TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Tengiz Pataraia 177329

EMBEDDED CONTROL SYSTEM DEVELOPMENT FOR AN AUTONOMOUS VEHICLE TEST PLATFORM

Master's thesis

Supervisor:	Raivo Sell
	Senior Research Scientist
Co-supervisor	Eduard Petlenkov
	Professor

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tengiz Pataraia 177329

SISSEJUHATUD KONTROLLISÜSTEEMI ARENDAMINE AUTONOOMSE SÕIDUKI KATSE PLATFORMILE

Magistritöö

Juhendaja:	Raivo Sell
	Venemteadur
Kaasjuht	Eduard Petlenkov
	Professor

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Tengiz Pataraia

06.05.2019

Abstract

Embedded control systems are one of the most challenging fields of software engineering Development process of such applications involves knowledge and understanding of several engineering disciplines. This thesis emphasises on one of the most distinctive applications of embedded systems – vehicle control system in robotics. The goal of this thesis is to give a technical overview of the vehicle platform, provide an extensive introduction on open and closed-loop control systems along with their practical implementation.

Some of the key takeaways from this study is closed-loop controller mathematics, modern practices and frameworks for programming ARM Cortex-M4 microcontrollers and creating industry standard electrical wiring and software flow diagrams of the control system.

Thesis presents an implementation of modern practices of mobile robot control system architecture such as modular control system for different actuators and sensors.

This thesis is written in English and is 47 pages long, including 6 chapters, 16 figures and 3 tables.

List of abbreviations and terms

DPI	*Dots per inch*
TalTech	Tallinn University of Technology
ARM	Acron RISC Machine
MCU	Microcontroller unit
UDP	User Datagram Protocol
CAN	Controller Area Network
UART	Universal Asynchronous Receiver-Transmitter
UGV	Unmanned Ground Vehicle
ATV	All-Terrain Vehicle
ROS	Robot Operating System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
ID	Identifier
I/O	Input/Output
HAL	Hardware Abstraction Layer
PID	Proportional Integral Derivative
SPL	Standard Peripheral Library
AI	Artificial Intelligence
ECS	Embedded Control System

Table of Contents

List of figures

List of tables

1 Introduction

Embedded systems have very diverse applications across multiple engineering disciplines. Range of these applications can be scaled from small consumer electronics such as digital watches or wearable devices to large stationary or dynamic complex systems such as space rockets, factory controllers, avionics or automotive industry controllers. The complexity of these embedded system processor units is as diversified as their application fields, however, in this thesis, we will be dealing with, microprocessors and microcontroller process units for medium and large scale embedded safety critical systems.

The thesis will contemplate different stages of development of the embedded software control system for autonomous and semi-autonomous all-terrain vehicle platform. The main aim of this study is to introduce the concept of a modular control system and emphasise more on the development of the embedded software for each module.

The primary goal of technical development of this thesis is to create embedded applications for different control units, design the most optimal control system for each driver, and create a codebase for the controller board that will ease the software creation of other control units on the specified vehicle platform for other students in future.

Secondary goal of the thesis is to describe different communication protocols on the vehicle, explain the working principle of high and low-level software on this autonomous vehicle. Describe existing hardware on the vehicle and create new efficient electrical diagrams for some sensors and actuators.

After the completion of the first prototype we should analyse how does our developed embedded application interact with higher level software and the physical vehicle which in this case is different driver units. Test, tune and validate the control system and compare it to other development methods used in the industry.

1.1 Project background and motivation

The main reason and one of the biggest motivation behind the development of new control system architecture for Uku was the project called ISEAUTO [5] which is the first self-driving car project development in Estonia. The project aim was to create a fully autonomous self-driving vehicle. The lifespan of this project was more than a year and lasted from June 2017 to March 2019. This period included development of mechanical structure, control system hardware, firmware and high-level software development. Due to the project complexity and short deadlines, the development processes of each primary subsystem such as high-level software (AI), control system hardware and software should have happened parallelly to each other.

Thanks to modern technologies it is easy to do comprehensive tests of autonomous vehicles in the laboratory in different virtual environments; however, developing the safety-critical system such as autonomous vehicles requires different means of testing and validation.

Let us further break down the development concept. At the beginning of the ISEAUTO project, there was an urgent need of having a test platform that would have functionalities precisely similar to the project so that controlling both, test platform ATV and the final car platform would be precisely similar to one another. Robot Uku, in this case, was the perfect mobile robot that could be upgraded to modern standards and support technical requirements set for the final autonomous bus platform.

Due to the aim and motivation of the development process, final product and architecture of Uku's control system are influenced by the final ISEAUTO self-driving car control system. Differences from one another are caused due to the different controllers, electronic circuits, control units, mechanical body and other reasons related to the use case and technical differences of each vehicle. Even though development process was going independently from one another main structure of the final control system for both bus and the ATV platform is derivative from one another except the fact that Uku's control system has highly modular control system approach comparing to the final bus platform.

One of the objective outcomes of this project was writing a research paper called "Modular smart control system architecture for the mobile robot platform" for

MMM2019 conference. This paper emphasizes on research of current technologies in today's mobile robotics industry.

1.2 Authors contribution

Following thesis thoroughly describes high-level software and hardware architecture of the vehicle, however main emphasis is on the development of embedded application and control system for this vehicle.

Besides the understanding of this complex system and taking part in the integration of the high-level software, the most significant contribution of this thesis and technical development of this project is creating firmware for controllers for this vehicle. Below is the list of the main contribution of this thesis:

- Explore existing drive, motor and hardware systems on the vehicle
- Chose general software frameworks for control unit such as Standard Peripheral Library versus Hardware Abstraction Layer Library (SPL vs HAL Libraries)
- Configure general data bus (CAN bus) for the controller module (STM32F303)
- Take part in ROS and vehicle integration
- Understand and be able to run most of the functionalities of ROS and configure high-level software on PC so that it controls the embedded software as expected
- Construct general embedded software architecture according to standards for the main drive controller module and create a codebase for future projects
- From the general codebase create a separate control system for each drive system
- Create new electrical wiring diagrams
- Create a software PID control system for controller units
- Tune PID controllers for most optimal autonomous and manual driving

To support these statements, a big part of the following text will be dedicated to the general description of the vehicle software and hardware architecture and working principles.

1.3 Outline of the thesis

Chapter 2: General theory and history of control systems, importance such systems in many fields of engineering. Provides the overview of modular control systems, the reasoning behind choosing this type of architecture and its advantages over other control architectures

Chapter 3: Gives the technical background necessary to understand the mobile robot vehicle platform.

- Describes functionalities of each motor drivers and presents them as control objects.
- Sub-chapter presents a microcontroller unit as a control module. Gives a thorough overview of the capabilities of this microcontroller unit (MCU) and reasoning behind choosing it
- Control system requirements, communication protocols

Chapter 4: Explains the overview of general software architecture on all levels of hardware. Connection diagram of all devices and message flow between primary vehicle components.

Chapter 5: General lifecycle of the control system development. The reasoning behind choosing the Proportional Derivative control. Step testing and tuning.

Chapter 6: The Last chapter concludes the full project development and presents several objective outcomes from it.

2 Control systems

First theoretical analysis of control systems has been done by James Clerk Maxwell in 1868 in the paper “On Governors” [1]. In this research, Maxwell studied several types of Governors which by that time were already successfully used in various products such as windmills and steam engines as a mechanical speed, pressure and distance regulators. Even though Governors have been used in industry, the theoretical basis of this device was not yet defined. For the first time in history, Maxwell introduced a distinction between proportional and integral control. Maxwell turned the analysis of Governors into the stability analysis of linearised systems. This method at a time was a breakthrough because the dominant method of stability study was energy conservation.

A governor is a part of a machine by means of which the velocity of the machine [...] is kept nearly uniform, not with standing variations in the driving power or the resistance. Most governors depend on the centrifugal force of a piece connected with a shaft of the machine. When the velocity increases, this force increases, and either increases the pressure of the piece against a surface or moves the piece, and so acts on a break or a valve. [1]

This reference above is an excerpt from Maxwell’s paper “On Governors” indicates that the emphasis of the study was to achieve zero-error regulation in the presence of constant uncertainties and therefore highlights the fact that “control theory” deserves paternity of Maxwell.

When we think of governors today, we can say that they are ancestors cruise control systems in today’s automotive industry.

Theory of Maxwell has gone through many iterations of mathematical and scientific advancements. Furthermore, the latest development of electronics and computer technologies created a possibility to solve problems with modern day control systems far more sophisticated and advanced then Maxwell’s Governors did. Some of the greatest achievements of human history like landing on a moon and coming back on earth safely on an Apollo mission became possible thanks to advanced control systems such as

Kalman filter [2]. One of the outstanding engineering achievements in the 21st century was the autonomous precision landing of space rockets achieved by private company SpaceX [3].

2.1 Embedded control systems

Having mentioned some of the backgrounds histories of foundations of the control system and some of the outstanding achievements of human history in control engineering we can start analysing the basis of control engineering more closely. Let us discuss the main building blocks of control science why is it useful and how is it connected to the topic of this research.

Roland S. Burns in his book Advanced Control Engineering describes: "Fundamentals to any control system is the ability to measure the output of the system, and to take corrective action if its value deviates from some desired value." [3]. Alternatively, to put it in other words control theory solves the problem of generating desired output from the plant which is the system that we want to control.

Nowadays most used control systems are embedded systems in wide ranges of applications. Embedded systems in most cases are complex systems; they consist of multiple subcomponents that can monitor a broad range of activities and development of such systems include the cooperation and joint effort of software and hardware designers.

This thesis will discuss control systems built on top of the embedded systems and authors contribution is mainly designing embedded software application.

The thesis will present two basic types of the control loop, open-loop and closed-loop control systems. Closed loop control systems, in general, are more sophisticated and precise because their input depends on the system output and therefore plant continually trying to adjust to the set position; therefore every closed-loop process has three main tasks: to measure, to compare and to adjust.

Open-loop control systems, on the other hand, are expected to operate without feedback and therefore are more likely to experience deviations from the setpoint because of any disturbances from the environment.

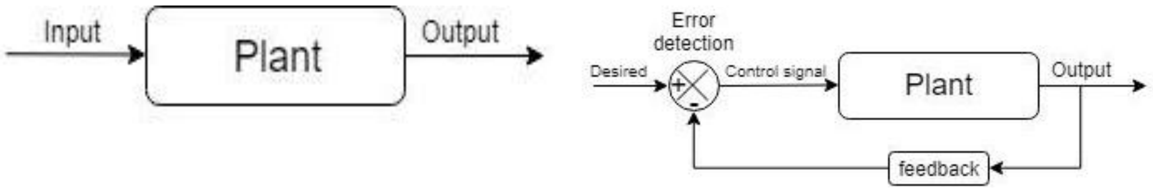


Figure 1 Control Loops

2.2.1 Embedded control system challenges

Besides the extensive use of embedded systems in the industry, there are multiple challenges that need addressing while designing these systems. Many research papers make their focus on problems such as flexibility of control systems, security and reliability reconfigurability and robustness cost efficiency of these systems [4].

Control system design presented in this paper will try to resolve some of these issues based on the modular architecture concept.

2.2 Modular architecture concept and novelty of the idea

Modular architecture concepts have been defined and used previously in many types of research and publications about mobile robotics. Development of this concept started more than a decade ago, and the reason behind this was to find a certain degree of overlap between two different fields of robotics. Modules in this concept would be components that would enable the transfer of the expertise developed in one research project to another.

Despite its popularity and thorough research in mobile robot industry this concept is not yet fully formalised. Professor S Virk describes concept of the “module” in his paper “CLAWAR: Modular Robots for the Future ” as follows: “*A module for mobile robots is described as any functionally complete device, or sub-assembly, that can be independently operated and can be readily fitted and connected to, or in combination with, additional modules to comprise a complete and functionally reliable system.*” [5] According to this definition, module should be plug and play component that can be reused in different systems and projects with minimal effort of integration. Let us take an example of a simple analogue sensor (potentiometer). Today if we need to integrate an analogue sensor into our system we have to go through several steps. First, we should use a separate microcontroller unit and then program it so that we can get readings out of it. This is one of the best and very well tested practice in the robotics industry today; however modular system concept cannot be considered as a “module”. According to the definition above analogue sensor can be considered as a module if it has a particular type of data bus on which the whole robot operates. This type of communication would make even the most straightforward analogue sensor independently operational and usable in every system that operates on that particular data bus.

Other than highly configurable nature of modular systems it is essential to note that they are more fault tolerant than traditional control systems; this is mainly the cause of the reduced dependency on a single control unit. This feature is crucial for safety-critical robots when for example the single unit is damaged in a system it does not cause the whole system malfunction and other parts of the machine to continue working as instructed. The system also enables to double most critical modules and create redundancy in between each subsystem.

3 Description of the vehicle

Robot mechanical platform was built several years ago in Tallinn University of Technology as a multi-purpose off-road vehicle capable of withstanding harsh weather conditions and suited to drive in an off-road environment. The robot has gone many iterations of minor development through these years and has been used as research material for Tallinn University of Technology students, lecturers and professors for various engineering disciplines. See the picture of the mobile platform below in figure 2.

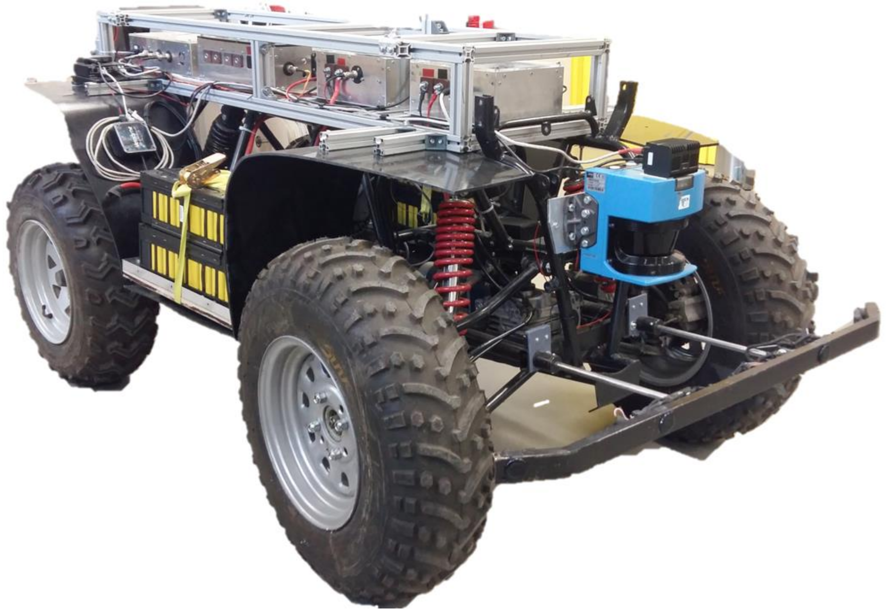


Figure 2 Mobile platform Uku

During the last, decade there were many changes to the mechanical body of the robot. Multiple experiments were done on this robot platform with different sensor sets and software algorithms. However, the central brain of the robot was still running on an old Windows machine. On the hardware side, all controllers, firmware and communication protocols of the machine stayed similar to its initial design. After many years, technology has advanced, and Uku was no longer capable of catching up with modern trends of technology, and therefore some hardware and software parts needed redesign from scratch.

3.1 Equipment layout on the vehicle

The main mechanical body of the robot Uku was built on the base of an all-terrain vehicle (ATV). The robot is equipped with three main driving electrical motors. The most powerful motor which is used to drive the primary vehicle is attached to the rear wheels and is controlled with its dedicated motor driver. Relatively smaller motor is placed in the front of the robot and is dedicated for steering. The smallest drive unit of the robot drive system is linear actuator which is used for breaking and parking operations in case the rear motor regenerative break is not able to entirely stop the robot.

Drive actuator controller units in the vehicle are distributed across the vehicle in separate boxes where they are enclosed together with the motor drivers and driver “motherboards”. See the picture of the general layout of the vehicle below in figure 2.

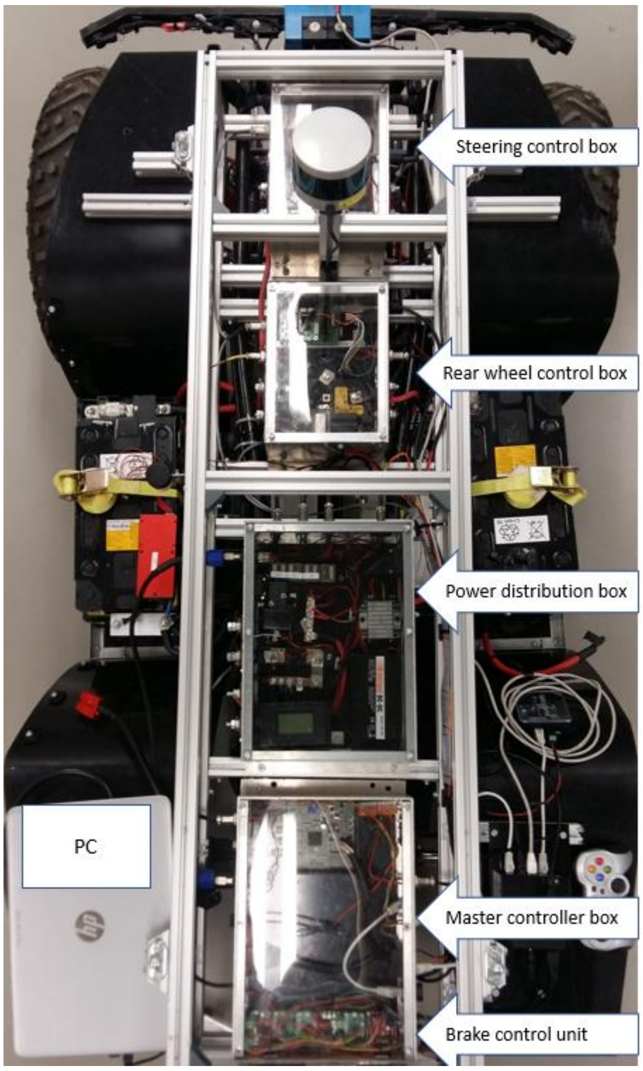


Figure 3 General layout of the vehicle

The robot is also equipped with multiple sensory units. Sensors on this robot can be separated into two groups. Ones that are connected directly to the control system and ones that are directly connected to the robot's central computing unit. The first group of sensors such as potentiometers and encoders are dedicated to the separate control system units and do not send their measurements to the central computing unit. On the other hand, the second group of sensors are main perception tools for robot and therefore are more sophisticated and require their software drivers for the computer. Sensors that belong in this group are 2d cameras, 3d and 2d laser scanners (Lidars), depth sensors, sonar sensors, GPS, gyroscope, and so forth. Full list of the actuator and sensory units can be seen below in the table.

Table 1 List of sensors and actuators on the robot

	Type	Manufacturer	Quantity	Purpose
Drive actuator	DC motors		1	• Responsible for main drive system • Regenerative braking
Steering actuator	DC motors	Lynch Motor Company Ltd	1	• Steering operations
Brake actuator	Linear DC motors	Elero GmbH	1	• Brake • Parking mode
Sensors	2d Lidar	Sick Rplidar	2	• Safety • obstacle detection/avoidance

	3d Lidar	Velodyne VLP16	1	● 3d mapping ● localisation ● navigation
	Sonar sensors	Saab	5	● safety ● obstacle avoidance/detection
	GPS	Xsense	1	● Localisation ● Navigation

3.2 Technical requirements

Software architecture details were set based on technical requirements of the ISEAUTO project the communication protocol between controllers on Uku would have to be similar to those in the final self-driving car. On the other hand, there was more freedom to choose the layout of controllers on the robot and make a new robust architecture. Other than requirements from the self-driving car project, the main focus was to create a safe, easily reconfigurable and scalable system where each module/microcontroller has a task to control each locomotion unit across the platform separately.

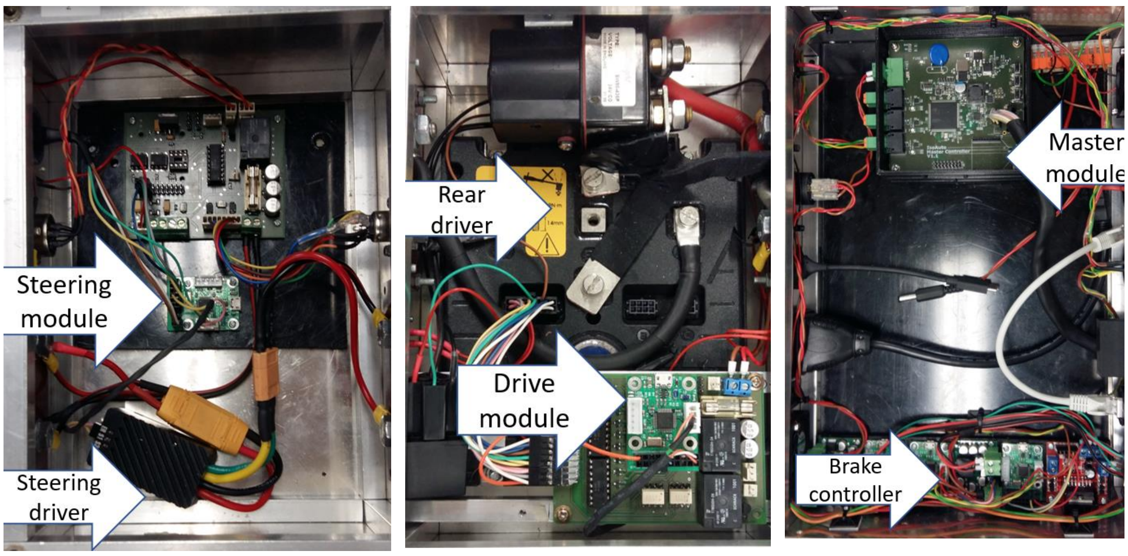
Previously communication on Uku between computer and low-level controllers were using serial protocols; however today serial protocols are not as fast and do not fulfil requirements for real-time applications. This time communication protocol from computer to the lower level systems happens with Universal Datagram Protocol (UDP). Messages from the Linux computer are sent to one controller that can receive UDP messages, and then this control unit distributes them to other controllers using Controller Area Network bus (CAN bus). The controller that translates UDP messages to CAN messages is called Master controller and has a role of the buffer from computer to the drive controllers, and therefore it does not do any calculations or interaction with other hardware.

The development process included choosing the correct controllers and writing firmware for it. However, each locomotion unit and sensor had a separate driver, and therefore they needed separate approach and development. All in all, three locomotion units needed separate firmware; all of them had dedicated controllers and control system development was the primary task of this thesis. Other than these controllers there were tasks with secondary priority such as controlling the light of the robot, reading sonar sensors and transferring values to the computer and reading encoder values to measure the speed of the robot.

3.3 Motor drivers as control objects layout

Control objects in this project were DC drive units, all of them different from one another with their I/O characteristics and load handling capabilities. As mentioned earlier there were three central motor drive units on the vehicle one for the linear dc motor for vehicle braking, one for the steering dc servo motor and one for the dc servo drive motor for rear wheels.

Each control module and other microelectronics are enclosed into the separate water-resistant control boxes. These boxes themselves connect with thin CAN bus wires to each other and with power cables to the motors and battery units. Even though all the STM32 microcontroller modules are the same layout inside all boxes are different. Figure 5 below shows the general picture of all control boxes.



a. Steering control box

b. Drive control box

c. Master and brake control box

Figure 4 Controller box layouts

3.2.1 Rear wheel motor driver

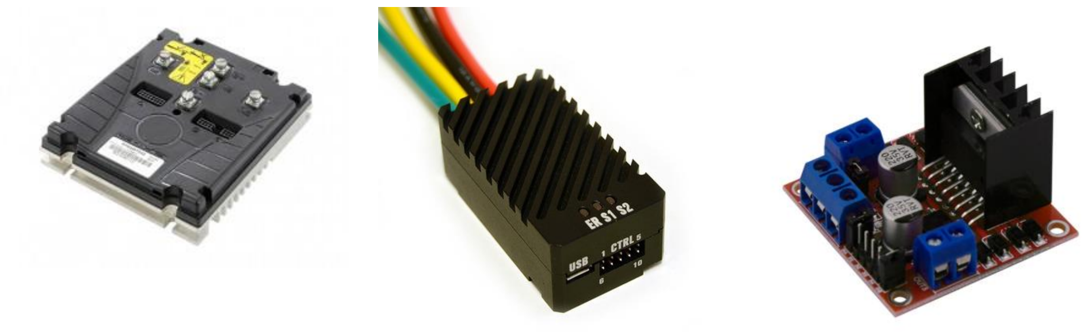
Rear wheel motor is the most significant DC motor on the vehicle and is equipped with a “Sigma“ drive system which is one of the most advanced controllers on the vehicle. This drive system is designed for heavy load applications such as industrial trucks, golf cars and material handling equipment. The unit itself is built to withstand shock vibration and extreme temperature environments. Control algorithms of this driver ensure efficient, smooth acceleration and deceleration without overloading DC motor with high torque. Motor control algorithms also ensure safe reversing and motor regenerative braking while moving with fast speeds. See the picture of the brake controller in figure 4. A.

3.2.2 Front motor driver

The front motor is equipped with 60A motor controller from RoboClow family of motion controllers. This drive unit is the very versatile programmable unit with different control options. The driver also supports multiple feedback devices and therefore can be used as an open-loop or a closed-loop speed or position control units. In our application feedback functions of this driver is not utilize; however, a closed loop control system is implemented on our controller for more flexible control options. See the picture of the river below in figure 4. B.

3.2.3 Brake motor driver

Finally, the third controller is a simple H-bridge with the peak current up to 2 amperes. This controller is used for controlling a linear actuator which is cheap easy to implement and easy to change in case it brakes. The braking system for this vehicle is used only when motor regenerative braking is not enough to stop the vehicle or in cases when we want to park the vehicle on a downhill. See the picture of the controller in figure 4. C.



A. Rear drive controller link

B. Front drive controller link

C. Brake controller link

Figure 5 drive controllers

3.2.4 Electrical connections

One of the contribution while designing the control system for the vehicle was to take part in the design process of the electrical diagrams for each control module. Priorities for designing electrical diagrams were choosing right electrical components and came up with the safe and efficient installation of all control modules, switches sensors and other vehicle equipment.

Below in figure 6, you can see the general control system wiring layout. As it is presented on the figure, the power source for all the modules on the vehicle is lithium ion battery pack which is later distributed into two power lines: one 5 volt line for microcontrollers and one 12 volt power line for the motors, lights and other high power consumption devices.

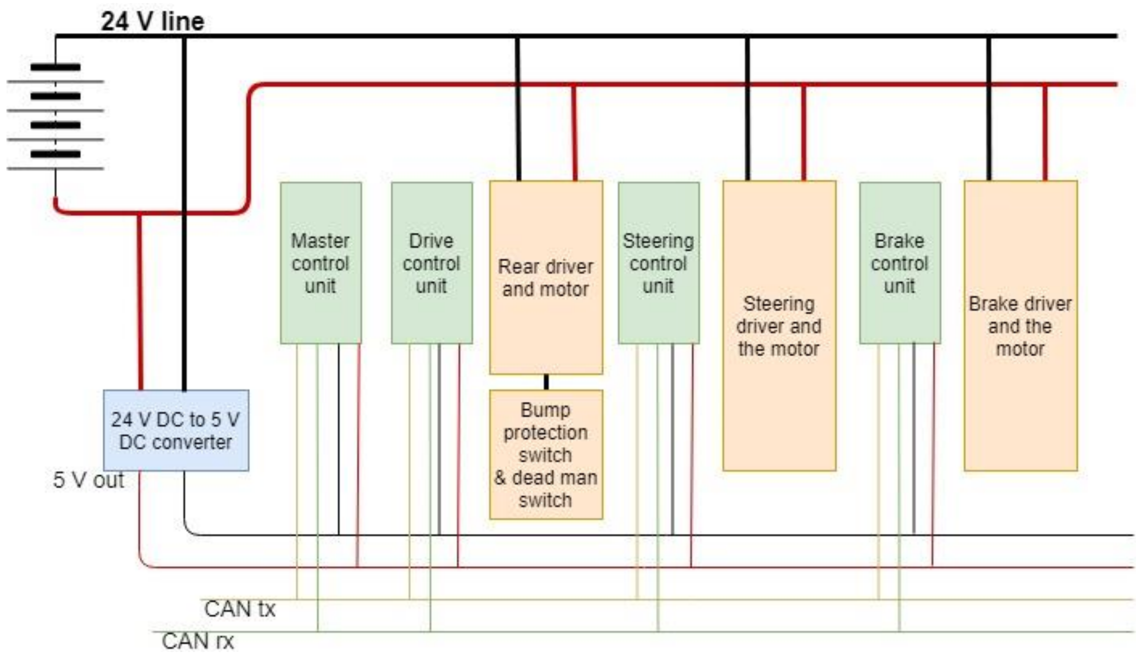


Figure 6 Electrical wiring

Power line for microcontrollers was separated from the battery pack by DC/DC step down converter which converts 24 volts into 5 volts of electrical potential. Connecting all controllers on the same power line means that startup and shut down for all controllers will happen simultaneously. Arranging microcontrollers on one line also enables to power on and off all microcontroller units simultaneously from one control switch (push button) on a power line.

CAN bus and power cables for microcontrollers are enclosed in one 4-strand circular wire, one twisted pair for the CAN bus and one twisted pair for the power line. End of each line was enclosed in industry standard “male” connectors which would directly plug into each control box from outside into the wall mounted “female header”. Control from the wall mounted header extends on the other side of the wall and directly connects to each microcontroller unit. This type of mechanical structure uses the controller box as a buffer which reduces the shock impact in case of rough road conditions on the wires and increases the overall safety of the vehicle.

Mechanical design of this connector allows connection of male and female headers only in one direction which reduces the risk of reverse connection, protects the vehicle from water and standardises the electrical circuit. Building such small details in the vehicle improve the overall integrity and quality of the whole control system and the vehicle in the long run.

3.4 Control system equipment

Modular architecture concept was applied to the base system Uku in several steps. The first step was getting the prototype of the master controller that would ensure delivery and translation of UDP messages from the computer to the other controllers on the CAN bus. A prototype of the master controller was designed and programmed in the scope of ISEAUTO project final car, and therefore this thesis does not include the development of firmware for this controller. The second step included the creation of the concept of the modular control system, integration and programming of all motor and sensor drivers on separate controller units.

3.2.1 UDP and CAN bus protocols

As mentioned earlier control system of Uku runs on two communication protocols (data bus): Universal Datagram Protocol (UDP) and Controller Area Network bus (CAN bus). Unlike other communication protocols, UDP does not have any flow control feedback and error correction mechanisms. Therefore, it is used for applications where fast and continues data exchange is required; such applications are video and audio streaming and other multimedia applications.

The main advantage of UDP is having no retransmission and flow control concepts. In scenarios where the vehicle is moving, and there is a small fault between the computer and the vehicle control it is crucial to continue working without error checking and data retransmission. System data retransmission in such scenarios might cause car deviation from the path and result in a crash because of data delivery latency.

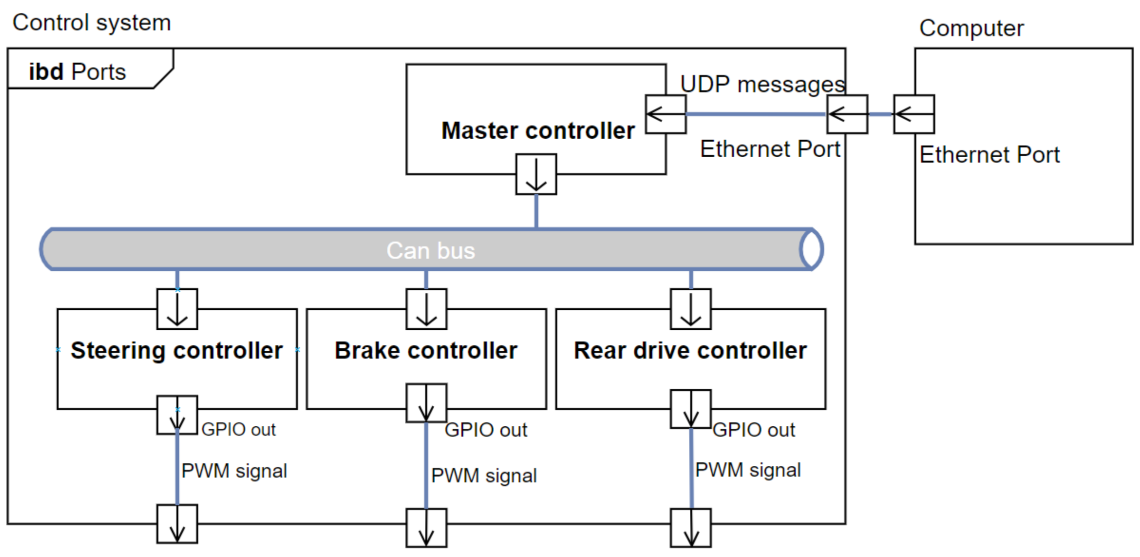


Figure 7 Internal block diagram [6]

3.2.2 UDP frame

UDP messages from the computer come into predefined frame. The current configuration of the UDP frame contains 12 bytes and includes ID of the message and set value of the message. Message IDs on UDP and CAN bus are the same. Below on table 2, you can see frame structure. It is important to note that frame length is scalable depending on the number of modules, control units and control variables available on a bus.

Table 2 UDP message frame example

Byte#	**Datatype**	**Data**	**Endian**
1	uint8_t	Speed ID	Big endian
2-5	char[4]	Speed value	Big endian
6	uint8_t	Steering wheel ID	Big endian
7-10	char[4]	Steering wheel value	Big endian

11	uint8_t	Brake ID	Big endian
12	uint8_t	Brake value	Big endian

3.5 Microcontroller board description

All control units on the drive system are based on one hardware which runs on ARM Cortex M4 MCU (microcontroller unit) with the serial number STM32F303CBT6. This specific hardware was developed before the start of the project outside the university by one Estonian private company. Initially, this controller was custom design for the back wheel driver controller board to fit on top of the motherboard pinouts of the drive control unit. For the simplicity of the whole system, this very hardware was used as a controller for other drive units as well. Main advantages of this microcontroller unit is that it is compact, has 27 available input-output pins and supports several communication interfaces such as CAN, I2C, SPI, UART and USB. Out of these protocols CAN bus is the data bus that we will be using and discussing in this study. Out of 27 I/O pins hardware design of the board enables us to program 21 I/O pin and reconfigure it based on the use case. Figure 7 below represents the ability of each MCU pin outlined in green and one use case of each pin outlined in red.

The nature of this control system design ended up being useful while developing the firmware for the drive system and sensors in many ways. The first reason was that because all the drive systems would have to be programmed on the same microcontroller unit that would mean less redundant work while setting up communication protocols for different microcontrollers. Let us review two separate use case where we need to set up ADC channel (analogue to digital converter) for one sensor and PWM light control for front lights which are both controlled from the PC through the master controller. This would mean that many pins on the MCU would stay unchanged such as CAN RX/TX pins, USART1 TX/RX (Universal Synchronous/Asynchronous Receiver/transmitter) pins, other LED (light-emitting diode) indicators which are on the board already. In such case only a single pin would have to be programmed in one case it is PWM on let us say PB0 and ADC on PA0.

This concept was used for the design of most of the control units on this control system where some pins such as CAN bus RX/TX baud rates, update frequencies and LED indicators are same through the whole control system. Another advantage of this type of design is that it also enabled us to create a specific codebase which is easily reconfigurable and understandable for future students and can be easily modified in case of some driver updates or other sensor edition.

Due to the research environment in the universities, this aspect of having one codebase and one controller for multiple sensors and drivers is crucial for laboratory workers and academic stuff for several reasons. One main reason other than its functionality is that students who write their thesis or a course project based on some university equipment are working in laboratories in short periods of times where this time in most cases is spent in understanding the system and getting environment and set up such as software and hardware ready for the actual work. Therefore having a ready codebase and instructions where most communication protocols are configured plays a huge role in the speed of development for future projects and make the process more automated for research assistants.

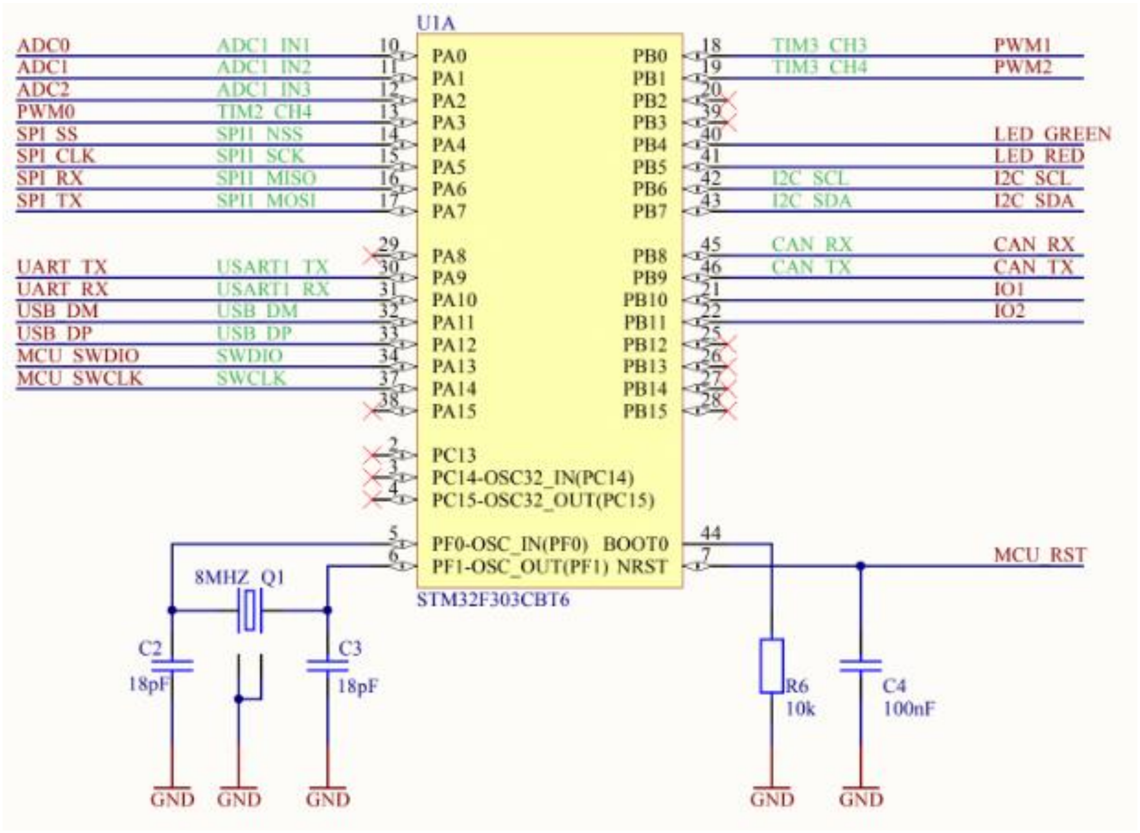


Figure 8 available GPIO pins on the controller

4 General vehicle software architecture

Overall control and autonomous operations of the vehicle happens on three hardware and two software levels. Three-level hardware control represents the base level controllers/modules that are directly connected to sensors and actuators, the second level is the master controller, and the third is PC. Two level software control is represented firstly by high-level software in PC which takes care of autonomous, manual driving and decision making and secondly firmware which runs master and drive controllers. The simplest way to understand the working principle of the whole system we should understand how the different hardware units are interconnected. Figure 5 shows the diagram of the hardware layout and is one of the best representations of the whole control system. The top part of the figure shows high functional sensor sets which are perception tools for the robot and is used for vehicles autonomous driving. Next component of the system is PC which is the central brain of the vehicle and runs on Ubuntu operating system which itself is one of the most popular Linux distribution based on Debian.

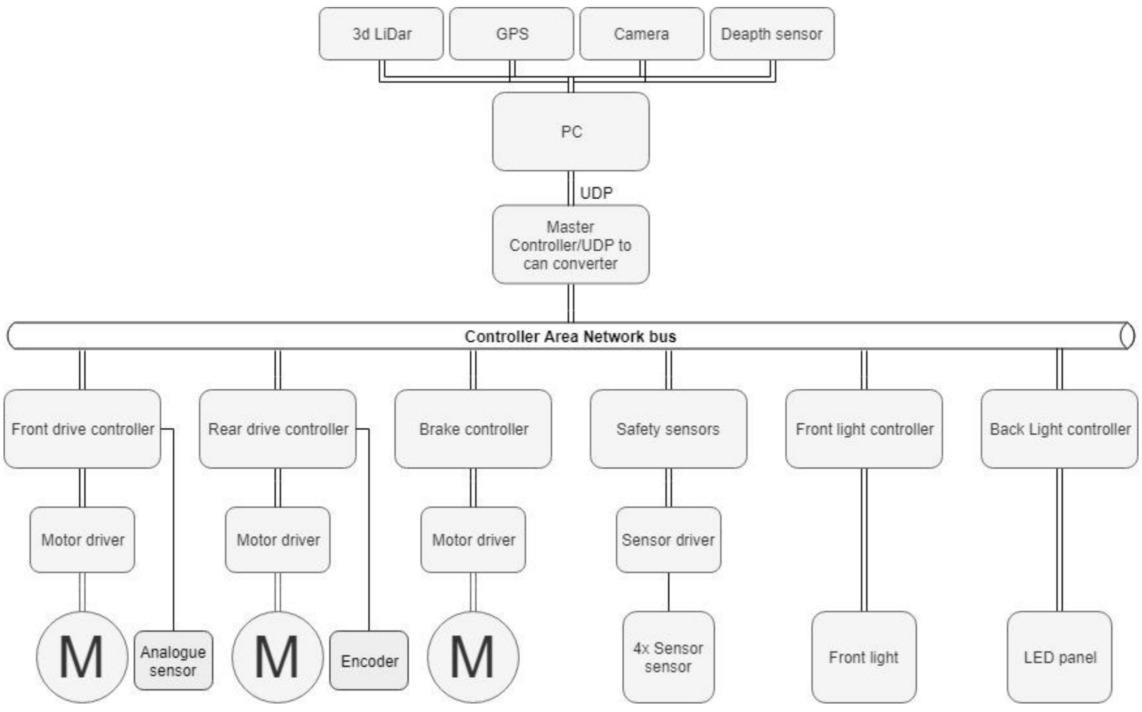


Figure 9 hardware architecture block diagram

4.1 Robot Operating System

Nowadays many new cutting-edge robot technologies are based on the Robot Operating System. Our robot Uku is no exception, and it is taking advantage of ROSs core features. As it is open-source software that allows inter-process communication and provides all the tools to build a robust system while keeping sanity on all levels of communication. ROS being a robotic middleware it provides excellent tools to organise complex robotic systems and have communication easily on both low-level firmware and higher-level hardware abstraction layer. One of the significant advantages of ROS is that one can have communication between many independent nodes on the number of independent architectures, e.g. Arduino publishing and laptop subscribing different streams of data. ROS itself is a master-slave communication where we have one ROS master and many different slave nodes. One of the main tasks of ROS is to run multiple processes simultaneously on different devices, pass messages between processes, manage packages from all processes, provide a different driver for many types of sensors. In pursuance ROS to tackle these problems, it uses several simple paradigms:

- Nodes - executables that publish or subscribes streams of data in the form of message, topic or service.
- Master- registers all nodes and directs them to each other
- Messages- a data structure that is published into the topic
- Services- contain request and response message information.

4.2 Autoware

Software development of Uku went through multiple iterations of development during the last one year. The development process was done mainly by the software team of ISEAUTO project; however software integration to the real platform was the result of the joint effort of both teams.

The core of the software stack used in ISEAUTO was based on one of the most popular open-source software for self-driving vehicles called Autoware [7]. This software is developed and runs on the Ubuntu operating system which is Linux based distribution and ROS which is a middleware for sensors hardware and the vehicle controllers.

Autoware solves many problems based on the latest autonomous vehicle trends and technologies. Main features of Autoware can be divided into three separate sections, perception, decision making and mission planning. These three sections could as well be divided into other subsections of computational tasks such as localisation, object detection, mission and motion planning, intelligent algorithms for decision making and so forth.

As mentioned earlier Autoware is based on the Robot Operating System which provides a strong foundation of the whole software architecture. Autoware benefits from ROS in several ways. Firstly after installation of Autoware, installing multiple sensor drivers separately is no longer necessary because it already has integrated and field tested many ROS sensor drivers which are maintained by the ROS community. Other then sensor drivers Autoware successfully utilises many software packages such as Gazebo for 3D simulations, OpenCV for computer vision, segmentation and tracking, PCL (Point Cloud Library) for LiDAR data processing, Cuda (Compute Unified Device Architecture) libraries for parallel computing for GPU image processing for computers that have NVIDIA GPUs.

Diagram figure 6 represents the diagram of the core software blocks running on the main computer of the vehicle. Diagram itself is the derivative of Authorware's software stack description figure from the paper “Autoware on Board” [7] and is edited based on the latest use case of the Robot Uku.

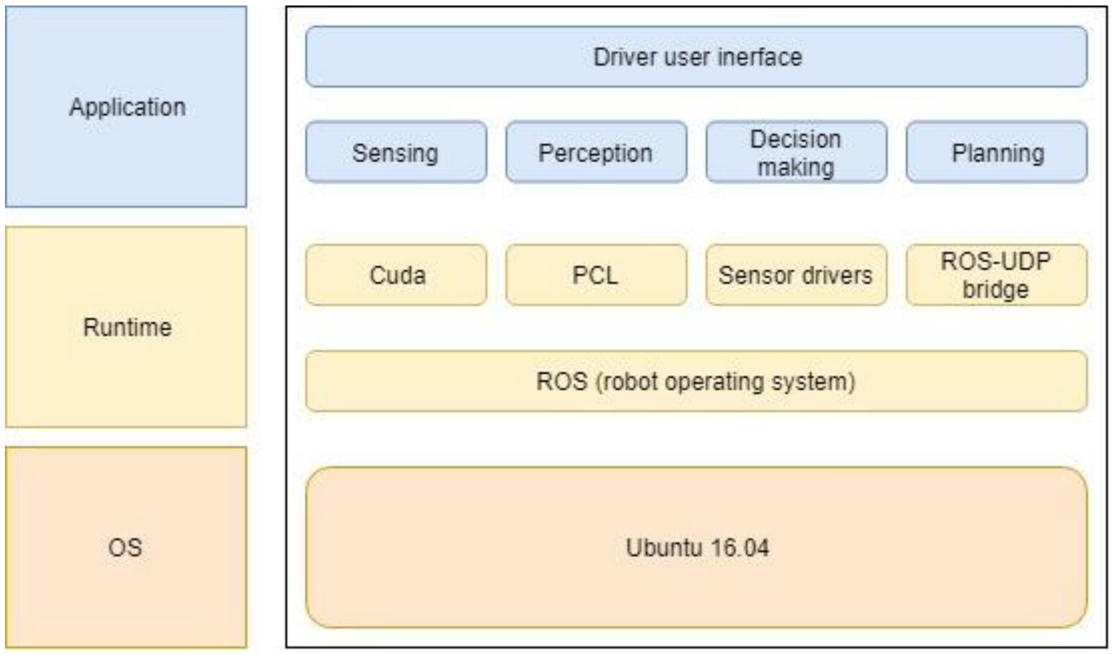


Figure 10 Software diagram of the vehicle

Understanding of higher level software helps to imagine the whole system better and see how commands are generated for electronic control units. Figure 6 describes a software stack diagram of the whole system; however, since we are dealing with control systems, we are most interested in several blocks from this diagram.

Control of the vehicle from upper-level software happens with two different message types: velocity command and steering angle command. These two messages are set points for control units of the vehicle control system. This means that if Autoware commands to decelerate high-speed control system should be able to autonomously decide to decelerate with the help of motor regenerative brakes or using brake controller. Generation of these two messages happens is done by motion planning node. Another block of software that takes part in all control operations is “ROS-UDP bridge”. This stack of software was built as part of the final ISEAUTO bus platform. The main task of this node is to translate ROS messages from motion planning node to UDP messages and send it out through ethernet terminal to low-level controllers.

4.3 Embedded control system software architecture

This sub-chapter will present the general software flow diagram which includes specific function of each module and explains similarities of error handling processes across three drive systems: for steering, rear wheel and brake controllers.

Embedded control systems (ECS) in many research papers are represented as a separate class of embedded systems mostly because of their mission-critical activities and dynamic requirements of the control software [8]. Development of ESCs in the, for example, automotive or avionics industry is a prolonged process. Robustness and reliability is the cornerstone for these ECS applications and therefore software development, testing and validation process happens parallelly in multiple iterations of the development process which can last years and sometimes decades. Usually, transport vehicles do not support embedded application Over the Air Updates (OTA), and therefore their firmware update is done once in many years during the vehicle maintenance process.

Even the embedded application designed in this project is supposed to run on a single vehicle it is still able to cause some damage since it operates in public places and therefore the vehicle can be considered as a critical safety system.

4.2.1 Drive system flow diagrams

Starting the embedded application design process is always related to the thorough understanding of the physical environment and electrical components to which the controller is interacting.

Various electronic components such as analogue sensors, encoders, motor drivers need the entirely different configuration of MCU I/O pins, internal and external clocks, PWM settings and control logic setup. Even though all the drive control software needed different setup of peripherals, some software blocks are similar on all control panels.

Let us start a control **flow description** by looking at the figure 11 diagram. As said earlier all control units are on the same power line and therefore their startup time is simultaneous. The graph below is constructed to emphasise this principle and show three different process flow which is initiated from a single point when the vehicle platform was powered up.

Software blocks in this diagram are divided into several units. CAN bus receiving and message handling unit, message value handling unit and finally value conversion unit into different PWM signals depending on the corresponding motor driver type. One of the primary aggregates of this figure is that: due to the modular architecture concept, initial process and error handling for CAN bus messages on each control unit are similar to one another. The main difference of control module software is the configuration of peripherals and the way these control processes are handled.

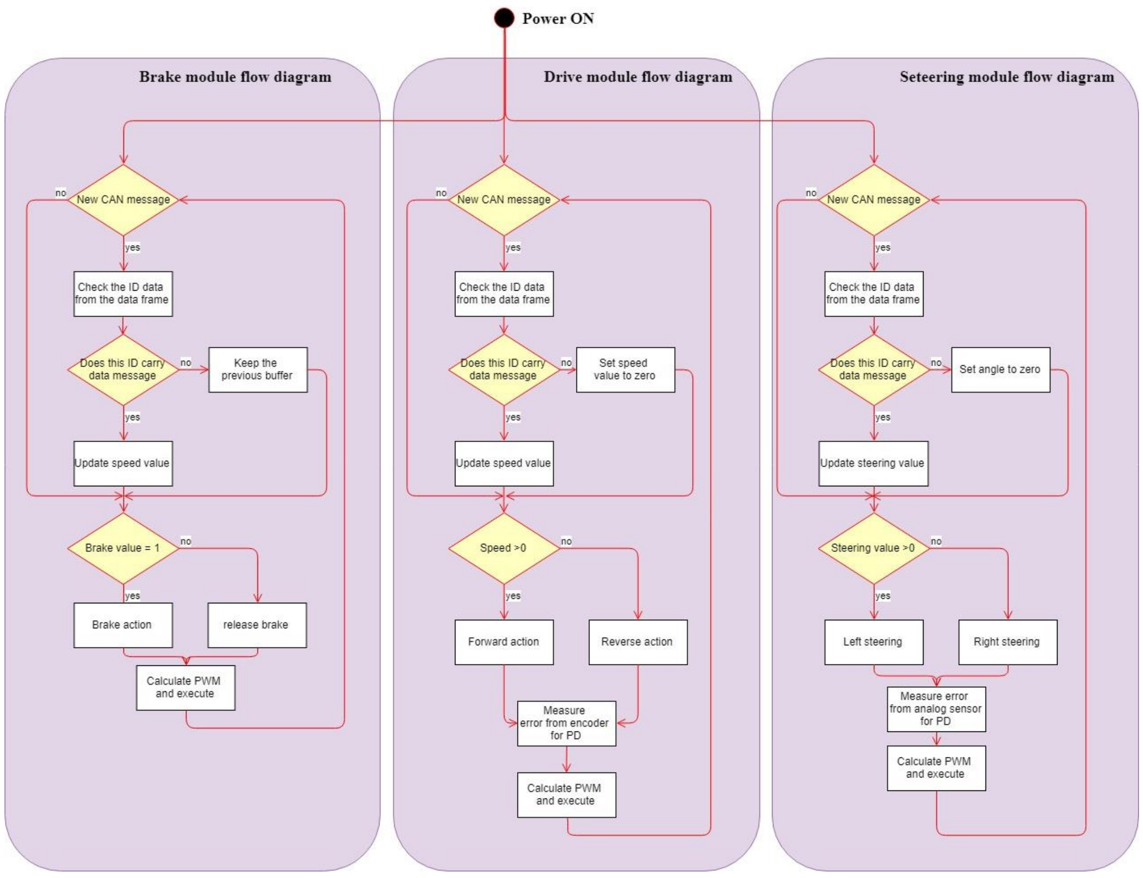


Figure 11 Flow diagram of three control unit

Brake module control flow represents an open loop control system and is easiest to handle because the controller receives Boolean 1 or 0 message whether or not brake should be pressed or not.

Steering and drive control units, on the other hand, are more sophisticated closed-loop control systems and include feedback message receive and pwm calculation functions. Proportional Derivative control process will be thoroughly described in next chaptar of this thesis.

These flow diagrams do not show all the initialisation processes, but it instead presents control loop after CAN bus message is received. The figure also does not include a diagram of how PID control processes but control flow is represented instead in blocks of software operations.

5 Vehicle control Implementation

This chapter presents each step of the firmware development of the modular control system for each module. The outcome of the overall control system for this vehicle is to design control processes, find sufficient control algorithms and determine needed control loop components for each module to guarantee the smooth control of the vehicle in both autonomous and manual mode. The main focus for the control process design was to reduce the variability of the process value, increase efficiency to make processes less maintenance dependent and ensure overall safety.

Process control as every field in engineering and science has some internationally defined terms that help us define specific processes. These definitions are rather trivial; nevertheless, they help to keep clarity while describing the control process.

- Set value - set point, the value that is desired to be maintained
- Measured variable - plant output, a variable which should be kept at the designated set point
- Error- Difference between the measured variable and the set value.
- Control algorithms - mathematical expressions of the control function that we are trying to achieve
- Disturbance - undesired change from the environment.

5.1 General procedure for control system design

In general, the first step of the design process is to establish the system goals and problems it solves after the resolution of the project. The second step is to assign variable to the hardware we desire to control. The third step is to write specifications and accuracy requirements and finally, based on the precision requirements chose the proper feedback devices and in some cases actuator units. After setting these practical guidelines, part includes the setup of the hardware and the software environment on a test bench. See if the given microcontroller can satisfy the set requirements and then configure GPIO and communication protocols.

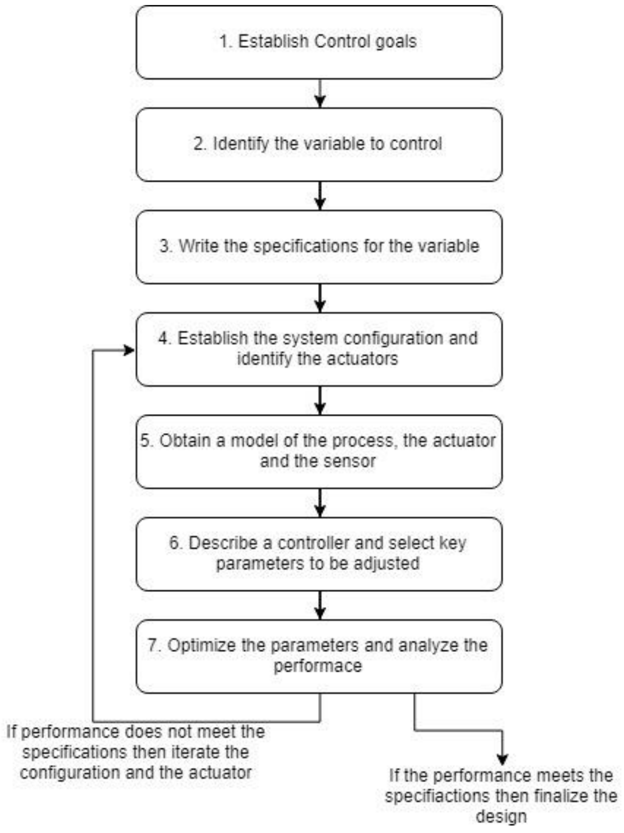

1. Establish Control goals
2. Identify the variable to control
3. Write the specifications for the variable
4. Establish the system configuration and identify the actuators
5. Obtain a model of the process, the actuator and the sensor
6. Describe a controller and select key parameters to be adjusted
7. Optimize the parameters and analyze the performace
If performance does not meet the specifications then iterate the configuration and the actuator
If the performance meets the specifiactions then finalize the design

Figure 12 control system setup guidelines

More precise guidelines of the design and implementation of control systems are shown in figure 6 above. These guidelines of the control system development which was presented in the book by Mutambara in the book "Design and Analysis of Control Systems" [9].

5.2 Choice of PD control

In most cases, PID control of the object was sufficient, easy to implement and robust way of control. One of the focus of designing the control system to create a somewhat unique system so that different motor controllers would use the same PID but with different gain parameters and coefficients.

Main advantages of creating a PID controller is that it is easy to implement, highly reliable, easy to test/troubleshoot and easy to understand. PID itself is an acronym, and it stands for the Proportional Integral Derivative. These terms describe how the error is handled during control operations. This means that in proportional controller error will be multiplied by a constant number. Let us call this number Kp. In the integral path, the error is multiplied by Ki and then integrated. In final derivative action, the error would be multiplied by the Kd and differentiated. After all these mathematical operations result is summed to produce controller output.

It is important to note that having all actions of PID is not needed in most of the applications. Today industry implements four basic behaviour types of PID. P controller, PI controller, PID controller and PD controller. All these controller types have their advantages and disadvantages depending on the control object specifications like stable or unstable and so forth.

Comparative study of PD, PI and PID controllers have been done published in international journal of engineering and science (IJES) [10]. Based on this study using a combination of proportional and derivative action (PD) was most reasonable for multiple reasons. PD action characteristics in this study were derived from the tests on a single joint robot which can be considered a similar system to a steering controller of our test platform. Based on the results of this study PD action has a fast response to the set-point, smaller steady-state error, Large disturbances and noise present in the system are handled, oscillation cancelling and damping of overshoots [10]. Characteristics described in this test results were most suitable for our system because of the disturbances from the feedback sensors and critical consequences in case of an overshoot of the setpoint which was handled by the PD controller.

5.3 Proportional derivative control implementation

From the control system set up guidelines description above, one of the main tasks of the process is the choosing of feedback sensors and test bench setup of each control plant. Control objects ware already chosen (see control object description subchapter) now we had to choose feedback sensors for each control process.

The mechanical structure of the steering was designed in a way that degree of rotation of the steering motor was restricted from 60 to - 60 degrees of rotation. A small degree of rotation enabled us to have a single analogue sensor (potentiometer) as a feedback mechanism where the analogue output pin is the indicator of the wheel angle position. On the other hand, the rear, the wheels control system used incremental mechanical encoder for feedback control which is attached to the rear shaft of the vehicle.

The PD control algorithm for each controller was designed similarly so that the implementation of the PD control becomes as easy as just changing the data input parameters and controller coefficient.

Let us review the example of steering and velocity control processes. Figure 7 below presents a control system block diagrams of steering and rear wheel controllers. Left, side of the figure shows the control of the rear wheels, and the right side shows the control of the steering operation. Difference between these control systems is only in terms of implementation and setup of the GPIO pins for PWM generation and sensor value readings.

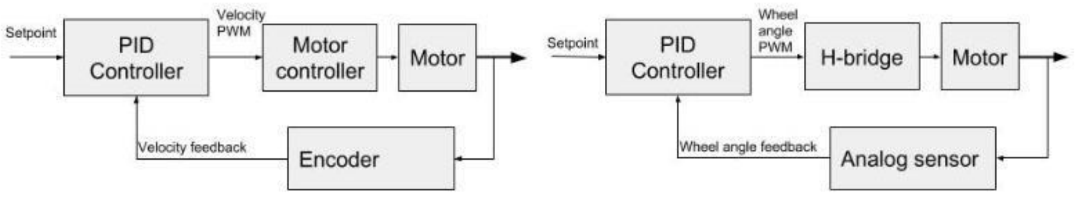


Figure 13 Control system block diagrams for steering and rear wheel drive

5.4 Proportional derivative control of the steering

After all components of the system were chosen now we had to configure microcontroller. As mentioned in the previous section steering controller uses the analogue sensor for feedback. To get analogue sensor readings to the microcontroller unit pure Analogue to Digital Converter (ADC) was set up on one GPIO pin which would measure an analogue change from the potentiometer output pin. After successfully integrating potentiometer to the microcontroller, it was necessary to determine what were the values of the potentiometer in different wheel positions to determine maximum angles vehicle mechanical structure can make and record corresponding potentiometer values.

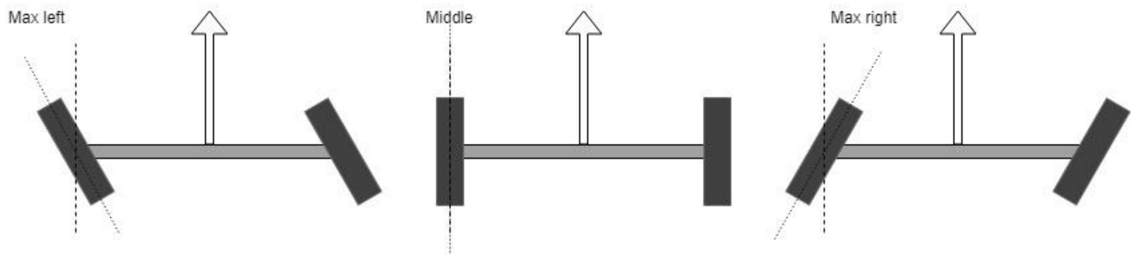


Figure 14 Wheel angle measurement positions

Most notable were measurements in three different steering position, extreme left, middle and extreme right. Top view wheel illustration of this three position is shown above in figure 10.

Table 3 Angle measurements

	Angle [rad]	potentiometer values from ADC
Max left	1	1865
Middle	0	1422
Max right	-1	1020

Measurement results are presented in table 3 above. After we know maximum and minimum angles of the vehicle, it is necessary to configure software on the ROS side to define steering angle limits for this specific vehicle to guarantee that computer sends steering floating values in the range of -1.0 and 1.0 radian. This will guarantee maximum precision of the vehicle control on later stages for both autonomous and manual control.

After knowing the threshold values of the UDP messages and their corresponding potentiometer readings, it is necessary to construct the formula that will map the range of UDP angle values to the range of potentiometer readings. This step is necessary for PD control implementation.

Mapping one range of values to another was achieved by linear interpolation using the Microsoft Excel program table 3 by drawing the trendline of the date given in the same table. After plotting the tradeline, it is not possible to export its corresponding equation using Microsoft Excel. Tradeline can be seen below in figure 9 and extracted equation (1)

$$y = 422.5 \times x + 1442.3 \ (1)$$

Where y is potentiometer value, and x is setpoints sent from the master controller which means that every time control signal is received from the master controller it is mapped in the potentiometer values and then these two values are used for proportional derivative control implementation and error measurement.

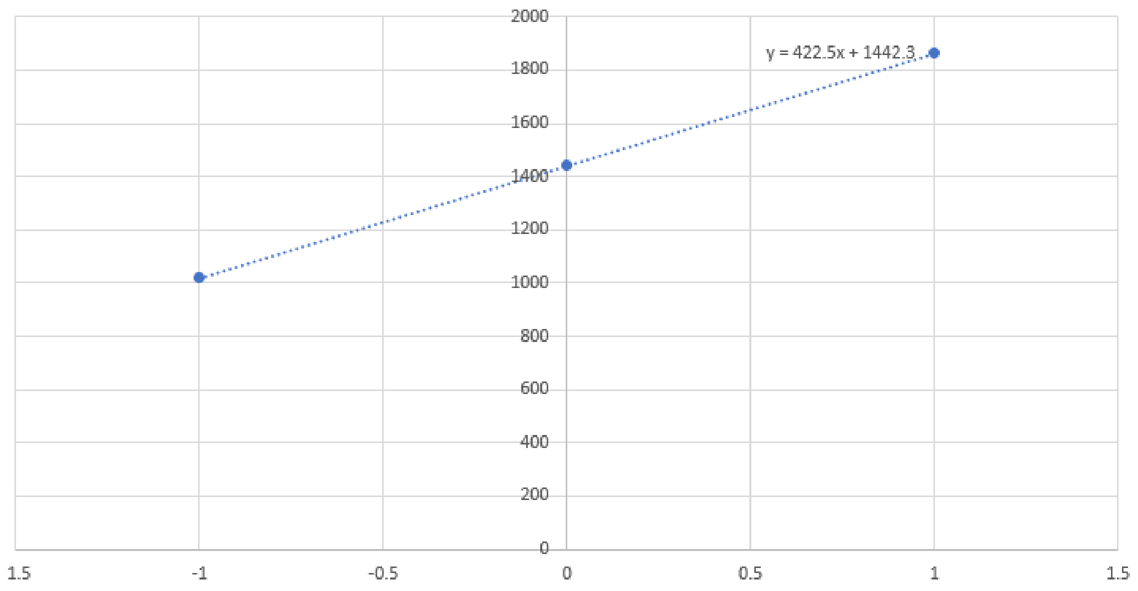


Figure 15 excel trendline for potentiometer angle mapping

The second step of steering control development was creating mathematical formulas for PD control based on the setup and control values that we have already defined. For more clearance let us list all control variables below:

- **Input**- master controller, commanded variable in the range from -1.0 to 1.0 radian and mapped to the potentiometer measurement range from 1020 to 1865 (see table 3)
- **Feedback**- potentiometer in the range from1020 to1865
- **The control signal to the drive controller**- pulse Width Modulation actuating signal which in our formula is PD control output

The equation below represents the PD control formula where u is a static characteristic kd is derivative gain, kp is proportional gain de and dt are changed in error and time, and e is the error.

$$u = k_p + e + k_d \frac{d_e}{d_t} (2)$$

Now let us break down this formula and analyse the proportional and integral action influence on control process. Proportional action generates a rapid response to the present changes in input by generating immediate changes in output and therefore speeds up the error elimination process. Derivative action offsets output signal by the amount proportional to the rate at which the input is changing and therefore protects the control process from overshooting. In short "if proportional action tells the output how far to move when an error appears derivative action tells the output how far to move when the input ramps" [11].

5.5 Proportional derivative control step testing results

A crucial part of the process control is properly tuning of the controller. Since our controller board built on STM32 microcontroller, it was possible to use third party software STM-studio to visualise and capture real-time values of each control variable. Our primary focus during testing was the comparison of set-value from the master controller and feedback value from the analogue sensor. As described in previous chapters we were given set values from -1.0 float value to 1.0 float radian value. Figure 11 below shows tradelines of these two variables. The blue line represents the angle request, and the green line represents analogue sensor feedback. The y-axes of the coordinate system on figure 11 represent angle values in radians, and x-axes represent time in milliseconds. Here we should note that analogue sensor feedback has many disturbances and therefore has an unstable tradeline which should also be encountered while tuning the vehicle wheels to avoid constant jitter of the steering wheels.

Test processes were validated by step-testing, and the primary goal of the test was to tune coefficients of the PD parameters to achieve the most optimal control speed and avoid overshoots. All testing procedures were conducted in the laboratory were test platform in the static position which in general influenced settling due to the high friction of standing wheels compared to the moving wheels. Example of figure 11 shows one of the most optimal results after tuning. This exact process shows one full turn of the wheels from the extreme left angle of the vehicle to the extreme right angle of the wheel and the whole settling time was around 8 000 milliseconds (8 seconds) which is a perfect result if we consider the fact that vehicle was standing.

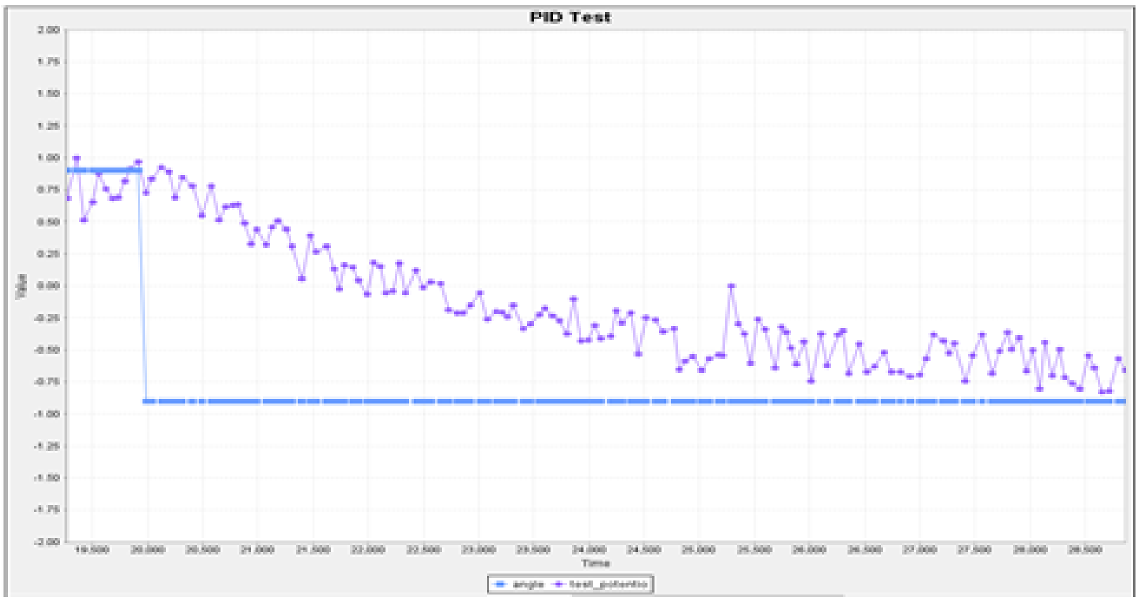


Figure 16 Step testing recording from one steering edge to another

Tuning and adjustment of the PD controller were done during the autonomous rides as well. Test results showed that steering control was one of the most important for autonomous mode driving. Control system with high derivative gains was creating an aggressive control of the steering and was causing vehicle deviation from the path. If settling time was too slow vehicle was unable to do sharp turns.

Tuning of the PD control for steering was the result of a trial and error method to sync high-level and low-level software together.

6 Summary

The thesis presents the development of an embedded control system for the self-driving car test platform. The author guides through the development and rapid prototyping cycles of the autonomous vehicle platform while keeping the main focus on creating an efficient embedded control system with the modular architecture. The study was conducted in collaboration with different parties from different engineering disciplines, and different faculties within the university. The main objectives from the beginning of the project were to create a test vehicle which would be operational in a short period and enable conducting outdoor tests.

In pursuance of, creating a background necessary to understanding technical development and main objectives of the thesis, a big part of the thesis is dedicated to the presentation of the technical equipment and mechanical structure of the vehicle. The thoroughly described high-level software architecture, low-level controller wiring and functional diagrams of the developed firmware gives an understanding on the scale of the whole project and makes it easy to comprehend the reasons behind authors choice for specific frameworks and control algorithms.

Carefully chosen components such as STM32 microcontrollers and frameworks such as HAL libraries gave a significant advantage while creating a robust control system in a timely manner. Wide range of peripheral and their configurability options on STM32 microcontrollers gave the possibility to have a single microcontroller unit for multiple sensors and drive units.

Obtained results from this working test platform was a standardised modular control system which brought significant advantage to the project of ISEAUTO and still remains fully functional system for other future projects. The created control system is flexible and easily reconfigurable for most of the students thanks to the modular architecture concept and created codebase for other module integration.

References

[1] J. Maxwell, “On Governors,” 5 March 1868.

[2] a. M. Leonard and F. S. Stanley, “Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry,” November 1985.

[3] R. S. Burns, Advanced Control Engineering, London, 2001.

[4] P. A. a. J. S. A. Crespo, Embedded Control Systems: From Design to Implementation, 2007.

[5] G. . S. Virk, “CLAWAR: Modular Robots for the Future,” 9-11 November 2002.

[6] R. Sell, E. Väljaots, T. Pataraia and E. Malayjerdi1, “Modular smart control system architecture for the mobile robot platform,” 2019.

[7] S. Kato, S. Tokunagay, Y. Maruyamay, S. Maeday, M. Hirabayashiz, Y. Kitsukawaz, A. Monrroyz and T. Andoz, “Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems,” April 2018.

[8] G. H. H. a. D. S. J. Jan F. Broenink, Software for Embedded Control Systems *.

[9] R. H. Bishop, Modern Control Systems, 2011.

[10] D. E. Oku and P. E. Obot, Comparative Study of PD, PI And PID Controllers for Control of a Single JOint System in Robot, 2018.

[11] T. R. Kuphald, Lessons In Industrial Instrumentation, 2015.

[12] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[13] A. P. L. o. S. Rockets, “Lars Blackmore”.

[14] R. Sell, A. Rassõlkin , M. Leier and J. P. Ernits , “978-1-5386-5413-2/18/$31.00 ©2018 European Union Self-driving car ISEAUTO for research andeducation,” 2018.

[15] G. H. H. a. D. S. J. Jan F. Broenink, “Software for Embedded Control Systems,” 2002.