orientation.z);
    pub_imu.publish(&imu_msg);
  }
  nh.spinOnce();
  delay(1);
}
```

Appendix E

Python code for the embedded computer Nvidia Jetson TX2

E.1 Ackermann drive node

Main parts of the code edited by author:
- Image processing
- Charging coil activation/deactivation
- Logging functions
- LiDAR mapping
- Joystick functions
- PID controllers
- Computer vision based path tracking controller
- Opportunity charging
- State observation

```python
#!/usr/bin/env python
'''
ackermann_drive_test.py:
    A ros script for automatic lane tracking of ackermann steering based robots
'''

__author__ = 'Alaa Khalaf'
__license__ = 'GPLv3'
__maintainer__ = 'Alaa Khalaf'
__email__ = 'alaahatem87@yahoo.com'

'Edited by Mats Jørgensen as a part of a master thesis 2021'

import time
import sys
import signal
# ROS Image message -> OpenCV2 image converter
from cv_bridge import CvBridge, CvBridgeError
import rospy
from ackermann_msgs.msg import AckermannDrive
from sensor_msgs.msg import Image, Joy
#import struct
from std_msgs.msg import Int64MultiArray, Float32, Int16
from sensor_msgs.msg import Imu
from nav_msgs.msg import Odometry
import matplotlib.pyplot as plt
from geometry_msgs.msg import PoseStamped
#from scipy import zeros, signal
from tf.transformations import euler_from_quaternion

# OpenCV2 for saving an image
import cv2
import numpy as np
from simple_trajectory import center2p, arc1pR, arc3p, distance, radius2p, purepur

#from lane_det1 import perspective_warp, pipeline, find_lanes, draw_lane #simulation
from lane_det_truck_thesis import pipeline3, pipeline4, find_lanes2, draw_lane, pipeline5
#from bluetooth_functions import try2connect, send_START, send_STOP, receive_ENERGY,
end_connection
import bluetooth
# Instantiate CvBridge
bridge = CvBridge()

#result = []
#data_sig = []


class lane_tracker:

    def __init__(self, args):
        # establish bluetooth connection
        bd_addr = "00:06:66:F3:79:D7"
        port = 1
        sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
        try:
            sock.connect((bd_addr, port))
        except:
            #pass
            print("Connection error, bluetooth failed to connect!")
            pass
            #wait for user input to reconnect
            #reconnect = input('try to reconnect? y or n: ')
            #if reconnect == 'y':
            #   return -1
            #   pass
            #else:
            sys.exit(1)

        self.lock_image = False
        self.ready = False

        self.lock_image1 = False
```

```
72          self.ready1 = False
73
74          self.battery_voltage = 0.0
75          self.battery_current = 0.0
76          self.charger_current = 0.0
77          self.active_coil = 0
78
79          self.X_now = 0.0
80          self.Y_now = 0.0
81
82          self.dist_x = 0.0
83          self.dist_y = 0.0
84
85          self.offset_total = 0.0
86          self.counter = 0
87          self.current_offset = 0.0
88
89          self.power_old = 0.0
90          self.time_old = None
91          self.power_integral = 0.0
92          self.power_now = 0.0
93
94          self.old_speed = 0
95          self.old_steering_angle = 0
96          self.raw_img = np.zeros((640,480,3))
97
98          self.raw_img1 = np.zeros((640,480,3))
99
100         self.processed_img = np.zeros((640,480,1))
101         self.processed_img1 = np.zeros((640,480,1))
102
103         self.end_log = 0
104         #
105         self.dist_truck = 0.0
106         self.old_dist_truck = 1.0
107         self.dist_stamp = -5.0
108         self.hall_sensor = 0.0
109         self.servo_fb = 0.0
110
111         self.iTerm = 0.0
112         self.error = 0.0
113         self.error_last = 0.0
114         self.derivate = 0.0
115         self.lt_speed_output = 45.0
116         self.speed_pid = 0.0
117
118         self.iTerm_lane = 0.0
119         self.error_last_lane = 0.0
120         self.pos_pid = 0.0
121         self.lane_tracker = 320.0
122         self.filenumber4 = 1
123         self.time_now3 = 0.0
124         self.time_stamp3 = 0.0
125         self.ref_pos = 0.0
126
127         self.iTerm2  = 0.0
128         self.error3 = 0.0
129         self.ang = 0.0
130         self.ang_output = 0.0
131         self.error_last2 = 0.0
132         self.ang_output_2 = 0.0
133         self.error_last_last = 0.0
134         self.ang_last = 0.0
135
136         self.log_mode = 1
137         self.filenumber3 = 1
138         self.time_now2 = 0.0
139         self.time_stamp2 = 0.0
140         self.ref_vel = 0.0
141
142         self.driving_mode = 0
143         self.simple_lane_follow_mode_button = 0.0
```

```
144         self.manual_mode_button = 0.0
145         self.set_speed_up = 0.0
146         self.set_speed_down = 0.0
147         self.manual_mode = 0.0
148         self.auto_mode = 0.0
149         self.ref_vel_man = 0.4 # 0.4
150         self.ref_vel_mode = 2 # 1 = man, 2 = auto
151 
152         self.filtered_signal = 0.0
153         self.delta = 0.0
154         self.time_stamp_pulse = rospy.get_time()
155 
156         self.time_stamp_bus = 0.0
157         self.bus_mark = 1
158         self.time_mark = 0
159         #self.time_mark2 = 1
160         self.bus_static = 0
161         self.bus_mode = 0 # 0 = dynamic, 1 = busmode
162 
163         self.led_color = 0
164 
165         ##########################
166         self.state = -1 #initial state
167         ##########################
168         if len(args)==1 or len(args)==2:
169             self.max_speed = float(args[0])
170             self.max_steering_angle = float(args[len(args)-1])
171             cmd_topic = 'ackermann_cmd'
172         elif len(args) == 3:
173             self.max_speed = float(args[0])
174             self.max_steering_angle = float(args[1])
175             cmd_topic = '/' + args[2]
176         else:
177             self.max_speed = 0.0
178             self.max_steering_angle = 0.449260 *2
179             cmd_topic = 'ackermann_cmd'
180 
181         self.speed = 0.0
182         self.speed_in = 0.45
183         self.steering_angle = 0
184 
185         self.max_steering_angle = 0.33
186 
187         self.charger_mark = 0
188         self.yaw = 0.0
189         self.charg_detected = 0
190         self.charg_speed = 0
191         self.test_mark = 0
192 
193         image_topic = "image_raw"
194         image_topic1 = 'image_raw1'
195         battery_voltage_topic = "CAN_bus/battery_voltage"
196         #battery_current_topic = "CAN_bus/battery_current"
197         charger_current_topic = "CAN_bus/charge_current"
198         charger_power_topic = "CAN_bus/charging_power"
199 
200 
201 
202         self.camera_sub = rospy.Subscriber(image_topic, Image, self.image_callback, buff_size
    = 640*480*3+5,queue_size=1)
203         self.camera_sub1 = rospy.Subscriber(image_topic1, Image, self.image_callback1,
    buff_size= 640*480*3+5,queue_size=2)
204         self.battery_voltage_sub = rospy.Subscriber(battery_voltage_topic, Int64MultiArray ,
    self.can_bv_callback, queue_size=1)
205         self.charger_current_sub = rospy.Subscriber(charger_current_topic, Int64MultiArray ,
    self.can_charger_current_callback, queue_size=1)
206         self.dist_truck_sub = rospy.Subscriber("dist_data", Float32, self.dist_callback,
    queue_size=1)
207         self.drive_pub = rospy.Publisher(cmd_topic, AckermannDrive,queue_size=1)
208         self.led_pub = rospy.Publisher('led_mode', Int16, queue_size=1)
209         self.imu_sub = rospy.Subscriber("odom", Odometry, self.odom_callback)
210         self.hall_sensor_sub = rospy.Subscriber("halleffect_data", Float32, self.hall_callback
```

```
210 , queue_size=1)
211         self.joy_sub = rospy.Subscriber("joy", Joy, self.joystick_callback, queue_size=1)
212         self.servo_sub = rospy.Subscriber("servo_fb_data", Int16, self.servo_callback,
    queue_size=1)
213         self.slam_sub = rospy.Subscriber("slam_out_pose", PoseStamped, self.slam_vel_callback
    , queue_size=1)
214 
215         rospy.Timer(rospy.Duration(0.01), self.pub_callback, oneshot=False)
216         rospy.Timer(rospy.Duration(0.01), self.pid_controller, oneshot=False)
217         rospy.Timer(rospy.Duration(0.01), self.pid_servo, oneshot=False)
218 
219         rospy.loginfo('ackermann_self-drive_node initialized---version:THESIS')
220         filenumber = 1
221         self.time_zero = rospy.get_time()
222         self.time_stamp = rospy.get_time()
223         filename = "cycle"+ str(filenumber)+"power.txt"
224         f1 = open(filename,"w+")
225         filename2 = "accelration"+str(filenumber)+".txt"
226         f2 = open(filename2,"w+")
227 
228 
229         while not rospy.is_shutdown():
230             if (self.lock_image == False and self.lock_image1 == False and self.ready == True
    and self.ready1 == True):
231                 self.lock_image = True
232                 self.lock_image1 = True
233                 self.ready = False
234                 self.ready1 = False
235 
236 ########## add a state machine to determine whether the charger is found
237         # 1. check for the yellow strap
238         # 2. if found : charger found (send bluetooth start) and switch to charger found state

239         # 3. if in charger found state: check for the yellow strap
240         # 4. if found : find the lane (yellow pipeline), else jump to no charger state and
    send bluetooth stop
241         # 5. in no charger: look for the lane (black pipeline)
242                 charger_exists, self.processed_img = pipeline4(self.raw_img, self.raw_img1)
243 
244 #handle driving
245                 if charger_exists==1 :#and self.battery_voltage <=7.95:
246                     if self.state!=1:
247                         self.state = 1
248                     yellow_lane_not_found = self.send_controls(self.processed_img) #track the
    charging marker
249                     if self.charger_mark == 0 and pipeline5(self.raw_img1)==1:
250                         self.charger_mark = 1
251                         self.dist_stamp = self.dist_truck
252                     if yellow_lane_not_found == -1:
253                         #self.charg_detected, self.processed_img = pipeline3(self.raw_img)
254                         self.charg_detected, self.processed_img = pipeline3(self.raw_img, self
    .raw_img1)
255                         #self.processed_img1 = pipeline3(self.raw_img1)
256                         self.send_controls(self.processed_img)
257 
258                 elif charger_exists==0:
259                     # lane tracking
260                     self.state = 0 #normal lane tracking
261                     #self.charg_detected, self.processed_img = pipeline3(self.raw_img)
262                     self.charg_detected, self.processed_img = pipeline3(self.raw_img, self.
    raw_img1)
263                     self.send_controls(self.processed_img)
264 
265 
266 
267 #handle communication with charger
268 
269                 if self.state == 1:
270                     #if self.charger_mark == 1 and self.active_coil == 0 and self.bus_static
     == 1: # static
271                     if self.charger_mark == 1 and self.active_coil == 0 and (self.dist_truck-
    self.dist_stamp) >= 0.32: #0.32
```

```
272                         self.active_coil = 1
273                         self.time_stamp = rospy.get_time()
274                         self.bus_static = 0
275                         if self.battery_voltage <=7950:
276                             sock.send("11\r\n")
277                             rospy.loginfo(' \033[31;1m \n\n\n Coil 1 energized!\n\n\n \033[0m
     ')
278 
279                     #if self.active_coil == 1 and (rospy.get_time()-self.time_stamp)>=4.0: #
    static
280                     if self.active_coil == 1 and (self.dist_truck - self.dist_stamp) >= 0.89
    :  #0.89
281                         sock.send("0\r\n")
282                         self.active_coil = 2
283                         self.charger_mark = 0
284                         rospy.loginfo(' \033[31;1m \n\n\n BT stop sent!\n\n\n \033[0m ')
285 
286                     if self.active_coil == 2 and (self.dist_truck-self.dist_stamp)>=1.07: #1.
    07
287                         self.active_coil = 3
288                         if self.battery_voltage <=7950:
289                             sock.send("12\r\n")
290                             rospy.loginfo(' \033[31;1m \n\n\n Coil 2 energized!\n\n\n \033[0m
     ')
291 
292                     if self.active_coil == 3 and (self.dist_truck-self.dist_stamp)>=1.51: #1.
    51
293                         self.active_coil = 0
294                         sock.send("0\r\n")
295                         rospy.loginfo(' \033[31;1m \n\n\n BT stop sent!\n\n\n \033[0m ')
296                         rospy.loginfo(' \033[31;1m Battery_energy = %0.2f \033[0m ', self.
    power_integral)
297                         f1.write('%0.6f %0.4f\n'%(time_now-self.time_zero,self.power_now))
298                         self.power_integral = 0.0
299                 else:
300                     if self.active_coil!=0:
301                         self.active_coil=0
302                         sock.send("0\r\n")
303                         rospy.loginfo(' \033[31;1m \n\n\n BT stop sent - Something went wrong
    with the charging cycle!!!\n\n\n \033[0m ')
304             #integrate power
305             time_now = rospy.get_time()
306             self.power_now = self.battery_voltage * self.charger_current
307             #f2.write("%0.6f %0.4f %0.4f\n"%(time_now-self.time_zero,self.X_now,self.Y_now)) #
    IMU
308             f2.write("%0.6f %0.3f %0.3f\n"%(time_now-self.time_zero,self.dist_x,self.dist_y))
    #Lidar
309             if self.time_old is not None and self.state==1:
310                 #write power to file
311                 #f1.write('%0.6f, %0.4f,%0.4f, %0.4f\n'%(time_now-self.time_zero,self.
    power_now,self.battery_voltage, self.charger_current))
312                 f1.write('%0.6f %0.4f\n'%(time_now-self.time_zero,self.power_now))
313                 delta_t = time_now-self.time_old
314                 self.power_integral += (self.power_old + self.power_now)/2*delta_t
315                 self.end_log = 1
316             self.time_old = time_now
317             self.power_old = self.power_now
318             if self.state == 0 and self.end_log == 1:
319                 f1.close()
320                 f2.close()
321                 filenumber += 1
322                 filename = "cycle"+ str(filenumber)+"power.txt"
323                 f1 = open(filename,"w+")
324                 filename2 = "accelration"+str(filenumber)+".txt"
325                 f2 = open(filename2,"w+")
326                 self.end_log = 0
327             #self.filter_sig()
328 
329     def image_callback(self, msg):
330         #print("from callback 1",self.lock_image)
331         #print('callback')
332             try:
```

```
333             # Convert ROS Image message to OpenCV2
334                 self.raw_img = bridge.imgmsg_to_cv2(msg, "bgr8")
335                 self.ready = True
336             except CvBridgeError, e:
337                 print(e)
338 
339     def image_callback1(self, msg):
340         #print("from callback 1",self.lock_image)
341         #print('callback')
342             try:
343             # Convert ROS Image message to OpenCV2
344                 self.raw_img1 = bridge.imgmsg_to_cv2(msg, "bgr8")
345                 self.ready1 = True
346             except CvBridgeError, e:
347                 print(e)
348 
349     def can_bv_callback(self,msg):
350         offset = 0.075
351         self.battery_voltage = msg.data[1]/1000.0-offset
352         if msg.data[1] <7000:
353             rospy.loginfo(' \033[31;1mBattery voltage too low! \033[0m ')
354             sys.exit(1)
355 
356     def can_battery_current_callback(self,msg):
357         self.battery_current = msg.data[1]/1000.0
358 
359     def can_charger_current_callback(self,msg):
360         if self.state == 0:
361             self.offset_total += (msg.data[1])/1000.0 #average offset value = 12/1000.0amps
362             self.counter += 1
363             self.current_offset = self.offset_total/self.counter
364 
365         else:
366             self.charger_current = msg.data[1]/1000.0-self.current_offset #subtract average
    offset value
367             if self.counter == 100:
368                 self.counter = 1
369                 self.offest_total = self.current_offset
370 
371     def odom_callback(self,data):
372 
373         self.X_now = data.pose.pose.position.x
374         self.Y_now = data.pose.pose.position.y
375 
376 
377     def slam_vel_callback(self,data):
378 
379     # Get current position
380         self.dist_x = round(data.pose.position.x, 3)
381         self.dist_y = round(data.pose.position.y, 3)
382 
383     def dist_callback(self,data):
384         self.dist_truck = data.data
385 
386     def hall_callback(self,data):
387         self.hall_sensor = data.data
388 
389     def servo_callback(self, data):
390         self.servo_fb = -((0.449260/575) * (1636-data.data))
391 
392 
393     def joystick_callback(self,data):
394 
395         self.simple_lane_follow_mode_button = data.buttons[0] # blue button
396         self.manual_mode_button = data.buttons[3] # orange button
397         self.set_speed_up = data.buttons[4] # LB button
398         self.set_speed_down = data.buttons[6] # LT button
399         self.manual_mode = data.buttons[8] # back button
400         self.auto_mode = data.buttons[9] # start button
401 
402         if self.set_speed_up is 1 and self.ref_vel_man < 1.0:
403             self.ref_vel_man += 0.1
```

```
404         elif self.set_speed_down is 1 and self.ref_vel_man > 0.0:
405             self.ref_vel_man -= 0.1
406 
407         if self.manual_mode is 1:
408             self.ref_vel_mode = 1
409         elif self.auto_mode is 1:
410             self.ref_vel_mode = 2
411 
412         if self.manual_mode_button is 1:
413             self.driving_mode = 0
414         elif self.simple_lane_follow_mode_button is 1:
415             self.driving_mode = 3
416 
417     def filter_sig(self):
418         global result
419         global data_sig
420 
421         data_sig.append(self.hall_sensor)
422         if len(data_sig) == 10000:
423             del data_sig[0]
424         b, a = signal.butter(3, 0.07)
425         z = signal.lfilter_zi(b, a)
426         result = zeros(len(data_sig))
427         for i, x in enumerate(data_sig):
428             result[i], z = signal.lfilter(b, a, [x], zi=z)
429         self.filtered_signal = result[-1]
430 
431     def pid_controller(self, event): #speed
432 
433         Kp = 1.47 # P speed gainPI = 1.11, PID = 1.47
434         Ki = Kp/0.73 # I Integral action PI = 1.21, PID = 0.73
435         Kd = Kp*0.045 # D derivate
436         dt = 0.01
437         #global f3
438 
439         if self.ref_vel_mode is 1:
440             self.ref_vel = self.ref_vel_man
441         elif self.ref_vel_mode is 2:
442             self.ref_vel = self.speed
443 
444         #logging
445 
446         #if self.driving_mode is 3 and self.log_mode is 1:
447         #   filename3 = "PIDcontrolNew"+str(self.filenumber3)+".txt"
448         #   f3 = open(filename3,"w+")
449         #   self.time_stamp2 = rospy.get_time()
450         #   self.log_mode = 2
451         #elif self.driving_mode is 0 and self.log_mode is 2:
452         #   f3.close()
453         #   self.log_mode = 1
454         #   self.filenumber3 += 1
455 
456         #filter_cur_vel = self.filter_sig()
457 
458         #self.error = self.ref_vel - self.filtered_signal
459         self.error = self.ref_vel - self.hall_sensor
460 
461         self.iTerm += (Ki * self.error * dt)
462 
463         self.derivate = (self.error - self.error_last) / dt
464         self.speed_pid = Kp * self.error + self.iTerm# + Kd * self.derivate
465 
466         if self.speed_pid > 1.0:
467             self.iTerm -= (self.speed_pid - 1.0)
468             self.speed_pid = 1.0
469         elif self.speed_pid < 0.0:
470             self.iTerm += (0.0 - self.speed_pid)
471             self.speed_pid = 0.0
472 
473         self.lt_speed_output = 90.0 * (1.0 - self.speed_pid)
474         self.error_last = self.error
475 
```

```
476         #logging
477 
478         #if self.driving_mode is 3:
479         #   self.time_now2 = rospy.get_time()
480             #try:
481             #   f3.write('%0.6f %0.2f %0.2f %0.4f %0.4f\n'%(self.time_now2 - self.time_stamp2
    , self.dist_truck, self.ref_vel, self.hall_sensor, self.filtered_signal))
482             #except NameError, ValueError:
483             #   filename3 = "PIDcontrolNew"+str(self.filenumber3)+".txt"
484             #   f3 = open(filename3,"w+")
485             #   self.time_stamp2 = rospy.get_time()
486             #   self.log_mode = 2
487 
488     def pid_lane_tracker(self, event):
489 
490         Kp = 1.5 # P speed gain PI = 1.11, PID = 1.47
491         Ki = 0.2 # I Integral action PI = 1.21, PID = 0.73
492         Kd = 0.1 # D derivate
493         dt = 0.01
494         global f4
495 
496         if self.driving_mode is 3 and self.log_mode is 1:
497             filename4 = "PIDlane"+str(self.filenumber4)+".txt"
498             f4 = open(filename4,"w+")
499             self.time_stamp3 = rospy.get_time()
500             self.log_mode = 2
501         elif self.driving_mode is 0 and self.log_mode is 2:
502             f4.close()
503             self.log_mode = 1
504             self.filenumber4 += 1
505 
506         #ref_pos = 0
507         #lane_center, dummy, dummy2 = find_lanes2(img,10,1)
508         cur_pos = -((0.44/320) * (self.lane_tracker - 320))
509 
510         error = self.ref_pos - cur_pos
511 
512         self.iTerm_lane += (Ki * error * dt)
513 
514         derivate = (error - self.error_last_lane) / dt
515         pos_pid = Kp * error# + self.iTerm_lane# + Kd * self.derivate
516 
517         if pos_pid > 0.44:
518             self.iTerm_lane -= (pos_pid - 0.44)
519             self.pos_pid = 0.44
520         elif pos_pid < -0.44:
521             self.iTerm_lane += ((-0.44) - pos_pid)
522             self.pos_pid = -0.44
523         else:
524             self.pos_pid = pos_pid
525 
526         self.error_last_lane = error
527 
528         if self.driving_mode is 3:
529             self.time_now3 = rospy.get_time()
530             try:
531                 f4.write('%0.6f %0.2f %0.2f %0.4f\n'%(self.time_now3 - self.time_stamp3, self.
    dist_truck, self.ref_pos, self.pos_pid))
532             except NameError, ValueError:
533                 filename4 = "PIDlane"+str(self.filenumber4)+".txt"
534                 f4 = open(filename4,"w+")
535                 self.time_stamp3 = rospy.get_time()
536                 self.log_mode = 2
537 
538     def pid_servo(self,event): #steering angle
539 
540         Kp = 0.51 # P speed gainPI = 0.51
541         Ki = 2.04 # I Integral action PI = 2.04
542         Kd = 0.05 # D derivate
543         dt = 0.01
544         max_steering_angle = 0.44
545 
```

```
546         #logging
547 
548         global f5
549         if self.driving_mode is 3 and self.log_mode is 1:
550             filename = "PIDservo"+str(self.filenumber4)+".txt"
551             f5 = open(filename,"w+")
552             self.time_stamp3 = rospy.get_time()
553             self.log_mode = 2
554         elif self.driving_mode is 0 and self.log_mode is 2:
555             f5.close()
556             self.log_mode = 1
557             self.filenumber4 += 1
558 
559 
560         self.error3 = self.delta - self.servo_fb
561         #error = self.ref_vel_man - self.servo_fb  #tuning
562         self.iTerm2 += Ki * self.error3 * dt
563 
564         derivate = (self.error3 - self.error_last2) / dt
565         self.ang = Kp * self.error3 + self.iTerm2# * Kd * derivate
566         self.ang_output_2 = self.ang_last + Kp * (self.error3 - self.error_last2) + Ki * self.
    error3 * dt  # +Kd*(self.error3-2*self.error_last+self.error_last_last)/dt
567 
568         if self.ang >= max_steering_angle:
569             self.iTerm2 -= self.ang - max_steering_angle
570             self.ang_output = max_steering_angle
571         elif self.ang <= -max_steering_angle:
572             self.iTerm2 += (-max_steering_angle) - self.ang
573             self.ang_output = -max_steering_angle
574         else:
575             self.ang_output = self.ang
576 
577         self.ang_last = self.ang_output_2
578         self.error_last2 = self.error3
579         self.error_last_last = self.error_last2
580 
581         #logging
582 
583         if self.driving_mode is 3:
584             self.time_now3 = rospy.get_time()
585             try:
586                 f5.write('%0.6f %0.2f %0.2f %0.4f\n'%(self.time_now3 - self.time_stamp3, self.
    dist_truck, self.delta, self.ang_output))
587             except NameError, ValueError:
588                 filename = "PIDservo"+str(self.filenumber4)+".txt"
589                 f5 = open(filename,"w+")
590                 self.time_stamp3 = rospy.get_time()
591                 self.log_mode = 2
592 
593 
594     def send_controls(self,img):
595         fpsi = 1/300
596         min_distance_allowed = 2e-2*640/0.21 #2cm
597         speed_factor = 1.0
598         p2 = [320,20]
599 
600         led_off = 0
601         led_green = 1
602         led_green_blink = 2
603         led_blue = 3
604         led_blue_blink = 4
605         led_red = 5
606         led_red_blink = 6
607 
608         #process the image to prepare it to find_lanes
609 
610         #find the lanes and export the points to the steering angle finder
611         if self.state == 0:
612             #self.led_color = led_blue
613             lane_center, window_y, single_lane_direction = find_lanes2(img,10,1)
614         elif self.state == 1:
615             #self.led_color = led_red
```

```
616             lane_center, window_y, single_lane_direction = find_lanes2(img,10,2)
617 
618         if  single_lane_direction == -1: # or (abs(single_lane_direction)==10): # tuning
619             rospy.loginfo(' \033[31;1mInvalid image: lane lines not found! \033[0m ')
620             #print ("\n\n\nInvalid image: lane lines not found!\n\n\n\n")
621             self.lock_image = False
622             self.lock_image1 = False
623             self.speed = 0.40 #0.45
624             #self.steering_angle = self.ang_output
625             self.steering_angle = 0
626             self.pub_callback
627 
628         elif single_lane_direction == 1:
629             p0 = [320,480+0.365*480/0.44] ###0.365### #28cm wheel base,23.4cm blind area in
    front of the truck, 480/0.44 is the scale image scale in pixel/m
630             #rad, center,nxtp1,nxtp2,dummy_dist,dummy_delta = arc3p( p0,[lane_center[-1],800-
    window_y[-1]],[lane_center[-2],800-window_y[-2]])
631             if self.state == 1:
632                 #rad, direc, dummy_delta1 = center2p(p0[0],p0[1],lane_center[-1],window_y[-1
    ]) #offset calibrated manually
633                 #rad, direc, dummy_delta2 = center2p(p0[0],p0[1],lane_center[0],window_y[0])
634                 #factor = 0.3
635                 #self.delta = factor*dummy_delta1+(1-factor)*dummy_delta2
636                 p1 = [328, -0.38 * 480/0.37] #-0.46, 0.45
637                 fac = 0.42 #0.39, 0.45, 0.56
638                 if len(lane_center) >= 3:
639                     self.delta = radius2p(p1, [lane_center[0],lane_center[1],lane_center[-1]-0
    ], [window_y[0],window_y[1],window_y[-1]], self.speed, fac)
640                     #self.delta = radius2p(p1, lane_center[-6:], window_y[-6:], self.speed,
    fac)
641                 else:
642                     self.delta = radius2p(p1, lane_center, window_y, self.speed, fac)
643 
644             else: #regular lane -two sides available
645                 #rad, direc, dummy_delta1 = center2p(p0[0],p0[1],lane_center[-1],window_y[-1
    ]) ##### baseline ##### -35
646                 #rad, direc, dummy_delta2 = center2p(p0[0],p0[1],lane_center[0],window_y[0
    ]) #-35
647                 #factor = 0.6
648                 #self.delta = factor*dummy_delta1+(1-factor)*dummy_delta2
649                 p1 = [320, -0.35 * 480/0.37] #-0.35, 0.46
650                 fac = 0.53 #0.55, 0.45, 0.53
651                 if len(lane_center) >= 6:
652                     self.delta = radius2p(p1, lane_center[-6:], window_y[-6:], self.speed, fac
    )
653                 else:
654                     self.delta = radius2p(p1, lane_center, window_y, self.speed, fac)
655                 #self.delta = radius2p(p1, lane_center[-5:], window_y[-5:], self.speed, 0.48)
656                 #delta = radius2p(p1, [lane_center[0],lane_center[-1]], [window_y[0],window_y
    [-1]])
657 
658             if self.charg_detected == 1:
659                 self.charg_speed = 1
660 
661             if self.bus_mode == 1:
662                 if self.state == 1:
663                     if (self.dist_truck-self.dist_stamp) >= 1.07 and (self.dist_truck-self.
    dist_stamp) <= 1.21:
664                         self.test_mark = 1
665                     #if self.active_coil == 1 and self.power_now >= 30.0 and self.
    battery_voltage <= 8.85 and self.bus_mark == 1:
666                     #if self.test_mark == 1 and self.active_coil == 1 and self.battery_voltage
     <= 8.85 and self.power_now >= 30.0 and self.bus_mark == 1:  #dynamic
667                     if self.test_mark == 1 and self.battery_voltage <= 8.85 and self.bus_mark
     == 1: #static
668                         self.speed_in = 0.0
669                         if  self.time_mark == 1:
670                             self.time_mark = 0
671                             self.time_stamp_bus = rospy.get_time()
672                         if rospy.get_time()-self.time_stamp_bus >= 3.5: #bus-charging seconds
673                             self.bus_mark = 0
674                         if rospy.get_time()-self.time_stamp_bus >= 1.0: #static
```

```
675                             self.bus_static = 1
676                         #elif self.active_coil == 1: #dynamic
677                         elif (self.dist_truck-self.dist_stamp) >= 0.72 and (self.dist_truck-self.
    dist_stamp) <= 1.4:
678                             self.speed_in = 0.21
679                             self.charg_speed = 0
680                         else:
681                             self.speed_in = 0.35/speed_factor
682                     elif (self.dist_truck - self.dist_stamp) < 4.9:  #smoother descent of charging
     station
683                         self.speed_in = 0.4/speed_factor
684                         self.charg_speed = 0
685                     elif self.charg_speed == 1:
686                         self.speed_in = 0.4/speed_factor #0.45
687                     else:
688                         self.speed_in = 0.5/speed_factor #0.5
689                         self.bus_mark = 1
690                         self.time_mark = 1
691                         self.test_mark = 0
692                 elif self.bus_mode == 0:
693                     if self.state == 1:
694                         self.speed_in = 0.39
695                     elif (self.dist_truck - self.dist_stamp) < 4.9:  #smoother descent of charging
     station
696                         self.speed_in = 0.39/speed_factor
697                         self.charg_speed = 0
698                     elif self.charg_speed == 1:
699                         self.speed_in = 0.39/speed_factor #0.45
700                     else:
701                         self.speed_in = 0.48/speed_factor #0.5
702 
703                 duration = rospy.Duration.from_sec(fpsi)
704                 self.speed = self.speed_in
705                 self.steering_angle = self.delta
706                 #self.steering_angle = self.ang_output
707                 #self.steering_angle = self.ang_output_2
708 
709             elif (abs(single_lane_direction)==10):
710                 #print('her',np.sign(single_lane_direction))
711                 self.speed = self.speed_in
712                 if self.state == 0:
713                     self.steering_angle = self.max_steering_angle * np.sign(single_lane_direction)
714                 else:
715                     #self.steering_angle = self.ang_output
716                     self.steering_angle = (self.max_steering_angle * np.sign(single_lane_direction
    ))/2.12
717                 #*abs(self.steering_angle/self.max_steering_angle)
718 
719         def pub_callback(self, event):
720             ackermann_cmd_msg = AckermannDrive()
721             #ackermann_cmd_msg.speed = self.speed
722             ackermann_cmd_msg.speed = self.lt_speed_output
723             ackermann_cmd_msg.steering_angle = self.steering_angle
724             led_msg = Int16()
725             led_msg.data = self.led_color
726             self.drive_pub.publish(ackermann_cmd_msg)
727             self.led_pub.publish(led_msg)
728             self.lock_image = False
729             self.lock_image1 = False
730 
731             if (self.old_dist_truck != self.dist_truck):
732                 self.print_state()
733                 self.old_dist_truck = self.dist_truck
734 
735 
736         def print_state(self): #'\x1b[1M\r'
737             #sys.stderr.write('\x1b[2J\x1b[H')
738             rospy.loginfo(
739                       '\033[34;1mSt.: \033[32;1m%d, '
740                       '\033[34;1mBat_v: \033[32;1m%0.3f, '
741                       '\033[34;1mSp: \033[32;1m%0.2f m/s, '
742                       '\033[34;1mDist: \033[32;1m%0.2f m, '
```

```
743                     '\033[34;1mRef: \033[32;1m%0.2f m/s ',
744                     self.state, self.battery_voltage, self.hall_sensor, self.dist_truck, self.
    ref_vel)
745
746     def finalize(self, signal, frame):
747         rospy.loginfo('Halting motors, aligning wheels and exiting...')
748         ackermann_cmd_msg = AckermannDrive()
749         ackermann_cmd_msg.speed = 0
750         ackermann_cmd_msg.steering_angle = 0
751         led_msg = Int16()
752         led_msg.data = 0
753         self.drive_pub.publish(ackermann_cmd_msg)
754         self.led_pub.publish(led_msg)
755         f1.close()
756         f2.close()
757         f3.close()
758         f4.close()
759         sys.exit()
760
761 if __name__ == '__main__':
762     rospy.init_node('simple_lane_tracking_node')
763     simple_lane_tracker = lane_tracker(sys.argv[1:len(sys.argv)])
764     #signal.signal(signal.SIGINT, finalize)
765     #rospy.spin()
766
767
```

E.2 Car cmd node

Main parts of the code edited by author:
- Obstacle detection
- Joystick commands
- Driving modes

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Car Controller node for SINTEF RC Truck with wireless charging
# By Jon Eivind Stranden @ NTNU 2019

'Edited by Mats Jørgensen as a part of a master thesis 2021'
from math import tan, sqrt, atan2
import numpy as np
import os
import rospy
import matplotlib.pyplot as plt
#from datetime import datetime
#from matplotlib import pyplot
#from matplotlib.animation import FuncAnimation
#from random import randrange
from std_msgs.msg import Float32, Float64, Int16, Int64MultiArray
from sensor_msgs.msg import Joy, LaserScan
from geometry_msgs.msg import Twist, PoseStamped
from ackermann_msgs.msg import AckermannDrive
from filters import *
from scipy import zeros, signal

### Settings ##################################

wheelbase_length = 0.28 # axle length L in m
max_steering_angle = 0.449260 # rad 0.449260

# set speed of vehicle (0=max, 90=min, 91-180=reverse)
car_velocity_straight_speed = 70
car_velocity_cornering_speed = 70
car_velocity_charge_speed = 70

obstacle_detector_enabled = True # obstacle detector enable/disable
obstacle_detect_dist = 0.425 # meters
obstacle_detect_fov = 1 # degrees field of view

### Variables #############

# Joystick variables
joy_steering_angle = 0
joy_vel = 90
gear_selected = 0
dead_switch = False
dead_switch_lock = True

# Driving mode variable
driving_mode = 0
# Modes:
# 0 = manual, use blue button to log a new path
# 1 = SLAM with path tracking to follow logged path
# 2 = AI controller for following marked lanes without logged path

# For calculating SLAM velocity
current_velocity = 0.0
prev_dist_x = 0
prev_dist_y = 0

# Move_base variables
move_base_velocity = 0.0
move_base_steering_angle = 0.0

# Pure_pursuit variables
pp_steering_angle = 0

# AI controller variables
dnn_steering_angle = 0.0

# CV lane keeper
lane_center_offset_cm = 0.0

# Waypoint logger status
```

```
 73 wp_logger_active = False
 74
 75 # Charger status
 76 charger_active = False
 77
 78 # Obstacle detector
 79 obstacle_detected = False # detect variable
 80 current_steering_angle = 0.0
 81
 82 # For terminal menu
 83 class bcolors:
 84     HEADER = '\033[95m'
 85     OKBLUE = '\033[94m'
 86     OKGREEN = '\033[92m'
 87     WARNING = '\033[93m'
 88     FAIL = '\033[91m'
 89     ENDC = '\033[0m'
 90     BOLD = '\033[1m'
 91     UNDERLINE = '\033[4m'
 92     WHITE = '\033[96m'
 93
 94
 95 def joy_callback(data):
 96
 97     global joy_steering_angle
 98     global joy_vel
 99     global gear_selected
100     global dead_switch
101     global dead_switch_lock
102     global wp_logger_active
103     global max_steering_angle
104     global driving_mode
105     global ref_vel_man
106     global ref_vel_mode
107     global steering_selected
108     global rst_obst
109
110     # Get joystick values
111     joy_steering_angle = (data.axes[2]*max_steering_angle) # right stick
112     joy_vel = -(data.axes[1]*90/1.6) + 90 # left stick
113     gear_val = data.axes[5] # arrow buttons
114     dead_switch_button = data.buttons[5] # RB button
115     dead_switch_lock_button = data.buttons[7] # RT button
116     slam_tracker_mode_button = data.buttons[1] # green button
117     ai_mode_button = data.buttons[2] # red button
118     manual_mode_button = data.buttons[3] # orange button
119     set_speed_up = data.buttons[4] # LB button
120     set_speed_down = data.buttons[6] # LT button
121     simple_lane_follow_mode_button = data.buttons[0] # blue button
122     manual_mode = data.buttons[8] # back button
123     auto_mode = data.buttons[9] # start button
124     steering_ang = data.axes[4] # tuning steering
125     rst_obst = data.buttons[10] # left joy button
126
127     #print(data.axes[1])
128
129     if set_speed_up is 1:
130         ref_vel_man += 0.1
131     elif set_speed_down is 1:
132         ref_vel_man -= 0.1
133
134     if manual_mode is 1:
135         ref_vel_mode = 1
136     elif auto_mode is 1:
137         ref_vel_mode = 2
138
139     if steering_ang is 1:
140         steering_selected += 0.1
141     elif steering_ang is -1:
142         steering_selected -= 0.1
143
144     # Set dead switch or dead switch lock
```

```
145     if dead_switch_button is 1:
146         dead_switch = True
147     elif dead_switch_lock_button is 1:
148         dead_switch_lock = not dead_switch_lock
149         dead_switch = not dead_switch
150     elif dead_switch_lock is False:
151         dead_switch = False
152 
153     # Setting driving mode from joycontroller
154     if manual_mode_button is 1:
155         driving_mode = 0
156     #print('Manual mode')
157     elif slam_tracker_mode_button is 1 and wp_logger_active is False:
158         driving_mode = 1
159     elif ai_mode_button is 1 and wp_logger_active is False:
160         driving_mode = 2
161     elif simple_lane_follow_mode_button is 1 and wp_logger_active is False:
162         driving_mode = 3 #new lane tracker
163 
164     #print(driving_mode)
165 
166     # Gear selector
167     if gear_val > 0 and gear_selected < 2:
168         gear_selected+=1
169     elif gear_val < 0 and gear_selected > 0:
170         gear_selected-=1
171 
172 def dnn_controller_callback(data):
173 
174     global dnn_steering_angle
175     global max_steering_angle
176 
177     dnn_steering_angle = (data.angular.z*max_steering_angle)
178 
179 
180 def slam_vel_callback(data):
181 
182     global prev_dist_x
183     global prev_dist_y
184     global current_velocity
185 
186     # Get current position
187     dist_x = round(data.pose.position.x, 3)
188     dist_y = round(data.pose.position.y, 3)
189 
190     #print(dist_x)
191     #print(dist_y)
192 
193     # Calculate current speed from SLAM odometry
194     current_velocity = sqrt( (dist_x - prev_dist_x)**2 + (dist_y - prev_dist_y)**2 ) / 0.1 # [
    m/s]
195 
196     prev_dist_x = dist_x
197     prev_dist_y = dist_y
198 
199 
200 def move_base_callback(data):
201 
202     global move_base_velocity
203     global move_base_steering_angle
204     global wheelbase_length
205 
206     move_base_velocity = data.linear.x
207     move_base_steering_angle = atan2(wheelbase_length * data.angular.z, move_base_velocity)
208 
209 
210 def pure_pursuit_callback(data):
211 
212     global pp_steering_angle
213     pp_steering_angle = data.angular.z
214 
215 def lane_tracker_callback(ackermann_cmd):
```

```
216
217     global lt_steering_angle
218     global lt_speed
219     global ref_vel_aut
220
221     lt_steering_angle = -ackermann_cmd.steering_angle
222     ref_vel_aut = 0.0
223     lt_speed = ackermann_cmd.speed
224
225 def lane_center_offset_callback(data):
226
227     global lane_center_offset_cm
228     lane_center_offset_cm = data.data
229
230 def wp_logger_status_callback(data):
231
232     global wp_logger_active
233
234     wp_status = data.data
235
236     if wp_status is 1:
237         wp_logger_active = True
238     else:
239         wp_logger_active = False
240
241 def charger_status_callback(data):
242
243     global charger_active
244
245     charger_status = data.data
246
247     if charger_status is 1:
248         charger_active = True
249     else:
250         charger_active = False
251
252 def laser_callback(data): #obstacle detection
253
254     global obstacle_detector_enabled
255     global obstacle_detect_dist
256     global obstacle_detect_fov
257     global obstacle_detected
258     global current_steering_angle
259
260     num_detected = 0
261
262     # Detection area changes with steering angle
263     steering_offset = int((current_steering_angle-90)/5)
264
265     ranges = np.array([])
266
267     ranges = np.append(ranges, data.ranges[0:45-0])
268     ranges = np.append(ranges, data.ranges[314-0:359])
269     #ranges = np.append(ranges, data.ranges)
270
271     for r in ranges:
272         #print (r)
273         if r < obstacle_detect_dist:
274             num_detected += 1
275
276     if obstacle_detector_enabled is True:
277         #print(num_detected)
278         # Trigger if more than n degrees field of view is obstructed
279         if num_detected > obstacle_detect_fov:
280             obstacle_detected = True
281         elif obstacle_detected == True and rst_obst == 1:
282             obstacle_detected = False
283         #else:
284         #   obstacle_detected = False
285
286
287 def can_bus_callback(data):
```

```
288
289     global charger_active
290
291     if data.data[1] > 0:
292         charger_active = True
293     else:
294         charger_active = False
295
296
297 def steering_angle_to_servo(angle):
298
299     # Convert radian angle to valid servo output
300     global max_steering_angle
301
302     servo_output = -(angle/max_steering_angle)*90 + 90
303
304     return servo_output
305
306
307 def gear_selector_to_servo(gear):
308
309     # Gear select to valid servo output
310     if gear is 2:
311         gear_cmd = 180.0 # Top gear
312     elif gear is 1:
313         gear_cmd = 90.0 # Mid gear
314     else:
315         gear_cmd = 0.0 # Low gear
316
317     return gear_cmd
318
319 def controller():
320
321     global joy_steering_angle
322     global dnn_steering_angle
323     global pp_steering_angle
324     #
325     global lt_steering_angle
326     global lt_speed
327     global lt_speed_output
328     global ang_output
329     #
330     global move_base_steering_angle
331     global joy_vel
332     global move_base_velocity
333     global current_velocity
334     global current_steering_angle
335     global gear_selected
336     global dead_switch
337     global wp_logger_active
338     global charger_active
339     global lane_center_offset_cm
340     global obstacle_detected
341     global sliding_window_median
342     global car_velocity_straight_speed
343     global car_velocity_cornering_speed
344     global car_velocity_charge_speed
345     global driving_mode
346
347     # For reverse hack
348     set_reverse = False
349
350     # Set up node
351     rospy.init_node('car_interface', anonymous=True)
352
353     # Publisher
354     controller_pub = rospy.Publisher('car_cmd', Twist, queue_size=10)
355     vel_pub = rospy.Publisher('velocity_real', Float64, queue_size=10)
356     led_pub = rospy.Publisher('led_mode', Int16, queue_size=10)
357
358     controller_msg = Twist()
359     vel_msg = Float64()
```

```
360     led_msg = Int16()
361
362     # Subscriber
363     rospy.Subscriber("joy", Joy, joy_callback)
364     rospy.Subscriber("ackermann_cmd", AckermannDrive, lane_tracker_callback)
365     #rospy.Subscriber("pf/viz/inferred_pose", PoseStamped, slam_vel_callback) # For move_base
366     rospy.Subscriber("slam_out_pose", PoseStamped, slam_vel_callback)
367     rospy.Subscriber("cmd_vel", Twist, move_base_callback)
368     rospy.Subscriber("pure_pursuit", Twist, pure_pursuit_callback)
369     rospy.Subscriber("car_vision/lane_center_offset_cm", Float32, lane_center_offset_callback)
370     rospy.Subscriber("dnn_controller", Twist, dnn_controller_callback)
371     rospy.Subscriber("wp_logger_active", Int16, wp_logger_status_callback)
372     rospy.Subscriber("aruco_charger_status", Int16, charger_status_callback)
373     rospy.Subscriber("scan", LaserScan, laser_callback)
374     rospy.Subscriber("CAN_bus", Int64MultiArray, can_bus_callback)
375     #rospy.Subscriber("halleffect_data", Float32, PIDControl)
376     rospy.Subscriber("dist_data", Float32, dist_callback)
377     #rospy.Subscriber("servo_fb_data", Int16, PIDControlServo)
378
379     rate = rospy.Rate(10) # hz
380
381     # LED modes:
382     led_off = 0
383     led_green = 1
384     led_green_blink = 2
385     led_blue = 3
386     led_blue_blink = 4
387     led_red = 5
388     led_red_blink = 6
389
390     time_zero = rospy.get_time()
391
392     while not rospy.is_shutdown():
393         # Manual control / logging mode (use blue button), joycontroller
394         if driving_mode is 0:
395             # Update status LED on car
396             #if wp_logger_active is True:
397             #   led_msg.data = led_blue
398             #elif wp_logger_active is False:
399             #   if charger_active is True:
400                 #   led_msg.data = led_red
401                 #else:
402                 #   led_msg.data = led_green
403             # Set velocity and steering commands for manual mode
404             if dead_switch is True:
405                 controller_msg.linear.x = joy_vel
406                 controller_msg.angular.z = steering_angle_to_servo(joy_steering_angle)
407                 #controller_msg.angular.z = steering_angle_to_servo(ang_output)
408                 #controller_msg.angular.z = steering_angle_to_servo(steering_selected)
409             else:
410                 controller_msg.linear.x = 90
411                 controller_msg.angular.z = 90
412             # Set gear select
413             controller_msg.linear.z = gear_selector_to_servo(gear_selected)
414
415         # SLAM with Pure pursuit controller
416         elif driving_mode is 1 and wp_logger_active is False:
417
418             # Update status LED on car
419             #if charger_active is True:
420                 #led_msg.data = led_red_blink
421             #else:
422                 #led_msg.data = led_blue_blink
423
424             # Set velocity and steering commands for autonomous mode
425             if dead_switch is True and obstacle_detected is False:
426
427                 # Dynamic speed control
428                 if abs(pp_steering_angle) > 0.1:
429                     # used this when cornering
430                     controller_msg.linear.x = car_velocity_cornering_speed
431
```

```
                elif charger_active:
                    # use this when charger is active
                    controller_msg.linear.x = car_velocity_charge_speed

                else:
                    # use this when going strait
                    controller_msg.linear.x = car_velocity_straight_speed

                # Steering control
                controller_msg.angular.z = steering_angle_to_servo(pp_steering_angle)

            else:
                controller_msg.linear.x = 90
                #controller_msg.angular.z = 90

            # Set gear select
            controller_msg.linear.z = gear_selector_to_servo(gear_selected)

        # AI controller
        elif driving_mode is 2 and wp_logger_active is False:

            # Set status led
            #led_msg.data = led_green_blink

            if dead_switch is True and obstacle_detected is False:

                # Dynamic speed control
                if abs(pp_steering_angle) > 0.1:
                    # used this when cornering
                    controller_msg.linear.x = car_velocity_cornering_speed

                elif charger_active:
                    # use this when charger is active
                    controller_msg.linear.x = car_velocity_charge_speed

                else:
                    # use this when going strait
                    controller_msg.linear.x = car_velocity_straight_speed

                controller_msg.angular.z = steering_angle_to_servo(dnn_steering_angle)

                    # AI controller
        elif driving_mode is 3 and wp_logger_active is False:
            #print('automatic lane detection active!')
            if dead_switch is True and obstacle_detected is False:
                controller_msg.linear.x = lt_speed
                controller_msg.angular.z = steering_angle_to_servo(lt_steering_angle) #
    steering angle
                #controller_msg.angular.z = steering_angle_to_servo(ang_output)
            else:
                controller_msg.linear.x = 90
                if obstacle_detected is True:
                    print('Obstacle ahead, emergency stop!!')
                #controller_msg.angular.z = 90

            controller_msg.linear.z = gear_selector_to_servo(gear_selected)

        # Current real velocity
        vel_msg.data = low_pass_filter_median(round(current_velocity, 2))

        # Store steering angle
        current_steering_angle = controller_msg.angular.z

        # Publish msgs
        controller_pub.publish(controller_msg)
        vel_pub.publish(vel_msg)
        led_pub.publish(led_msg)

        rate.sleep()


if __name__ == '__main__':
```

```
503     try:
504         controller()
505     except rospy.ROSInterruptException:
506         pass
507
508
509
```

E.3 Lane detection

Main parts of the code edited by author:
- Warping functions 2 and 4
- Pipeline functions 3, 4 and 5
- Find lane function

```python
import numpy as np
np.set_printoptions(threshold=np.inf)
import os, glob
import cv2
import matplotlib.pyplot as plt

def undistort_image(img):
    mtx = np.array([[708.4340756775686, 0, 317.9663110540382], [0, 724.3038241696117, 274.0865876256384], [0, 0, 1]])
    dist = np.array([0.09702126218642344, -0.1836878268546886, 0.01359685879119158, 0.007942342964235989, 0])
    undist = cv2.undistort(img, mtx, dist, None, mtx)
    return undist

def perspective_warp2(img,
                      dst_size=(640,480),
                      src=np.float32([(0.225,0.5625), (0.7266, 0.5625), (0.0546875,0.95), (0.90625,0.95)]),
                      dst=np.float32([(0.078125,0.0),(0.921875,0.0),(0.0546875,1.0),(0.921875,1.0)])):
    img_size = np.float32([(img.shape[1],img.shape[0])])
    src = src* img_size
    dst = dst * np.float32(dst_size)
    M = cv2.getPerspectiveTransform(src,dst)
    warped = cv2.warpPerspective(img, M, dst_size)
    return warped,M

def perspective_warp3(img,
                      dst_size=(640,480),
                      src=np.float32([(45,0), (560, 0), (45,480), (560, 480)]),
                      dst=np.float32([(0,0),(640,0),(0,480),(640, 480)])):
    M = cv2.getPerspectiveTransform(src,dst)
    warped = cv2.warpPerspective(img, M, (640,480))
    return warped

def perspective_warp4(img,
                      dst_size=(640,480),
                      src=np.float32([(150,270), (487, 270), (0,455), (640, 455)]),
                      dst=np.float32([(47,0),(585,0),(35,480),(594, 480)])):
    M = cv2.getPerspectiveTransform(src,dst)
    warped = cv2.warpPerspective(img, M, (640,480), borderMode=cv2.BORDER_CONSTANT, borderValue=(145,145,145))
    return warped

def perspective_warp5(img,
                      dst_size=(640,480),
                      src=np.float32([(45,0), (560, 0), (45,480), (560, 480)]),
                      dst=np.float32([(0,0),(640,0),(0,480),(640, 480)])):
    M = cv2.getPerspectiveTransform(src,dst)
    warped = cv2.warpPerspective(img, M, (640,480))

    return warped

def inv_perspective_warp2(img,
                      dst_size=(640,480),
                      src=np.float32([(241.8,208.25), (361.0, 208.25), (90,480), (507.48,480)]),
                      dst=np.float32([(40,0),(600,0),(40,480),(600,480)])):
    img_size = np.float32([(img.shape[1],img.shape[0])])
    src = src* img_size
    dst = dst * np.float32(dst_size)
    M = cv2.getPerspectiveTransform(src, dst)
    warped = cv2.warpPerspective(img, M, dst_size)
    return warped

def pipeline2(img):
    plt.imshow(img)
    plt.show()
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thresh = 120
    img_binary = cv2.threshold(img_gray, thresh, 255, cv2.THRESH_BINARY)[1]
    plt.imshow(img_binary,cmap = 'gray')
    plt.show()
```

```
    inverted = cv2.bitwise_not(img_binary)
    imgp,matrix = perspective_warp2(inverted)
    plt.imshow(imgp,cmap = 'gray')
    plt.show()
    imgp = cv2.bitwise_not(imgp)
    plt.imshow(imgp,cmap='gray')
    return imgp

def pipeline3(img, img2): #look for the lanes and yellow mark
    x0 = perspective_warp4(img)
    merg = np.concatenate((x0,img2), axis = 0)
    #plt.imshow(merg)
    #plt.show()
    img_gray = cv2.cvtColor(merg, cv2.COLOR_BGR2GRAY)
    dark_yellow =  (23,100,100)
    light_yellow =  (30,255,255)
    mask1 = cv2.inRange(cv2.cvtColor(merg, cv2.COLOR_BGR2HSV), dark_yellow, light_yellow)
    th3 = cv2.adaptiveThreshold(img_gray,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
        cv2.THRESH_BINARY,151,+20)
    if mask1.sum() > 500000:
        return 1, th3
    else:
        return 0, th3

def pipeline4(img,  img2): #look for the charging station
    x0 = perspective_warp4(img)
    merg = np.concatenate((x0,img2), axis = 0)
    dark_green = (45,35,45)
    light_green = (90,255,255)
    mask1 = cv2.inRange(cv2.cvtColor(merg, cv2.COLOR_BGR2HSV), dark_green, light_green)
    #plt.imshow(mask1)
    #plt.show()
    if mask1.sum()>1600000:
        return 1, mask1
    else:
        return 0, mask1

def pipeline5(img): #look for the red mark
    x,y = perspective_warp2(img)
    dark_red = (90,90,150)
    light_red = (150,130,220)
    mask1 = cv2.inRange(x, dark_red, light_red)
    if mask1.sum()>50000:
        return 1
    else:
        return 0

def get_hist(img):
    hist = np.sum(img[img.shape[0]//2:,:], axis=0)
    return hist

def draw_lane(pointsx,pointsy): #img,window_y,lane_center
    plt.plot(pointsx,pointsy,'rx')
    return

def find_lanes2(img,slices,lane):
        #fig, axs = plt.subplots(slices)
        right_lane = np.zeros(slices+1)
        left_lane = np.zeros(slices+1)
        sx = int(img.shape[0]/slices)
        sy = img.shape[1]
        stencil = np.zeros([sx,sy])
        copy = np.zeros_like(img)
        index_l=0
        index_r=0
        #window_y = np.linspace(0+sx/2,img.shape[0]-sx/2,slices) #window center (vertical)
        window_y = np.linspace(img.shape[0]-sx/2,0+sx/2,slices)
        for i in range(slices):
            stencil = img[i*sx:i*sx+stencil.shape[0],0:stencil.shape[1]]
            #imgplot = plt.imshow(stencil)
            #plt.show()
            histogram = get_hist(stencil)
```

```
            #fig.suptitle('Sliced Image Histograms')
            #axs[i].plot(histogram)
            if lane == 2:
                threshold = 2500
                left_lane[index_l]=None
                right_lane[index_r]=None
                for x in range(len(histogram)):
                    if histogram[x]>= threshold:
                        left_lane[index_l]=x
                        break
                index_l+=1
                for x in range(len(histogram)-1,0,-1):
                    if histogram[x]>= threshold:
                        right_lane[index_r]=x
                        break
                index_r+=1
            elif lane==1:
                #b,a = signal.butter(3,0.1)
                #filterhistogram = signal.filtfilt(b,a,histogram)
                minima = argrelextrema(histogram, np.less)
                if len(minima[0])>1 and max(histogram[minima[0][0]],histogram[minima[0][1]])<
    5000:
                    threshold = 8000 #8000
                else:
                    if min(histogram)!=max(histogram):
                        threshold = 8000 #8000
                    elif min(histogram) == max(histogram) or min(histogram)>5000:
                        threshold = -1000 #handles one lane only
                left_lane[index_l]=None
                right_lane[index_r]=None
                for x in range(len(histogram)):
                    if histogram[x]<=threshold:
                        left_lane[index_l]=x
                        break
                index_l+=1
                for y in range(len(histogram)-1,0,-1):
                    if histogram[y]<=threshold:
                        right_lane[index_r]=y
                        break
                index_r+=1
            min_lane_width = 150
            valid_points = 0
            points_found = 0
            new_point_r = 0
            new_point_l = 0
            new_point_sl_r = 0
            new_point_sl_l = 0
            sl = 0 #single lane detector
            #find the lane centerline -begin
            lane_center = []
            yout = []
            single_lane_x = []
            single_lane_y = []
            direction = 1
            l=0
            r=0

        for i in range((slices)-1,-1,-1):
            if right_lane[i]==right_lane[i] and left_lane[i]==left_lane[i]:
                valid_points+=1
                if (right_lane[i] - left_lane[i])> min_lane_width:
                    points_found +=1
                    #print(points_found)
                    lane_center.append((right_lane[i] + left_lane[i])/2)
                    yout.append(window_y[i])
                elif (right_lane[i] - left_lane[i])< min_lane_width and right_lane[i] >
    right_lane[i-1] and points_found > 0 and new_point_l == 0:
                    new_point_r += 1
                    lane_center.append(((right_lane[i+new_point_r] + left_lane[i+new_point_r
    ])/2)-(right_lane[i+new_point_r]-right_lane[i]))
                    yout.append(window_y[i])
                elif (right_lane[i] - left_lane[i])< min_lane_width and left_lane[i] <
```

```
left_lane[i-1] and points_found > 0 and new_point_r == 0:
                    new_point_l += 1
                    lane_center.append(((right_lane[i+new_point_l] + left_lane[i+new_point_l
])/2)+(left_lane[i]-left_lane[i+new_point_l]))
                    yout.append(window_y[i])
                elif new_point_r > 0:
                    new_point_r += 1
                    lane_center.append(((right_lane[i+new_point_r] + left_lane[i+new_point_r
])/2)-(right_lane[i+new_point_r]-right_lane[i]))
                    yout.append(window_y[i])
                elif new_point_l > 0:
                    new_point_l += 1
                    lane_center.append(((right_lane[i+new_point_l] + left_lane[i+new_point_l
])/2)+(left_lane[i]-left_lane[i+new_point_l]))
                    yout.append(window_y[i])
                else:
                    single_lane_x.append((right_lane[i] + left_lane[i])/2)
                    single_lane_y.append(window_y[i])
                    sl+=1
        if valid_points == 0:
            direction = -1
            return [],[],-1
        elif valid_points == sl:
            lane_center = single_lane_x
            yout = single_lane_y
            if lane == 1:
                if right_lane[-1] < right_lane[-2]:
                    r=1
                    direction = -10
                    #print ("right")
                else:
                    l=1
                    direction = 10
                    #print ("left")
            elif lane ==2:
                if right_lane[0] < right_lane[1]:
                    r=1
                    direction = -10
                    #print ("right")
                else:
                    l=1
                    direction = 10
                    #print ("left")
        #plt.show()
        return lane_center, yout, direction

```

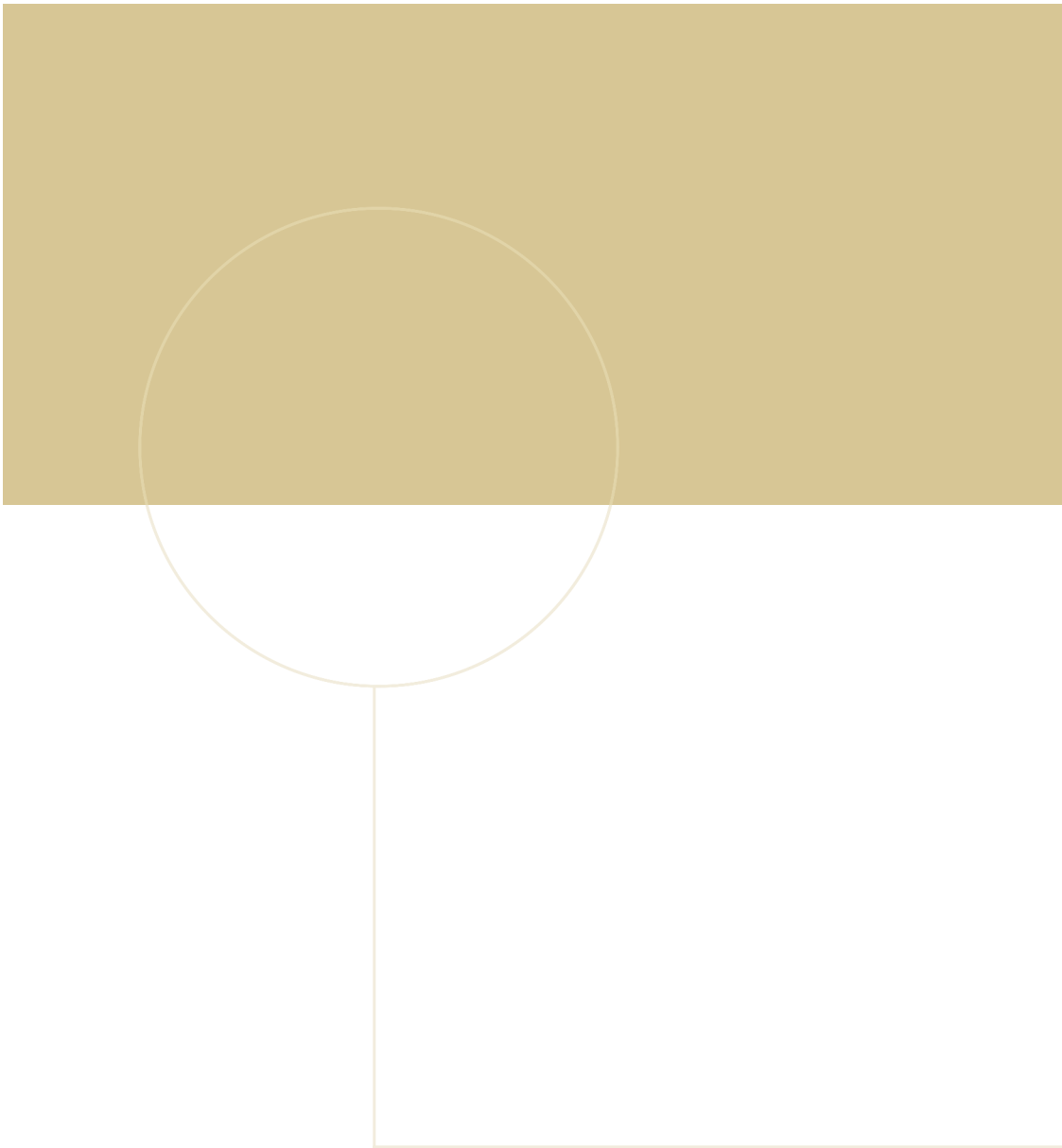
E.4 Simple trajectory

Main parts of the code edited by author:
- radius2p function
- pure pursuit function
- target index function
- arc3p function

```python
#!/usr/bin/env python
#import sys
#sys.path.remove('/opt/ros/kinetic/lib/python2.7/dist-packages') # in order to import cv2 under python3
#import cv2
#sys.path.append('/opt/ros/kinetic/lib/python2.7/dist-packages') # append back in order to import rospy

import numpy as np
np.set_printoptions(threshold=np.inf)
import os, glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import math
debug = False

l = 0.28
k = 2
lfc = 0.5
max_s_a = 0.44

def distance(p1,p2):
    return math.sqrt(pow(p2[0]-p1[0],2)+pow(p2[1]-p1[1],2))

def arc1pR(  x1 ,y1,x2,direction,radius ):

    centerx = x1+direction*radius
    centery = y1
    number_of_points = 100
    points = np.linspace(x1,x2,number_of_points)
    y = np.zeros(number_of_points)
    dist = 0
    for i in range(number_of_points):
        y[i] = math.sqrt(radius*radius-math.pow(points[i]-centerx,2))+centery
        dist+= distance([points[i],y[i]],[points[i-1],y[i-1]]) if (i>0 and i<=number_of_points-2) else 0
    if debug :plt.plot(points,y)
    end = len(points)
    return [points[end-2],y[end-2]],[points[end-1],y[end-1]],dist

def center2p(  x1 ,y1, x2, y2):
    l =0.28*480/0.547
    centery = y1
    centerx = (x1*x1-x2*x2-y2*y2-y1*y1+2*y1*y2)/(2*x1-2*x2+1e-10)
    #print('centerx',centerx, 'centery',centery)
    radius = abs(x1-centerx)
    #print('centerx',radius)
    direction = np.sign(centerx-x1)
    delta = direction*math.atan2(l,radius)#*180/3.14
    delta = np.clip(delta, -max_s_a, max_s_a)
    #print ('steering angle = ',direction*delta)
    #print('delta', delta)
    return radius, direction, delta

def radius2p (p0, x, y, v, fac):
    l = 0.28 * 480/0.37
    r = 0.0
    x_dir = 0.0
    offset = 1.3
    for i in range(len(x)):
        x2 = (p0[0]*p0[0]-x[i]*x[i] + (p0[1] - y[i])*(p0[1] + y[i] - 2*p0[1])) / (2*p0[0] - 2*x[i]+1e-10)
        x_dir += x[i]
        r += abs(x2-p0[0])
    radius = r / len(x)
    x_direction = np.sign((x_dir / len(x))-p0[0])
    if x_direction == 1:
        radius = radius * offset
    delta = (x_direction*math.atan2(l,radius)/fac)
    delta = round(delta,5)
    delta = np.clip(delta, -max_s_a, max_s_a)
    return delta
```

```
 69
 70 def purepur(x,y,v,yaw,p):
 71     ind = target_index(x,y,p,v)
 72     if ind < len(x):
 73         tx = x[ind]
 74         ty = y[ind]
 75     else:
 76         tx = x[-1]
 77         ty = y[-1]
 78         ind = len(x)-1])
 79     alpha = math.atan2(ty - p[1],tx - p[0]) - yaw
 80     Lf = k * v + lfc
 81     delta = -math.atan2(2.0 * l * math.sin(alpha)/ Lf, 1.0)
 82     delta = np.clip(delta, -max_s_a, max_s_a)
 83     return delta
 84
 85 def target_index(cx,cy,p,v):
 86
 87     dx = [p[0]-icx for icx in cx]
 88     dy = [p[1]-icy for icy in cy]
 89     d = [abs(math.sqrt(idx ** 2 + idy ** 2)) for idx,idy in zip(dx,dy)]
 90     ind = d.index(min(d))
 91     L = 0.0
 92     Lf = k * v * lfc
 93     while Lf > L and (ind+1) < len(cx):
 94         dx = cx[ind] - cx[ind-1]
 95         dy = cy[ind] - cy[ind-1]
 96         L += math.sqrt(dx ** 2 + dy ** 2)
 97         ind += 1
 98     return ind
 99
100 def arc3p(p1,p2,p3):
101     x1 = p1[0]
102     y1 = p1[1]
103     x2 = p2[0]
104     y2 = p2[1]
105     x3 = p3[0]
106     y3 = p3[1]
107     midpoint1 = [(x2+x1)/2,(y2+y1)/2]
108     midpoint2 = [(x3+x2)/2,(y3+y2)/2]
109     a1 = -(x2-x1)/((y2-y1+1e-10))
110     a2 = -(x3-x2)/((y3-y2+1e-10))
111     b1 = midpoint1[1]-a1*midpoint1[0]
112     b2 = midpoint2[1]-a2*midpoint2[0]
113     centerx = (b2-b1)/(a1-a2+1e-10)
114     centery = a1*centerx+b1
115     radius2 =   math.pow((p3[0]-centerx),2) + math.pow((p3[1]-centery),2)
116     number_of_points = 100
117     points = np.linspace(x1,x3,number_of_points)
118     y = np.zeros(number_of_points)
119     y2 = np.zeros(number_of_points)
120     dist = 0
121     dist2 = 0
122     for i in range(number_of_points):
123         y[i] = -math.sqrt(radius2-math.pow(points[i]-centerx,2))+centery
124         y2[i] = math.sqrt(radius2-math.pow(points[i]-centerx,2))+centery
125         dist+= distance([points[i],y[i]],[points[i-1],y[i-1]]) if (i>0 and i<=number_of_points
    -2) else 0
126         dist2+= distance([points[i],y[i]],[points[i-1],y[i-1]]) if (i>0 and i<=
    number_of_points-2) else 0
127     end = number_of_points -1
128     if y1>=0:
129         if (y1>centery):
130             nxt1 = [points[end-2],y2[end-2]]
131             nxt2 = [points[end-1],y2[end-1]]
132             if debug :plt.plot(points,y2)
133             direction = 1
134         else:
135             nxt1 = [points[end-2],y[end-2]]
136             nxt2 = [points[end-1],y[end-1]]
137             if debug :plt.plot(points,y)
138             direction = -1
```

```
139     else:
140         if (y1>centery)>0:
141             nxt1 = [points[end-2],y[end-2]]
142             nxt2 = [points[end-1],y[end-1]]
143             if debug :plt.plot(points,y)
144             direction = 1
145         else:
146             nxt1 = [points[end-2],y2[end-2]]
147             nxt2 = [points[end-1],y2[end-1]]
148             if debug :plt.plot(points,y2)
149             direction = -1
150     delta = -np.sign(centerx-x1)*math.atan2(l,math.sqrt(radius2))
151     return math.sqrt(radius2), [centerx, centery], nxt1,nxt2,dist,delta
152
153
154
```



NTNU

Norwegian University of
Science and Technology