Bachelor's Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Slip detection for F1/10 model car

Jan Dusil

Supervisor: Ing. Michal Sojka, Ph.D.
May 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Dusil** Jméno: **Jan** Osobní číslo: **457007**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Systémy a řízení**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Detekce smyku pro model auta F1/10

Název bakalářské práce anglicky:

Slip detection for F1/10 model car

Pokyny pro vypracování:

1. Seznamte se se soutěží modelů autonomních aut F1/10.
2. Objednejte díly a sestavte z nich dva nové soutěžní modely.
3. Navrhněte a vyrobte mechanické díky a elektroniku tak, aby výsledek byl univerzálnější než díly doporučované organizátory soutěže (desku napájení menší a vyrobitelná v České republice, flexibilnější mechanická konstrukce, víc poloh pro LiDAR, …).
4. Zprovozněte jednotku inerciální navigace (IMU) a pokuste se na základě dat z ní a jiných částí systému detekovat smyk.
5. Výsledky důkladně otestujte a zdokumentujte.

Seznam doporučené literatury:

1. Martin Vajnar: Model car for the F1/10 autonomous car racing competition, diplomová práce ČVUT, 2017
2. InvenSense Inc.: MPU-9250 Product Specification Revision 1.0, 2014
https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdfa

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Michal Sojka, Ph.D., vestavěné systémy CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **24.01.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Michal Sojka, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to thank my supervisor Ing. Michal Sojka, Ph.D. for valuable advices on my thesis. Also, I would like to thank to the F1/10 team members, Bc. David Kopecký and Bc. Jaroslav Klapálek for patience and help throughout the work process on this thesis.

Last but not least, I would like to thank my parents and my friends that helped me and supported me, not only during my studies, but during my whole life.

Declaration

I declare that I have worked on my thesis separately and that I have listed all the information sources used in accordance with a Methodical Guideline on Ethical Principles in Preparation college final thesis.

V Praze, 24. May 2019

Abstract

This bachelor thesis describes the modification of a racing car model. The aim of this thesis was to make the new models more effective, universal and so they would obey new rules of F1/10 competition. The whole process of the car modelling was thoroughly described including essential theoretical background regarding the robotic operating system (ROS), F1/10 competition and more. Implementation of two algorithms for slip detection is described in the second part of this work. This implementation included configuration and tests of inertial measurement unit (IMU). The outcome of this thesis are two new car models which were already used on projects in our research centre. Moreover, this thesis contributes as both theoretical and practical background for future application concerning slip detection phenomena

Keywords: F1/10 car model, F1/10 competition, ROS, slip detection, IMU, EKF

Supervisor: Ing. Michal Sojka, Ph.D.
Department of Telecommunication Engineering Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

Abstrakt

Tato bakalářská práce pojednává o modifikaci závodního modelu auta. Cílem této modifikace bylo zefektivnění stávajícího modelu tak, aby nové modely byly více univerzální a aby splňovaly nová pravidla soutěže F1/10. Celkový proces návrhu nových modelů byl důkladně okomentován a popsán včetně nezbytných teoretických základů týkajících se robotického operačního systému (ROS), soutěže F1/10 a dalších. V druhé části práce je popsán a zdokumentován vývoj dvou algoritmů pro detekci smyku. Tento vývoj zahrnoval konfiguraci a testování jednotky inerciální navigace (IMU). Výsledkem této práce jsou dva nové modely závodních aut, které byly již využity na projektech v našem výzkumném centru. Daším přínosem práce je teoretický a praktický základ pro budoucí využití ohledně problematiky detekce smyku.

Klíčová slova: F1/10 model auta, soutěž F1/10, ROS, detekce smyku, IMU, EKF

Překlad názvu: Detekce smyku pro model auta F1/10

Contents

Figures

Tables

Chapter 1

Introduction

In today's rapidly developing world, robots are improving their significance in human life. With the quick growth of technology, new challenges in the field of automation appear accordingly. One of these challenges is an autonomous vehicle control.

Luckily, nowadays, the process of solving these challenges is much easier than in the past because of easy-accessible technology. Most importantly, the price reduction of computer embedded systems is the key factor why we can create more precise models or simulate different situations and conditions with a relatively small budget. As a result of that, many international competitions have emerged over the past years to help to popularize different research fields.

One of those competitions is the F1/10 Autonomous Racing Competition which focuses, according to the organizers, "on creating a meaningful and challenging design experience for students. The competition involves designing, building, and testing an autonomous F1 race car at 1/10th the size capable of speeds over 60 km/h" [1]. In 2017, a team of students from the Department of Industrial electronics decided to take part in the competition, and they have won the next-years race in Porto.

Following their tremendous achievement, the department has decided to build upon it and continue with the work. This thesis covers my contribution to the new concept of the F1/10 model, which is being prepared for future competitions and other projects. My goal was to enhance the previous design both in the mechanical and electrical part of the model by building two new car models. Moreover, my contribution to the project also included integrating the Inertial Measurement Unit (IMU) sensor into the system and carrying out experiments on it to calibrate it properly for the mounted vehicle.

Lastly, the biggest challenge was to do a research about a current state of the vehicle slip detection methods and to try to implement at least one of these approaches to enhance the perception of the car model and to lay knowledge base for future research.

The thesis is divided into three main chapters followed by conclusion

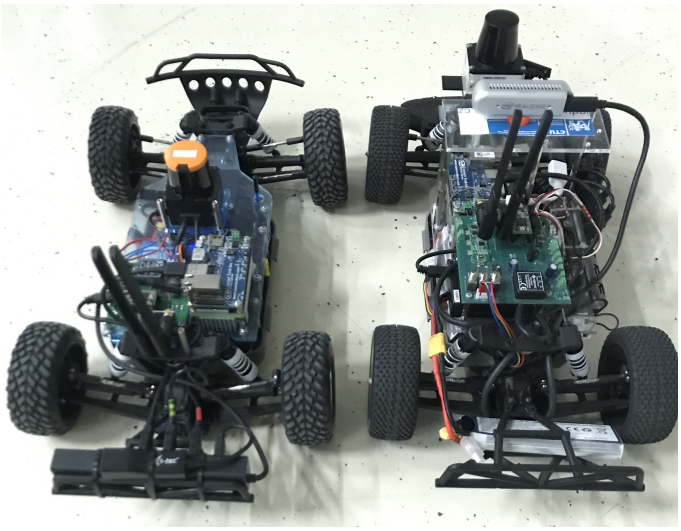


Figure 1.1: New vs. old car model.

(without the Introduction). The first two chapters 2 and 3 cover information about the current state of our F1/10 project. Particularly in chapter 2, I introduce the background topics related to the tasks done in this thesis. In chapter 3, I describe the improvements that I have made on the two new car models.

Later on, chapter 4 is about the side-slip angle estimation from the broad literature overview to two implemented methods and their evaluation. The conclusion and proposal of future-work topics are in chapter 5.

All abbreviations used in this thesis are in appendix B.

Chapter 2

Background

This chapter provides a theoretical background for vehicle construction. Moreover, it introduces the car model from several points of view.

Firstly, I go through the F1/10 competition rules which are important to and understand because the car construction must obey them.

Secondly, this chapter gives the car model overview from both the hardware and software points of view.

2.1 F1/10 competition

Since the first competition, the organizers have introduced several changes regarding the car model. I have compared the older version of the rules with the latest version from February 2019 to highlight the most significant and the most interesting ones.

Firstly, the new rules state that there is no limit on the number of sensors as long as they meet the class restrictions. That brings massive space for solving car identification challenges. For example, one could think about using the wheel velocity sensors to improve the slip angle detection, or another benefit might be in using multiple Light Detection And Ranging (LiDAR) sensors to enhance the vision of the car model. Luckily, the capability class for the processor increased as well to NVIDIA Jetson TX2 or similar which supports the usage of multiple sensors.

Secondly, the organizers have divided vehicles into two new classes:

- F1/10 Restricted Class
- F1/10 Open Class

As the names indicate, the open-class competition is a derivative of the restricted-class one with almost no restrictions

Thirdly, the revised rules present new race classes:

- Time Trial Race
- Head-to-head Race

Time trial class follows the original concept of the race. The goal is to go through the presented track as many times and as fastest as possible without a crash. Both the fastest lap and the highest number of laps are awarded.

The head-to-head race is a new type of competition when there are multiple vehicles at once running through the track. The challenge is to be the fastest without any vehicle-with-vehicle collisions.

To summarize, the newly-revised rules bring many exciting features and challenges. Unfortunately, compared to the original rules, the new ones are far less clear and also bring many doubts and questions for example, about the judgment during the races, specifically during the head-to-head competition. Moreover, in my opinion, the new rules could be a bit confusing for new teams that do not have experiences with it.

2.1.1 F1/10 Restricted Class

1. A 1/10 scale rally car chassis equivalent to the Traxxas model 74054 type is allowed. Example in [2].
   - Use this example for reference in terms of dimensions. See further description for restrictions on motor ratings.
   - Four-wheel drive and two-wheel drive versions are both allowed > in this class.
2. Only the use of stock tires, or equivalent - in size and profile, is allowed. Example in [3].
   - No special traction modifications are allowed, this includes:
     - Applying any liquids or gels of any kind to the stock tires.
     - Using alternate racing tires.
3. Use of NVIDIA Jetson TX2 or an equivalent capability processor or anything of the lower spec is allowed.
4. Use of Hokuyo 10LX or an equivalent LiDAR range sensor or anything with a lower spec is allowed.
5. Multiple LiDARs are allowed, as long as they are all equivalent to, or lower spec than, the Hokuyo 10LX.
6. There are no restrictions on the use of cameras, encoders, or custom electronic speed controllers.
7. Use of Brushless DC motor equivalent to Vellineon 3500 or anything of the lower spec is allowed. Example in [4].

2.1.2 F1/10 Open Class

1. Car dimensions should be within 10% of the dimensions of the car required in Restricted class
2. Only electric drive motors are allowed.

2.2 The Robotic Operating System

2.2.1 Architecture overview

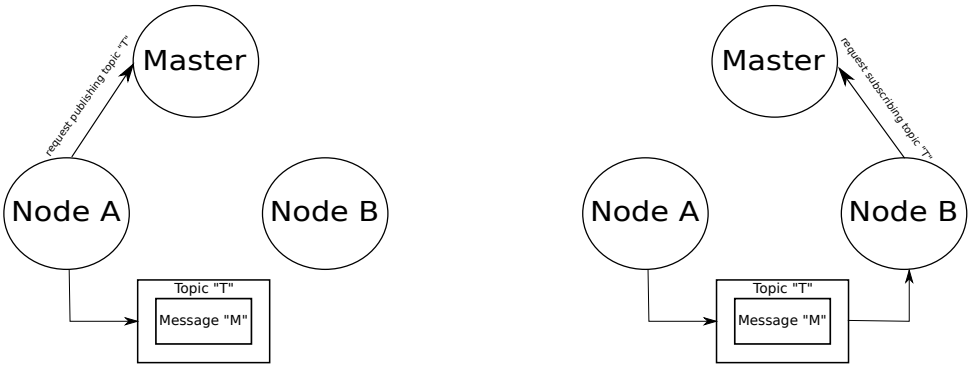
One of the main and most important components of the car model software architecture is the Robotic Operating System (ROS). It is a well known robotic tool which is widely used in robotics projects, and it is also recommended by the F1/10 competition organizers in their tutorial [5] (ROS kinetic distribution particulary). According to the official site [6], the ROS "is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.

Why? Because creating a truly robust, general-purpose robot software is hard. From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own.

As a result, ROS was built from the ground up to encourage collaborative robotics software development. For example, one laboratory might have experts in mapping indoor environments and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, and yet another group might have discovered a computer vision approach that works well for recognizing small objects in clutter. ROS was designed specifically for groups like these to collaborate and build upon each other's work."

ROS architecture is based on a graph concept. The essential part in this plays the **ROS Master node** which "provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other, they communicate with each other peer-to-peer. The Master also provides the Parameter Server. The Master is most commonly run using the roscore command, which loads the ROS Master along with other essential components"[7].

Nodes communicate through ROS by publishing **messages** to **topics**. A message is a data structure that consists of data fields each defined by proper standard primitive type (integer, float, etc.) or even arrays. Every user has



(a) : Node A requests for publishing topic.

(b) : Node B requests for subscribing topic.

Figure 2.1: ROS graph architecture example

an option to either use standardized message structures like “odometry” or “imu”. On top of that, there is a possibility to define custom messages very easily by creating “.msg” files.

Lastly, it is important the **ROS topics** . Based on the ROS wiki the topics “Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data subscribe to the relevant topic; nodes that generate data publish to the relevant topic. There can be multiple publishers and subscribers to a topic.

Topics are intended for unidirectional, streaming communication. Nodes that need to perform remote procedure calls, i.e., receive a response to a request, should use services instead. There is also the Parameter Server for maintaining small amounts of the state”[8]. Those topics are transported via TCP/IP-based and UDP-based message transport. According to [8], the desired transport method is negotiated by nodes at runtime.

The described architecture is shown in the 2.1 as a simple example. Node “A” requests for publishing some message “M”. A topic “T” is created after confirming with the master node but no data are being sent because there is no subscriber. Then, if node B requests for subscription the two nodes are “connected” with topic “T”.

2.2.2 Software development

With the knowledge of the ROS architecture, the implementation of new functionalities on top of ROS is pretty straightforward. The users have three options when it comes to the programming language as ROS currently supports Python, C++, and JAVA. Each new functionality is presented as a package which has a simple “package.xml” definition file where the programmer defines new ROS nodes to be added into the system. Furthermore,

it is important to mention that there are two building tools in ROS - catkin, and rosbuild.

Our ROS project uses catkin to compile and build the packages and most of the nodes are written in Python.

2.2.3 Recording files

Every implementation process consists of a debugging part. For simple repairs only console logging is sufficient. But when it comes to deeper scrutiny, it is useful to be able to simulate certain conditions or situations multiple times. For this, ROS has rosbags. "Bags are typically created by a tool like rosbag, which subscribe to one or more ROS topics, and store the serialized message data in a file as it is received. These bag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics.

Using bag files within a ROS Computation Graph is generally no different from having ROS nodes send the same data, though you can run into issues with timestamped data stored inside of message data. For this reason, the rosbag tool includes an option to publish a simulated clock that corresponds to the time the data was recorded in the file.

The bag file format is very efficient for both recording and playback, as messages are stored in the same representation used in the network transport layer of ROS"[9]

2.2.4 Data visualisation

ROS package called rviz is a useful tool for 3D data visualization of the ROS topics [10]. This tool helps significantly during the debugging process. I have used it during the implementation stage of this thesis (see chapter 4). Particularly, I used the data from the ROS node, which provides odometry data to the position during my experiments. This node was developed by my colleagues from F1/10 team. Moreover, I use ROS hector_slam package that provides are mapping based on data from LiDAR [11].

2.2.5 Used packages

In this subsection, I provide the list of ROS packages that I used during the second part of my work (see section 4.5). For clarification, these packages are only related to my work. Our F1 / 10 model works with many other packages that are not associated with this work. Therefore there are not listed here.

- razor_imu_9dof - this package provides ROS driver for Sparkfun Razor IMU 9DOF sensors. Moreover, it also contains the necessary firmware that needs to be loaded onto the board. More about the IMU configuration in section 4.5.1.

- imu_filter_madgwick - this package is used as one of the methods to filter the raw data from IMU. It works on a fusion algorithm which was originally developed by a student as a part of his Ph.D. research and later transformed into a ROS package. Even though it has received thousands of downloads, it is currently unfortunately not maintained nor updated [12].

- vesc_to_odom - is a ROS package written in C++, provided by the mit-racecar team [13]. The package takes data from VESC and computes information about the robot odometry.

2.3 Model overview

This section serves as a quick overview of the vehicles hardware components that are relevant to this thesis. Namely, the overview presents components that are used for processing data and for car model's perception. Figure 2.2 shows simplified graph of hardware connections.

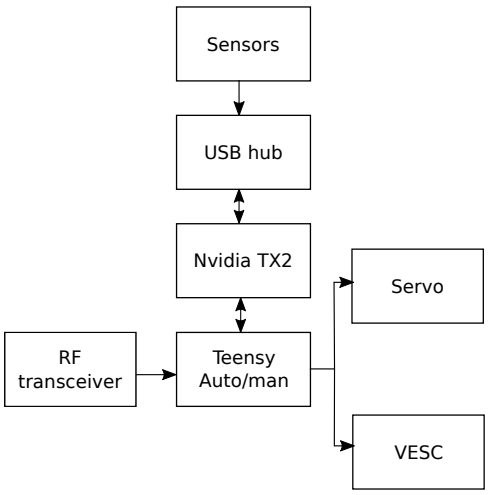


Figure 2.2: Simplified hardware connection overview.

2.3.1 Processing

- NVIDIA Jetson TX2 - All the processing and calculations related to the algorithms developed on our car platform such as localization run on NVIDIA Jetson TX2. It is a state-of-art embedded system [14]. It is mounted on the car on an orbitty carrier board [15] which provides the necessary interfaces to communicate with the Jetson like Ethernet etc.

- Teensy 3.2 - is a USB-based microcontroller that comes with Cortex-M4 96MHz central processing unit (CPU). Its purpose is to control the

behaviour of the electronic speed controller, and SERVO of the car model by received radio frequency (RF) signals coming from the antenna and messages coming NVIDIA board using its build firmware.

2.3.2 Perception

- LiDAR - is the main on-car sensor used for perceiving the environment surrounding. It uses the laser pulses to collect range data by receiving the reflected light. Our car platform features LiDAR that is prescribed by the F1/10 competition rules - Hokuyo UST-10LX. This LiDAR works on 40 Hz scanning frequency with semiconductor laser diode light source. It can measure from 0.02m up to the 10m range and 270° view angle (± 40mm accuracy). The communication is through standard Ethernet interface. The LiDAR data are mainly used for navigation algorithms.
- Camera - is the second vision sensor of the car model. In this occasion, the rules do not limit camera class. We use Intel RealSense Depth Camera D435, which uses stereo vision to calculate depth. Specifically, the built-in infrared projector is used to shoot irregular patterns of dots which are later interpreted with the depth sensors. The vision range is from 0.1m up to approximately 10m. Main advantages are output frame rate up to 90fps, easy mounting, and global shutter type of sensor providing simultaneous image scan. The communication interface is USB-C cable.
- IMU - is the key sensor of this thesis. The current competition rules state no restrictions when it comes to this sensor. Therefore, we freely chose to use SparkFun 9DoF Razor IMU M0. It can be divided into three sub sensors - accelerometer, gyroscope, and magnetometer. It measures linear acceleration, angular rotation velocity and magnetic field vectors along its three axes. This IMU comes with onboard microprocessor Atmel's SAMD21G18A which is Arduino-compatible, 32-bit ARM Cortex-M0+ microcontroller.
- VESC - is an open-source electronic-speed-controller designed by Benjamin Vedder [16]. The VESC controls DC-brushless motor of the car model. Controlling the VESC is done by pulse-width modulation (PWM) signal from the computer unit (Teensy).

Chapter 3

Car model design

This chapter contains information about the procedure of model construction. Firstly, I introduce the motivation for the construction of the new models and bill of materials for hardware and electrical components. The second part describes the designing of the printed circuit board (PCB) and chassis boards.

In the end, the utilization of the new car models is briefly commented.

3.1 Motivation

The first ever F1/10 model in our research team was created in 2017 by my colleague Martin Vajnar as a part of his master's thesis [17]. This original model was built based on the documentation provided by the f1tenth organization [18]. This detailed, manual is undoubtedly an excellent start for the newly formed teams; however, it is not much universal. For example, more positions for LiDAR or camera can be beneficial in custom model architectures.

The first requirement for the model draft was to have multiple LiDAR positions since our model, uses algorithms based on data from it. Moreover, the original design had rather a hectic cable setup, and the batteries powering the electronics were in unstable positions on the car. Also, the original bill of materials (BOM) mostly refers to parts that are only available overseas or worse out of stock. Because of that, a BOM revalidation was made to find easy-accessible parts that could be all bought within a few weeks at most. Finally, the car power board which is used to "provide a stable voltage source for the car and its peripherals since the battery voltage drops as the battery runs out" [18] had several unused electrical components. Following that, my goal was to remove those parts and design a new PCB less spacious on the car chassis.

Name	Description
Orbitty Carrier for NVIDIA Jetson TX2	Controller
Focbox Motor Controller - VESC	Controller
Intel RealSense Depth Camera D435	Sensor
Teensy USB Board, Version 3.2	Controller
SparkFun 9DoF Razor IMU M0	Sensor
USB Hub	Miscellaneous
Hokuyo UST-10LX Scanning Laser Rangefinder	Sensor
Traxxas Slash 1:10 VXL 4WD TQi TSM RTR	RC car
Traxxas Slash 1:10 VXL 4WD TQi RTR Mike Jenkins	RC car

Table 3.1: BOM - hardware components.

Name	Quantity	Description
Polarized capacitor	4	100$[nF]$
Non-polarized capacitor	10	100$[nF]$
Non-polarized capacitor	2	330$[nF]$
Polarized capacitor	2	330$[\mu F]$
3-pin male header	8	-
Red LED	6	-
Resistor	6	360$[\Omega]$
Switch	4	SLIDE SPDT 100 $[mA]$ 12$[V]$
DC/DC converter	2	30 $[W]$ 12$[V]$
Term blocks	8	2.54 mm, 2pos.
IC reg. linear	2	12 to 8$[V]$ LDO
LiPo 3S battery	2	5100$[mAh]$
LiPo 2S battery	2	4000$[mAh]$

Table 3.2: BOM - electrical components.

3.2 PCB design

The original design of the power board was too big to fit on the chassis next to the NVIDIA JETSON [19]. Assuming we do not want to widen the chassis board due to stability issues and unnecessary additional mounting, the perfect opportunity to fit on the board the previously mentioned components next to each other was to shorten the PCB design. As mentioned in the previous section, after mounting the car based on the tutorial [20] it was discovered that some of the PCB components were not necessary. It was namely the LIPO protection, several thermal blocks and several switches. The new design was derived from the previous PCB provided by the organizing team and was made using Altium designer software. The final design is in figure 3.1.

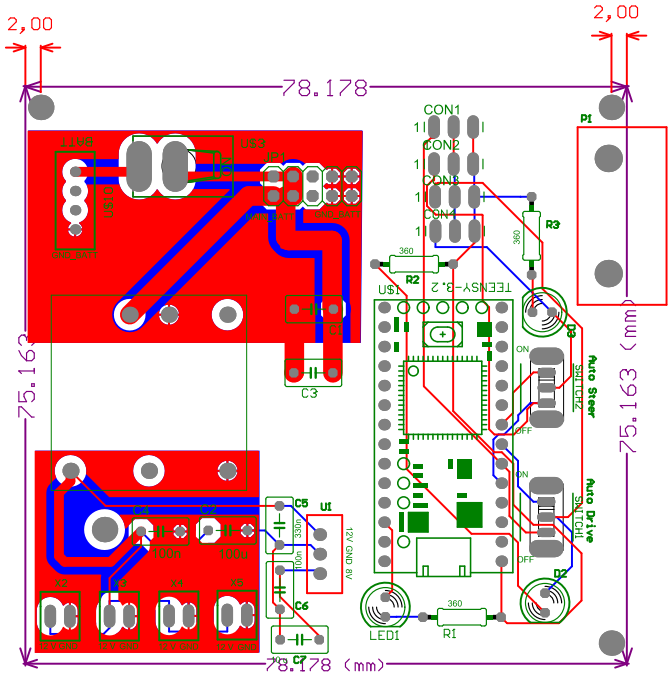


Figure 3.1: Power board design.(dimensions in mm)

3.3 Chassis boards design

The chassis design was the second step after getting the new PCB made. At the start, I used a drawing in AutoCAD format .dwg from the organizers [20] as a template. To get new possible positions for the LiDAR, I have created a separate board for its mounting. It is designed in a way so that the LiDAR could be put either on top of it or upside down. Furthermore, I did several changes to the board that is mounted directly to the car. Namely, I changed the positions for the IMU, the power board, and the VESC. An important task was both putting the VESC under the board and moving the PCB next to the NVIDIA JETSON, which made more space on top of the board. The more space there is left, the more future improvements can be made without much work like adding new sensors or changing the positions of the current ones.

In the second step, I have created a different board for our second car. This car has a differently shaped chassis, which is also higher from the ground compared to the first car. Because of that, I had to do a slightly different design of the board. The most significant advantage of the second car is that all of the elements could easily fit under the board, creating even more space on top of it.

All of my designs are in D.1, D.2, D.3, D.4 D.5 and respectively and are part of the CD contents as well. The boards are made of 4 mm plexiglass and shaped with laser cut.

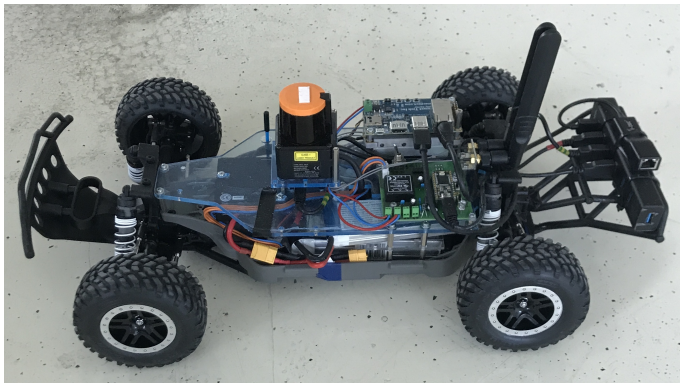


Figure 3.2: New F1/10 model.

3.4 Car model integration

All of the necessary equipment is described in tables 3.1 and 3.2. One of the constructed models 3.2 was used in master's thesis of my colleague[21] from the f1tenth project. The work required a fully functional model with high computational requirements which proved full functionality of the car model. Moreover, the car has also been used during student semestral projects.

Chapter 4

Slip detection

The ability to move the driving responsibility from the driver to the vehicle controller has been one of the most significant challenges of the 21. Century. During this process, we focus, among others, on connecting common Active Safety Systems (ASS) [22] that are already available on standard cars like Anti-lock Braking System (ABS), Electronic Stability Control (ESC) and many others so that these systems would be able to produce one controller creating an autonomous vehicle.

Slip estimation is one of the challenges in the vehicle control field. Its estimation and prevention is an important step towards controlling the vehicle dynamics in terms of suitable yaw rate, avoiding dangerous conditions, or improving car performance. Moreover, slip plays an important role during vehicle localization and position estimation.

In this chapter, the slip detection techniques are discussed. Firstly, I introduce the types of vehicle slips. The longitudinal slip is only described theoretically with some mentioned estimation methods found in the literature. Afterwards, the main focus moves on the side-slip estimation, introducing common approaches towards it. Finally, I describe and evaluate the implementation of two algorithms dealing with the side-slip estimation.

4.1 Longitudinal slip/Wheel slip

The wheel slip is a well-known vehicle phenomenon. Its measurement/detection is, for example, a key for the ABS systems which prevents the tires from locking during the braking maneuver. Generally, we can divide the wheel-slip motion into two situations - during acceleration and braking.

The first situation typically occurs during high acceleration maneuver. The transformed wheel force that is pushing the vehicle is higher than road friction transformed force. This situation puts the wheels into pure spinning motion without noticeable longitudinal movement.

On the other hand, during aggressive braking maneuver one could observe

that the wheel goes into a locked mode when the vehicle slips due to the inertial force of the car being higher than the static friction force of the soil. We can define the longitudinal slip ratio with the equation as follows [23]:

$$\lambda = \begin{cases} \frac{r\omega - v_x}{r\omega}, & r\omega > v_x, \quad \text{for acceleration} \\ \frac{r\omega - v_x}{v_x}, & r\omega < v_x, \quad \text{for deceleration} \end{cases} \tag{4.1}$$

where v_x is the longitudinal vehicle velocity of the vehicle centre of gravity (CG), r is the wheel radius, and ω is the angular velocity of wheels.

There is a wide range of motivational aspects to estimate and control the wheel slippage. Wheel slippage can, for instance, significantly affect the odometry predictions of the vehicle since it gives the wheel encoders an accumulative error. Another unwanted impact could be energy consumption when wheel slippage occurs during acceleration. All in all, the wheel slippage interferes with many automation problems such as localization and position estimation algorithms and that is why it has gained considerable interest.

I have reviewed several approaches to wheel slip detection. In [24], the proposed algorithm is a predictive algorithm that takes the motor armature current and wheel velocities as input. The authors take the wheel-soil interaction into account only partly. However, their algorithm is simple and showed promising results. Vinkó and Ákos [25] propose a solution using extended Kalman filtration (EKF) on data from inertial sensors that are placed on each wheel. In my opinion, the most promising approach was used in [23]. The method is based on dynamic vehicle modeling. The advantage is that only the longitudinal dynamics are taken into account why simplifies the model significantly. As expected, the dynamic approach showed good results, following the measured or expected values.

4.2 Lateral slip/ Side-slip

The second type of vehicle slip is side-slip. It is defined as follows:

$$\beta = \arctan\left(\frac{V_y}{V_x}\right) \tag{4.2}$$

where the V_y and V_x are the lateral and longitudinal velocity respectively, and β is the side-slip angle. In common words, the vehicle side-slip angle (VSA) is the angle between the vehicle heading and the velocity vector CG.

VSA estimation plays a key part in improving vehicle stability performance. More specifically, "nowadays, vehicle control systems such as rear wheel steering, active steering, direct yaw moment control through active differentials or torque vectoring, advanced traction controls, and the above mentioned ESC (in all its forms) are used in conjunction to extend the vehicle performance and stability envelopes."[26]

Apart from the stability control, one of the main reasons for studying the VSA estimation is the lack of sensors for its measurement. Even though

there are ways of measuring the VSA, these sensors lack reliability (GPS), or the cost is way too high for commercial cars not to mention our car model (optical sensors).

In our case, the goal is to summarize and study the VSA estimation methods and also to implement one of the approaches to improve the car model perception which could be possibly used later during the next F1/10 competition by fusing the estimates with other car model control algorithms.

4.3 Common approaches

As mentioned in the previous section, the biggest problem with the VSA measurement is that a suitable sensor in terms of cost, reliability, accuracy, and robustness to environmental conditions does not yet exist [26, 27, 28]. For our car model, the GPS measurements are not usable because the testing and the competition are indoors. Unfortunately, even the indoor GPS which we possess does not provide measurements with sufficient accuracy and is thus unusable.

The following sections 4.3.1 and 4.3.2 introduce two most common approaches towards VSA estimation[28].

4.3.1 Observer-based estimation

Observer-based estimation is the most used method for VSA estimation [26]. It relies on the knowledge of the vehicle model as a reference. Those models could be either kinematic or dynamic (see section 4.4 for more details about vehicle models).

The main drawback of this method is its computational burden, which depends on the complexity of the vehicle model and also on the choice of the observer. On the other hand, this method provides better results and is far more robust than the second method (see section 4.3.2). The literature overview on the VSA estimation from Chindamo, Lenzo and Gadola [26] recognizes three main observers: the Luenberger observer (LO), the sliding-mode observer (SMO), the Kalman Filter (KF) and their variants. LO and SMO are deterministic observes, which means that these observers deal worse with modelling errors and noises of the input variables. Kalman-filter-based observes are more complicated to implement but can deal better with the stochastic behaviour of the model and its variables.

This thesis includes the implementation of two observer-based methods - LO and extended Kalman filter (EKF) 4.5).

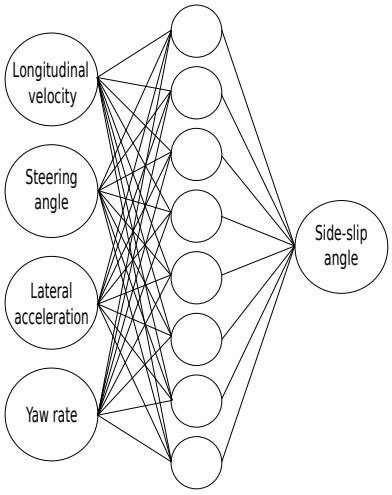


Figure 4.1: Typical neural-network design for VSA estimation.

4.3.2 Neural network-based estimation

Neural network-based estimation is the second recognized [26] method. Its biggest advantage is that it does not need any specification for the car model. Figure 4.1 shows typical design of such network. It only takes the car as a black box system and uses just relationships between input and output variables taken from low-cost sensors to estimate the VSA.

Nonetheless, this approach still has several major drawbacks, the biggest being probably its need to redo the training procedure every time the environment changes. Although it is being considered as the second choice for VSA estimation, the interest in this approach has recently risen, being encouraged by the availability of high-speed computational units. One of the the examples can be found in [29] or [30].

4.4 Vehicle modelling

Vehicle modelling is considered as one of the most important tasks for any car development. The main motivation for modelling is that with the model, simulations, testing and implementation become easy, faster and more available.

The model complexity can vary a lot, depending on the chosen tasks that are to be accomplished. The majority of the vehicle models that have been designed are either domain specific, highly complex, or generalized. One of the most common model simplifications is the assumption of a single track model or bicycle model, which is also used in this thesis. “The modelling is based on a series of simplifications:

- The velocity of the vehicle’s centre of gravity is considered to be constant

along the longitude of its trajectory.

- All lifting, rolling and pitching motion will be neglected.
- The vehicle's mass is assumed to be concentrated at the centre of gravity S.
- The front and the rear tires will be represented as one single tire on each axle. , The imaginary tire contact points V and H, which the tire forces are to act upon, lie along the centre of the axle.
- The pneumatic trail and the aligning torque resulting from the slip angle of the tire will be neglected.
- The wheel-load distribution between front and rear axle is assumed to be constant.
- The longitudinal forces on the tires, resulting from the assumption of a constant longitudinal velocity, will be neglected"[31].

In this section, I describe two commonly considered vehicle modelling approaches - kinematic and dynamic.

4.4.1 Kinematic model

Kinematic modelling is, compared to the dynamic, much simpler because the vehicle motion has no reference to forces. Therefore, many parameters and forces are not considered, such as those regarding tires. That simplifies the model a lot because estimating the tire forces is a non-trivial task.

On the contrary, the simplifications have their drawbacks. Apart from the fact that the model is less accurate, "the main issue of VSA estimation using a kinematic model is that it does not work when the vehicle yaw rate is relatively small or zero"[26]. Because of that, a bare kinematic model without any modification would not be sufficient. The chosen methods for VSA estimation (see section 4.5) are based on kinematic models shown in 4.2.

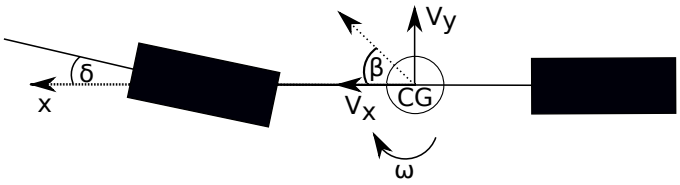


Figure 4.2: Kinematic bicycle model.

4.4.2 Dynamic model

Dynamic models provide a far more accurate description of the vehicle. On the other hand, the modelling of highly non-linear tire models could be very

tricky. The results of the VSA estimation can be significantly worsen if the tire model does not reflect the actual conditions. Moreover, parameters like tire wear and road conditions can also have negative effects on the estimation if not modelled. These issues are commonly addressed and modelled with linear models such as Pacejka's.

4.5 Implementation

This section describes the steps during the VSA estimation algorithm implementation. At first, I introduce the methods for IMU noise filtration. Then, the car model odometry model is briefly explained. In the end, I propose two VSA estimation methods.

4.5.1 IMU calibration and noise filtering

As specified in 2.3.2, the SparkFun 9DoF Razor IMU M0 (4.3) is used on the car platform. Before using the sensor, it is necessary to flash the internal SAMD21 microprocessor with a firmware. That is done with Arduino IDE. The used firmware is a part of the razor_imu_9dof (see 2.2.5) ROS package that handles the data from IMU and publishes the raw data in topic "/imu".

The second step before using the IMU is its calibration. Thanks to the ROS wiki [32] the process of calibration is pretty straightforward. The calibration is done for each sub-sensor accelerometer, gyroscope, and magnetometer separately. The goal is to find the influence of gravity on those sensors and edit the IMU configuration file accordingly.

The third step in the configuration is noise filtration. In the models that are introduced later in this thesis I use three outputs variables of the IMU sensor - yaw $\omega_z[rad/s]$, longitudinal acceleration $A_x[m/s^2]$ and lateral acceleration $A_y[m/s^2]$. Because of that, I only focused on filtrating those data. Three filtration methods are introduced in this thesis:

- Low-pass filter - an exponential moving average (EMA) filter is used. The output of the filter is a weighted sum of the new sensor measurement and the previous filter output value. It is defined as [33]:

$$x(k)_{filtered} = (1 - \alpha) * x(k-1)_{filtered} + \alpha x_{sensor} \tag{4.3}$$

  where α <0,1> is the filtering coefficient.

- Madgwick filter - is a filter that is a part of ROS package imu_filter_madgwick. The package is based on of code on code by Sebastian Madgwic [34].

- Kalman filter - the filter is designed so that the state vectors are taken as the measurements from the IMU that I want to filter. Firstly, I

define the state-transition matrix A_k, the observation matrix H_k, the covariance matrix of the process noise Q_k and the covariance matrix of the observation noise R_k, the observation z_k and the measurement states x_{k_raw}. Covariance matrices Q_k and R_k were tuned during the implementation of the non-linear observer (see sub-section 4.5.2).

$$x_k = \begin{bmatrix} A_x & A_y & \omega_z \end{bmatrix}^T \tag{4.4}$$

$$A_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad H_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$Q_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}, \quad R_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

$$x_{k\_raw} = \begin{bmatrix} A_{x\_raw} & A_{y\_raw} & \omega_{z\_raw} \end{bmatrix}^T \tag{4.7}$$

$$z_k = H_k x_{k\_raw} \tag{4.8}$$

The prediction phase consists of two computations. We predict the state vector and the error covariance using the previous computations:

$$\begin{aligned} \hat{x}_{k|k-1} &= A_k \hat{x}_{k-1|k-1} \\ P_{k|k-1} &= A_k P_{k|k-1} A_k^T + Q_k \end{aligned} \tag{4.9}$$

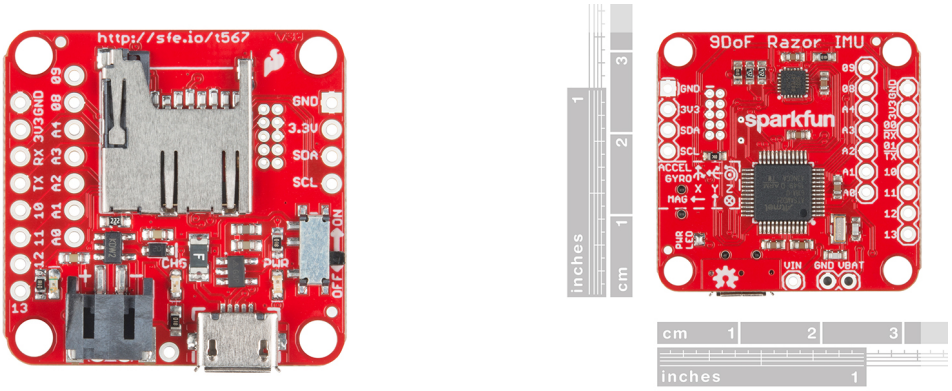
Since the dynamics of the change of the state vectors are unknown we take the previous filtered value as a guess (A_k and H_k are identity matrices). In this model the Kalman filter behaves practically like an enhanced low-pass filter.
The second step is the update of the Kalman gain K_k, the measurement residual $\tilde{y}$, the innovation covariance matrix S_k, the error covariance matrix estimate $P_{k|k}$ and the state estimate $\hat{x}_{k|k}$:

$$\begin{aligned} \tilde{y} &= z_k - H_k \hat{x}_{k|k-1} \\ S_k &= H_k P_{k|k-1} H_k^T \\ K_k &= P_{k|k-1} H^T S_k^{-1} \\ P_{k|k} &= (I - K_k H_k) \hat{P}_{k|k-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k \tilde{y}_k \end{aligned} \tag{4.10}$$

4.5.2 Non-linear observer

The vehicle state observer is based on work of Selmanaj, Corno, Panzani and Savaresi [28]. Firstly, we need to define the kinematic model 4.2 as follows:

$$\begin{aligned} A_x &= \dot{V}_x(t) - \omega_z(t) V_y(t) \\ A_y &= \dot{V}_y(t) + \omega_z(t) V_x(t) \end{aligned} \tag{4.11}$$



(a) : IMU bottom view.

(b) : IMU top view.

Figure 4.3: IMU.

where A_x and $A_y[m/s^2]$ are the longitudinal and lateral accelerations respectively, V_x and $V_y[m/s]$ are the longitudinal and lateral velocities and $\omega_z[rad/s]$ is the yaw rate.
The equations in formula 4.11 can be rewritten into state-space model:

$$\begin{bmatrix} \dot{V}_x \\ \dot{V}_y \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \omega_z(t) \\ -\omega_z(t) & 0 \end{bmatrix}}_{A} \begin{bmatrix} V_x \\ V_y \end{bmatrix} + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{B} \begin{bmatrix} A_x(t) \\ A_y(t) \end{bmatrix} \tag{4.12}$$

Based on this kinematic model the following state observer was introduced in [35]:

$$\begin{bmatrix} \dot{\hat{V}}_x \\ \dot{\hat{V}}_y \end{bmatrix} = (A - KC) \begin{bmatrix} \hat{V}_x \\ \hat{V}_y \end{bmatrix} + B \begin{bmatrix} A_x(t) \\ A_y(t) \end{bmatrix} + KV_x \tag{4.13}$$

$$K = \begin{bmatrix} 2\alpha|\omega_z(t)| & (\alpha^2 - 1)\omega_z(t) \end{bmatrix}^T \tag{4.14}$$

where $\hat{V}_x$ and $\hat{V}_y$ are the longitudinal and lateral velocity estimates, V_x is a measured reference of the longitudinal velocity that is taken from the car model odometry, and α is a constant parameter. This observer secures stability during cornering. However, it has two major drawbacks:

- "Longitudinal velocity is updated only on cornering, although on straight maneuvers the reliability of the velocity measure through wheel angular velocities is higher
- For straight maneuvers, the observability is lost. In these conditions, the observer trivially integrates the accelerations. Small measurement offsets and errors combined with long straight maneuvers, which are common, can cause the filter divergence"[28].

The main contribution of [28] is the modification of the state matrix A and the observer gain matrix K.

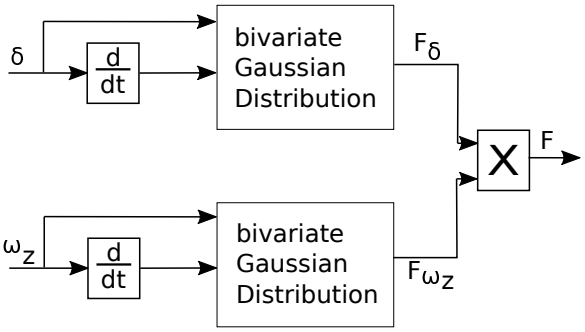


Figure 4.4: Heuristic force computation.

The state matrix includes new element $F(t)$ which is a heuristic force. Its main purpose is to drive the lateral velocity estimation(and thus the side slip angle estimate) to zero during straight driving. "The scheduling of F is based on the idea that, as the vehicle curves, the kinematic model is observable and the closed loop observer performs correctly; on the other hand, during straight driving, F should be large to drive the side-slip estimation to zero"[28]. It has two components F_δ and F_{ω_z}, which are computed from the corresponding measurements of steering angle and yaw rate. This stabilizing force is computed as bivariate Gaussian Distribution with formula 4.15:

$$F_i = e^{-0.5\left(\frac{i^2}{\sigma_i^2} - \frac{di^2}{\sigma_{di}^2}\right)} \quad i = \omega_z, \delta \tag{4.15}$$

where σ is the standard deviation. The derivatives are approximated using backward rectangular rule. The frequency of the published data from IMU is 50 Hz so the time step $T = 0.02[s]$:

$$\dot{x} = \frac{x[k] - x[k-1]}{T} \tag{4.16}$$

The observer gain matrix is redesigned with three parameters α_0, α_1 and α_2. α_1 and α_2 multiplicated with the yaw rate update the observer state on cornering while the α_0 provides the update of the longitudinal velocity estimate $\hat{V}_x$ also during straight driving.

The redesigned observer is defined as follows:

$$\begin{aligned} A &= \begin{bmatrix} 0 & \omega_z(t) \\ -\omega_z(t) & -F(t) \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \\ \begin{bmatrix} \dot{\hat{V}}_x \\ \dot{\hat{V}}_y \end{bmatrix} &= \underbrace{\begin{bmatrix} -\alpha_0 - \alpha_1|\omega_z(t)| & \omega_z(t) \\ -(\alpha_2+1)\omega_z(t) & -F(t)| \end{bmatrix}}_{(A_K C)} \begin{bmatrix} \hat{V}_x \\ \hat{V}_y \end{bmatrix} \\ &\quad + B \begin{bmatrix} A_x(t) \\ A_y(t) \end{bmatrix} + \underbrace{\begin{bmatrix} \alpha_0 + \alpha_1|\omega_z(t)| \\ \alpha_2\omega_z(t) \end{bmatrix}}_{K} V_x \end{aligned} \tag{4.17}$$

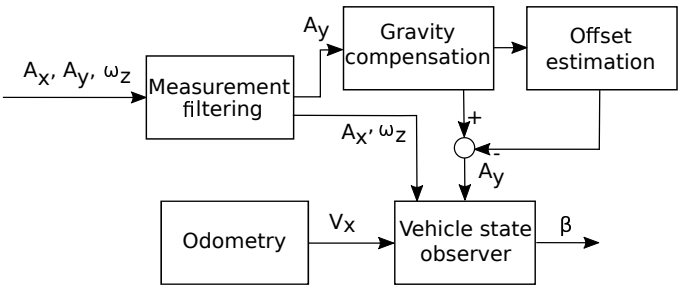


Figure 4.5: Observer model.

In the last step, I discretized the model given by 4.17 using Euler approximation method:

$$x[k+1] \approx (I + AT)x[k] + TBu[x] \tag{4.18}$$

Then the discretized model is defined as:

$$\begin{aligned}\begin{bmatrix}\hat{V}_x[k+1]\\ \hat{V}_y[k+1]\end{bmatrix} = \begin{bmatrix}1-\alpha_0-\alpha_1|\omega_z[k]| & \omega_z[k]\\ -(\alpha_2+1)\omega_z(t) & 1-F[k]|\end{bmatrix} T \begin{bmatrix}\hat{V}_x[k]\\ \hat{V}_y[k]\end{bmatrix}\\ + TB\begin{bmatrix}A_x[k]\\ A_y[k]\end{bmatrix} + T\begin{bmatrix}\alpha_0+\alpha_1|\omega_z[k]|\\ \alpha_2\omega_z[k]\end{bmatrix} V_x\end{aligned} \tag{4.19}$$

The car model rolls and gravity acceleration can significantly influence the lateral acceleration measurement. Firstly, we address the offset estimation. Formula 4.20 sums two equations. The first describes the dynamics of the lateral velocity while the second equation is the sensor output model:

$$\begin{cases}\dot{V}_y(t) = A_y(t) - \omega_z(t)V_x(t)\\ A_y^{off}(t) = A_y(t) + \Delta A_y(t) + \epsilon_y(t)\end{cases} \tag{4.20}$$

where $A_y^{off}(t)$ is the processed sensor output, $A_y(t)$ is the real lateral acceleration, $\Delta A_y(t)$ is a time-varying sensor offset and $\epsilon_y(t)$ is a zero mean noise. Combining the presented formulas in 4.20 we obtain the equation for sensor offset:

$$\Delta A_y(t) = A_y^{off}(t) - \omega_z(t)V_x(t) - \dot{V}_y(t) - \epsilon_y(t) \tag{4.21}$$

We can simplify 4.21 with two assumptions. Only mean values of the quantities in 4.21 would be considered thus we can neglect $\epsilon_y(t)$ which has zero mean by definition. Secondly, "the mean values of the derivative of the lateral velocity on a certain time interval can be related to the vehicle lateral velocity with the following"[28]:

$$E[\dot{V}_y] = \frac{1}{t_f - t_0}\int_{t_0}^{t_f} \dot{V}_y(t)dt = \frac{V_y(t_f) - V_y(t_0)}{t'_f - t_0} \tag{4.22}$$

Assuming a generic route with straight finishing and starting the derivative of $\dot{V}_y$ is zero on average and we can ignore it. The final equation for offset estimation is defined as:

$$\Delta A_y(t) = A_y^{off}(t) - \omega_z(t)V_x(t) \tag{4.23}$$

Lastly, I address the gravity compensation. The formula is taken from [28] and is derived from the vertical car dynamics. The roll angle is computed from the quaternions that are part of IMU output.

$$A_y^{off}(t) = A_{y\_measure} - gsin(\theta) \tag{4.24}$$

where g is the gravitational acceleration and θ is the roll angle.
The overall design of the observer is shown in 4.5. The estimation of VSA at each step k is computed as:

$$\hat{\beta}[k] = \frac{\hat{V}_y[k]}{\hat{V}_x[k]} \tag{4.25}$$

4.5.3 Extended Kalman filter

The second implemented approach is EKF based on [36]. Again I assume the kinematic bicycle model defined by state space model:

$$\begin{aligned} \dot{x} &= Ax + Bu + Gw \\ A &= \begin{bmatrix} 0 & \omega_z \\ -\omega_z & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ G &= \begin{bmatrix} -V_x & -1 & 0 \\ V_y & 0 & -1 \end{bmatrix}, \quad w = \begin{bmatrix} \omega_{z,n} & A_{x,n} & A_{y,n} \end{bmatrix}^T \end{aligned} \tag{4.26}$$

where $\omega_{z,n}$, $A_{x,n}$ and $A_{y,n}$ are the yaw rate, longitudinal acceleration and lateral acceleration measurement noises. G and w are the disturbance input matrix and the process noise vector respectively.

However, the process noise vector is unknown and thus neglected:

$$\dot{x} = Ax + Bu \tag{4.27}$$

Since the car model uses sensor with output rate of 50 Hz the kinematic model is discretized using forward rectangular rule ($T = 0.02\ [s]$):

$$\frac{(z-1)}{T}x = Ax + Bu \tag{4.28}$$

The discrete state-space model is obtained as:

$$\begin{aligned} x_{k+1} &= \Phi_k x_k + \Gamma u_k + \Gamma_1 w_k \\ \Phi_k &= \begin{bmatrix} -V_x & -1 & 0 \\ V_y & 0 & -1 \end{bmatrix}, \Gamma = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}, \quad \Gamma_1 = \begin{bmatrix} -V_{x|k}T & -T & 0 \\ -V_{y|k}T & 0 & -T \end{bmatrix} \end{aligned} \tag{4.29}$$

Because Γ_1 contains the velocity states we define the EKF estimate as follows:

$$\hat{x}_{k+1} = \Phi_k \hat{x}_k + \Gamma u_k + L_k(y_k - C\hat{x}_k) \tag{4.30}$$

where L_k is the estimation Kalman gain matrix defined as:

$$L_k = \begin{cases} P_k H^T R^{-1}, & |\omega_z| \geq \omega_{z,th} \\ 0, & |\omega_z| < \omega_{z,th} \end{cases} \tag{4.31}$$

where R is the variance of measurement noise and P_k is the estimate covariance matrix described as:

$$P_k = M_k - M_k H^T (H M_k H^T + R)^{-1} H M_k \tag{4.32}$$

finally the matrix M_k matrix is the update law for P_k:

$$M_{k+1} = \Phi_k P_k \Phi_k^T + \Gamma_{1,k} Q \Gamma_{1,k}^T \tag{4.33}$$

in which the Q_k is the process noise covariance matrix.
As mentioned before, the kinematic model becomes unobservable when the yaw rate it too small (smaller than the threshold value). The model in [36] proposes lateral velocity estimate based on cornering stiffness parameters of the tyres which are unknown in our car model. Therefore, during the occasion of potential unobservability the estimate for the lateral velocity from the non-linear observer is used (see section 4.5.2).

The estimate of VSA is computed at every step as follows:

$$\hat{\beta}_k = \arctan\left(\frac{\hat{V}_{y,k}}{\hat{V}_{x,k}}\right) \tag{4.34}$$

4.5.4 ROS integration

The two presented algorithms are managed in one ROS package called sparkfun_9do_razor_m0. The package is included on the CD as an attachment. The architecture overview of the implemented node is shown in 4.6.

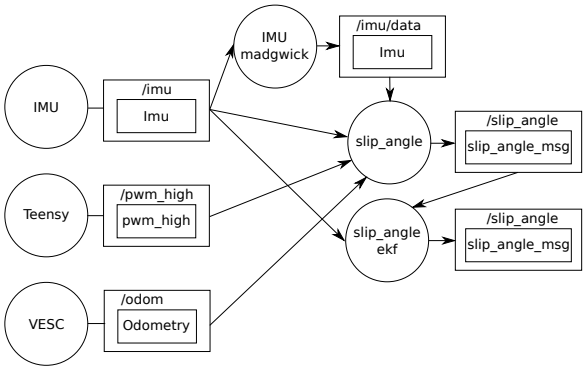


Figure 4.6: Graph of the ROS implementation architecture.

The NL observer receives data from VESC which are handled through vesc_to_odom package (see section 2.2.5), data from IMU and data from

teensy. The data from Teensy consist constants of PWM to calculate the steering angle.

The EKF observer works with data from imu and also receives data from the NL observer about the lateral velocity. That is wrapped inside slip_angle_msg.

4.6 Evaluation

This section documents the evaluation of the above-mentioned methods that were implemented. Firstly, I compare the IMU filtration methods. Secondly, results from four proposed experiments are shown and described. Table 4.1 shows tuning parameters of the non-linear observer and threshold value for the vehicle yaw rate. Those parameters were found experimentally. More specifically, the parameters a_o-a_2 were found by tuning the difference between the longitudinal velocity estimate and the longitudinal velocity from the odometry.

Unfortunately, during those experiments, no reference to the vehicle side-slip angle is available.

Parameter	Value	Unit
a_0	5	[-]
a_1	12	[-]
a_2	7.5	[-]
$\omega_{z,th}$	0.1	$[rad/s^2]$

Table 4.1: Tunning parameter values.

4.6.1 IMU noise filtering

The first experiment is a simple case when the car is standing still. The values for longitudinal acceleration and lateral acceleration are presented in 4.7. We can see that the best result provide the Kalman filter and the low-pass filter while the Madgwick filter almost copies the noisy output from IMU. Because of that, this filter is not later used. The yaw rate during this experiment was not significantly noisy; thus, is not presented. Moreover, the lateral acceleration shows $-0.05\ [m/s^2]$ offset. Both of the mentioned models 4.5.2 and 4.5.3 are updated with this information.

4.6.2 Straight drive

The second experiment is straight drive maneuver. The longitudinal velocity and the steer angle are fixed using ROS node which can interpret user-specified

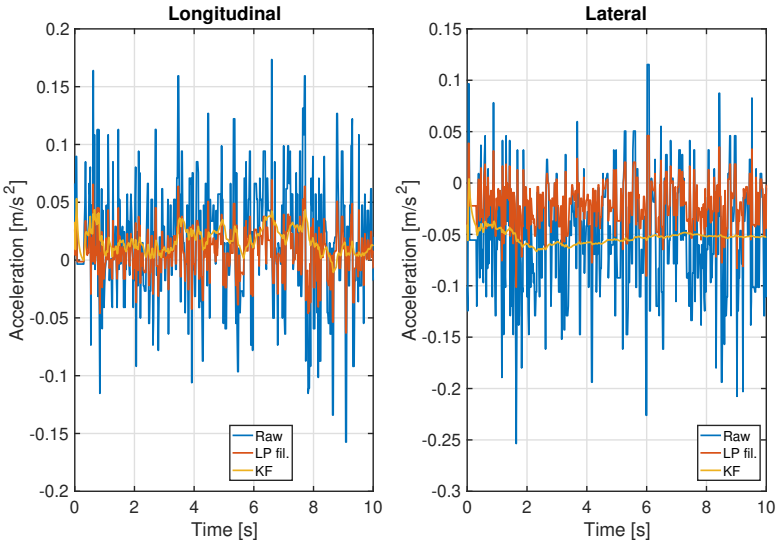


Figure 4.7: Longitudinal and lateral acceleration - stopped car.

driving values (velocity, steering angle) values and send them to the Teensy controller.

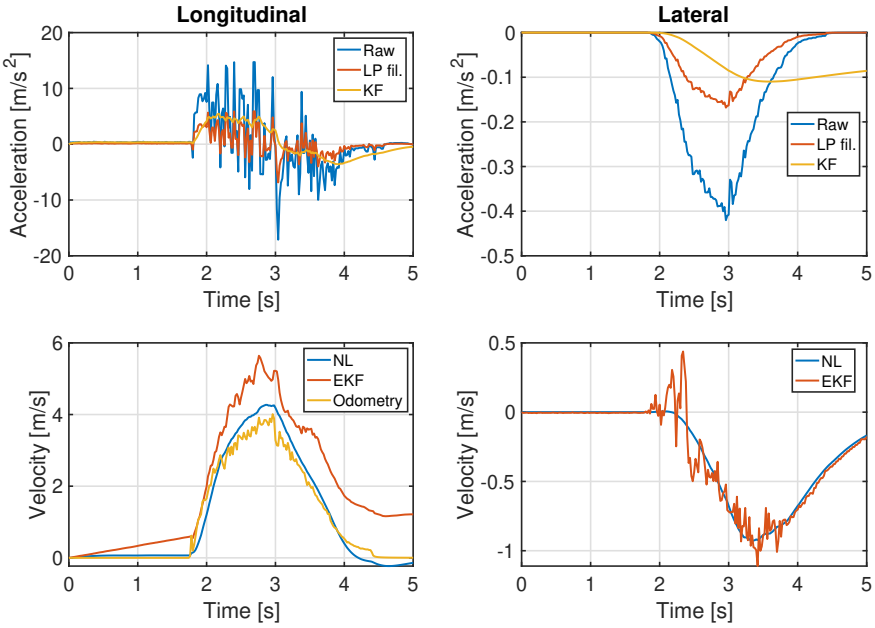


Figure 4.8: Longitudinal and lateral measurements and estimations - straight drive maneuver.

The vehicle model accelerates to approximately three seconds and is then stopped manually by the RF controller. We can see from the left figures in 4.8 that the longitudinal acceleration and the longitudinal velocity have the expected courses. Mainly, both estimates (NN and EKF) follow the reference taken from odometry.

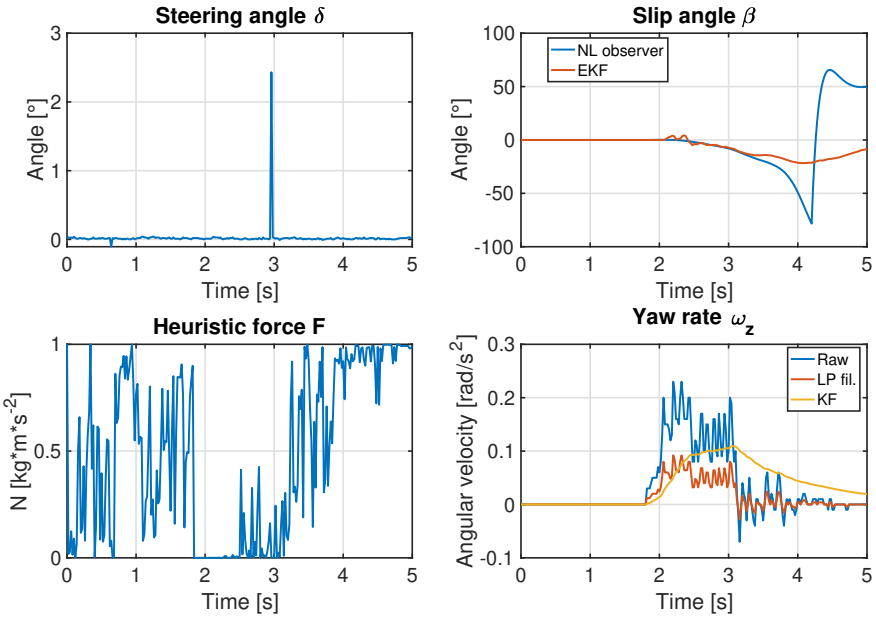


Figure 4.9: Angular measurements - straight drive maneuver.

On the contrary, the lateral velocity is highly off. Ideally, the lateral acceleration and the estimate for the lateral velocity (down-left graph) would be near zero, but IMU shows insignificantly high values, even though the raw data are modified with the offset and gravity compensations. This could be caused by several aspects. Possibly, the offset estimation or gravity compensations are not properly calculated. Moreover, the mounting correction of the IMU is not included in the model, which could also be a possible explanation for those outputs.

This erroneous behaviour of the lateral velocity estimation significantly influences the side-slip angle estimation shown in 4.9. During breaking maneuver, the lateral velocity estimate exceeds the longitudinal velocity, which causes divergence of the non-linear observer, and the observability is lost. The EKF-based observer behaves more realistic but still shows a not negligible angle. That is because when the yaw rate is smaller than 0.1 $[rad/s]$ threshold value the estimate for the lateral velocity is taken from the non-linear estimation (see section 4.5.3).

Figure 4.9 also shows the yaw rate and the heuristic force. The scheduling is done right because when the yaw rate increases the force converges to zero, which is correct behaviour (see section 4.5.2).

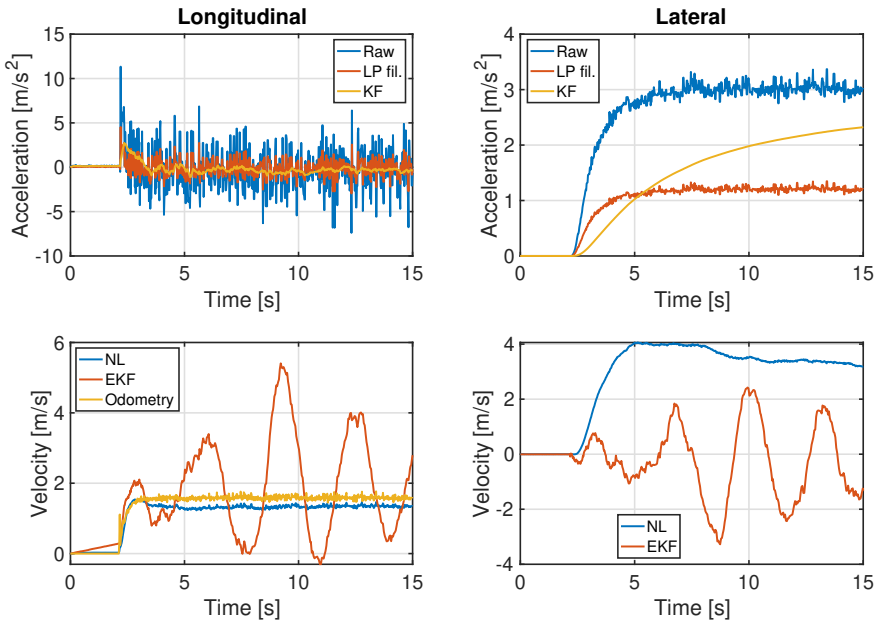


Figure 4.10: Longitudinal and lateral measurements and estimations - fixed circling maneuver.

4.6.3 Circling without drift

The second test is a circling maneuver. Again, the velocity and the steering angle are fixed with special ROS node. The values are chosen so that the car would not slip and smoothly rotate in the same circles. Fig 4.10 shows the measurements and estimations of accelerations and velocities, respectively. Down left figure shows that the longitudinal velocity estimate converges to the odometry measurements. On the other hand, the EKF estimation is highly oscillative for both-axis velocities, and thus the slip-angle estimation behaves the same way.

For the non-linear observer, the disturbing factor of the side-slip estimation is again the lateral velocity which reaches high values for a relatively small turn.

Figure 4.11 documents, among others, the lateral slip angle estimation. The EKF estimation is unfortunately unusable as the information is lost with the velocities estimation error. The non-linear shows more realistic behaviour, even though the angle to which the estimation converges is around 70 deg, which is very high. This possible explanation for this could be the before mentioned neglected mounting correction. The IMU is mounted closer to the front wheels than to the back ones. Thus the estimation indicates more the lateral slip angle of the front wheels than of a whole vehicle.

Finally, 4.11 shows the fixed steering angle, the heuristic force, and the yaw rate. Again, we can confirm that the heuristic behaves as it should; when the yaw rate or steering angle differ from zero, it should converge to zero.

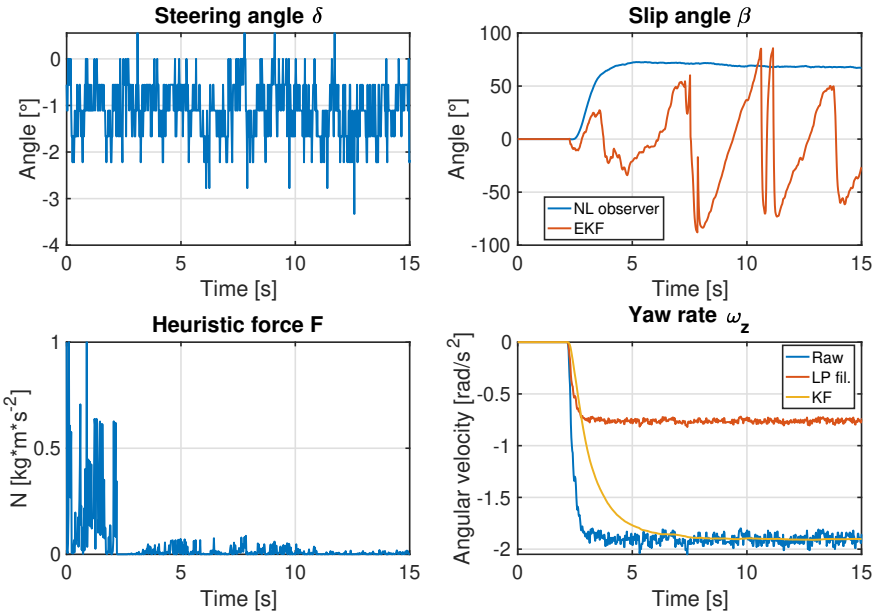


Figure 4.11: Angular measurements- fixed circling maneuver.

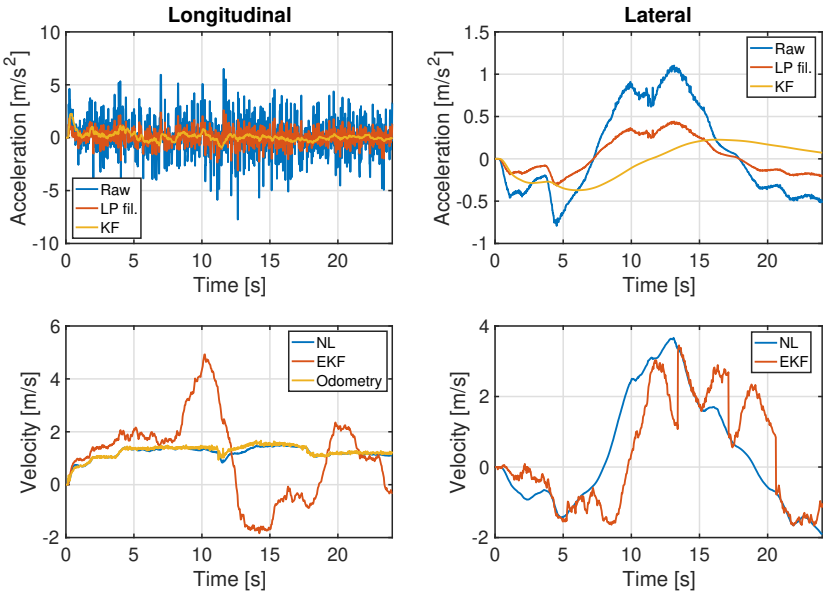


Figure 4.12: Longitudinal and lateral measurements and estimations - track lap.

4.6.4 Track lap

This experiment is one track lap at a track built in our research centre (map of track in 4.14b). During this experiment, the car model is controlled with the RF controller.

Although the floor in the laboratory where the test was run is very slippery, the lateral slip angle from the non-linear estimator is not very accurate. In

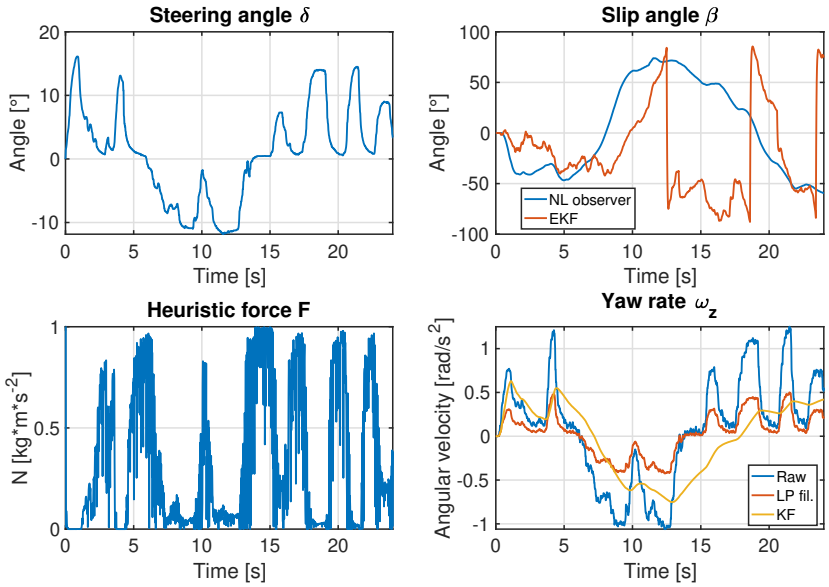


Figure 4.13: Angular measurements - track lap.

particular, when comparing the values from 4.13 of the slip angle and the yaw rate or steering angle, the estimate is very "slow" and does not converge to zero when the car is in straight drive motion.

The EKF estimator shows false values, specifically for the longitudinal velocity estimation. Because of that, the estimation of the side-slip angle cannot be taken as valid.

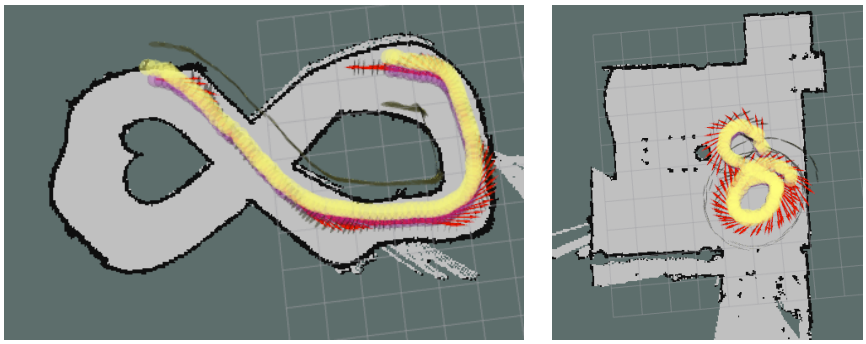
4.6.5 Circling with drift

In this experiment, the car model is controlled manually with the RF transceiver. The velocity and steering are higher than in the previous similar experiment (4.6.3) making the circle trajectory to extend as the car model is influenced by the side-slip (see 4.14b).

As shown in 4.16, the car model is circling approximately twenty seconds to the right and then about ten seconds to the left. The VSA estimation from the NL observer is stable. However, it converges to high values. That could be influenced by the mounting of the IMU, which may be too front-positioned. Namely, we can see in 4.15 that the values of the lateral velocity estimate are again very high.

The EKF observer shows no valuable response from which we could make any conclusion. During experiments with straight driving motion, the assumption was that the lateral velocity from the NL observer influences the EKF so much that it can cause its divergence. However, during strictly turning motion, the two observers are not connected at all. That means that the EKF is probably badly tuned or there is a bug in the implementation.

The heuristic force corresponds with the yaw rate and steering angle as it should.



(a) : Path visualization of the vehicle model during track lap.

(b) : Path visualization of the vehicle model during circle drifting.

Figure 4.14: Maps of the experimental environment with car model visual reference. Visualized with RViz.

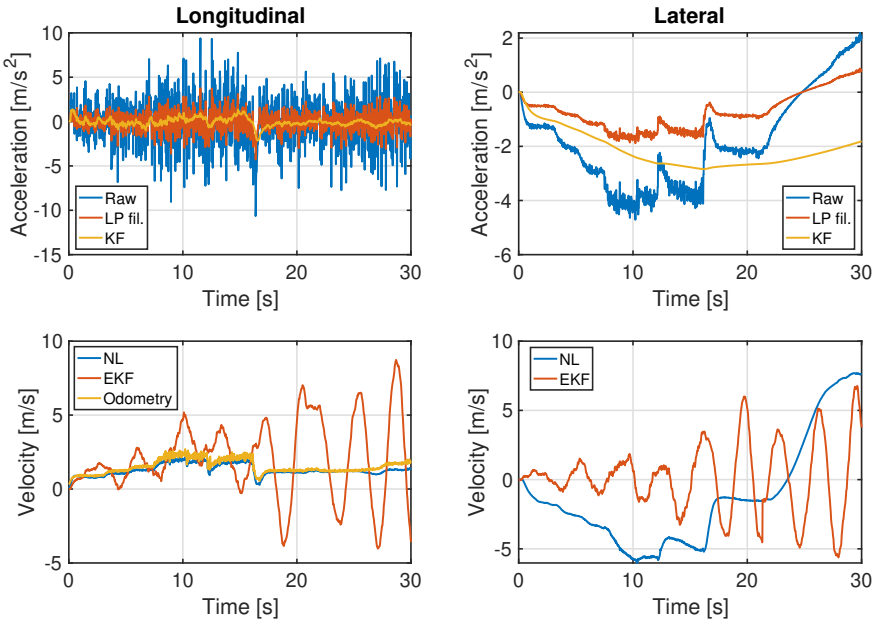


Figure 4.15: Longitudinal and lateral measurements and estimations - circling with drift.

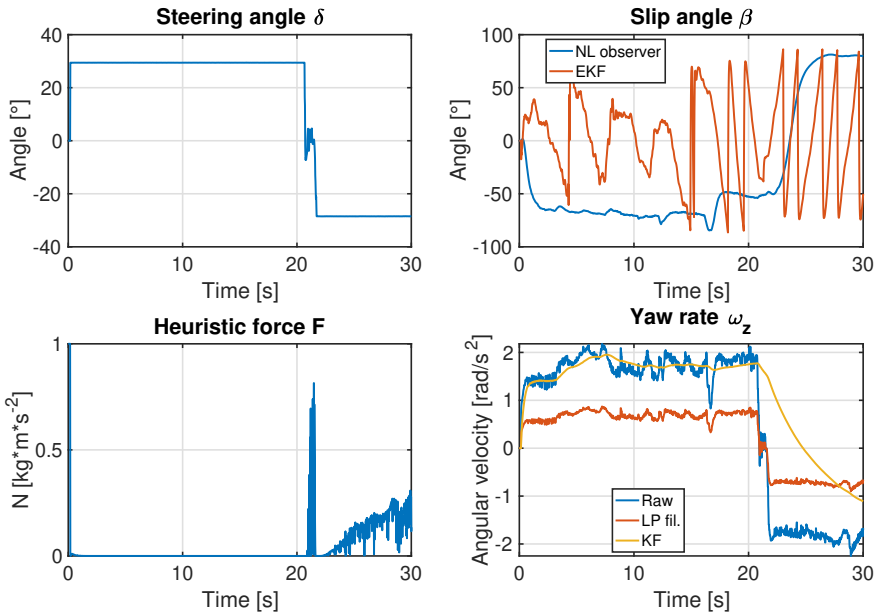


Figure 4.16: Angular measurements- circling with drift.

Chapter 5

Conclusion

The aim of this thesis was to construct and modify two new models for the F1/10 competition and for other research purposes. These modifications can be divided into two parts - mechanical and software. As the car model was explicitly built for the competition, the first step for both of these parts was to get familiar with the competition rules.

Firstly, I worked out a new design for two unique car models. These designs included remodelling of the power board and remodelling of the chassis boards. All modifications provided more space on top of the car which met the requirements for more possible positions for sensors on top of it.

The second part of this thesis focuses on research on vehicle slipping motion. The aim was to familiarize with the slipping estimation problem and to propose and implement estimation algorithm. I have implemented two methods on side-slip estimation - NL observer and EKF observer. During the process I had to study the car platform itself from hardware (IMU, Teensy etc.) to software (ROS) components. Mainly, the IMU configuration and noise filtering was highly studied, tested and described in the thesis.

Then, the observes were tested on set of experiments in our research laboratory. These tests were properly documented and depicted. The NL observer showed promising results even though it was based on a simple kinematic model. On the other hand, the EKF observer was not successfully implemented, partially because the original EKF observer was modelled for more dynamic approach.

To summarize, this thesis contributes to the F1/10 project of our research team with two new vehicle models, IMU configuration overview and knowledge base for on-the-vehicle VSA estimation.

5.1 Future work

This chapter introduces several improvements that could be done to enhance the model and the implemented methods used for VSA estimation. The main sensors that could enhance the estimation are:

- Ultrasonic sensors
- Wheel velocity sensors
- IMU

Mainly, the wheel velocity sensors could provide better longitudinal velocity estimation. Other possibility, might be in adding more LiDARs or using ultrasonic sensors to improve the car model localization.

Furthermore, the main improvements could be done in modelling of the car. More specifically, the dynamic approach could be used to improve the card model and its control.

Finally, the used inertial measurement unit (see section 2.3.2) proved to be very inaccurate. More accurate IMU sensor would be key improvement to the algorithms presented in this thesis.

Appendix A

Bibliography

[1] F1/10 Team, "Gallery." http://f1tenth.org/gallery.htm, 2019. [Online; accessed 24-May-2019].

[2] Traxxas, "Traxxas 1/10 scale ford fiesta st rally." https://traxxas.com/products/models/electric/ford-fiesta-st-rally. [Online; accessed 24-May-2019].

[3] Traxxas, "Tires overview." https://traxxas.com/products/parts/7473R. [Online; accessed 24-May-2019].

[4] Traxxas, "Velineon 3500 brushless motor." https://traxxas.com/products/parts/motors/3351Rvelineon3500motor. [Online; accessed 24-May-2019].

[5] F1/10 Team, "Installing ros." http://f1tenth.org/build.html#installros, 2019. [Online; accessed 24-May-2019].

[6] Open Source Robotic Foundation, "About ros." http://www.ros.org/about-ros/. [Online; accessed 24-May-2019].

[7] ROS community, "Ros master." http://wiki.ros.org/Master. [Online; accessed 24-May-2019].

[8] ROS community, "Ros topics." http://wiki.ros.org/Topics. [Online; accessed 24-May-2019].

[9] ROS community, "Ros bags." http://wiki.ros.org/Bags. [Online; accessed 24-May-2019].

[10] ROS community, "rviz package." http://wiki.ros.org/rviz. [Online; accessed 24-May-2019].

[11] ROS community, "hector_slam." http://wiki.ros.org/hector_slam. [Online; accessed 24-May-2019].

[12] ROS community, "imu_filter_madgwick." http://wiki.ros.org/imu_filter_madgwick. [Online; accessed 24-May-2019].

[13] mit-racecar team, "mit-racecar - vesc_to_odom package." https://github.com/mit-racecar. [Online; accessed 24-May-2019].

[14] NVIDIA, "Jetson TX2 devkit." https://developer.nvidia.com/embedded/buy/jetson-tx2-devkit. [Online; accessed 24-May-2019].

[15] Connect Tech Inc., "Orbitty Carrier for NVIDIA Jetson TX2." http://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1/. [Online; accessed 24-May-2019].

[16] Benjamin Vedder, "VESC – Open Source ESC." http://vedder.se/2015/01/vesc-open-source-esc/. [Online; accessed 24-May-2019].

[17] Martin Vajnar, "Model car for the F1/10 autonomous car racing competition," Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2017.

[18] F1/10 Team, "Build." http://f1tenth.org/build.html#, 2019. [Online; accessed 24-May-2019].

[19] F1/10 Team, "Power board to the chassis." http://f1tenth.org/build.html#mountingthepowerboardtothechassis, 2019. [Online; accessed 24-May-2019].

[20] F1/10 Team, "Official F1/10 competition github repository." https://github.com/mlab-upenn/f1tenthpublic, 2019. [Online; accessed 24-May-2019].

[21] David Kopecký, "Localization and advanced control for model cars," Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2017.

[22] Alice Matthews, "Active and passive automotive safety systems." https://automotive.electronicspecifier.com/safety/active-and-passive-automotive-safety-systems, 2017. [Online; accessed 24-May-2019].

[23] K. Nam, Y. Hori, and C. Lee, "Wheel slip control for improving traction-ability and energy efficiency of a personal electric vehicle," *Energies*, vol. 8, no. 7, pp. 6820–6840, 2015.

[24] Z. Zhu, K. Yuan, W. Zou, and H. Hu, "Current-based wheel slip detection of all-wheel driving vehicle," in *2009 International Conference on Information and Automation*, pp. 495–499, June 2009.

[25] Vinkó, Akos, "Wheel slip detection by using wheel-mounted inertial sensors," 09 2014.

[26] D. Chindamo, B. Lenzo, and M. Gadola, "On the vehicle sideslip angle estimation: A literature review of methods, models, and innovations," Applied Sciences, vol. 8, no. 3, p. 355, 2018.

[27] B. L. Boada, M. J. L. Boada, V.Diaz, "Vehicle sideslip angle measurement based on sensor data fusion using an integrated ANFIS and an Unscented Kalman Filter algorithm," Mech. Syst. Signal Processing, 72–73 (2016), pp. 832-845.

[28] Donald Selmanaj, Matteo Corno, Giulio Panzani and Sergio M. Savaresi, "Vehicle sideslip estimation: A kinematic based approach," Control Eng. Pract., 67 (2017), pp. 1-12.

[29] S. Melzi, E. Sabbioni, "On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results," Mech. Syst. Signal Processing, (2011), pp. 2005-2019.

[30] Daniel Chindamo, Marco Gadola, "Estimation of Vehicle Side-Slip Angle Using an Artificial Neural Network,"

[31] R. B. Dieter Schramm, Manfred Hiller, *Single Track Models*. Springer, Berlin, Heidelberg, 2014.

[32] ROS community, "razoz_imu_9dof package." `http://wiki.ros.org/razor_imu_9dof`. [Online; accessed 24-May-2019].

[33] Matlab Arduino, "MATLAB Arduino Tutorial 4 - Filtering Noise out of 3-axis Accelerometer Data in Real-time." `https://www.youtube.com/watch?v=TeKk3DjN_gs`, note =.

[34] ROS community, "Ros kinetic wiki - imu_tools package." `http://wiki.ros.org/imu_tools?distro=kinetic`, 2019. [Online; accessed 24-May-2019].

[35] J. Farrelly and P. Wellstead, "Estimation of vehicle lateral velocity," in *Proceeding of the 1996 IEEE International Conference on Control Applications IEEE International Conference on Control Applications held together with IEEE International Symposium on Intelligent Contro*, pp. 552–557, Sep. 1996.

[36] B.-C. Chen and F.-C. Hsieh, "Sideslip angle estimation using extended kalman filter," *Vehicle System Dynamics*, vol. 46, no. sup1, pp. 353–364, 2008.

Appendix B

Glossary

Acronym	Meaning
ABS	Anti-lock Braking System
ASS	Active Safety Systems
CG	Centre of Gravity
CPU	Central Processing Unit
EKF	Extended Kalman Filter
ESC	Electronic Stability Control
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
LO	Luenberger Observer
LP	Low-pass
NL	Non-linear
PCB	Printed Circuit Board
PWM	Pulse-Width Modulation
RF	Radio frequency
SMO	Sliding-Mode Observer
USB	Universal Serial Bus
VSA	Vehicle Side-slip Angle

Appendix C

CD contents

Directory name	Description
`/thesis.pdf`	Bachelor thesis in pdf format.
`/code/sparkfun_9dof_razor_m0/`	ROS package including two presented algorithms as ROS nodes written in python.
`/chassis_boards/`	All Autocad drawings of chassis boards.
`/pcb/`	PCB designs of power board - original and new for comparison.
`/videos/`	Illustrative videos of the autonomously driven experiments.

Appendix D

Chassis boards design

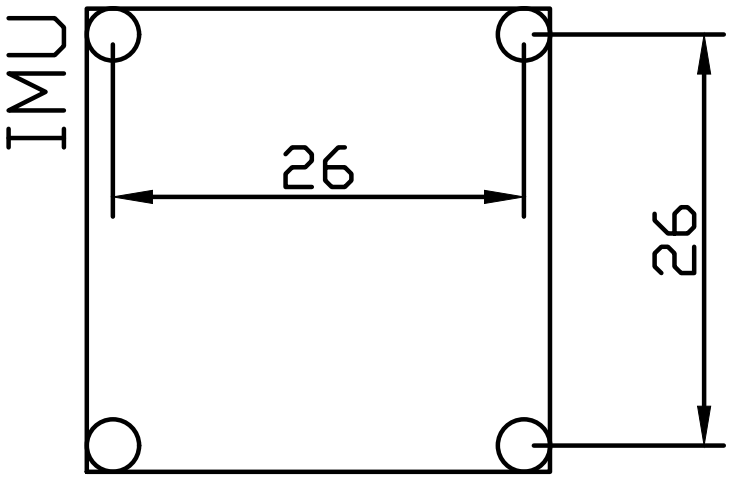


Figure D.1: IMU board design. (dimensions in mm)

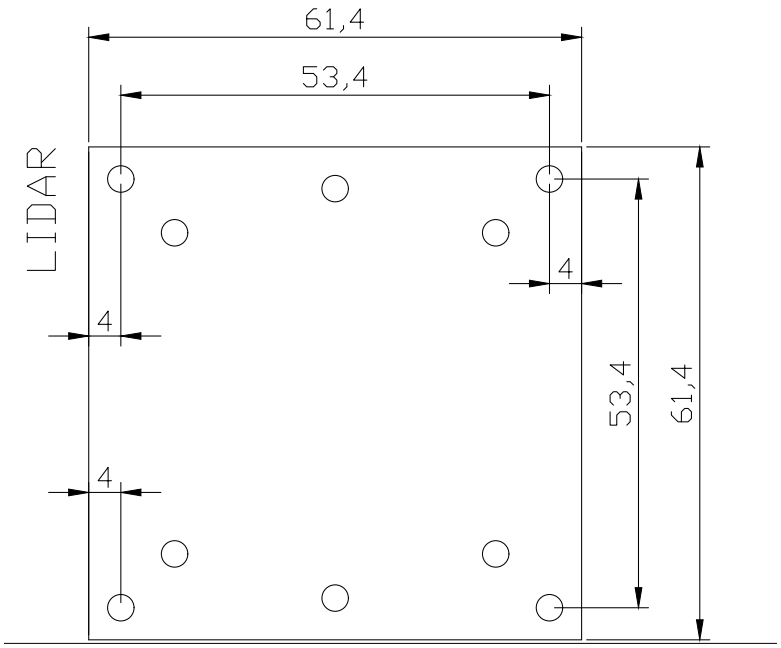


Figure D.2: LiDAR board design. (dimensions in mm)

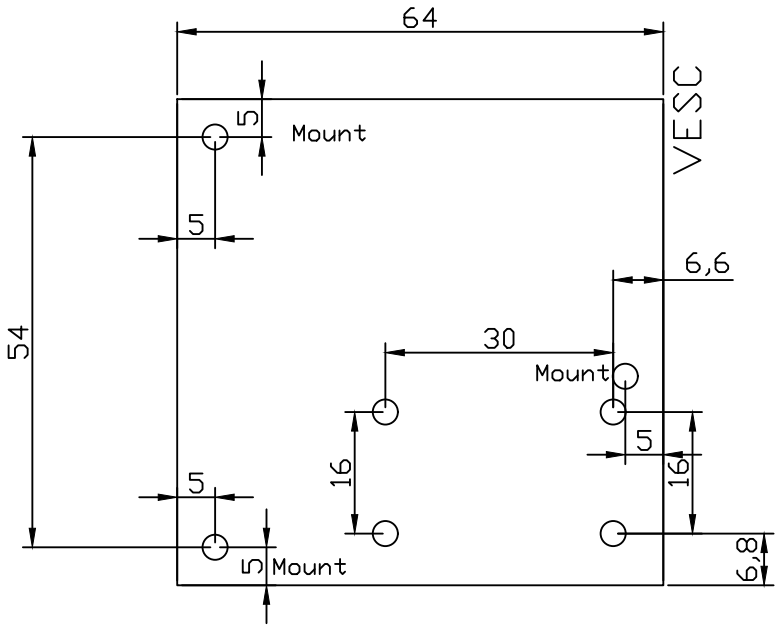


Figure D.3: VESC board design. (dimensions in mm)

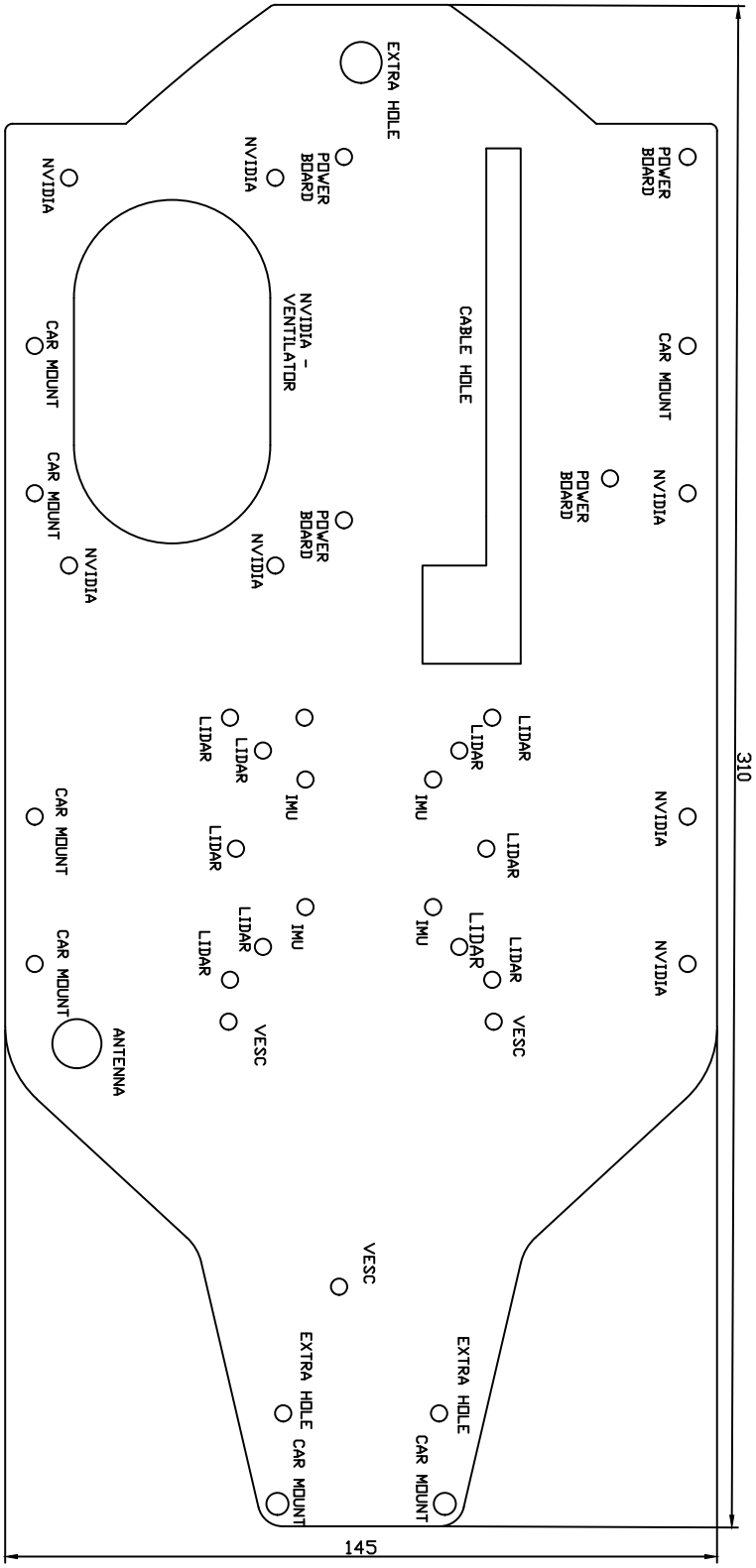


Figure D.4: Board design for the smaller car model.(dimensions in mm)

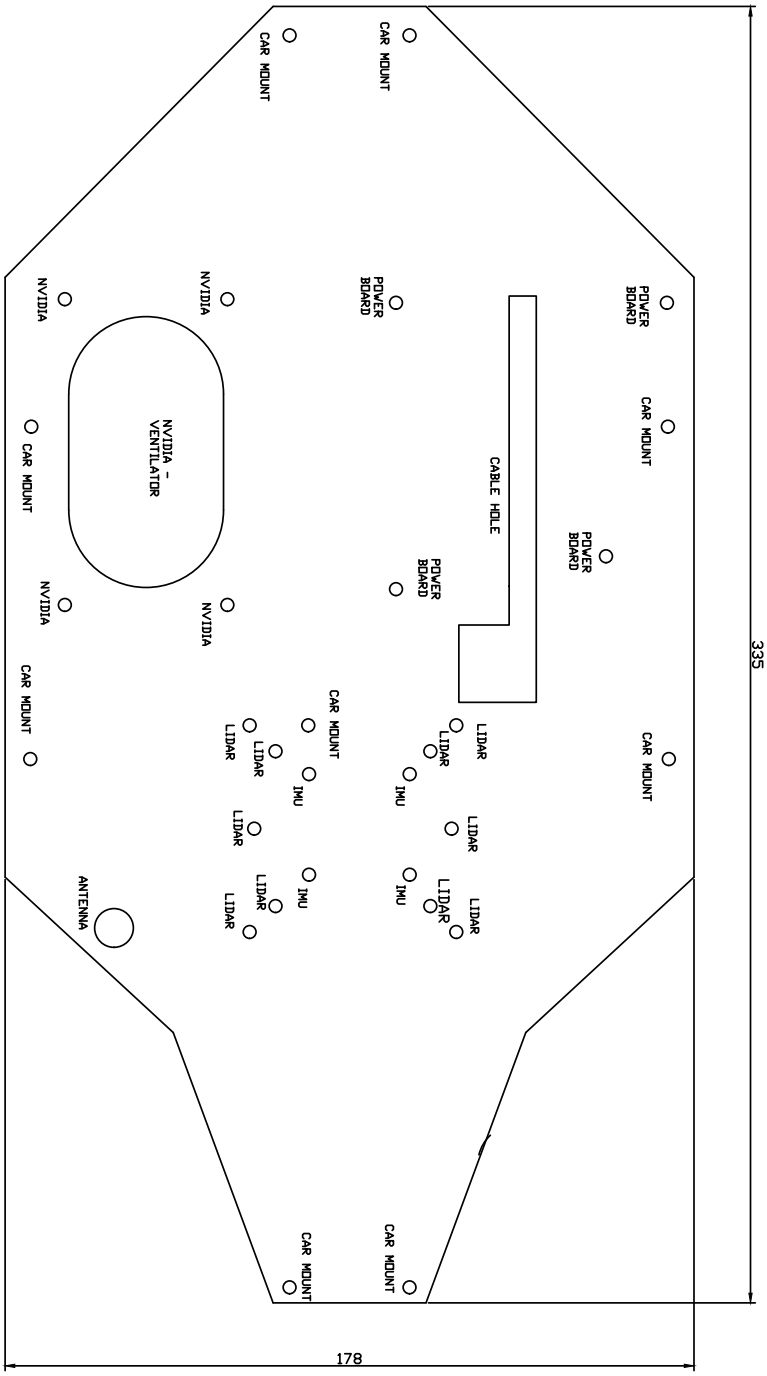


Figure D.5: Board design for the bigger car model.(dimensions in mm)

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Dusil Jan** Personal ID number: **457007**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Systems and Control**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Slip detection for F1/10 model car

Bachelor's thesis title in Czech:

Detekce smyku pro model auta F1/10

Guidelines:

1. Familiarize with the F1/10 autonomous competiton.
2. Order parts and mount two new racing models.
3. Design and make mechanical parts and electronics so the result would be more universal than the one proposed ones by the f1tenth competition organizers.
4. Configure inertial measurement unit (IMU) and try to implement a method to detect slip based on data from it.
5. Thoroughly test and document the results.

Bibliography / sources:

1. Martin Vajnar: Model car for the F1/10 autonomous car racing competition, diplomová práce ČVUT, 2017
2. InvenSense Inc.: MPU-9250 Product Specification Revision 1.0, 2014
https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdfa

Name and workplace of bachelor's thesis supervisor:

Ing. Michal Sojka, Ph.D., Embedded Systems, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **24.01.2019** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

Ing. Michal Sojka, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature