Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Development of verification platform for overactuated vehicles

Tomáš Rutrle

Supervisor: Ing. Tomáš Haniš, Ph.D.
May 2020

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Rutrle Tomáš** Personal ID number: **478067**

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

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

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Development of verification platform for overactuated vehicles

Bachelor's thesis title in Czech:

Vývoj verifikační platformy pro přeaktuovaná vozidla

Guidelines:

The goal of the thesis it to develop and assemble the sub-scale vehicle platform for verification of control algorithms for over-actuated vehicle. The thesis will address following points:
1) Mechanical design and construction of over-actuated vehicle platform with independently steered wheels.
2) Assembly of HW for purpose of control algorithms deployment (sensors, communication, computational power and actuators)
3) Development of control algorithms for over-actuated vehicle
4) Testing of developed verification means and control algorithms

Bibliography / sources:

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini – Vehicle Dynamics – Duisburg 2014
[2] Hans B. Pacejka - Tire and Vehicle Dynamics – The Netherlands 2012
[3] Franklin, Powell, Emami-Naeini: Feedback Control of Dynamics Systems. Prentice Hall, USA
[4] Robert Bosch GmbH - Bosch automotive handbook - Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. : Bentley Publishers

Name and workplace of bachelor's thesis supervisor:

Ing. Tomáš Haniš, Ph.D., Department of Control Engineering, FEE

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

Date of bachelor's thesis assignment: **28.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. Tomáš Haniš, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
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

Acknowledgements

I would like to express my appreciation towards my supervisor Tomáš Haniš. Not only for his continuous and patient guidance in regards of this thesis, but also for his non-work related support.

My personal gratitude belongs to my supporting family, especially to my parents. This thesis is a product of their care and support for which I am extremely grateful.

Declaration

I, Tomáš Rutrle, hereby declare that this thesis is a product of my own work and that to the best of my knowledge all information sources have been listed in accordance with the methodical instructions for observing the ethical principles in the preparation of a university thesis.

Prague, May 2020 .....

Abstract

The main focus of this thesis is on the development of an overactuated platform suitable for the verification of four wheel steering control algorithms. The mechanical construction is presented as well as the instrumentation of the platform including sensor deployment, data acquisition and actuator control. Furthermore, the validation of the implemented sensors takes place followed by vehicle states acquisition necessary for dynamic control using sensor fusion. At last, simple kinematic feed-forward control algorithms are presented to showcase the performance of the platform.

Keywords: verification platform, overactuated vehicle, four wheel steering, dynamic vehicle control, complementary filter

Supervisor: Ing. Tomáš Haniš, Ph.D.

Abstrakt

Hlavním cílem této bakalářské práce je vývoj přeaktuované platformy vhodné pro verifikaci řídících algoritmů vozidel se čtyřmi nezávisle zatáčejícími koly. Nejprve je představena mechanická konstrukce vozidla spolu s instrumentací platformy, imlementací senzorů, získáváním dat a řízením aktuátorů vozidla. Nasazené senzory jsou dále validovány a s využitím senzorové fúze jsou získány kličové stavy vozidla potřebné k dynamickému řízení. Na závěr jsou vyvinuty a otestovány kinematické řídící algoritmy za účelem ověření funkčnosti platformy.

Klíčová slova: verifikační platforma, přeaktuované vozidlo, zatáčení čtyřmi koly, dynamické řízení vozidla, komplementární filtr

Překlad názvu: Vývoj verifikační platformy pro přeaktuovaná vozidla

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

With the current state of the automotive industry new challenges and possibilities arise. Vehicles are pushed towards being fully autonomous. Topics such as drive-by-wire or decision making algorithms are being discussed more than ever. But to develop any new system takes time and resources. That is why often the first step in a development cycle is a creation of a mathematical model on which the system properties are studied.

The second step would be the development of a verification platform. That is to say an experimental platform on which the theoretical results are to be verified. Four wheel steering has been already introduced by a number of commercial car manufacturers, but to a limited extend. The advantages of such systems lie in maneuverability as much as in controllability of the vehicle. But by introducing a fully independent four wheel steering and proper control algorithms we hope to show these qualities can further improve.

1.2 Thesis goals

The goals of this bachelor thesis are the following:

- **Design and construction of over-actuated vehicle platform with independently steered wheels.** In the first part of the project a commercially available electrical 1:5 remotely controlled car will be mechanically modified to allow for four wheel steering. These and other modifications will then be made in order to prepare the platform for instrumentation.
- **Assembly of hardware for purpose of control algorithms deployment.** Multiple on-board processing units are to be implemented to enable easy control algorithm deployment using Matlab & Simulink. In this phase we will implement actuators controllability, sensor data logging and mutual communication between all on-board electronic control units(ECU).

- **Development of control algorithms for over-actuated vehicle.** Finally control algorithms are to be developed and implemented. Using these algorithms we are going to verify the properties of the platform.

1.3 Thesis outline

This thesis will be divided into five chapters. The first chapter will consist of research and state of the art analysis. What is overactuation and four wheel steering, where and how is it being utilized? In the second chapter the mechanical construction of the platform will take place. Here the process of modification of the platform to allow for four wheel steering will be described. This will be followed by the instrumentation chapter. In this chapter the overall architecture regarding the logical and computational units will be explained, together with individual implementation of all three processing units. Furthermore the mutual communication channels between those will be installed as well as all the sensors and data acquisition. The fourth chapter will be of experimental character. With the instrumentation and data acquisition finished, experiments will be made to validate the functionality of the on-board sensors. The acquired data will also have to be processed to obtain desired vehicle states necessary for the vehicle control. The last chapter will focus on feed-forward control design to verify the functionality of the system as a whole.

Chapter 2

Topic analysis

2.1 Overactuated vehicles

A vehicle moving in a three dimensional space has six degrees of freedom. Given a Cartesian space , three of these degrees would be translational $[x, y, z]$ a three rotational $[roll, pitch, yaw]$. But if we approximate a vehicle as a rigid body moving across a two dimensional Cartesian plane, let's say defined by the x and y axis, it is obvious such body can be absolutely described by three general coordinates. In other words it has three degrees of freedom which are the two $[x, y]$ coordinates and the rotation along the z axis - yaw rate. Traditionally motion control systems contain a number of actuators equal to the degree of free rigid-body modes [4]. In a standard road vehicle one of the actuators is a mechanical linkage between the steering wheel and the front axle, the other one is the combination of accelerator and brake pedals. The pedals control the force applied along the x axis, the steering wheel controls both the y axis forces as well as the yaw torque. With these two actuators controlling three degrees of freedom, standard vehicle control is thus undetermined [1]. This means that the system cannot achieve any position with arbitrary orientation and vice versa. Overactuated vehicle is defined as vehicle with larger number of control inputs than the given number of degrees of freedom [1]. By introducing four independent steering inputs, this condition of overactuation is thus met.

The first commercially available vehicle equipped with four wheel steering (4WS) technology was a 1988 Honda Prelude which used purely mechanical systems to determine the steer angles of the rear axle. For lower steering angles the rear wheels would turn in the same direction as the front ones. But by increasing the steering angle the rear wheels would start rotating in the opposite direction. Currently a considerate number of car manufacturers do have models with 4WS at their disposal. One of the more notorious examples is the Volkswagen Group 4WS system available in a number of their high end sport / luxury cars. The systems are no longer mechanical and are much more sophisticated, yet the idea and functionality remains the same. At lower speeds, agility is of preference and the rear axle counter steers relative to the front one, virtually shortening the wheel-base of the vehicle. At higher

speeds where stability is important and thus the rear axle steers in the same direction as the front axle. The rear steering actuators are controlled by an on-borad computer and some of the input quantities are ground speed, steer angle or longitudinal and lateral acceleration of the vehicle [5]. Such system where no mechanical connection between the steering wheel and the steered wheel can be found is often referred to as a "Steer by wire" system.

2.2 State of the art

Commercial vehicles capable of four wheel steering are already widely available, but it is fair to say that in the current state the potential of the technology is far from met. Although even the limited implementation of the 4WS technology has its advantages as mentioned before, in the hands of an experienced driver the available 4WS technology bears no significant improvements during high speed maneuvers [6]. But commercial vehicles do not represent the current state of the technology development. To understand the state of the art situation, we will have to look at some conceptual vehicles.

2.2.1 Protean360

One of the examples of the 4WS technology currently in development is the Protean autonomous EV. Protean as a company specializes in a in-wheel motor technology. Since 2018 a steering wheel module capable of full 360 degree rotation has been in development, this module is shown in the following figure 2.1

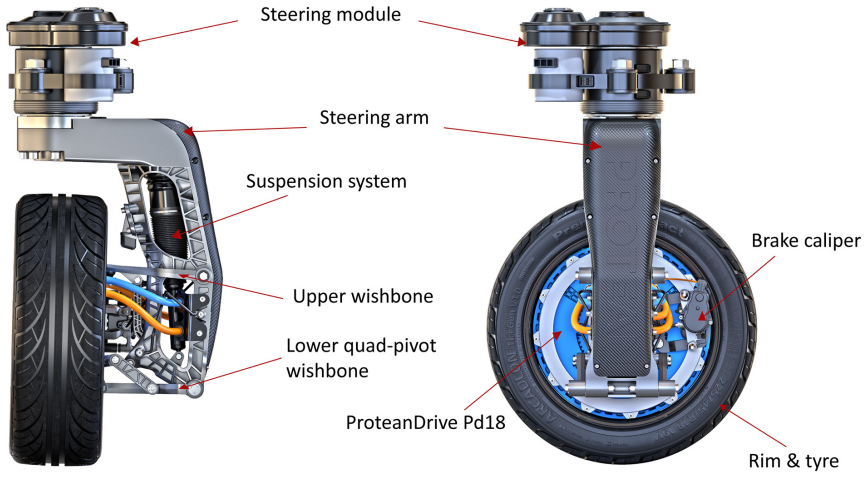


Figure 2.1: Protean360 drive and steer module, adopted from [7]

To cite the CTO of the company: "Part of its attraction is that it allows creation of a very flexible, totally flat-floor vehicle platform that can be adapted for various uses including commuting and delivery of goods and services, so the range of vehicles supported will be varied" [7]. The project promises to enable development of EVs of unmatched maneuverability, as such vehicles could rotate along the z axis within its own footprint.

2.2.2 Jeep Hurricane

Although not capable of turning its wheels by 360 degrees, the Jeep Hurricane sparked interest in many off-road enthusiasts in 2005. The Hurricane shown in 2.2 was meant to be an extreme vehicle showcasing the engineering and off-road capabilities of a four wheel independently steered offroader. Despite not making it into commercial production, several prototype units were produced and shown to the public. An interesting fact about the prototype was its engine situation. Or rather two engine situation, since the Hurricane hosted two V8s, one in the front and one in the rear. This was not crucial for the steering capabilities though. When it comes to the steering situation, the concept was capable of three modes. First one was the counter steer, where the rear axle steered in the opposite direction of the front axle. In the second mode, the rear wheels followed the angle of the front ones, moving the car to the side without changing the heading. The last mode enabled all the wheels to "toe in" resulting in the Hurricane rotating in place [8].

Figure 2.2: Jeep Hurricane concept, adopted from [8]

Chapter 3

Mechanical construction

3.1 Construction outline

This part of the thesis will focus on the mechanical design and construction of the platform. Firstly a commercially available high end RC car will be chosen, which will then be repurposed as a base platform for the project and modified for four wheel steering. The goal is to mount all needed instrumentation on the RC cars platform so it will fit into the original bodywork. Doing so will preserve the rigidity and body properties of the RC car.

The car will be divided into two physical levels, effectively dividing the drive-train and higher voltage parts of the car from the instrumentation part. The lower level will consist of:

- Four steering mechanisms including the servos and their power and control unit
- Brushless DC motor (BLDC motor) with an electronic speed controller mounted to a central differential
- Remote controller receiver unit
- Lithium Polymer (LiPo) batteries as a power source for the motor and servos

The upper level will host all the processing units of the car. This layer will then be easily accessible so any modifications to the instrumentation can be done without any significant mechanical operations.

3.2 About the base platform

RC cars are produced in several standardised scales with the 1:5 scale being the largest one. This makes the scale the best fit for this project. Although gas powered platforms are also common, an electrical version is the obvious choice due to simple maintenance and controllability of a BLDC motor. The chosen model is a four wheel drive Losi®1:5 DBXL-E which is showcased in the following figure 3.1.

(a) : Top view **(b) :** Side view

Figure 3.1: The Losi®1:5 DBXL-E which will serve as a base platform for this project, adopted from [9]

Properties of the stock RC car as provided by the manufacturer [9]:

- Length, width, height: 844, 501, 308 mm
- Weight: 12.5 kg
- 4 mm thick aluminium platform
- Motor: Brushless, sensorless, 800 Kv
  - "Kv" referring to constant angular velocity the motor will have with no load when 1 Volt is applied
- 8S Dynamite Fuze 160A motor speed controller

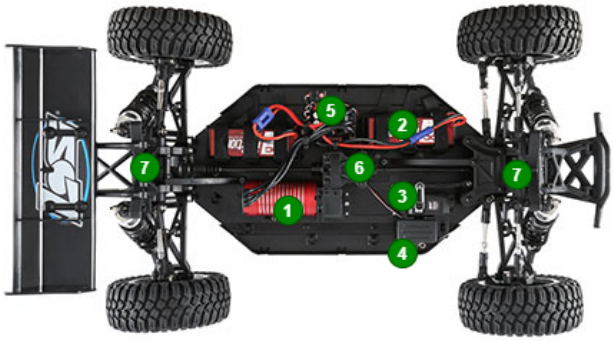


Figure 3.2: Part diagram of the Losi®1:5 DBXL-E which will serve as a base platform for this project, adopted from [9]

Parts provided with the car as shown in figure 3.2:

- **1.** Brushless 800 Kv motor
- **2.** 4S 14.8 V LiPo batteries
- **3.** Losi S900S steering servo
- **4.** Radio receiver
- **5.** 160 A motor speed controller
- **6.** Central mount differential
- **7.** Front and rear axis differentials

3.3 Front axle steering

The first step of the mechanical construction will be the reconstruction of the front axle. One of the advantages of using a platform primarily focused on hobbyists and RC car enthusiasts is that the car can be fully disassembled and every spare part can be bought and changed separately. For this reason the natural way to go about the platform conversion is to use the parts and mechanisms already available on the car. In other words, copy the steering design that has been implemented by the manufacturer and apply it on each wheel.

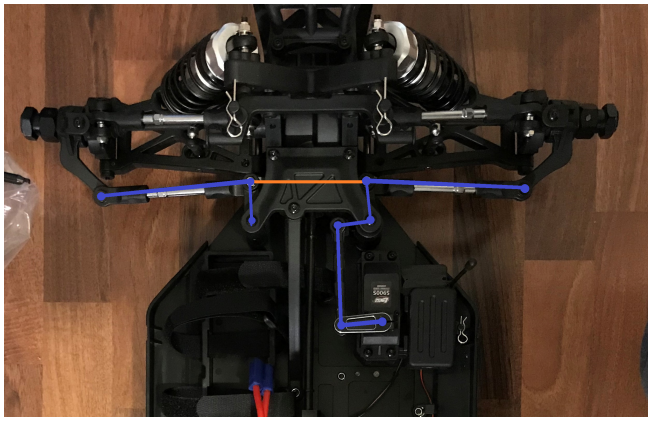


Figure 3.3: Losi®1:5 DBXL-E front axis steering linkages

Figure 3.3 showcases the front axle of the car. The linkages and pivots leading from the servo to the wheel are highlighted in blue and the orange linkage represents the physical connection between the left and right wheels. By simply removing this connection we end up with independently steering front axle, the removed connecting rod is shown in figure 3.4. We will leave the servo and control arms of the front right wheel as they are and mirror copy the layout for the front left wheel.

Figure 3.4: Connecting rod and steering pivot from the front axle which are to be replaced in order to allow for independent steering

Firstly, one of the pivots will need to be replaced. The old part, to be seen in figure 3.4, will be replaced with a more complex part which is shown disassembled and assembled in figure 3.5. The new introduces a spring loaded shock damper and allows for servo connection. In case an unwanted stress is applied to the steering mechanism against the servos will, this stress is not transferred to the servo motor directly, but instead it is largely reduced by the damper. In other words the damper allows for some rotation when enough force is applied with the servo remaining static.

(a) : Disassembled

(b) : Assembled

Figure 3.5: New steering pivot which allows for servo control arm to be connected and introduces a shock damper

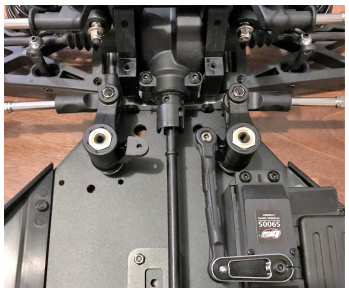


Figure 3.6: Front axle independent steering pivots ready for servo connection

The replacement of the steering pivot is shown in figure 3.6. In this state the front axle is ready for the second steering servo to be mounted.

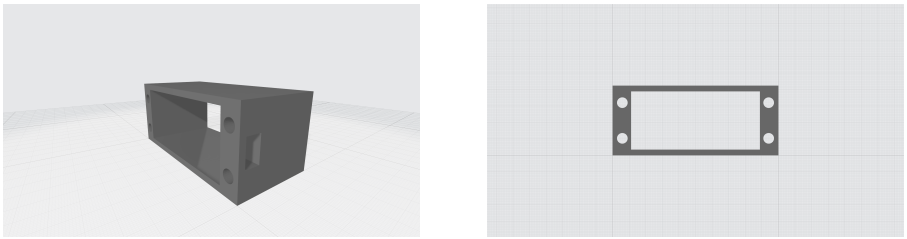
3.3.1 Steering servo and servo case

In this subsection the used steering servo will be described. The one used for this project is a Losi S900S for 1:5 scaled models and the servo is showcased in figure 3.7. The servo motor specifications as provided by the manufacturer are [10]:

- Length, width, height: 65, 57, 30 mm
- Weight: 200 g
- Type: Digital with pulse width modulation decoding
- Torque: 30 kg-cm
- Angular speed: 60° in 0.21 s

Figure 3.7: Losi S900S steering servo

In order to mount the servo on the platform a case had to be designed. With four 4 mm holes and known dimensions, the final case design is shown in figure 3.8. This design was created in an open source computer aided design (CAD) freeware called *FreeCAD* as all the other models in this thesis will be. The case was designed in such a fashion that the servo tightly slides into it and the screw holes then sit on top of the holes shown in figure 3.8(b). An $M4$ threaded rod will then be inserted into these openings and using these rods the servo will be bolted and secured straight to the base platform.



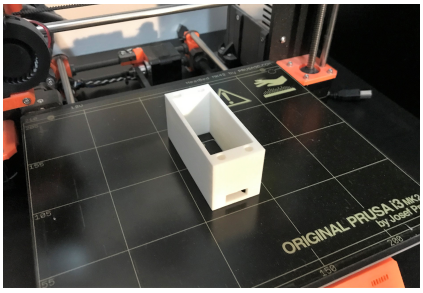
(a) : Combined view **(b) :** Top view

Figure 3.8: Servo case 3D model

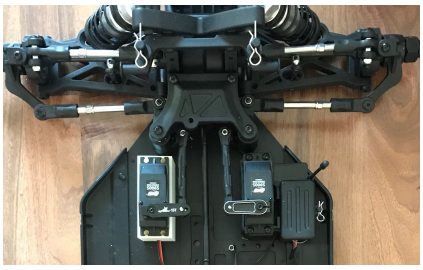
The case is 3D printed using a polylatic acid (PLA) filament. This material was used due to its suitable properties for the application. Its overall strength and stiffness make it an ideal material for printing parts where no flex is expected and the material is rated to start deforming at 60°C and higher, which should not pose any issues. PLA is also the most user friendly 3D printing material, making it easy to reproduce any custom made parts used on this car [11]. The settings used for the slicing and printing of this part were:

- Material: PLA
- Extruder width: 0.15 mm
- Temperatures: 215 / 60 °C
- Infill: hexagonal, 80%
- Supports: none

The printed servo case is showcased in figure 3.9(a). The mounting point of the new servo is than measured relatively to the front right servo, as shown in figure 3.9(b), in order to ensure same kinematic properties of the individual steering chains.



(a) : Printed servo case

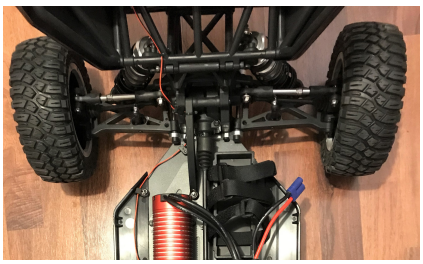


(b) : Position of the front left servo motor

Figure 3.9: With the front left servo motor in place the car is now capable of front axle independent steering

3.4 Rear axle steering

After the front axle finished the next step is to reconstruct the rear axle. In this part of the construction we will follow the same formula as we used for the front axle. That means identifying the parts that will need to be replaced and modified to allow for servo connections. Although it became obvious after the initial inspection that the rear axle will demand more modifications in order to make servo controlled steering possible, the transformation will still be made. The original build of the rear axle is shown in figure 3.10.



(a) : Top view of the rear axle

(b) : Unmounted rear axle

Figure 3.10: The original rear axle of the LOSI RC car

First the rear axle is disassembled into separate parts to determine which parts will need to be replaced. The first part is the rear wheel drive shaft. The original shaft was intended only for one DOF relative to the rear axle suspension. The new shaft which is used for the front axle as well enables another DOF relative to the steering angle of the wheel. The comparison of the two shafts is shown in figure 3.11.

Figure 3.11: Comparison of rear (top) and front (bottom) wheel drive shafts. For the rear axle the former had to be replaced with the latter.

The next step is to replace the mounts which host the drive shaft and wheel bearings of the rear wheels, so a steering rod can be connected.

3.4.1 Rear differential mount

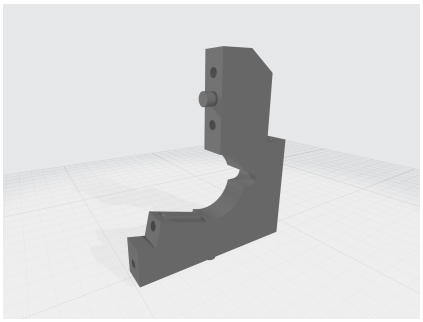
As shown in figure 3.9(b) the steering rack is hidden under a cover which also doubles as a part of the construction as it holds the steering pivots in place. Figure 3.12(a) shows the covering part. In order to mount this cover to the rear axle, the differential mounts will have to be adjusted so the cover can sit on top of them. The original differential mounts and how it needs to be modified can be seen in figure 3.12(b).

(a) : Steering rack cover

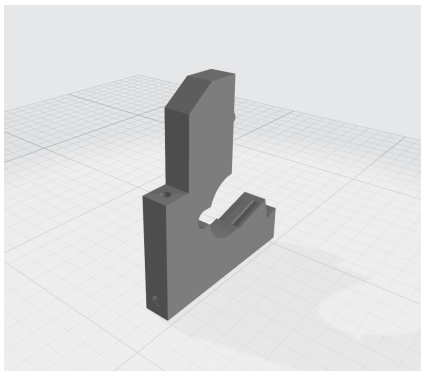
(b) : Original rear axle differential mount

Figure 3.12: Rear axle steering rack cover and differential mount. The highlighted red area shows how the mount will have to be changed

Simply cutting away the area and drilling a hole will not work, because the mount is not wide enough to hold a custom screw hole. The solution will be creating a custom model which is shown in figure 3.13.



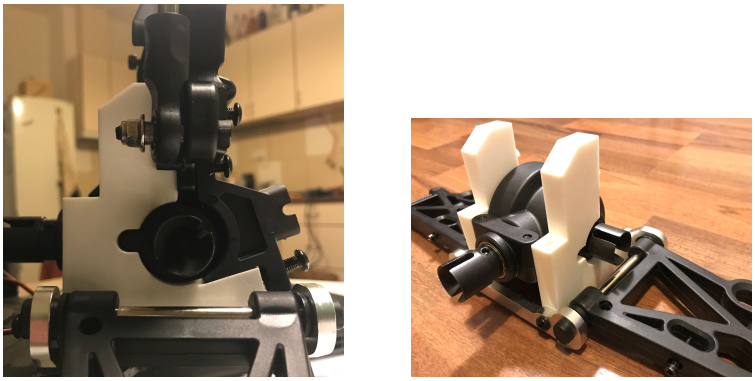
(a) : Front point of view exactly copies the original part



(b) : Rear point of view shows the desired cut out and screw hole

Figure 3.13: The new model of the rear differential holder

After printing the part with the same settings as used in section 3.3 the mounts were fitted in place of the old parts and figure 3.14 shows the result.



(a): (b):

Figure 3.14: New differential mounts printed and fitted

3.4.2 Rear steering linkages and servos

In this state the rear axle is mounted back onto the platform and is prepared for steering links to be added as shown in figure 3.15.

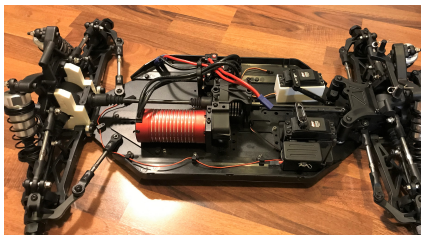
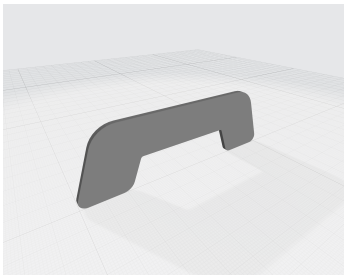


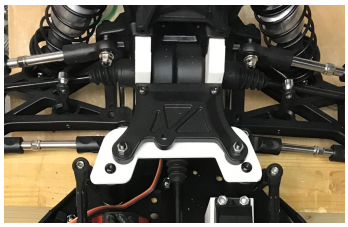
Figure 3.15: With the modified rear axle mounted to the platform the next step is to set up the rear steering links and servos

The same steering pivots as in 3.5 will be mounted. Due to some asymmetrical properties of the front and rear axles the position of the steering pivots will be slightly offset compared to the front axle. An extension to the steering rack cover from figure 3.12(a) will be designed. The idea is to create a thin but sturdy part which will extend the reach of the rack cover and will be able to host the steering pivots. This way the intended structure will remain preserved.

The model presented in the following figure 3.16(a) was printed with 100% infill so holes could be drilled after printing. In the picture 3.16(b) we can see the part mounted together with the steering pivots.



(a) : 3D model of the extension



(b) : Extension is printed and mounted

Figure 3.16: Rear steering rack cover extension

For the rear steering servos we will use the same servo case as previewed in figure 3.8. The position of the rear left servo is straight forward since there are no obstructions on the platform. For the rear right servo though, the possibilities are limited because of the motor. The servo will have to be mounted right next to the motor leaving just enough space for the car cover to close and the steering arm will be lifted to avoid contact with the motor. But as long as the steering arm is kept horizontal respective to the platform (as all the arms of the other three servos are), this modification will result in no difficulties.

With the rear servos mounted as shown in figure 3.17, the independent rear axle steering is now finished meaning that at this point the goal of all four wheels to be able to independently steer is achieved.

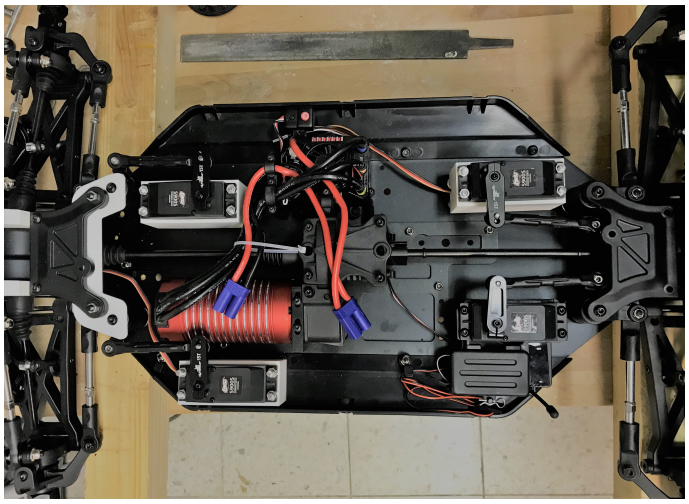


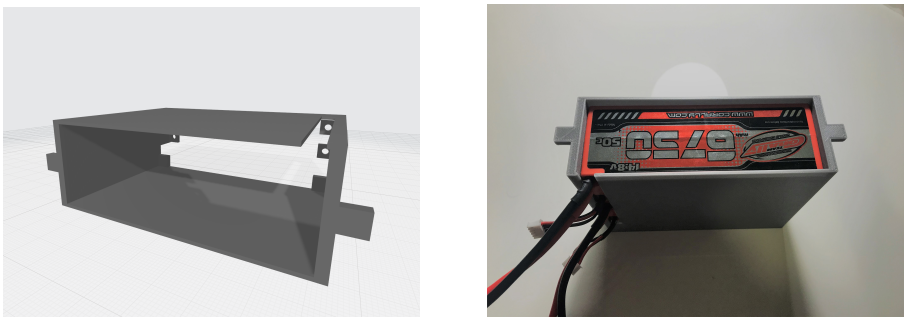
Figure 3.17: Rear axle steering is now finished and the base platform is fitted with four wheel independent steering

3.5 Miscellaneous hardware

The last step of the hardware construction will be to rearrange the bottom level of the platform and mount a second layer made of plexiglass which will be used for mounting all the desired computational units.

3.5.1 Battery box

First we will change the position of the motor controller in order to create enough space for one of the vehicle's batteries. The two LiPo batteries necessary for the speed controller to operate will be stacked on top of each other. The following figure 3.18 shows the model of the battery box.



(a) : 3D model of the battery box **(b) :** Batteries in the battery box

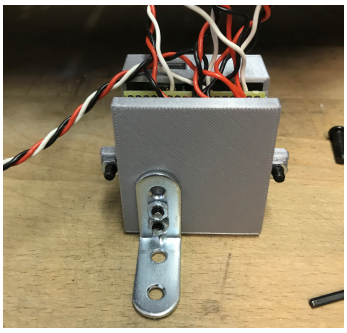
Figure 3.18: Model of the battery box

3.5.2 Servo control and power unit

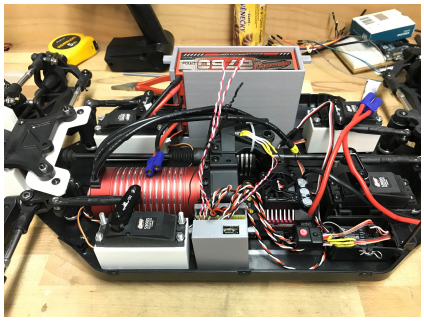
Originally the one servo was connected directly to the receiver unit which was powered at 6 V by the motor controller. The receiver also generates a PWM signal respective to the remote controller user input and this PWM signal was then fed directly to the servo motor. The end goal will be to decode the user input related PWM, run it through an algorithm and generate according PWMs for the four servo motors.

With this in mind a circuit board will be created to meet the following criteria: provide power for the servos, have four independent inputs for the PWMs and host an Arduino Nano used for digital and analog input reading. To fix the finished circuit board in place and protect it from elements a special enclosure will be designed. The modeled case is divided into two parts which can be bolted together and hold the board in between them selves. The case has two openings. One larger for all the necessary wires and one smaller specifically designed for the Arduino Nano USB connection.

The case itself will then be mounted to the platform using a steel mounting piece which is showcased in figure 3.19(a). In 3.19(b) we can see the case mounted together with shifted motor controller and mounted battery box. In this state all the hardware of the bottom level is finished and the platform is ready for the plexiglass installation.



(a) : Circuit enclosure is printed and ready to be mounted on the platform

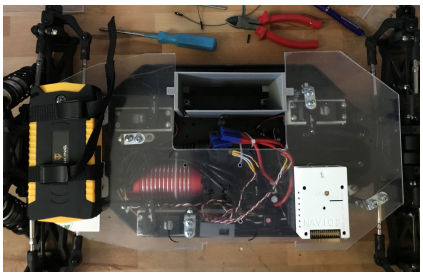


(b) : View of the platform in its current state

Figure 3.19: In this state all the hardware of the bottom level is finished

3.5.3 Plexiglass installation

As mentioned before, a second level will be mounted above the servomotors and other parts mounted directly to the chassis. For this a polystyrene sheet will be used as it offers great toughness mixed with flexibility and lightness. The shape of the sheet will copy the shape of the base platform in order to maximize space. Four mounting points will be added and the plastic sheet will be bolted on top of them as shown in figure 3.20(a).



(a) : The polystyrene sheet is bolted to four steel mounts

(b) : With the second layer in place, the car cover still closes all the way

Figure 3.20: Second layer for instrumentation is mounted

Chapter 4

Instrumentation

4.1 About instrumentation development

Before any decision about the instrumentation can be made, a list of requirements will have to be set. This will lead to organized progression with milestones along the way, as opposed to aimless development with no obvious short term goals. For this reason the chosen approach is to firstly create communication links between the individual computational units and then enable all their required processes.

4.1.1 Control architecture

The main goal of this chapter is to create a system which from a user's point of view behaves like a simple input-output system with a black box in the middle. To achieve this, the goals to meet go as follows:

- Simulink based control
- Remote controller user input reading
- Independent servo motors PWM control
- Enable user friendly control algorithm development
- GPS and IMU data reading
- Custom RPM sensors for wheels and motor
- Data logging

The following diagram 4.1 shows the goal control architecture, dividing the listed tasks into three levels by their accessibility by the user. That means that high level tasks will be solely implemented and controlled by the user. Medium level tasks will be fully operational and put in place and their modification will not be necessary to allow for full control of the platform. But these processes will still be easily accessible when needed, for example when a calibration is necessary. The lowest level of the control architecture will consist of self sustaining processes which would not benefit from additional user interference.

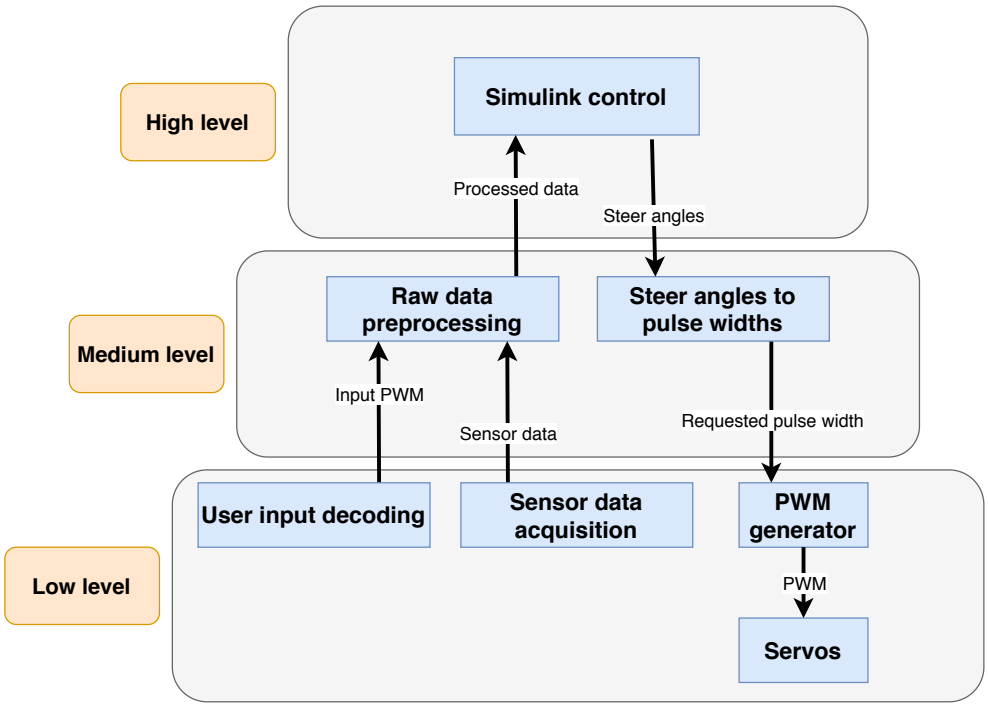


Figure 4.1: Control architecture of the platform

4.1.2 Processing units

Here a short overview of the processing units chosen for the platform will take place, including the reasons behind their selection.

Intel NUC

The Intel NUC7i7BNK mini computer will take the role of a brain of the platform. Its compact size, connectivity and powerful processor make it an ideal on-board control unit. The specifications of the mini PC are:

- 7th Gen *Intel*® i7 processor
- 16 GB DDR4 memory
- 512 GB SSD

The role of this unit will be to run a real-time Simulink model, which will run the car control algorithm and collect all data of the platform. Essentially all other processing units will be connected to the NUC and will work on a master-slave basis.

Navio2

Navio2 is an autopilot HAT for the Raspberry Pi mini computer. The following figure 4.2 shows the unit with all its components.

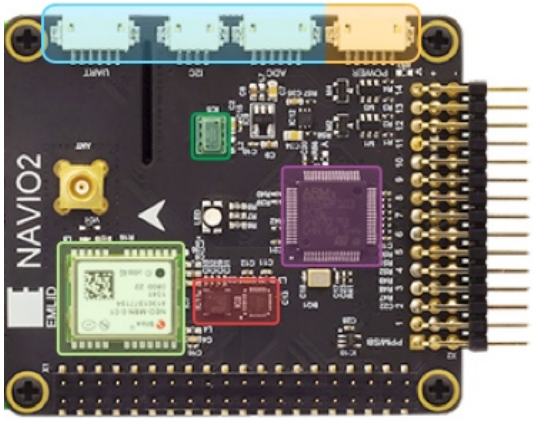


Figure 4.2: Navio2 - autopilot Raspberry Pi HAT, adopted from [12]

- GNSS receiver with external antenna
- Dual 9DOF IMU units
- Extension ports: ADC, I2C and UART
- Power supply with protection and power sensing
- Barometer
- RC I/O co-processor: Accepts PPM/SBUS input and provides 14 PWM output channels for motors and servos

The Navio2 unit will be used for two things. Firstly, all the sensorical data from the board will be acquired and sent to the NUC computer for processing and logging. Secondly, the PWM outputs of the board will be used to control the steering servos.

Arduino NANO

A small board in form of an Arduino NANO will be used for added I/O flexibility as more digital or analog I/O pins could and will be needed. This unit will be used for input PWM decoding and custom RPM sensor integration.

4.1.3 Communication diagram

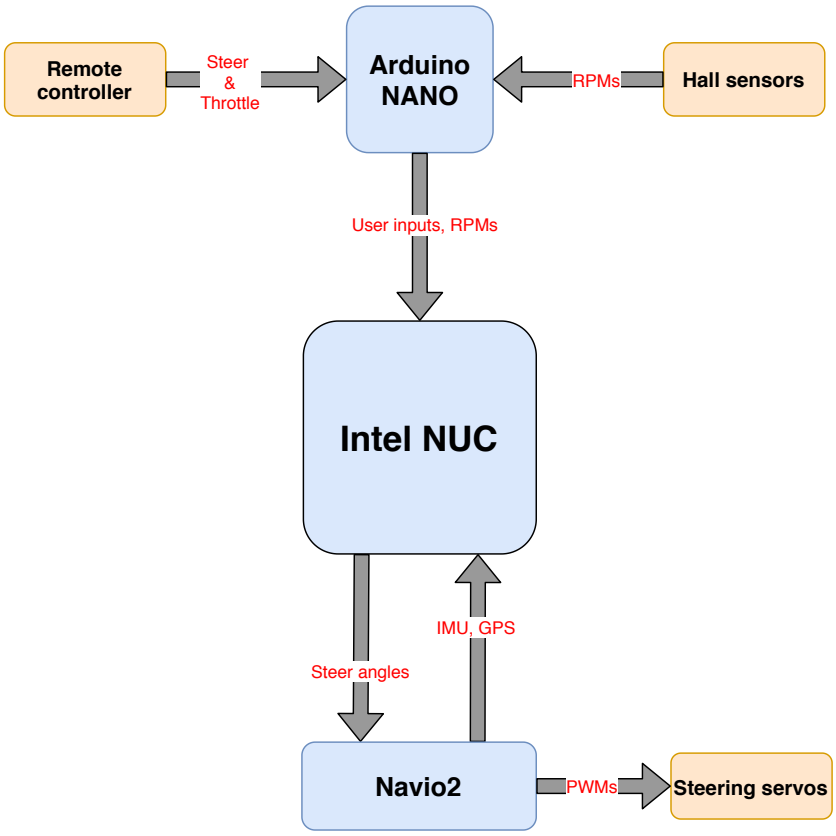


Figure 4.3: Communication diagram of the processing units used

The interconnections between the individual units including the key information to be transferred via these connections is showcased in diagram 4.3.

4.2 Arduino NANO

4.2.1 Receiver signal decoding

In this section of the thesis the process of the integration of the Arduino NANO board will be described. The main motivation to use this board was the need to decode a PWM signal from the receiver. Although the Navio2 unit is capable of PPM signal reading, in order to use that feature it would be necessary to convert the available 5 V PWM signals to 3.3 V PPM signal. There are commercially available converters capable of such task, but due to the flexibility and ease of use, the NANO was chosen for the decoding [13].

4.2.2 RPM sensors

When it comes to the actual car control algorithms development, one of the key information about the kinematics of the car are the revolutions of individual wheels and the motor. Since the BLDC motor used is a sensorless one, we will have to implement a RPM sensor for the motor as well.

One of the methods of measuring position, respectively rotation of a body would be to use the Hall effect principle. The principle states that when a conductor through which current is flowing in one direction is introduced perpendicular to a magnetic field, a difference in electrical potential can be measured transverse to the current path [14]. This effect is than used in so called Hall sensors which are going to be used to measure the RPM of the motor. This sensor was chosen due to the advantage of not being very sensitive to their surrounding conditions and being very easy to install and use consistently.

Our solution

The Hall sensor chosen for our application is a low noise, linear Hall Effect sensor with Analog Output with serial number A1324. The sensor is specified to draw maximum of 9 mA of current when operating on 5 V, while the 5 V supply pin of the Arduino NANO is rated at up to 0.5 A.

To measure the rotation, ten 1x3 mm neodymium magnets will be attached to the driven wheel of the central differential, as shown in figure 4.4.

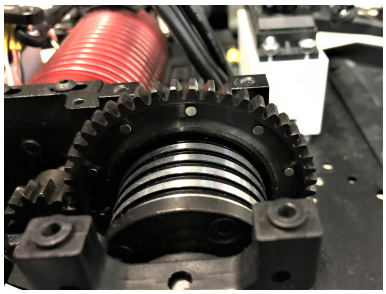
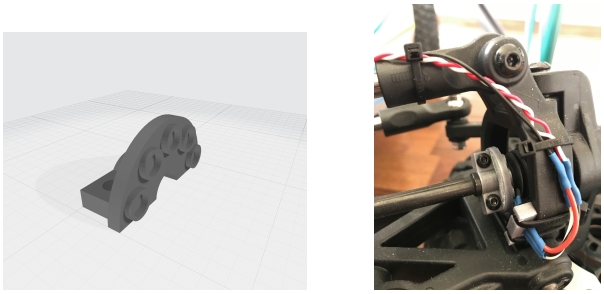


Figure 4.4: Neodymium magnets used for motor RPM sensing

The rotating magnets will cause pulses on the output of the Hall sensor which will be measured by the NANO. Two possibilities arise to calculate the actual number of rotations. Either count the number of pulses in a given time period or measure the time window between individual pulses. The second option will be implemented as it offers potentially higher frequency of updates and precision. Respectively, the higher the RPM of the motor the more frequent the measurements. On the other hand the first method has a fixed rate of measurements, which could come with compromises. For example, for a shorter time window the lower RPM values would not be detectable and for a larger time window, the accuracy may suffer due to averaging out the actual RPM in the given time period.

Wheel RPM measurement

To measure the rotations of the individual wheels the same sensor and technique will be used, since the NANO provides up to eight individual analog inputs. A small disc which will host the magnets will be mounted to the wheel axle. The Hall sensor will then be installed in such a fashion, that the vertical movement of the axis will not interfere with the readings. The model of the disc will be printed in two parts and bolted around the axle. The designed model can be seen in figure 4.5(a) along with the final installation shown in figure 4.5(b).



(a) : 3D model of a disc to hold the magnets

(b) : Installation of the disc and Hall sensor

Figure 4.5: Wheel RPM sensor implementation

4.3 Navio2

The Navio2 developed by EMLID is an extension HAT for the Raspberry Pi (RPi) unit. It allows the RPi to transform into a sensor and control module for all types of vehicles. An open source API is accessible with the product as well as a headless preconfigured Raspbian operating system which is required for the Navio2 board to fucntion properly [15].

For the purpose of this project a RPi 3 model B with a 64 GB micro SD card will be used. The specifications of this model are as follows:

- Quad Core 1.2 GHz Broadcom BCM2837 64bit CPU
- 1 GB RAM
- 2.5 A micro USB power port
- Micro SD port
- Connectivity: 4x USB type 2, ethernet, HDMI, 40 pin GPIO

Using the Python Multiprocessing library four processes will be created to collect, send and receive data. The first process will be in charge of IMU data acquisition, second process will collect the GNSS positional and velocity information, third process will gather this data from shared structures and send it to the Intel Nuc and the last process will receive steering angle data from the NUC and control the PWM duty cycles.

4.3.1 Dual IMU

The board contains two 9DOF inertial measurement units, the MPU9250 and LSM9DS1. Both of these combine a three axis accelerometer, gyroscope and magnetometer into one chip which is integrated into the board. An example code to test the individual IMUs is available with the API to showcase the functionality of the sensors and to display how to read their data. The following picture 4.6 shows the LSM sensor output with the values corresponding to $[x, y, z]$ axis.

```
Acc:  +0.007  +0.129 +10.179  Gyr:  +0.005  +0.057  +0.012  Mag:  -7.714 -86.652 +88.160
Acc:  +0.022  +0.086 +10.229  Gyr:  +0.004  +0.055  +0.012  Mag:  -7.946 -87.058 +88.566
Acc:  -0.108  +0.129 +10.136  Gyr:  +0.005  +0.050  +0.011  Mag:  -7.830 -87.290 +87.696
Acc:  -0.115  +0.100 +10.129  Gyr:  +0.005  +0.059  +0.017  Mag:  -7.656 -87.058 +89.030
Acc:  +0.007  +0.158 +10.114  Gyr:  +0.007  +0.056  +0.020  Mag:  -7.888 -87.638 +88.334
Acc:  -0.057  +0.144 +10.122  Gyr:  +0.004  +0.050  +0.013  Mag:  -7.482 -88.044 +88.450
Acc:  -0.100  +0.194 +10.043  Gyr:  +0.006  +0.060  +0.018  Mag:  -7.540 -87.638 +89.204
Acc:  -0.007  +0.151 +10.136  Gyr:  +0.006  +0.050  +0.011  Mag:  -8.062 -87.580 +88.044
```

Figure 4.6: Raw output of the LSM9DS1 sensor

By rotating with the unit along all three axis, we can determine the pitch, roll and yaw axis relative to the IMU output values which were defined prior. The mounting point of the Navio2 unit and the IMU axis orientation relative to the car are then showed in the following picture 4.7.

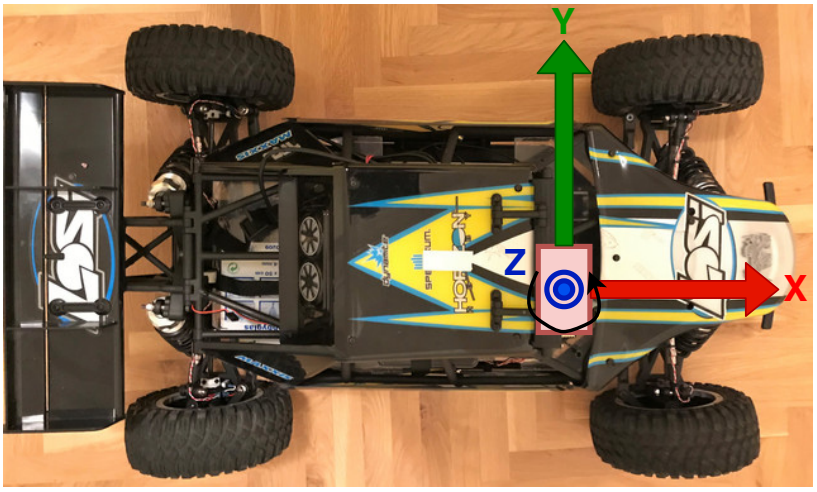


Figure 4.7: Axis orientation of the IMU unit

4.3.2 U-blox NEO-M8N module

The Navio2 features a NEO-M8N u-blox Global Navigation Satellite System (GNSS) module. The module supports concurrent receptions of up to three GNSS which improves availability and stability of signal and most improtantly accuracy. The performance of the NEO module is shown in the table A.1.

Positional information

The data from the NEO module are readable via UBX protocol messages [17]. An example code for UBX messages decoding is available in the Navio2 API. The protocol contains over a dozen message types ranging from positional information, to status updates and satellite information. To obtain the positional information, we will be interested in two types of messages. Firstly in the the "Geodetic Position Solution" (NAV-POSLLH) and secondly in the "Navigation Solution Information" (NAV-SOL). The structure of the messages and their contents are shown in tables A.2 and A.4.

Geographic coordinate systems

The positional information acquired via the previously mentioned messages will be acquired in two coordinate systems. The relation between the latitude, longitude, altitude geographical coordinate system and the "Earth centered, Earth fixed" (ECEF) system is similar to the relation between cartesian and spherical coordinates. The difference being that an established ellipsoid model is used instead of a perfect sphere. This so called "World geodetic system" is maintained by the National Geospatial-Intelligence Agency and the currently utilized model is the WGS-84. The position and orientation of the ECEF coordinate system relative to the ellipsoid is determined as follows [2]:

- The origin of the Cartesian system is at the centre of the ellipsoid
- The X axis lies in the equator of the ellipsoid and passes through the prime meridian (0 degrees longitude)
- The Y axis also lies in the equator but passes through the meridian of 90 degrees East
- The Z axis coincides with the polar axis of the ellipsoid

The separate coordinate systems are shown in the following figure 4.8, where $\lambda = longitude$, $\phi = latitude$.

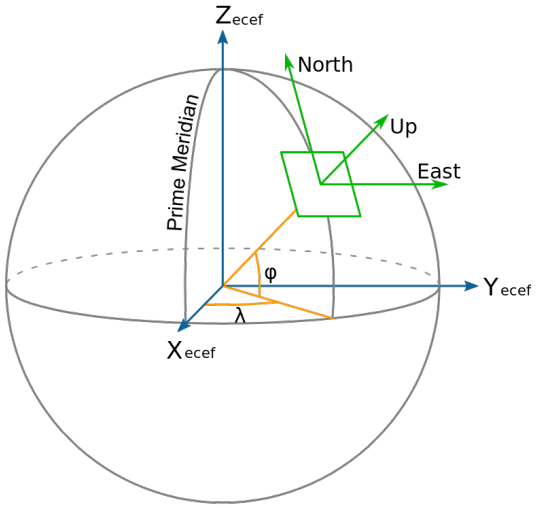


Figure 4.8: Geographic coordinate systems, adopted from [3]

Velocity information

To obtain velocity information from the NEO module, the "Velocity Solution in North, East, Down" (NAV-VELNED) message will be received, as per table A.3. This message contains information such as the ground speed of the vehicle and its heading with the combination of the velocity in the north and east direction. These will be useful later to determine the longitudinal and lateral velocities of the vehicle.

4.3.3 PWM servo control

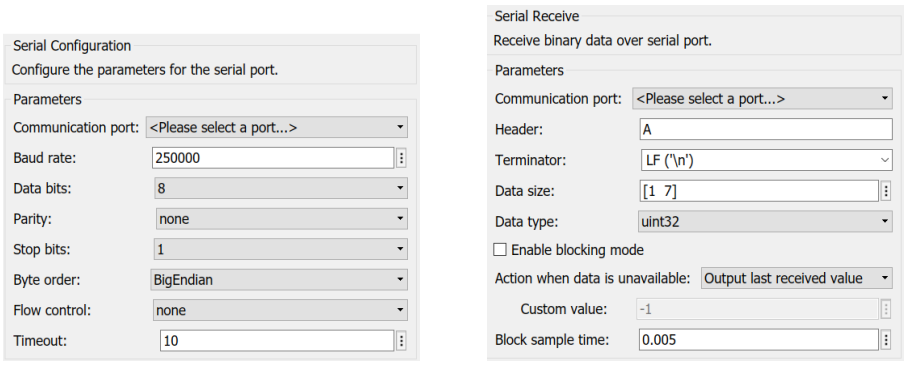
Apart from the sensor data, an essential feature of the Navio2 unit is the fourteen pin independent PWM output. First a frequency which is shared among all the outputs is set, for servomotors that frequency is typically 50Hz. Using the API functions we can set the PWM parameters easily. Duty cycle of these outputs is then adjusted separately. Pulse length between 1-2ms is enough to cover the whole angular extent of the servo motor. Important note from the manufacturer is the need for the Navio2 kernel driver of the PWM generator to receive data at least every 100ms. So it is necessary to update the duty cycle of the driver even when no change is required.

4.4 Intel NUC

The third and most important on-board computer will be the Intel NUC. In this section the integration of the NUC into the platform will be described. Both the Arduino Nano and the Navio2 units will be connected to the NUC and work on a one or two way communication basis, bringing the whole communication chain together and enabling the car to actually function.

4.4.1 NUC-NANO communication

Using the Instrument Control Toolbox we will be able to setup a serial communication between the NUC and the NANO. This toolbox allows us configure our serial port and have separate receive and send blocks. In this case we are only aiming for one way communication since there is no information that needs to be transferred from the NUC to the NANO. The serial configuration and receive blocks are shown in the following figure 4.9.



(a) : Nano serial configuration **(b) :** Settings of the receive block

Figure 4.9: NANO serial configuration

Seven values of a uint32 type will be received (rotations of motor and four wheels and user throttle and steering inputs). With one start bit, number of stop bits set to 1 and no parity, that makes for a 10 bit character and 40 bits per value. The total number of bits per message will then amount to $7 \times 40 + 20 = 300$ bits, where the "+20" bits represent the header and terminator of each message, which help to stabilize and control the data flow and decrease the number of incorrectly received characters. With the baud rate set to 250000 the link is able to transfer 250000 bits/s. With the current message size being 300 bits, the message transfer time should only be around 1.2 ms, making the 200Hz receive frequency well within safe boundaries. The following figure 4.10 shows logged RPM data from random accelerating a decelerating.

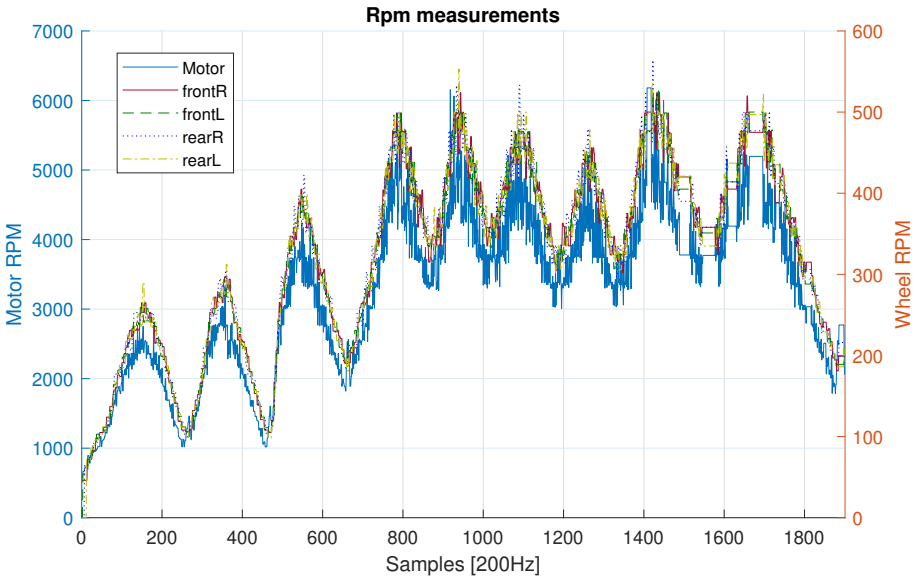


Figure 4.10: Logged RPM

Because the data was logged with all of the four wheels in the air, we can see that the RPM values of all wheels are, as expected, very similar. The noticeable noise of the motor RPM data may be caused by the imperfect distribution of the magnets used for the Hall sensor.

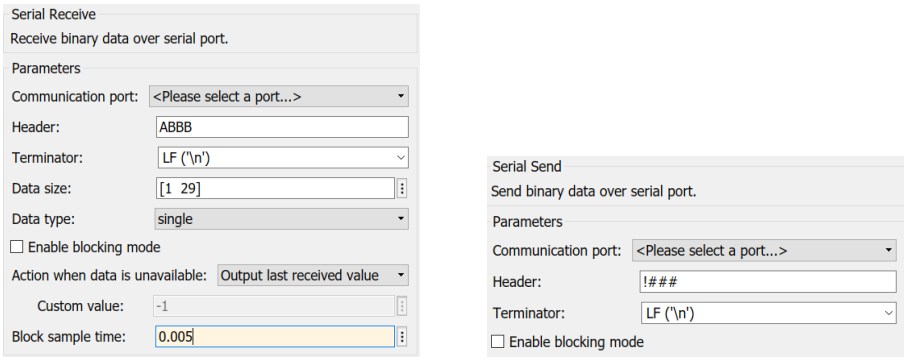
4.4.2 NUC-Navio2 communication

To setup a communication link between the NUC and the Navio2, USB to serial adapters will be used. Specifically a converter using PL2303HX micro controller which, with the correct drivers, allows for the user to treat the USB port as a serial port, essentially creating a bridge between the serial and USB protocols.

Using the same toolbox as in the previous section the serial port in the Simulink will be configured. This time data will not only be received but also sent, as the NUC needs to sends the steering angles to the Navio2. The serial configuration will be identical as in 4.9(a) and the settings of the serial Send/Receive blocks is shown in the following figure 4.11.

Twenty nine single precision values will be received with each message from the Navio2:

- 2x accelerometer, gyroscope, magnetometer
- longitude, latitude, EFEC position
- velNED, heading, speed (2D and 3D)



(a) : Navio2 receive block **(b) :** Navio2 send block

Figure 4.11: Navio2 serial configuration

With the whole message accounting to 1200 bits including the header and terminator, and the baudrate set to 250000 bits/s, one message takes around 4.8 ms to transfer. Although the receive block sample time is set to 5 ms, this is simply to make sure, the block never misses a message, as the receiver waits in case nothing is being transferred. The sending frequency is therefore determined by the sending party, which in this case is the Navio2. The configurable sending frequency is then set to 100Hz.

As for the data that needs to be sent to the Navio2, the sending frequency is decided by a zero order hold connected between the signal and the sending block and is set to 66 Hz. This value could seem low, but the real limitations in this case is the angular speed of the servo motors and a higher frequency would have no real impact on the behaviour of the car. With the steering angles being the most sensitive data to be sent or received, an algorithm will be implemented, to ensure no unintended steering angles will actually be applied. A four character header and a terminator are set to surround the actual message. On the receiving end, the following loop will run indefinitely.

```
Result: PWMs set to correct values
while True do
    head = receive 1 byte;
    if head == "!" then
        tail = receive 3 bytes;
        if tail == "###" then
            PWMs = receive 16 bytes;
            terminator = receive 1 byte;
            if terminator != "\n" then
                data corrupted, set previous PWMs;
            else
                continue;
            end
            if PWMs within boundaries then
                apply received PWMs;
            else
                unacceptable PWM values, set previous PWMs;
            end
        else
            wrong header, wait for terminator and try again;
        end
    else
        continue ;
    end
end
```

Algorithm 1: Steering angle receiving algorithm

4.4.3 Simulink model

By establishing the communication link with both the Navio2 and the NANO a Simulink model can be created. In this model the communication will be able to work in parallel as long as it is kept in a non blocking mode - the absence of data will not pause the whole simulation. Using the Real-time pacer block the simulation time will correspond with the real wall clock time, making the whole model more natural [18]. The current state of the outer layer of the model, which is divided into three sections, is shown in the following figure 4.12.

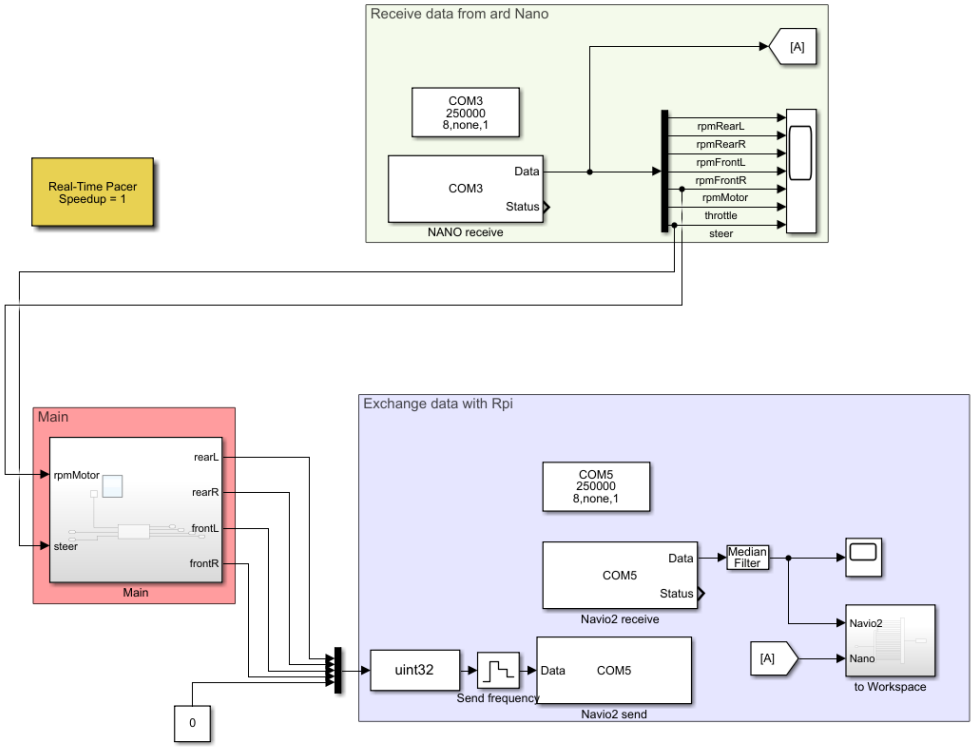


Figure 4.12: The outer layer of the Simulink model

The functions of the communication related blue and green blocks have been explained in previous sections, since these blocks only control and enable data flow and to-workspace logging. The main control algorithms will be inserted inside the main, red subsystem. This subsystem will have n inputs used by the control algorithm and four outputs in the form of steering angles.

Chapter 5

Experiments and data processing

With all the necessary instrumentation and sensors in place, the next step will be to validate and process the sensor data. The term "Ground truth" is used to refer to information provided by direct observation as opposed to information provided by measurement [19]. This information is then used for sensed data verification. Strictly speaking, acquiring such data requires a degree of toleration towards imprecision, since no measuring method is absolutely accurate. The main focus of this chapter will be on sensor data verification and processing. By designing a number of experiments it will be possible to determine the functionality and reliability of the data acquired from the on-board sensors. Followed by that, it will be necessary to acquire a given set of vehicle states required by system control algorithms using sensor fusion.

5.1 Experimental sensor data verification

5.1.1 GPS ground speed

To begin with, the GPS ground speed of the vehicle will be validated. The most intuitive way of verifying the ground velocity information measured by the GPS would be to travel a set distance with constant speed and measure the time it took the vehicle to cover said distance. The first issue with this method would be the fact, that it cannot be ensured, that the vehicle actually drives with a constant speed. And the second issue is, that even if such measurement was made successfully, it would only prove the GPS is capable of measuring constant long-lasting velocities. Another way of testing the performance of the GPS would be to use another known good sensor.

To verify the accuracy of the ground velocity data measured by the GPS, the wheel rotations will be used. Although the RPM values are also one of the measured states of the vehicle, the accuracy of these values can be ensured, due to the simplicity and resolution of the RPM sensors. Naturally minimal longitudinal wheel slip will be able to take place in such scenario, but that can be easily ensured with careful non-aggressive driving. To convert the measured revolutions of the wheels to the velocity of the vehicle, we will

average out the measurements of all four wheels. This is done to account for steering, where the left and right wheel rotations are different due to the axis differentials. Let $d = 0.185$ m be the diameter of the wheels and $avgRPM$ the average rotations per minute of the four wheels in a given moment. The ground velocity v [m/s] of the vehicle in this moment is then calculated as follows:

$$v = \frac{avgRPM \cdot d \cdot \pi}{60} \tag{5.1}$$

The following figure 5.1 shows the comparison of the acquired ground speed from the GPS and the calculated velocity from the wheel rotations.

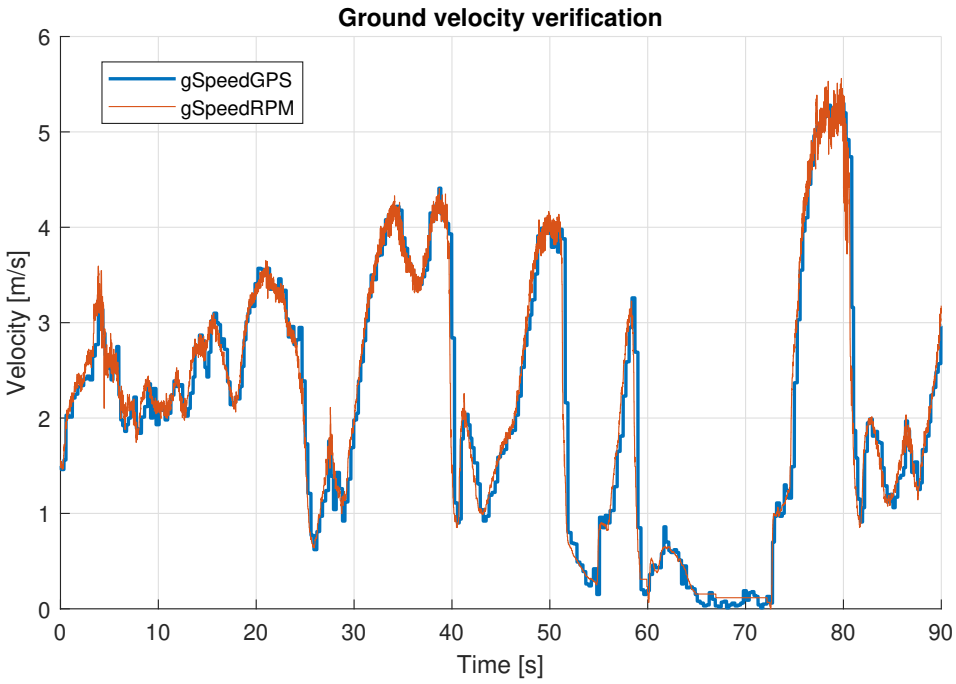


Figure 5.1: GPS ground velocity verification

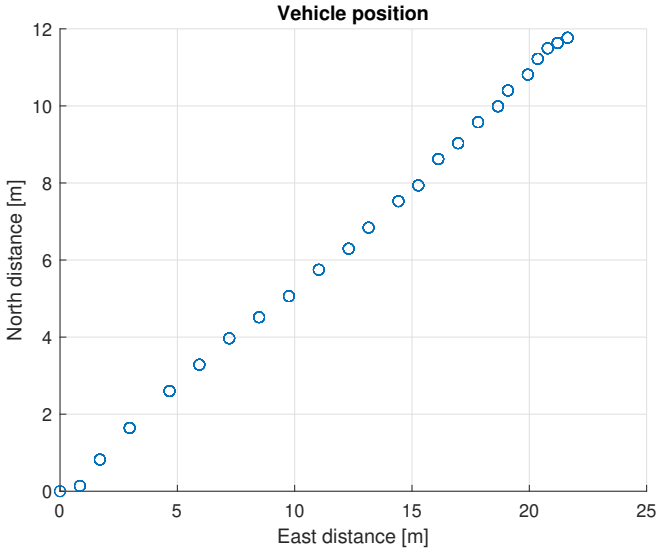
As the figure shows, the measured and calculated velocities follow each other closely, which can be interpreted as a successful verification of the GPS ground speed data. The lower and varying 3-6 Hz receiving frequency of the GPS is noticable compared to the frequent information available by the RPM sensors. On the other hand, compared to the wheel sensors, the GPS is capable of detecting null velocities.

5.1.2 Heading

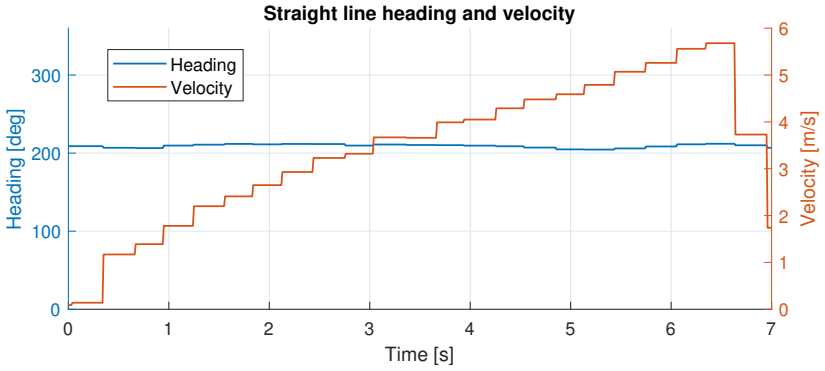
In the following experiment the behaviour of the measured heading will be tested. Heading or true heading as it is sometimes referred to, tells the angle between the direction in which the front of the vehicle is facing and true North. The value lies between 0 and 360 degrees with the orientation

being such that $0° == 360° \implies$ North and $90° \implies$ East. Ideally, if the vehicle were to travel sideways towards west direction and the nose of the vehicle faced north, the heading would remain zero. But as is the case with non-differential GNSS units, the heading is calculated from two positions and thus represents the direction of travel. With this in mind, we can perform certain experiments to determine whether the measured heading behaves as expected. While performing these, the lateral slip will have to be kept as low as possible.

Firstly, the car will travel in a straight line, altering only its speed while maintaining constant heading - within the realm of possibilities. The results are shown in 5.2.



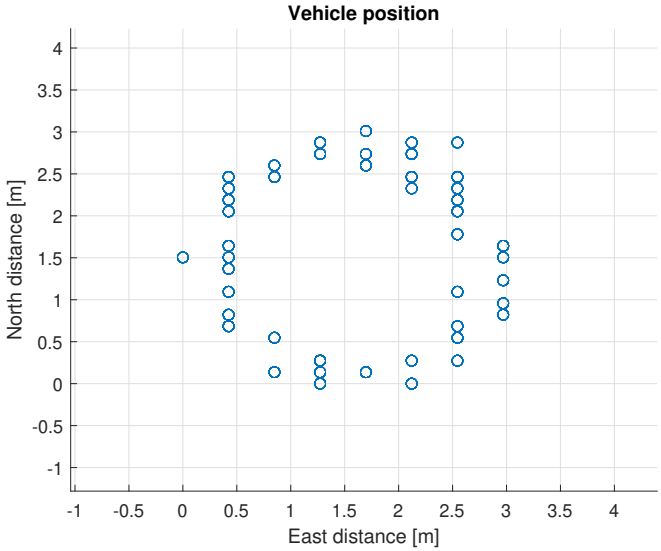
(a) : Vehicle position



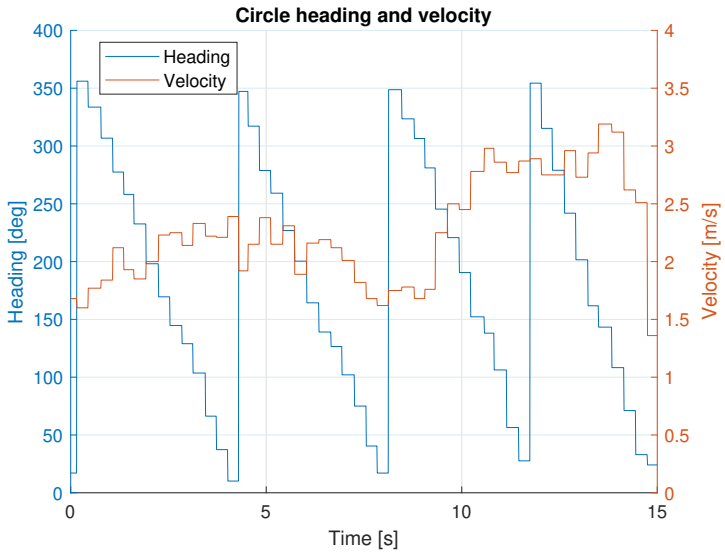
(b) : Vehicle heading and velocity

Figure 5.2: Straight line heading experiment

Secondly, the car will be driven in a circle with constant turning radius and slightly varying ground speed. In this case the heading of the vehicle should change linearly, proportional to the velocity. The results of this scenario can be seen in figure 5.3. As before, the vehicle position was calculated from the longitude and latitude positional data.



(a) : Vehicle position



(b) : Vehicle heading and velocity

Figure 5.3: Circle heading experiment

5.1.3 IMU validation

Accelerometer validation

To determine whether the data from accelerometer, gyroscope and magnetometer behave as expected, it is first important to understand, what exactly are each of these sensors measuring. An accelerometer is a sensor measuring acceleration forces in its rest frame, which is defined as a coordinate system in which the examined body is at rest. These forces may be static - gravity is perceived as an inertial force as for Einsteins equivalency principle. For this reason accelerometers do measure constant acceleration towards the Earth's surface which is equivalent to 1 $g == 9.81\ m/s^2$. Or the measured forces may be dynamic - caused by movement [20]. Typically a three axis accelerometer is used. These are capable of measuring objects orientation due to the influence of gravity in different axis of the sensor, as well as the movement of the unit.

In this section the performance of the accelerometer units on-board of the vehicle will be evaluated. Let's take the linear acceleration data in the x axis of the vehicle as defined by 4.7. The raw data from randomly driving is shown in the following figure 5.4.

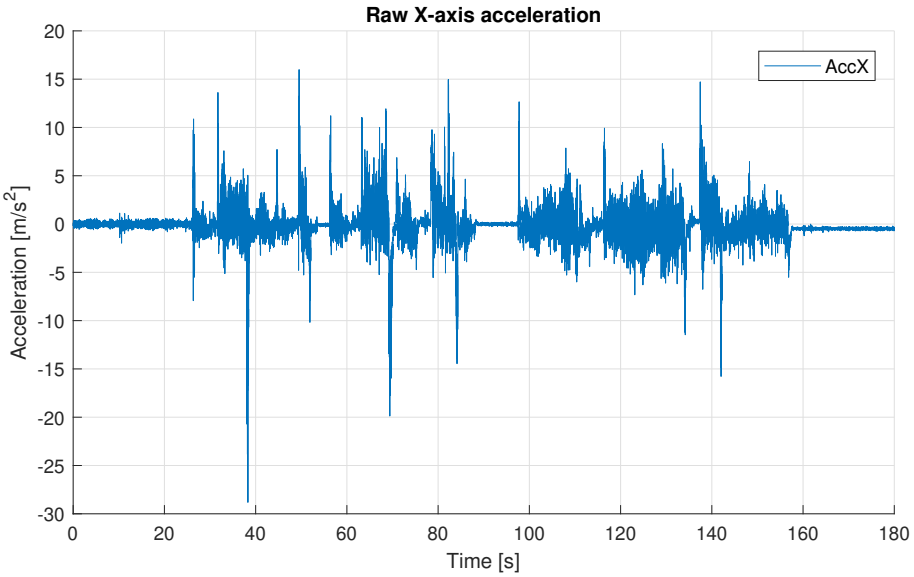


Figure 5.4: Raw acceleration data

This data does not really provide any useful information, at least not without some processing. But to determine whether the acceleration data at least partly reflects true behaviour of the vehicle we can integrate the acceleration to acquire velocity. The comparison between the ground truth velocity and the integrated velocity can be seen in figure 5.5. In short-term, the integrated acceleration actually correlates with the measured ground speed, but by looking at a longer window of data, the integration of the raw

acceleration data makes the velocity data unreliable due to noise and drifting zero value.

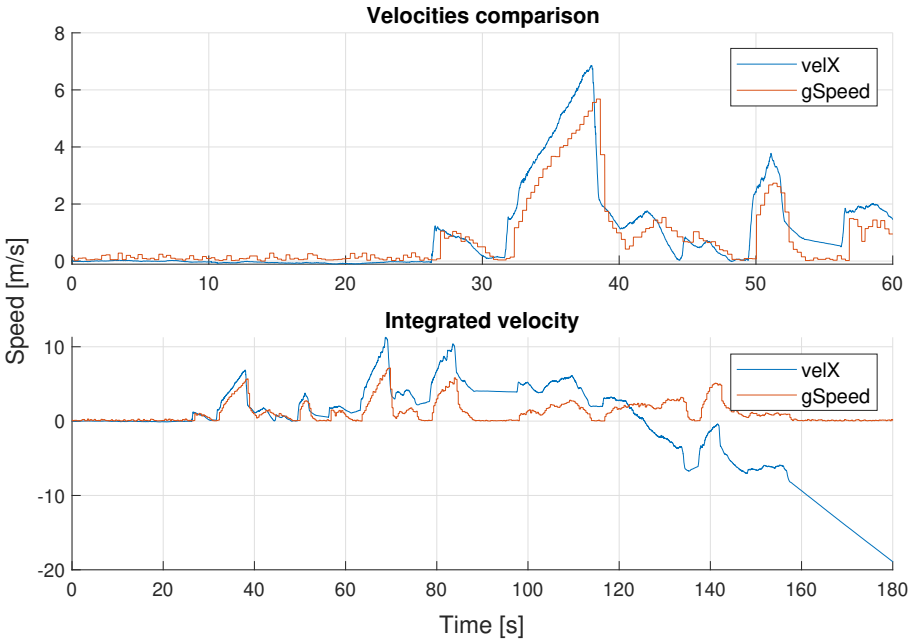


Figure 5.5: Integrated velocity

Gyroscope validation

A gyroscope is a device measuring angular velocity relative to it self, enabling us to potentially measure the roll, pitch and yaw of the vehicle. In this case we will mostly be interested in the the yaw of the vehicle, which corresponds with the z axis of the gyroscope.

By integrating the measured rotation along the z axis and offsetting it to match the initial value of the GPS heading, we can compare the resulting angle with the heading acquired by the GPS. The result of this process is shown in figure 5.6 Although the integrated angles correspond with "ground truth" heading pretty well when it comes to rapid changes, the zero offset error of the gyroscope causes the absolute angle value to drift when integrated.

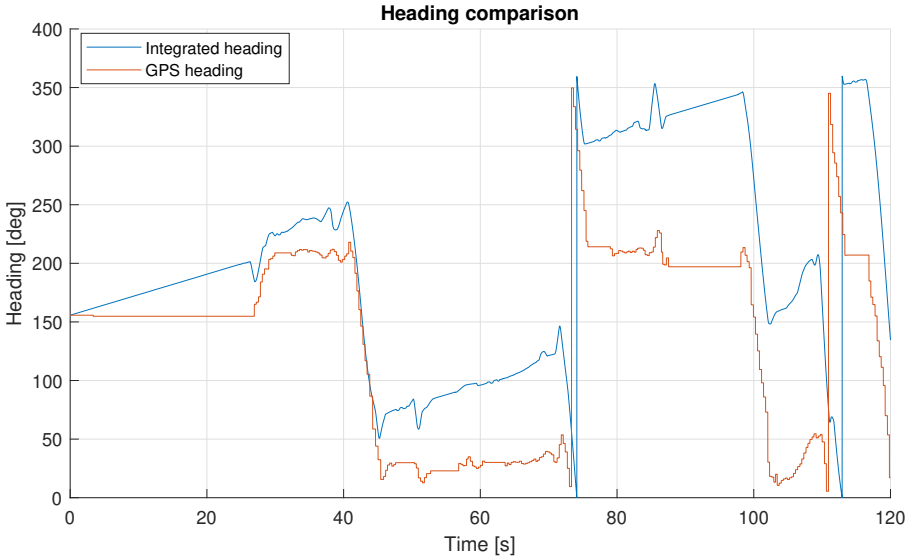


Figure 5.6: Integrated heading

5.2 Sensor data processing

The main motivation for this section is the need to acquire necessary vehicle states to allow for proper dynamical vehicle control. For example in order to calculate the lateral slip angle of a wheel, first its lateral and longitudinal velocities must be calculated. In other example, if we were to control the stability of the vehicle, it would be necessary to have a reliable information about the lateral speed of the vehicle. For these reasons the key states to be obtained from the available sensors are these two velocities - let's define them as V_x and V_y.

5.2.1 VelNED to V_x, V_y

The first step of acquiring an accurate information about the lateral and longitudinal velocities of the vehicle will be to use the available GPS data. As described in 4.3.2 the GPS measurements do not only dispose of the current ground velocity but also of so called *velNED* information. This stands for North, East, Down velocity which represent the speed of the module relative to the true North, East or even downwards direction for aerial navigation. For our purpose only the speeds relative to North and East will be necessary, we will define them as V_N, V_E. By combining these two velocities with the heading and steer angle information, it will be possible to determine the V_x, V_y of the vehicle. This is a kinematic solution, as minimal lateral wheel slip can occur for the vehicle heading information to be accurate. In order to have fully reliable information about the direction in which the vehicle is facing a differential GNSS system would be necessary.

Let the North and East direction determine one Cartesian coordinate system. This system will be viewed as static for an observer. Meaning, that the position and orientation of the vehicle will not change the orientation of the North, East system from the outsiders perspective. A second X, Y axis coordinate system will be defined in relation to the vehicle. This system will remain constant from the car's point of view but will change its orientation relative to the observer. Both of these systems are three dimensional though and their third Z axis is a shared one. This means, that the relation between these two systems is absolutely defined by a set rotation along this shared axis. This rotation being the combination of heading and steering angles of the vehicle, let it be θ. By using a rotation matrix along the Z axis, it is possible to find the magnitude of a vector from one coordinate system relative to the second coordinate system, as long as the angle between these two systems is known. This principle is shown in the following figure 5.7.

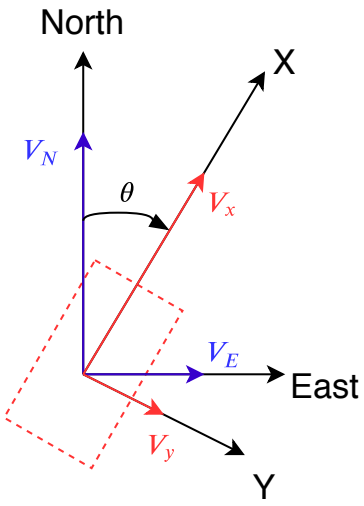


Figure 5.7: Coordinate system conversion

Let R_z be a rotation matrix along the Z axis, for a clockwise heading rotation θ, the vectors V_x, V_y are defined as follows.

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = R_z \begin{bmatrix} V_N \\ V_E \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} V_N \\ V_E \end{bmatrix} \tag{5.2}$$

To evaluate the results of this process an experiment will be conducted. By driving in such a fashion that none or minimal side slip occurs with varying speed, the transformation should ensure that the measured ground speeds corresponds with V_x while the lateral component remains close to zero. The resulting data can be seen in figure 5.8. As expected, the calculated velocity V_x matches the measured GPS ground speed very well. The lateral velocity on the other hand oscillates around zero. This may be caused be the inaccuracies of the V_N, V_E velocities and heading. For example after the 60 second mark where the value of V_y reaches its maximum, even the velocity

in X axis direction does not match the ground speed. This peak was was caused by the car driving in a circle several times.

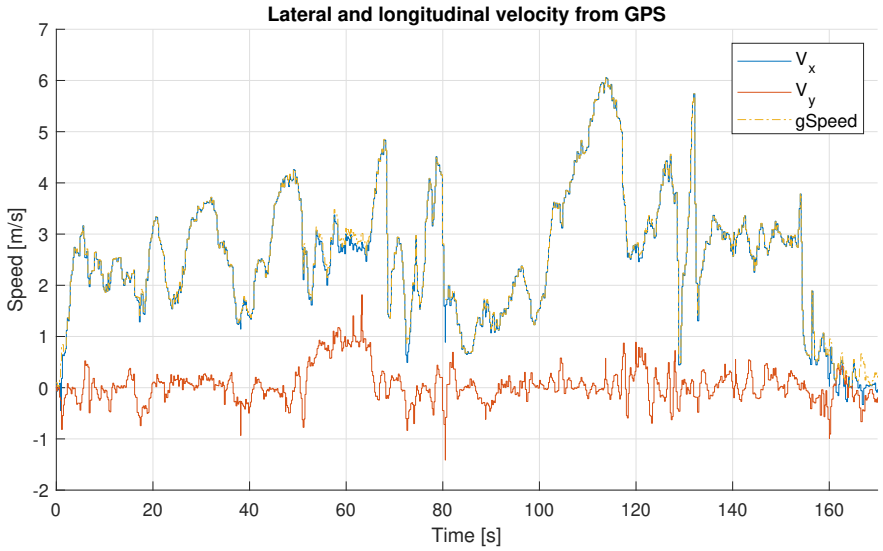


Figure 5.8: V_x, V_y from GPS

5.2.2 Complementary filtering

Although the acquired velocities V_x, V_y from the GPS can represent the actual state of the vehicle their main issue comes with the receiving frequency of the GPS module. This frequency is not only low, but also non-consistent. For this reason a method will have to be derived to acquire a smoother and faster responding velocity information.

Complementary filters are widely used for unification of two sensors measuring the same quantity or its derivative. The key idea is to combine two sensor signals both of which prioritise in either long term or short term accuracy. In other words, both sensors react optimally to different kind of inputs. As an example, let's take an accelerometer and gyroscope trying to measure the pitch of a vehicle. Were the vehicle to remain still for a prolonged periods of time, the accelerometer alone would get a very good readings due to gravity. But once the vehicle begins moving and vibrating, the accelerometer readings will become much less accurate. Gyroscopes on the other hand excel at determining the current rate of rotation, but in a still position they tend to have a zero offset - this was showcased in 5.6 [21].

The complementary filter itself takes the slow moving signal and passes it though a low-pass filter. The integrated high speed signal is then fed through a high-pass filter and both of the resulting signals are added. The requirement being, that the frequency response of both of these filters adds up to one at all frequencies [22].

Complementary V_x

In this case, the complementary filter will be used to enhance the information about the lateral and longitudinal velocities of the vehicle, which was acquired in the previous section 5.2.1. By taking the already acquired $V_{x,gps}$ signal and combining it with the acceleration in the X axis, the complementary filter can be set up. The working principle of the filter is shown in the next graphic 5.9.

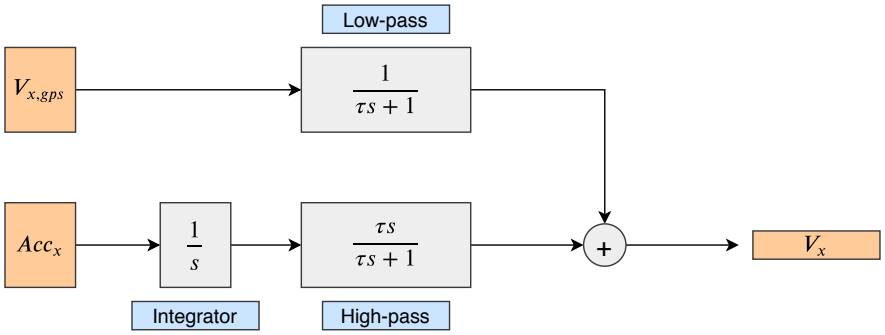


Figure 5.9: Complementary V_x principle

The cut off frequency of the system is defined as $\frac{1}{\tau}$. For the low-pass filter, any components of a signal with a frequency higher than $\frac{1}{\tau}$ will be suppressed, while the gain of the lower frequencies will be one. The logic of the high-pass is reversed. In this case the cut off frequency will be defined by the receiving frequency of the GPS, therefore $\frac{1}{\tau} = 6\ s \implies \tau \approx 0.17\ s$.

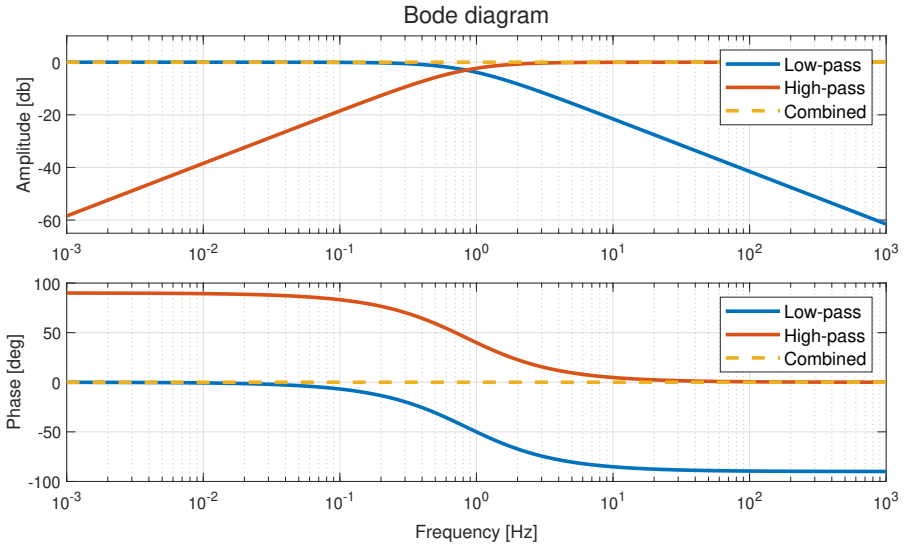


Figure 5.10: Bode diagram of the complementary filter

The Bode diagram showcasing the frequency response of the individual filters and the whole system, where $\tau = 0.17$, is shown in the figure 5.10. It is obvious that the condition of the combined gain being equal to one at all frequencies is met. The following plot 5.11 will compare the low frequency $V_{x,gps}$ data with the filtered velocity. Although the filtration causes a slight delay to the signal, the resulting smoother velocity signal is better for algorithmic control.

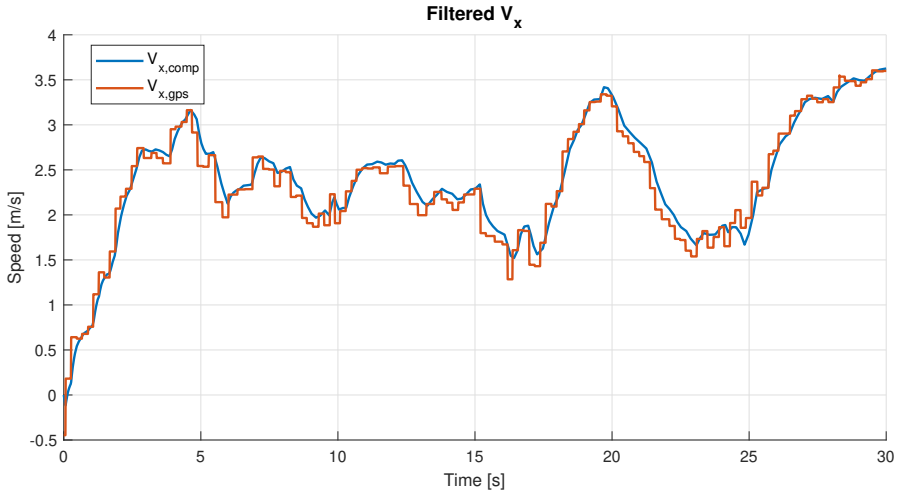


Figure 5.11: Complementary V_x

Complementary V_y

The process of complementing the known lateral velocity $V_{y,gps}$ with the accelerometer data will require one more step than the previous process of filtering the longitudinal velocity. As the accelerometer in the Y axis measures all acceleration forces impacting the accelerometer, it is necessary to distinguish between the desired lateral acceleration and the centrifugal acceleration. As an example, were the car to drive in a circle without actually slipping, the accelerometer would still measure acceleration forces working in the lateral direction, but these forces would not result in the vehicle to actually change its lateral velocity. This is called a centrifugal force and it will be necessary to subtract it from the measured data to correctly derive the final V_y [24].

Let us define F_c as centrifugal force, r as radius, v as velocity, ω as angular velocity and m as mass. The meaning of these quantities is shown in the following diagram 5.12.

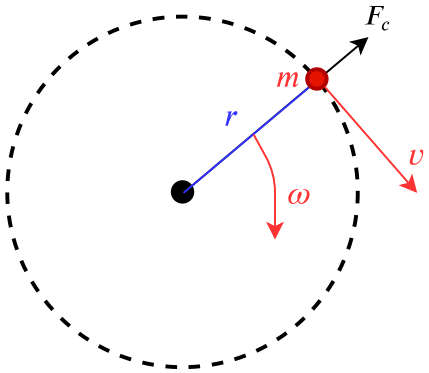


Figure 5.12: Centrifugal force

To derive the centrifugal acceleration a_c the following formulas are going to be used:

$$F_c = \frac{mv^2}{r} \tag{5.3}$$

$$v = \omega r \tag{5.4}$$

By substituting 5.4 into 5.3 for diameter we get:

$$F_c = \omega m v \tag{5.5}$$

And to acquire the desired centrifugal acceleration a_c we will use Newton's second motion law, and thus:

$$a_c = \omega v \tag{5.6}$$

In our case the angular velocity ω is directly measured by the gyroscope in the z axis and the forward velocity v is measured and filtered as per previous section.

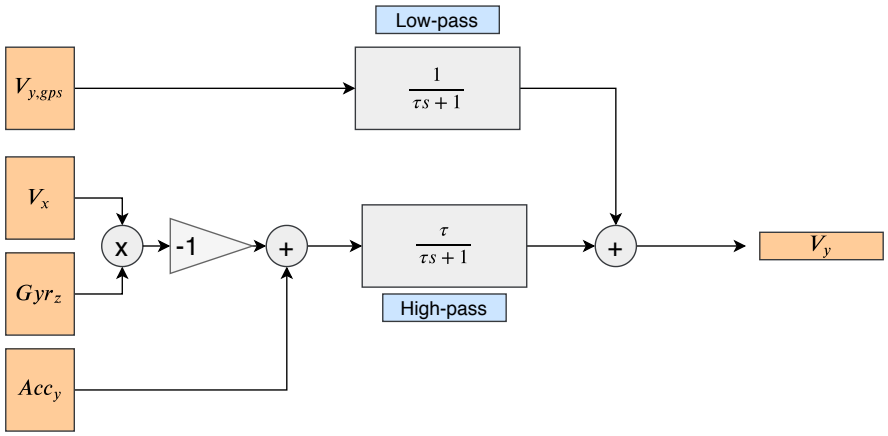


Figure 5.13: Complementary V_y principle

The process of the complementary filtering of the lateral velocity is visualized in figure 5.13. The effect of the complementary filter as well as the subtraction of the centrifugal acceleration can be observed in the following figure 5.14. To compare the effect of the subtraction, the plot consists of the values which did take the centrifugal acceleration into account - V_{y_1}, as well as the values that did not - V_{y_2}.

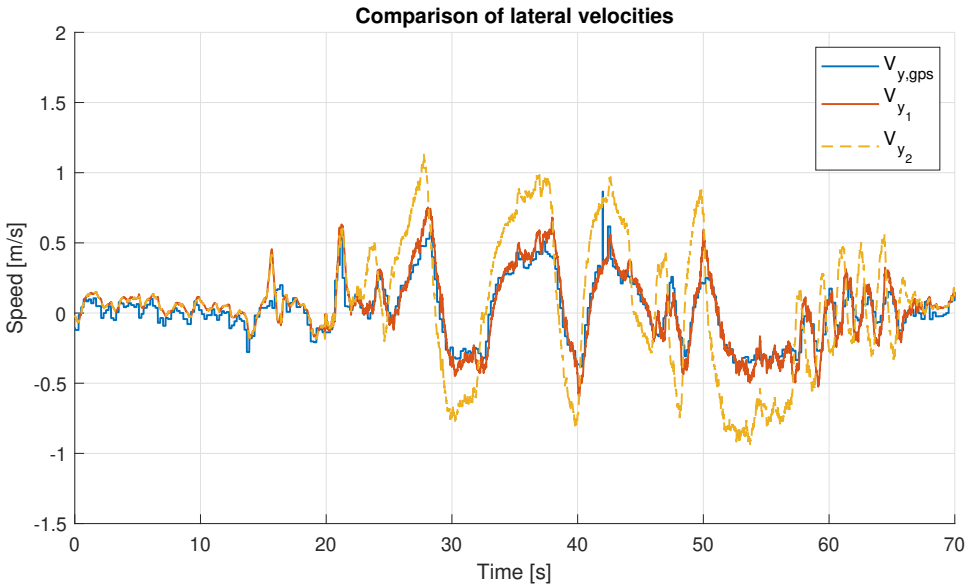


Figure 5.14: Complementary V_y comparison

As we can see the final filtered lateral velocity V_{y_1} follows the GPS velocity pretty well whilst providing smoother information with a much higher frequency. The effect of the centrifugal acceleration is also obvious, especially in higher speed turns where the V_{y_2} overshoots the actual lateral velocity.

To validate the accuracy of the acquired lateral and longitudinal velocities a high speed aggressive drive test will be performed. By comparing the GPS ground velocity information with the sum of the V_x, V_y vectors, we will be able to determine the accuracy. This comparison is shown in figure 5.15. It can be said, that the sum of these vectors indeed does correspond with the measured ground speed very well.

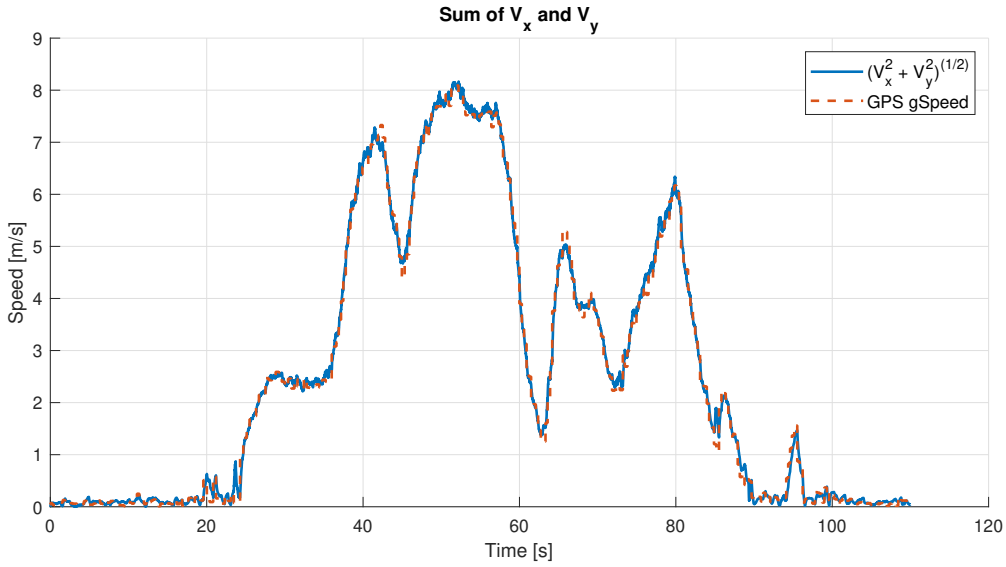


Figure 5.15: Sum of the acquired velocities compared to measured ground speed

Chapter 6

Vehicle control

Now that all the necessary vehicle states for automatic control are known, this chapter will focus on implementation of two simple control algorithms to verify the properties and functionality of the platform. To begin with, a simple rear axle steering control algorithm complementing the user steering angle input, will be implemented. The second control system will focus on the kinematic control of yaw rate and lateral velocity of the vehicle, with the prerequisite of null wheel slip angle.

6.1 Ground speed feed-forward

This simple control system will be inspired be the systems described in the introduction of this thesis 2.1. The key point being, that the rear axle of the vehicle steers in counter movement relative to the front axle in lower speeds to increase maneuverability. Above certain speed the rear axle will start complementing the front axle and in turn increasing the stability of the vehicle. To make the rear axle control smoother, the ground speed information will be used to determine the exact steer angle of the rear axle linearly. Furthermore two adjustable constants will be necessary. "Critical speed" V_c, marking the flex point where the rear axle stops counter steering and begins with the aligned steering. And a reduction factor $R_f \in [0, 1]$ which reduces the final rear steer angle.

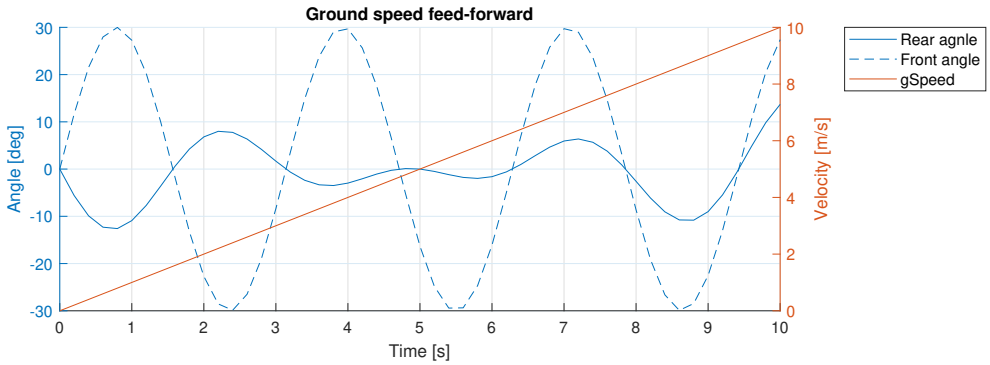


Figure 6.1: Ground speed feed-forward simulation

The figure 6.1 shows the inputs and output of the designed system for $V_c = 5\ m/s$, $R_f = \frac{1}{2}$ and linearly rising vehicle speed. It is shown, that for velocities lower than V_c the rear axle is steering in the opposite direction, but the effect is reduced as the velocity approaches V_c. As the vehicle speed rises, the rear axle steering angle starts complementing the front axle more distinctively.

The following figure 6.2 shows the effect of the control system with the same constants as mentioned before. It is clear, that the rear axle steering angles behave as expected. For example in the time window between 2-12 seconds, the front axle angle was constant and as the vehicle speed approached $5m/s$, the rear axle went from a counter steer to zero degrees. For velocities above $5m/s$, the rear axle started turning in the same direction as the front one.

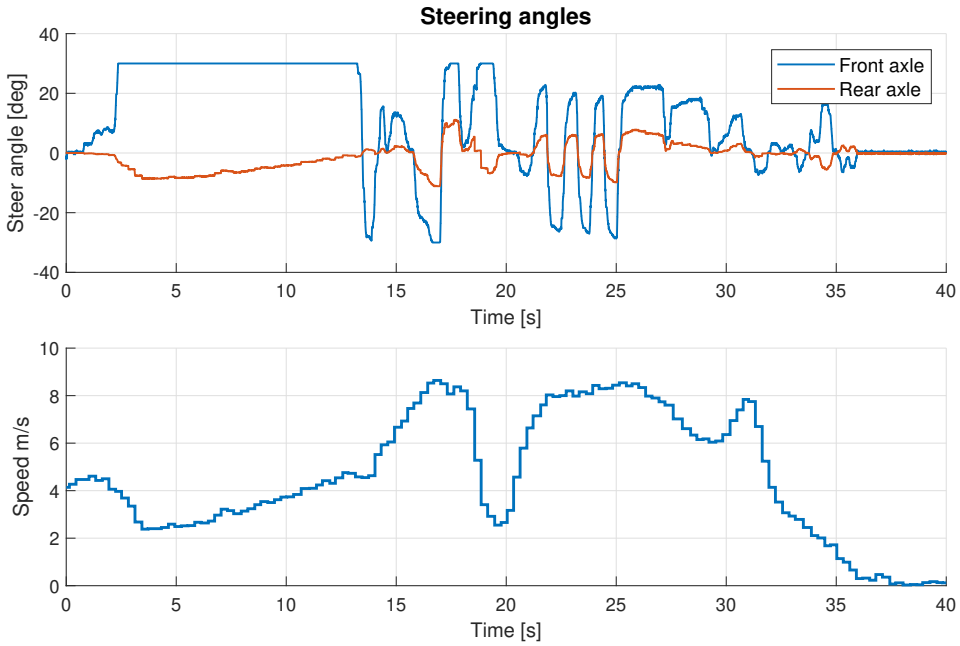


Figure 6.2: Ground speed feed-forward vehicle data for $V_c = 5\ m/s$, $R_f = \frac{1}{2}$

6.2 Yaw rate and lateral velocity control

The second control system will be designed to control the lateral velocity and yaw rate of the vehicle using kinematic control. That means only using geometrical analysis to determine the desired kinematics outputs - wheel angles in this case, without taking any external forces and torques into consideration. Therefore, the resulting control algorithm should be able to control the system, as long as these non considered quantities do not disturb the system too much for the control system to break down.

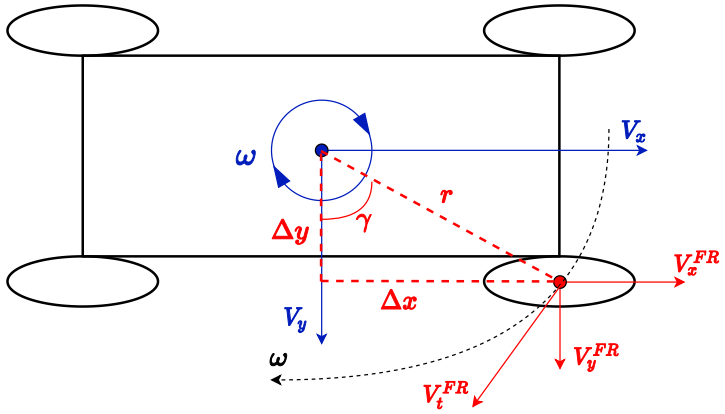


Figure 6.3: Vehicle states used for vehicle control: ω = yaw rate, V_x, V_y = longitudinal and lateral velocities measured, V_x^{FR}, V_y^{FR} = longitudinal and lateral velocities of the Front Right wheel, V_t^{FR} = tangential velocity

Let us define X^{ij}, $i \in \{Front, Rear\}$, $j \in \{Right, Left\}$ as a quantity described in the coordinate system of a given wheel. The following derivation will only be described for the front right wheel as the resulting equations will be universal for all four wheels.

To control the yaw rate and lateral velocity of the vehicle, firstly a geometrical description will have to be made. The figure 6.3 shows all the necessary quantities that will be needed to describe the system. The longitudinal velocity of the vehicle V_x is acquired as per chapter 5.2. The intervals $\Delta x, \Delta y$ are constant offsets between the IMU unit and the center of the front right wheel. With the system inputs being the yaw rate of the vehicle ω and the lateral velocity V_y, all information necessary to calculate the appropriate steer angle δ^{FR}, should be available.

To begin with, let us calculate the lateral and longitudinal velocities of the wheel V_x^{FR}, V_y^{FR} for a given ω and V_y. The hypotenuse r is equal to:

$$r = \sqrt{\Delta x^2 + \Delta y^2} \tag{6.1}$$

The angle γ thus equals:

$$\gamma = \arcsin\left(\frac{\Delta x}{r}\right) = \arccos\left(\frac{\Delta y}{r}\right) \tag{6.2}$$

The tangential velocity V_t^{FR} is defined as:

$$V_t^{FR} = \omega r \tag{6.3}$$

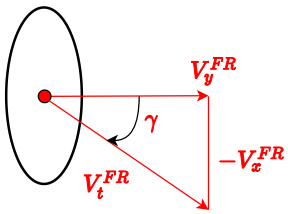


Figure 6.4: Wheel triangle similarity

From the triangle similarity shown in figure 6.4 it is obvious that:

$$\begin{aligned} -V_x^{FR} &= \sin(\gamma)\omega r \\ V_y^{FR} &= \cos(\gamma)\omega r \end{aligned} \tag{6.4}$$

By substituting the equation 6.2 into 6.4:

$$\begin{aligned} -V_x^{FR} &= \omega \Delta y \\ V_y^{FR} &= \omega \Delta x \end{aligned} \tag{6.5}$$

And thus the overall velocities in the centre of the wheel are calculated as follows:

$$\begin{aligned} V_{x,fin}^{FR} &= V_x - V_x^{FR} \\ V_{y,fin}^{FR} &= V_y + V_y^{FR} \end{aligned} \tag{6.6}$$

Finally to calculate the desired steer angle δ^{FR}:

$$\delta^{FR} = \arctan \frac{V_{y,fin}^{FR}}{V_{x,fin}^{FR}} \tag{6.7}$$

6.2.1 Yaw rate and lateral velocity control testing

With the derived relation between the yaw rate and lateral velocity of the vehicle and the steering angles, a feed-forward system can be designed. By measuring the exact offsets $\Delta x, \Delta y$ for each wheel, all four steering angles can be calculated. To showcase the functionality of the control system, two scenarios will be presented.

Firstly, the lateral velocity input will be set to zero and only the yaw rate of the vehicle will be controlled. The following figure 6.6 shows both the requested yaw rate of the vehicle and the user input. The measured yaw rate actually corresponds with the requested yaw rate pretty well, but an offset in the positive direction can be seen. This may be caused by mechanical imperfections as well as by measurement inaccuracies. The averaged out front and rear axle steer angles also do behave as expected, with the rear axle heavily counter-steering.

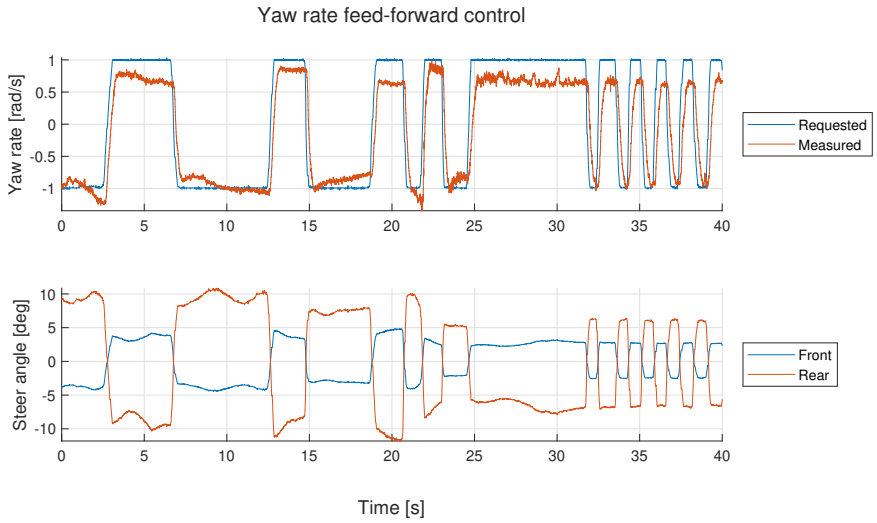


Figure 6.5: Yaw rate kinematic feed-forward control

Secondly, only the lateral velocity will be controlled, while the yaw rate is set to zero. The measured lateral velocity is actually responding to the desired V_y really well. In this case, all four wheels are to have the same steering angle, as the yaw rate of the vehicle is set to zero.

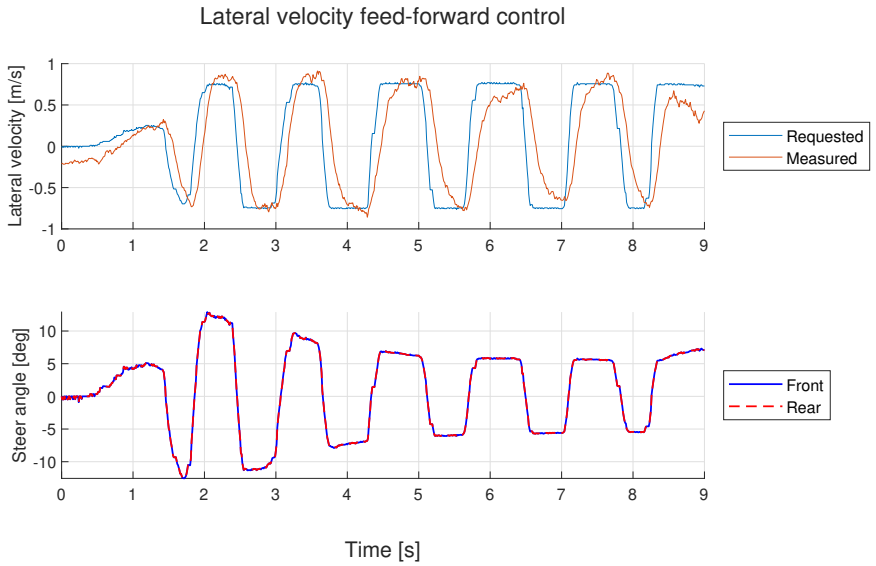


Figure 6.6: Lateral velocity kinematic feed-forward control

Chapter 7

Conclusion

I believe, that drive by wire systems with the combination of independently steered wheels are the next logical step in vehicle development. The main challenge lies in developing reliable and robust control algorithms suitable for such vehicles. The main focus of this thesis was on the development of an overactuated platform, which will be used for deployment and verification of said algorithms.

To begin with, a 1:5 vehicle platform was modified to allow for four wheel steering. This was followed by the instrumentation of the platform, including sensor deployment, actuator control and vehicle data logging. With the hardware part of the thesis finished, the next step was the validation of the functionality and precision of deployed instrumentation. This section includes experimental sensor data validation as well as an implementation of sensor fusing filters to acquire precise vehicle states. Here the challenge of a single antenna GPS module was faced. Without the accurate body heading information commonly available with dual antenna devices a kinematic solution to calculate the body heading of the vehicle had to be used. This in return decreases the accuracy of vehicle states measurements in more dynamic situations.

Lastly, two simple yet telling feed-forward kinematic control algorithms were developed and tested. These were used to showcase the overall performance of the platform and validate its capabilities. Overall, the goal of developing an easy to use rapid development platform suitable for drive by wire algorithms validation, was achieved.

Bibliography

[1] J. Edrén, "Motion modelling and control strategies of over-actuated vehicles", 2014, pp. 1-8

[2] Ordnance Survey, "A guide to coordinate systems in Great Britain", 2018, pp. 13, available at: https://www.ordnancesurvey.co.uk/documents/resources/guide-coordinate-systems-great-britain.pdf

[3] A diagram showing ECEF, ENU, Longitude and Latitude coordinates and the relationship between them, 2010, available at: https://cs.wikipedia.org/wiki/Soubor:ECEF_ENU_Longitude_Latitude_relationships.svg

[4] M. G. E. Schneiders, M. J. G. van de Molengraft and M. Steinbuch, "Benefits of over-actuation in motion systems," Proceedings of the 2004 American Control Conference, Boston, MA, USA, 2004, pp. 505-510 vol.1.

[5] Parry, Tommy. "Porsche's Rear Wheel Steering - How It Works and Operates." FLATSIXES, 29 Apr. 2018, available at: https://flatsixes.com/porsche-culture/porsche-factoids/rear-wheel-steering-history-operation/

[6] Allan Y. Lee, "Performance of Four-Wheel-Steering Vehicles in Lane Change Maneuvers", California Institute of Technology, pp. 161-173

[7] Birch Stuart, "Protean Electric aims for pirouetting autonomous EVs", SAE International ®, 12 Sept. 2019, available at: www.sae.org/news/2019/09/protean-electric-360-degree-steering

[8] Grabianowski Ed, "How the Jeep Hurricane Works" HowStuffWorks, 5 Feb. 2005, available at: https://auto.howstuffworks.com/jeep-hurricane.htm

[9] Losi: Desert Buggy XL-E homepage, available at: http://www.losi.com/Products/ProductGallery.aspx?ProdID=LOS05012T1

[10] Losi , "S900S 1/5 Scale Steering Servo w/Metal Gears5IVE-T", Product specification, available at: http://www.losi.com/Products/Features.aspx?ProdID=LOSB0884

[11] "PLA overview", SIMPLIFY3D, available at: https://www.simplify3d.com/support/materials-guide/pla/

[12] "Navio2 overview", ARDUPILOT, 2019, available at: https://ardupilot.org/copter/docs/common-navio2-overview.html

[13] xkam1x, Arduino-PWM-Reader, June 2018, https://github.com/xkam1x/Arduino-PWM-Reader

[14] "Hall Effect Principle - History, Theory Explanation, Mathematical Expressions and Applications." Electricalfundablog.com, 1 Apr. 2018, available at: https://electricalfundablog.com/hall-effect-principle-history-theory-explanation-mathematical-expressions-applications/

[15] EMLID, Navio2 documentation, available at: https://docs.emlid.com/navio2/

[16] Ublox, "u-blox M8 concurrent GNSS modules", Datasheet, March 25th 2020, pp. 6, available at: https://www.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_%28UBX-15031086%29.pdf

[17] Ublox, "Ublox-6 receiver description including protocol specification", April 18th 2013, Manual, available at: https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf?utm_source=en%2Fimages%2Fdownloads%2F\Product_Docs%2Fu-blox6_ReceiverDescriptionProtocolSpec_%28GPS.G6-SW-10018%29.pdf

[18] Gautam Vallabha (2020). Real-Time Pacer for Simulink, MATLAB Central File Exchange. Retrieved April 22, 2020, available at: https://www.mathworks.com/matlabcentral/fileexchange/29107-real-time-pacer-for-simulink

[19] "What Is Ground Truth? - Definition from Techopedia" ,Techopedia, available at: www.techopedia.com/definition/32514/ground-truth

[20] W3C Working Group Note, "Motion Sensors Explainer.", 30 Aug. 2017, available at: www.w3.org/TR/motion-sensors/#intro

[21] "Two Sensors One Filter", Equals zero, December 9th, 2010, available at: www.etotheipiplusone.net/?p=1081

[22] "Complementary Filter Design." Gait Analysis Made Simple, 14 July 2016, https://gunjanpatel.wordpress.com/2016/07/07/complementary-filter-design/

[23] Gupta, Lipi. "How to Calculate Centrifugal Force", Sciencing, 22 July 2019, available at: https://sciencing.com/calculate-centrifugal-force-5130895.html

[24] Pierre Pettersson, "Estimation of Vehicle Lateral Velocity", Lund University, November 2018, pp. 8-9, Available at: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8847658&fileOId=8859347

Appendix A

U-blox protocol messages specification

Parameter	Specification					
Receiver type	72-channel u-blox M8 engine GPS L1C/A, SBAS L1C/A, QZSS L1C/A, QZSS L1 SAIF, GLONASS L1OF, BeiDou B1I, Galileo E1B/C					
Accuracy of time pulse signal	RMS 99%	30 ns 60 ns				
Frequency of time pulse signal		0.25 Hz...10 MHz (configurable)				
Operational limits [1]	Dynamics	≤ 4 g				
	Altitude	50,000 m				
	Velocity	500 m/s				
Velocity accuracy [2]		0.05 m/s				
Heading accuracy [2]		0.3 degrees				
GNSS		**GPS & GLONASS**	**GPS**	**GLONASS**	**BeiDou**	**Galileo**
Horizontal position accuracy [3]		2.5 m	2.5 m	4 m	3 m	TBC[4]
NEO-M8N/Q						
Max navigation update rate	NEO-M8N	5 Hz	10 Hz	10 Hz	10 Hz	10 Hz
	NEO-M8Q	10 Hz	18 Hz	18 Hz	18 Hz	18 Hz
Time-To-First-Fix [5]	Cold start	26 s	29 s	30 s	34 s	45 s
	Hot start	1 s	1 s	1 s	1 s	1 s
	Aided starts [6]	2 s	2 s	2 s	3 s	7 s
Sensitivity[7]	Tracking & Navigation	–167 dBm	–166 dBm	-166 dBm	-160 dBm	-159 dBm
	Reacquisition	–160 dBm	–160 dBm	-156 dBm	-157 dBm	-153 dBm
	Cold start	–148 dBm	–148 dBm	-145 dBm	-143 dBm	-138 dBm
	Hot start	–157 dBm	–157 dBm	-156 dBm	-155 dBm	-151 dBm

Table A.1: Performance specifications of the NEO-M8N module, adpoted from NEO-M8 datasheet [16]

	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x01 0x02	28	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS Millisecond Time of Week
4	I4	1e-7	lon	deg	Longitude
8	I4	1e-7	lat	deg	Latitude
12	I4	-	height	mm	Height above Ellipsoid
16	I4	-	hMSL	mm	Height above mean sea level
20	U4	-	hAcc	mm	Horizontal Accuracy Estimate
24	U4	-	vAcc	mm	Vertical Accuracy Estimate

Table A.2: Geodetic Position Solution message, adpoted from Ublox datasheet protocol [17]

Message	**NAV-VELNED**				
Description	**Velocity Solution in NED**				
Firmware	Supported on u-blox 6 from firmware version 6.00 up to version 7.03.				
Type	Periodic/Polled				
Comment	**See important comments concerning validity of velocity given in section Navigation Output Filters.** -				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x01 0x12	36	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS Millisecond Time of Week
4	I4	-	velN	cm/s	NED north velocity
8	I4	-	velE	cm/s	NED east velocity
12	I4	-	velD	cm/s	NED down velocity
16	U4	-	speed	cm/s	Speed (3-D)
20	U4	-	gSpeed	cm/s	Ground Speed (2-D)
24	I4	1e-5	heading	deg	Heading of motion 2-D
28	U4	-	sAcc	cm/s	Speed Accuracy Estimate
32	U4	1e-5	cAcc	deg	Course / Heading Accuracy Estimate

Table A.3: Velocity solution message, adpoted from Ublox protocol datasheet [17]

	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x01 0x06	52	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS Millisecond Time of Week
4	I4	-	fTOW	ns	Fractional Nanoseconds remainder of rounded ms above, range -500000 .. 500000
8	I2	-	week	-	GPS week (GPS time)
10	U1	-	gpsFix	-	GPSfix Type, range 0..5 0x00 = No Fix 0x01 = Dead Reckoning only 0x02 = 2D-Fix 0x03 = 3D-Fix 0x04 = GPS + dead reckoning combined 0x05 = Time only fix 0x06..0xff: reserved
11	X1	-	flags	-	Fix Status Flags (see graphic below)
12	I4	-	ecefX	cm	ECEF X coordinate
16	I4	-	ecefY	cm	ECEF Y coordinate
20	I4	-	ecefZ	cm	ECEF Z coordinate
24	U4	-	pAcc	cm	3D Position Accuracy Estimate
28	I4	-	ecefVX	cm/s	ECEF X velocity
32	I4	-	ecefVY	cm/s	ECEF Y velocity
36	I4	-	ecefVZ	cm/s	ECEF Z velocity
40	U4	-	sAcc	cm/s	Speed Accuracy Estimate
44	U2	0.01	pDOP	-	Position DOP
46	U1	-	reserved1	-	Reserved
47	U1	-	numSV	-	Number of SVs used in Nav Solution
48	U4	-	reserved2	-	Reserved

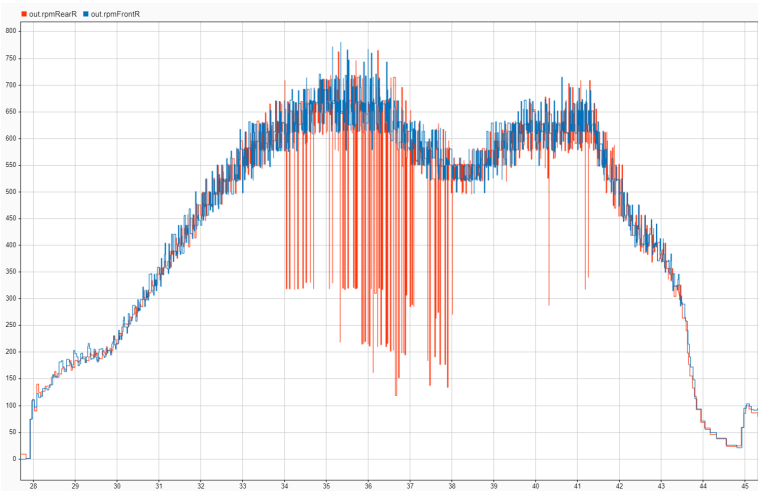
Table A.4: Navigation Solution Information message, adpoted from Ublox protocol datasheet [17]

Appendix B

Platform user manual

B.1 RPM sensors calibration

In case one or more of the wheel RPM sensors display behaviour similar to the red one shown in the following picture, the border values for detecting peaks from the HALL sensor need to be adjusted / calibrated.



This is caused by unwanted spacial shift of the HALL sensor and with the currently set values the sensor may stop recognizing peaks at higher RPM. To fix this issue, the two underlined values within the NANO source code **nano2nuc.ino** have to be modified. Firstly lift the wheels from the ground, secondly comment all prints within the code except the "*Serial.println(valFrontX)*". Follow by uploading the code, starting the plot monitor (with correct baud rate) within the Arduino IDE and spin the wheels. Now the peaks from HALL sensor are visible and the peak detection values can be adjusted. Note that the first value must be higher than the second.

```
94    //frontR RPM
95    valFrontR = analogRead(frontRpin);
96    //Serial.println(valFrontR);  // debug value
97    if(valFrontR > 530 && risedFrontR == false){
98      risedFrontR = true;
99      elapsed_time_FrontR = micros() - start_time_FrontR;
100     start_time_FrontR = micros();
101     rpmFrontR = ((1000000*60)/elapsed_time_FrontR)/10;
102     //Serial.println(rpmFrontR);  // debug value
103   }
104   else if(valFrontR < 510 && risedFrontR == true){
105     risedFrontR = false;
106   }
```

B.2 Raspberry Pi / Navio2

General information

The raspberry requires a login and password when powered up, these are:

- login: **pi**
- pswd: **raspberry**

When connecting to the rpi via ssh (*ssh pi@*IP**), the same password will be required.
When in root directory, two folders are visible. *Navio2* is the API with all directories and sample codes necessary. The folder *Raspberry* is the main working folder which includes the main control script *rpi_main.py*.
The unit is set to run the main script automatically upon reboot, this can be modified via the **sudo crontab -e** command called from root.
The navio2 LED is used to indicate the state of the program. Upon starting up, the default colour should be blue (cyan), once the Navio2 starts receiving valid PWM values, the LED turns green, if invalid values are received, the LED turns red.

GPS

If the GPS seems to be not working - only receives null values or the received values are extremely inaccurate, first make sure the GPS is used in an open space with clear view of the sky. Secondly try restarting the Navio2 unit and wait for several minutes (good indicator of a working GPS is when the ground speed value oscillates around 10 cm/s) at a frequency around 4 Hz. Sometimes the unit needs to be cold-started, which means it has to remain stationary with clear view of the sky for up to twenty minutes.

B.3 Intel NUC

COM ports

The brain of the car is the *ard_nuc_rpi_comm.slx* Simulink model. One communication channel is established with the Arduino, typically at COM3. The Navio2 is set to COM5 by default, but in case the serial to USB converter is not visible at neither COM4/5, open device manager and COM ports, most likely this message will be visible "PL2303HXA PHASED OUT SINCE 2012. PLEASE CONTACT YOU SUPPLIER". The converter needs a specific driver to work. Therefore right click the message, update driver, browse my computer, let me pick and now select the installed 2008 version. The converter should work now. Note, that Matlab has to be reset in order to notice the updated drivers.

Wheel angles calibration

If the car is turning even with centered steering input, use the steer trimmer of the remote controller.
If an angle of an individual wheel is set to 0 degrees but is offset, an offset constant has to be changed. This should only occur if the servo arm was unattached and attached at a different angle. The constants to change the offsets of individual wheels can be found in the MAIN section of the model in a sub-model called *Angles to PWMs & angles logging* - by changing these constants we simply shift the PWM center value for the servo motor.

LAN connectivity

For easier testing a LAN remote desktop connection can be established. For this **TeamViewer** (TV) software is used. The following method works upon booting up the NUC, the windows password can then be entered via TV.

1. Download TV - no account is needed
2. Connect your PC to the NUC using a LAN cable, LAN network is established and the NUC is set to have a static IP
3. On your PC enter said IP - **169.254.113.47**
4. TV will ask for connection password - this is also set statically to **123456**
5. Remote desktop should now be established

Note that the NUC TV should be set to accept LAN connections exclusively.

Appendix C

Attached CD contents

- Code
  - Arduino NANO
    - nano2nuc.ino
    - PWM lib
      - PWM.cpp
      - PWM.hpp
  - Intel NUC
    - ard_nano_nuc_comm.slx
    - Logged data
  - Navio2
    - rpi_nuc_communication.py
- 3D models
  - Obj files
  - FreeCad source
- Photo & Video