# The Development of a Self-Driving E-scooter

## *Utveckling av en självkörande e-scooter*

**Bachelor's Thesis 2023**


*Authors*
Hanna Ekhage
Armin Khorsandi
Oscar Rosman
Agnes Sundström
Franz Weijmer

*Supervisors*
Fredrik Bruzelius
Hadi Hajieghrary

*Examiner*
Jonas Sjöberg



CHALMERS


Dept. of Electrical Engineering
CHALMERS
Sweden, Gothenburg, Johanneberg


June 26, 2023

# Abstract

This report outlines the development of the hardware and software of an e-scooter with the aim of making it self-driving. The project is part of Project Autobike, which aims to improve traffic safety by designing bicycles that can be used in testing of safety systems in autonomous cars. The results show functional hardware and software, and simulations indicate that the e-scooter can balance on flat surfaces if the starting angles of roll and tilt are straight. While the e-scooter has shown promise during tests, further work is needed to achieve full balance. It may not be possible to balance the scooter fully outside of simulations because of its low center of mass.



# Sammandrag

Denna rapport beskriver förbättringarna till hårdvaran och mjukvaran av en e-skoter i syfte att göra den självkörande. Projektet är en del av Project Autobike som jobbar med att förbättra trafiksäkerheten genom att designa cyklar för att testa säkerhetssystem i bilar. Resultaten visar funktionell hårdvara och mjukvara, och simuleringar tyder på att e-skotern kan balansera då skotern kör på platt underlag och har en rak utgångsvinkel i roll och tilt. Även om e-skotern kunde balansera kortare stunder under tester, krävs ytterligare arbete för att uppnå full balans. Det kanske inte är möjligt utanför simulering med nuvarande e-skoter på grund av låg höjd på masscentrum.



***Keywords***— e-scooter, self-driving, testing, simulation

# Contents

# List of Symbols and Abbreviations

## Controller Parameters

| | |
|---|---|
| $\delta$ | Steering angle |
| $\delta_e$ | Effective steering angle |
| $\varphi$ | Roll/lean angle |
| $\beta$ | The body slip angle |
| $\psi$ | The yaw indicating the orientation of the e-scooter |
| $\dot{x}_{\mathrm{L}}$ | A state consisting of X and Y |
| $\theta$ | The direction of the bike |
| $v$ | Speed |
| $O$ | The point of rotation |
| $R$ | The radius of the driven path and the distance to the point of rotation from the CoM |
| $D$ | Inertia with respect to the xz-axes |
| $J$ | The moment of inertia of the scooter body with respect to the x-axis |

## Physical Parameters

| | |
|---|---|
| m | Mass of E-scooter |
| $a$ | Distance between the rear wheel and the frame's center of mass |
| $b$ | Wheelbase. Distance between wheel centers. |
| $c$ | Distance between the front wheel contact point and the extension of the fork axis |
| $\lambda$ | The angle of the fork axis |
| CoM | Center of Mass |
| $\mathrm{h}_{\mathrm{CoM}}$ | Height of center of mass |
| $X, Y$ | The position of the bike in global coordinates |
| $g$ | Gravitational constant |

# 1 Introduction

Autonomous automobiles are becoming more frequent in today's society [1]. To ensure the functionality of safety systems in these cars, numerous tests and simulations have to be done before they are let out in real-world traffic. Real traffic tests are performed with scenarios that accurately match possible situations to assure the safety and protection of the riders and the environment outside the automobile [2], [3]. The usage of e-scooters has increased in recent years, and thus, engineers have to consider e-scooters in the testing and development of safety systems as well [4].

A study conducted in Sweden found that e-scooter riders are drunk drivers to a greater extent than drivers of other vehicles [5]. In the US, between 2017 and 2021, e-scooter related emergency visits went up by almost 450% [6]. Because of this, safety between cars and e-scooters is now held in high regard.

Bicycles are already part of the development of safety features and are being introduced in testing to ensure safe reactions from cars by simply using a bike-shaped doll and pushing it into the car's trajectory [7]. Chalmers University has been developing a self-driving bike that can more accurately emulate bike rider movements, improving the safety system's functionality further [8].

This thesis aims to do the same for e-scooters, providing the automotive industry with self-driving e-scooters for testing car safety systems and ensuring increased road safety in the future.

## 1.1 Background - Project Autobike

Project Autobike is a joint project between Chalmers and Mälardalens University with the goal of developing self-balancing bicycles. The purpose of these bicycles is to simulate human behavior to improve traffic safety tests for cars. The Autobike project has several subprojects, and this project focuses on developing a self-driving e-scooter [9].

The development of the e-scooter started at Mälardalen's University, where an e-scooter and much of the hardware were acquired and put together. Mälardalen also made a simulation program and software to run the e-scooter. They were unable to achieve self-balancing for the scooter prior to sending it to Chalmers, where this bachelor continued the work [9].

In order for the scooter to retain realistic human driving characteristics, no gyro or support wheels are used. Instead, the scooter must rely on leaning and steering, like a human driver would.

## 1.2 Existing E-scooter

The project started with a group of students at Mälardalen's University [9] in the fall of 2022 and stretched over six months. A Xiaomi Mi Electric Scooter Pro 2 [10] was purchased and modified during the project, in accordance to Figure 1. In addition to the requirement of balancing by steering, another requirement was to modify the scooter in a similar way and methodology to the already existing bike projects so that they would be as similar as possible.

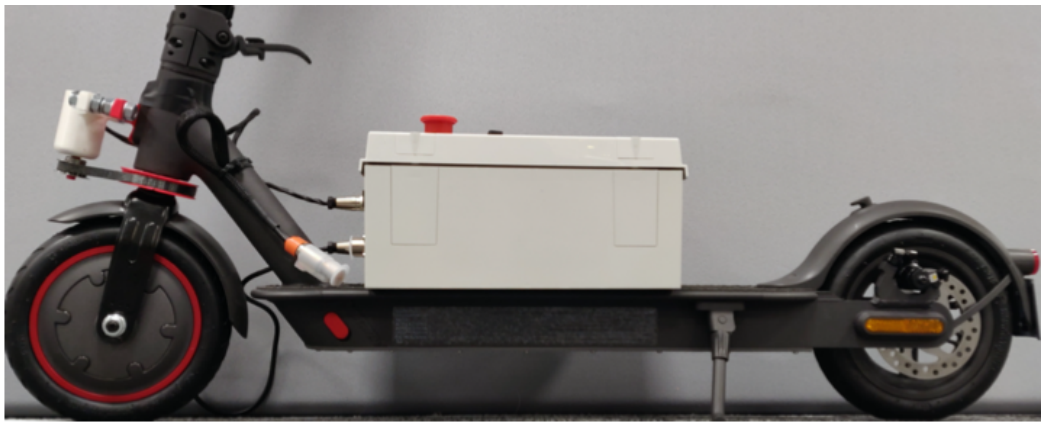

Figure 1: E-scooter after modifications from Mälardalen [9]

### 1.2.1 The Original Modifications Done to the Scooter.

The scooter had its own controller taken out, and the battery is instead connected to a cable coming out of the front fork of the scooter. The original battery **NED-1004-K**, and driving motor **JYX36300** of the scooter remained.

The controller box is the protective housing for all sensors, computer boards, and drivers. It is similar to the bicycle's controller boxes, sharing many of the same components but being different in shape and layout.

The two most integral components of all the controller boxes for the balancing of the vehicles are the **NI MyRIO-1900** and **IMU Pmod NAV 9**.

#### MyRIO-1900 [11]

The real-time embedded evaluation board by National Instruments, myRIO, is the "brain" and an important part of the self-driving e-scooter. It reads the IMU sensor values in real time, which is the sole reason why National Instruments LabVIEW is used for the software of the e-scooter. MyRIO uses the software to command the different motors through the computer boards, internal wiring, and the two drivers.

#### IMU-Pmod NAV 9 [12]

The IMU (Inertial Measurement Unit) is the main sensor used, providing pivotal data and information about the e-scooter's state. Mainly using the gyroscope and accelerometer functions of this particular IMU.

### 1.2.2 Software

Initially, the project incorporated both a real-time program and a simulation program. However, they were subsequently replaced with the goal of achieving software modularity across all autobike projects, including bicycles and e-scooter.

### 1.2.3 The E-scooter's Condition

The steering motor assembly, see Figure 2, was broken and the steering motor's encoder was not functional. The controller box had a faulty diode on the voltage regulator, which was noticed after the initial power-on. The rest of the hardware were functional but needed to be validated. All software, simulations, and CAD models were shared and functional.

Figure 2: Mount for the steering motor, with the same design as the one from Mälardalen.

## 1.3 Purpose and Goal

The goal of this bachelor's project was to continue the development of the e-scooter as it was already constructed. First to enable the scooter to balance it self at certain velocities, and then continue the development beyond that. Such as path tracking, etc.

The goals that were set up for this project are as follows:

- Validate the functionality of the e-scooter
  - Validating the forward motor function.
  - Validating the steering motor function.
  - Validating IMU sensor function.
  - Replicating the results from Mälardalen
- Compare controlling of the e-scooter in simulation to real life.
  - A simplified mathematical model of the e-scooter
  - Simulating the balancing tests to find the critical parameters, such as controller parameters and minimum stable velocity
  - Doing real tests on the e-scooter with the parameters, achieved with the simulations
  - Balancing the e-scooter on its own without external assistance, and just the scooter's own steering inputs alone.

## 1.4 Societal- and Ethical Aspects

The ethics behind self-driving vehicles is important to discuss, to understand what the technology can be used for in the future. In case of predicted bad outcomes, it may be necessary to evaluate whether the project is worth carrying out, or whether other solutions with better ethical outcomes need to be developed.

There are studies showing that many traffic accidents are the result of drunk driving on e-scooters [5]. The drivers are also prone to not wearing helmets, which exposes them to greater danger. Because of this, testing autonomous vehicles with e-scooters is very important. If the e-scooter riders are prone to making bad decisions, the car's safety systems have to be ready to act to reduce the consequences. In all traffic situations, it is important not to expect traffic to work according to the rules, and this has to be taken into consideration when programming the automotive safety features [13]. There is also an ethical dilemma in the question of who is responsible if a collision or accident happens [14]. The

public need to feel safe around the vehicles, but those who built them, should have trust in their product and that they feel assured they won't have to be held responsible for traffic accidents.

# 2 Methodology

The project was pursued by conducting a literature review, simulating various phenomena, improving hardware and then performing actual testing to see whether those phenomena held true.

The e-scooter's mathematical model is similar to that of an inverted pendulum, which is also true for the balancing problem of the bikes. To understand the kinematics and dynamics of the e-scooter, the master's thesis of Cécile Savoye, "Modeling and Path Tracking Control of an Autonomous Bicycle" [15], and the papers "Lateral Control of a Self-Driving Bike" from Chalmers University [16] and "Bicycle Dynamics and Control" from Lund's University [17] have been studied to understand the mathematical side of the project.

## 2.1 Mathematical Model

The mathematical model of the e-scooter is important to understand and incorporate to control and balance the e-scooter. The problem of balancing the e-scooter and similar vehicles is that of an inverted pendulum, where the tilt angle, $\varphi$, is influenced by the steering angle, $\delta$.

### 2.1.1 Planar motion

Kinematics is the study of motion without looking at its cause. The e-scooter exists within an XY-plane, with its state being expressed through $x_L = [X\ Y\ \Psi]^T$ where $\Psi$ is the angle the scooter is heading in, see Figure 3.

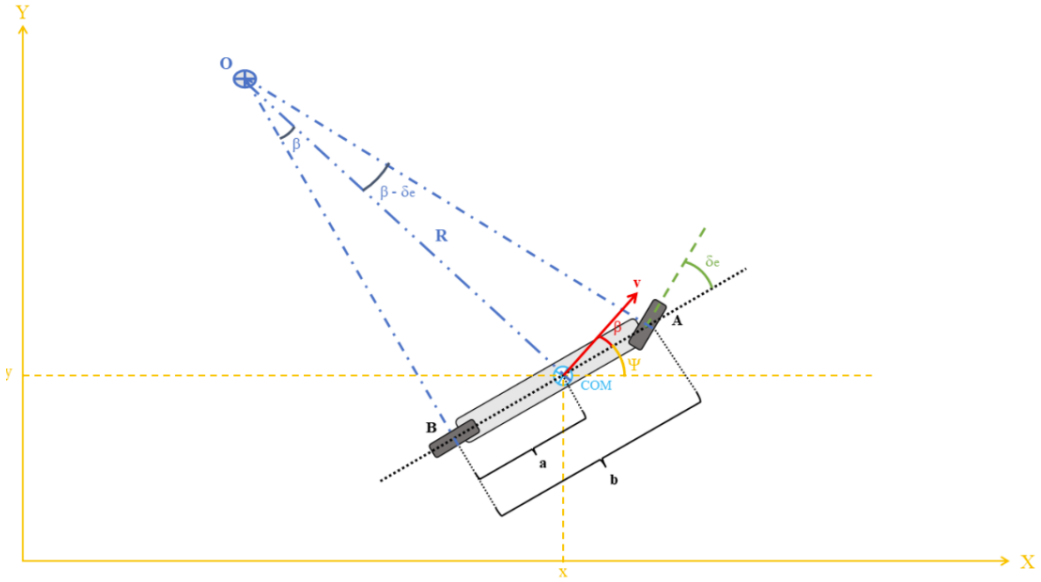


Figure 3: Visualisation of the path-controlling. A representing the front wheel of the e-scooter and B representing the rear wheel.

In the path-control, $O$ is the point of rotation for the scooter, with the distance between the CoM and it being $R$.

The angle of the velocity is $\beta$ which can be determined through

$$\beta = \arctan(\frac{a}{b} \cdot \tan(\delta_e)) \quad [°]. \tag{1}$$

With $x_L = [X\ Y\ \Psi]^T$ as the position of the scooter, the change of position, $\dot{x}_L$ is calculated:

$$\dot{x}_L = \begin{bmatrix} v \cdot \cos(\psi + \beta) \\ v \cdot \sin(\psi + \beta) \\ v \cdot \frac{\tan(\delta_e)}{b} \cos(\beta) \end{bmatrix}. \tag{2}$$

The direction of the scooter $\theta$ is calculated by

$$\theta = \Psi + \beta \quad [°]. \tag{3}$$

Since the distance $R$ is much greater than the wheelbase $b$, the system can be linearized. This results in the equations:

$$\beta = \frac{a \cdot \delta_e}{b} \quad [°], \tag{4}$$

$$x = [X\ \ Y\ \ \psi]^T, \tag{5}$$

$$u = [v \;\; \delta_e]^T \tag{6}$$

and

$$\dot{x} = \begin{bmatrix} v(\cos\psi \cdot \cos\beta - \sin\psi \cdot \sin\beta) \\ v(\cos\psi \cdot \sin\beta + \sin\psi \cdot \cos\beta) \\ \frac{v \cdot \tan(\delta_e)}{b} \end{bmatrix} = \begin{bmatrix} v(\cos\psi - \frac{a}{b} \cdot \delta_e \cdot \sin\psi) \\ v(\sin\psi + \frac{a}{b} \cdot \delta_e \cdot \cos\psi) \\ \frac{v}{b} \cdot \delta_e \end{bmatrix} =$$

$$= \begin{bmatrix} v \cdot \cos\psi \\ v \cdot \sin\psi \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{a}{b} \cdot v \cdot \sin\psi \\ \frac{a}{b} \cdot v \cdot \cos\psi \\ \frac{v}{b} \end{bmatrix} \delta_e . \tag{7}$$

The system is not truly linearized because of $\Psi$. This model has the in-parameters shown in equation 6 as those are the parameters that can be controlled through the driving motor and steering motor of the scooter.

### 2.1.2 Roll dynamics

Dynamics is the study of causes of motion where the balancing act is most important, in which the scooter and bike work as an inverted pendulum. See Figure 4 for illustration.

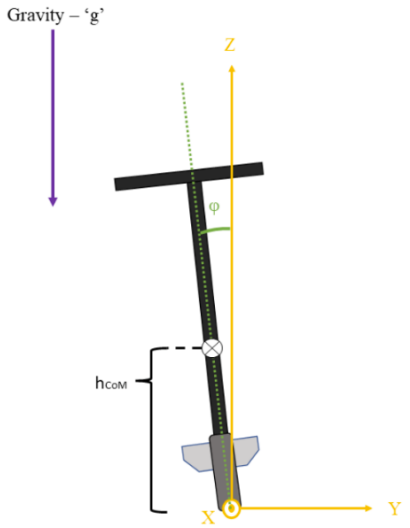


Figure 4: The balancing of the e-scooter.

The goal for balancing is to maintain $\varphi = 0°$, or for future path-tracking, for the $\varphi$ to equal a desired angle for turning.

The angular momentum of this inverted pendulum is calculated by

$$J\ddot{\varphi} - mgh\varphi = \frac{Dv\dot{\delta}}{b} + \frac{mv^2h\delta}{b}, \tag{8}$$

with the second-order equation has the transfer function between $\varphi$ and $\delta$ being

$$G_{\varphi\delta}(s) = \frac{v(Ds + mvh)}{b(Js^2 - mgh)}. \tag{9}$$

This is not a stable system, though. For that, the proportional feedback law

$$\delta = -k_2\varphi \quad [^\circ] \tag{10}$$

is implemented in equation 8, which leads to

$$J\ddot{\varphi} + \frac{Dv\dot{\delta}}{b} + (\frac{mv^2hk_2}{b} - mgh)\varphi = 0. \tag{11}$$

This is stable if

$$k_2 > \frac{bg}{v^2}, \tag{12}$$

which is most affected by the increasing velocity of the scooter $v$. This is also where a notable difference between a bike and an e-scooter becomes evident. The wheelbase $b$ of the bicycle is larger than that of the e-scooter, resulting in less stability for the bicycle and requiring a higher velocity to maintain balance.

This underlying logic drives the e-scooter's simulation and software.

### 2.1.3 Center of Mass

The CoM of the e-scooter is calculated by dividing it into parts in simplified form. The simplified parts are divided into the following parts:

- **Base -** Encompassing the battery, base and additional forks. Simplified as a homogenous box with its CoM in the middle of its height and length.
- **Wheels -** Separated by front and rear wheel. The CoM of each wheel is in the center of the wheel. Calculated together with the base in CoM calculations, since they are centered on the same y-value.
- **Steering bar -** In the shape of a cylinder, where it has been generalized to become homogenous.
- **Handle bar -** In the shape of a cylinder, where it has been generalized to become homogenous.

See Figure 5 for visualization. The e-scooter was divided into these parts since the CoM of the different parts affects the e-scooter's CoM in a well-distributed way. The parts can also be considered homogeneous.

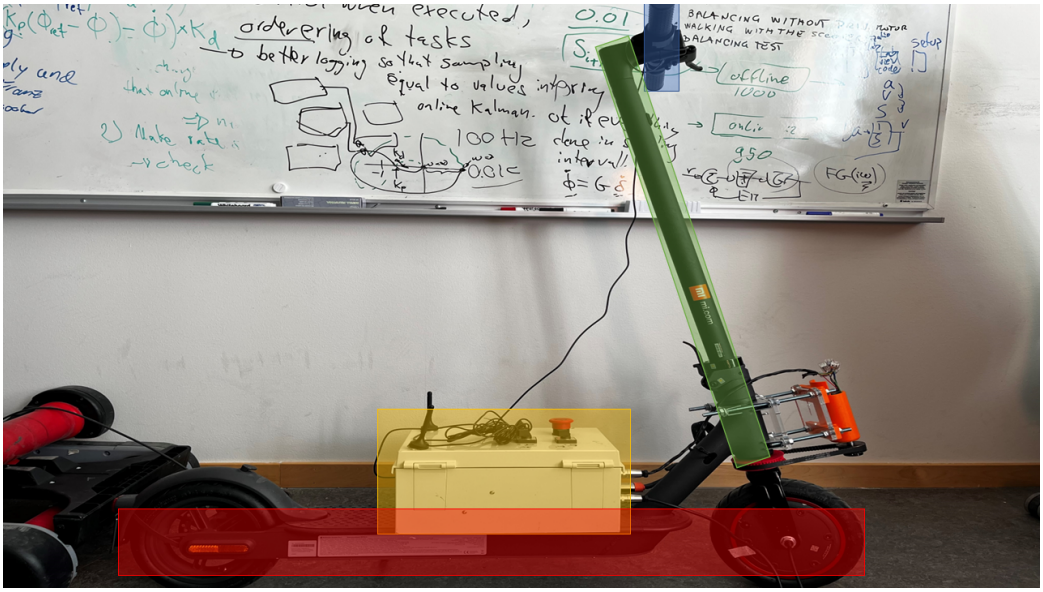


Figure 5: The e-scooter simplified, each box represents a term in the equation of the scooter's CoM position.

Because of mismatching values of the weight in previous work, each part was remeasured. This was done with a digital personal scale [18], with an accuracy of 0.1 kg. The lengths were also remeasured to get the correct calculations. The values are as follows:

| Steering bar (i = 1) | Steering handle (i = 2) |
|---|---|
| $m_{bar} = 0.9\text{kg}$ | $m_{handle} = 0.7\text{kg}$ |
| $l_{bar} = 0.7\text{m}$ | $l_{handle} = 0.43\text{m}$ |
| $\text{x} = x_{fork} - 0.35 * \cos(\lambda) = 0.7516$ | $x_{handle} = x_{fork} - 0.7 * \cos(\lambda) = 0.6832$ |
| $\text{y} = y_{fork} + 0.35 * \sin(\lambda) = 0.7632$ | $y_{handle} = y_{fork} + 0.7 * \sin(\lambda) = 1.1065$ |

Table 1: Measurements of the simplified steering bar and steering handle

| E-scooter | Control box (i = 3) | Base (i = 4) | Wheels |
|---|---|---|---|
| $m_{scooter} = 14.3$kg | $m_{box} = 3.7$kg | $m_{base} = 12.5$kg | r = 0.1079m |
| b = 0.88m | $x_{box} \approx 0.42$m | $x_{base} = 0.41$m | $m_{frontwheel} = 3.06$kg |
| $\lambda = 78.7°$ | $y_{box} \approx 0.23$m | $y_{base} = 0.15$m | |
| $x_{fork} = 0.82$m | | | |
| $y_{fork} = 0.42$m | | | |

Table 2: Measurements of simplified components and the complete e-scooter

The measurements of the steering bar, steering handle, control box, and base, together with wheels, in **??**,2 are put into equation 13 where CoM is calculated. The position of all of the centers of mass will be in coordinates, with the x part equaling a and the y part equal to $h_{CoM}$, and this is calculated by

$$(a, h_{CoM}) = \sum_{i=1}^{4} \frac{m_i(x_i, y_i)}{m_{scooter}} = (0.4401; 0.2352). \tag{13}$$

The inertia of the front wheel is also calculated in 14, as well as D, the inertia of the scooter which is calculated as

$$J = \frac{1}{2} \cdot m_{frontwheel} \cdot r^2 = 0.0178 \quad [\text{kg}\cdot\text{m}^2] \tag{14}$$

and

$$D = -J_{xy} = mah_{CoM} = 1.86 \quad [\text{kg}\cdot\text{m}^2]. \tag{15}$$

For alterations to the weight on the e-scooter, CoM has to be recalculated using equation 13.

## 2.2 Simulation

This project aimed to simulate the following:

- Simulate the e-scooter to determine the minimum stable speed and suitable controller values.
- Simulate the differences between the self-driving bikes and their parameters. What makes the e-scooter different?
- Simulate scenarios with different amounts of mass added to the e-scooter, changing its CoM and weight.

Knowing suitable speed and controller values is necessary for the testing. Simulating allows investigation of promising values, without risking to hurt the hardware. These values can be further examined, so they are well suited for real testing. Since there already exist working self-driving bicycles [15], a key interest was to determine how the differences between the bike and e-scooter vehicle dynamics impact the results.

The simulations are done using the previously mentioned simulation program from Chalmers and changing the parameter values for the intended phenomena that is being simulated for. All software is made to be modular and easily work with different parameters. The parameters are taken from table 1.

Increasing the velocity until stable for both scooter and bike, changing controller parameters to fit to the e-scooter and drawing conclusions, are advantageous steps in increasing the understanding of the problem.

### 2.2.1 Altering the Scooter CoM

The bicycle has already been validated to be able to balance itself [15]. Knowing the difference between the scooter and the bike is a way to figure out how to adjust for the scooter, and see if it is possible to balance it. In the simulation it is possible to recreate the scooter by feeding in the necessary parameters.

This is also partly why altering the CoM position is interesting to simulate. The height of the CoM and IMU are some of the largest differences between the bike and the scooter, and both are important for the balancing. Examining how changing them will impact the scooter in simulation is therefore of great interest.

To study the effects of altering the scooter CoM, the simulation ran the same trajectory of "ascending_sine". Running at the same speed of 2.5 m/s, the different models of altered CoM get put through different controller values to see how they impact the controller.

The different alterations to the scooter are as follows:

- The base scooter for reference with the same parameters.
- One alteration with 10 kg suspended 30 cm above the scooter bases's CoM.
- One alteration with 15 kg suspended to the same height.
- Another with 15 kg but suspended 50 cm above the scooter base's CoM.

All of the alterations keep the center off mass on the same $x_{pos}$. By keeping the same center of mass position, the distance from the rear wheel to the CoM, $a$, will remain the same. The values were then used in equation 13.

## 2.3 Hardware Alterations and Additions

In parallel with running simulations, the hardware of the scooter was altered. The broken parts were repaired or redesigned and more things were also added to the scooter to improve it.

### 2.3.1 Changed Components

The following components have been replaced:

- The previous steering motor was replaced to a SD3039, 12 V, spur-gear DC motor from **Transmotec** [19] with an encoder and 100:1 gear ratio. In order to make it compatible with the scooter, certain cables and couplings needed to be replaced.
- The entire steering assembly was redesigned and improved, both to fit the new steering motor and to strengthen it, to be able to handle the great stress that the belt pulley was creating.
- The previous PDB (power distribution board) was replaced with a DC-DC adjustable voltage step-down module (LM2596S) from **Whadda** [20].

The previous PDB was custom-made, and despite replacing a faulty diode on the board, it experienced another failure. Tests were made to decide whether the diode needed better soldering, but the decision was made to replace it entirely. The new PDB's only difference is that it is adjustable in output voltage.

### 2.3.2 Steering Assembly

See Figure 6 for the steering assembly. It consists of:

- 3D printed motor mount.
- Laser cut acrylic plates.
- Gang rods, nuts and rim.
- Two pulley gears, one of which 3D printed to fit around the steering shaft of the scooter.

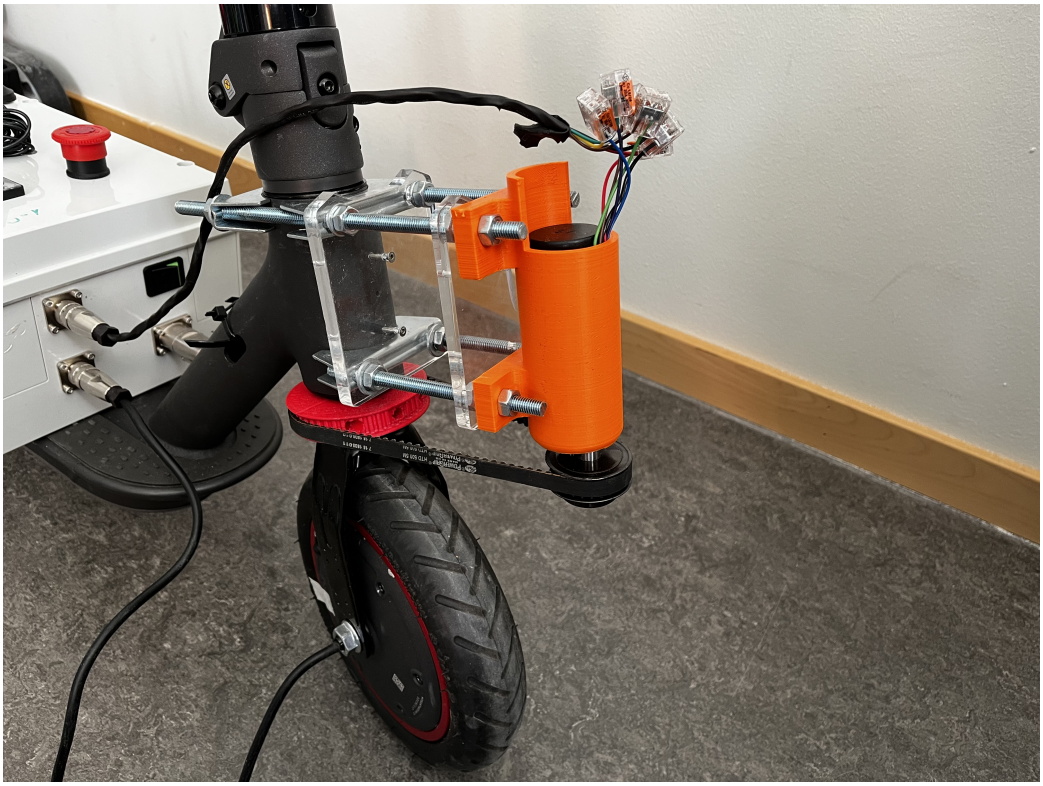

Figure 6: The steering assembly. All the respective parts can be seen.

By utilizing acrylic instead of 3D-printed parts in high-stress areas, this steering assembly is significantly stronger than its predecessor. Once subjected to the appropriate tension required to ensure proper pulley function without slipping, it can withstand the necessary stress.

The use of nuts and gang rods also allows for rigidity and adjustability. The position of the motor mount and steering motor could be adjusted through the nuts along the gang rods to ensure the right tension on the pulley. Previous iterations of the steering assembly did not use the rims which resulted in too much tension between the e-scooters steering bar and the plate. These tensions resulted in cracks in the plate. With the rims the forces are distributed and it can handle more force and tension.

The drawings for the new parts can be found in Appendix A.

### 2.3.3 Additions to the Hardware:

Certain additions to the hardware have also been made to the e-scooter, such as the inclusion of WiFi, GPS and box holders.

There was previously no solution to keeping the controller box on the scooter during actual testing. Four separate box mounts that hold onto the scooter were developed to keep the box's corners from moving. The four mounts clamp on the scooter base and retain the variability of the box's position on the scooter, allowing for future adjustments and additions. The mounts have been updated to be stronger so as not to snap, but they are still not sufficient at keeping the box in place during crashing.

The GPS **ZED-F9P** [21] was attached to the e-scooter to allow for future path-tracking implementation. The WiFi-router **RUT955** [22] was implemented to allow remote control of the scooter, letting the code to be run remotely and for test data to be easily downloaded through the WiFi connection.

## 2.4 Software

While alterations were done to the hardware, the software got analyzed, to be able to start testing. The aim of using software in this project was to communicate with the myRIO and run various component and system tests on the e-scooter. The software used is called LabVIEW, a graphical programming language developed by National Instruments where the user creates virtual instruments (VIs). The main reason why LabVIEW is used is the hardware, myRIO, which is also developed by National Instruments and has good communication with LabVIEW.

### 2.4.1 LabVIEW codes

There are three major VIs in LabVIEW, each for a different purpose and use. Every VI in LabVIEW has two main parts: the Front Panel and the Block Diagram. The Front Panel is the graphical user interface where the user can see the code results, input data and interact with the program. The Block Diagram is the part where the code is built and executed. A brief description of all three major VIs is given below.

- **Main.VI**: The VI where the code to run the e-scooter is created. It consists of several subVIs, which are shown in a better format in Figure 7 below.
- **Configuration.VI**: The parameters of the e-scooter and the motors, the control values and periods of each task, along with the trajectory matrix, are found there. These settings can be uploaded to myRIO once in order to not have to load and save them every time in the front panel of Configuration. VI. As of now, the configuration parameters are permanently stored in the myRIO and can be found in /c/ni-rt/startup/configuration.xml in the myRIO driver. To calibrate the IMU gyroscope. As of now, the accelerometer can be calibrated manually in Configuration.VI.
- **Calibration.VI**: To calibrate the IMU gyroscope. As of now, the accelerometer can be calibrated manually in Configuration.VI.

### 2.4.2 Main.VI

In this section, Main.VI is explained in more detail. The process is explained step by step, and a brief description of every subVI is given. Further information on the subVIs can be found in Appendix D.

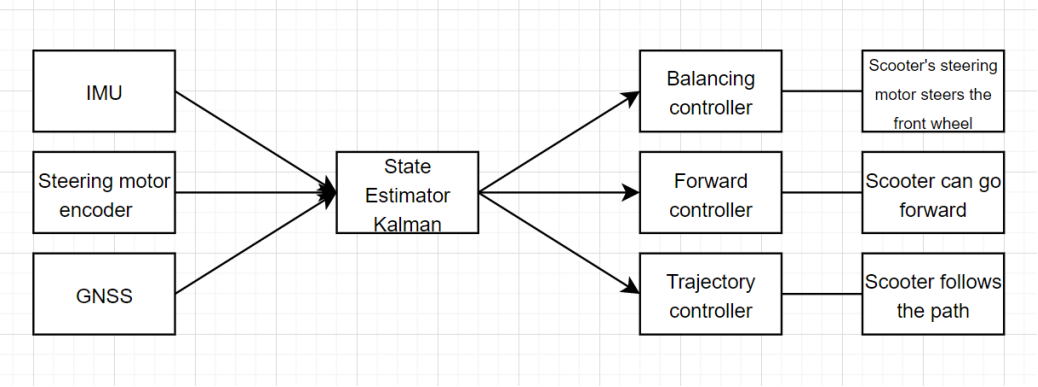


Figure 7: Main.VI working process step by step.

- IMU: Reads the accelerometer and gyroscope data of the IMU and outputs them.
- Steering motor encoder: Receives the encoder counter value from the encoder and, together with the steering motor configurations, outputs the steering angle in radians.
- GNSS: Receives the data from the GPS, performs the required calculations, and outputs the latitude in degrees and radians, the longitude in degrees and radians, the speed in meters per second, and the hand length in radians. As can be seen in Figure 7, the outputs of the IMU, steering motor encoder, and GNSS are going into the state estimator.
- State Estimator Kalman: The inputs coming into the state estimator are noisy measurements. To reduce the size of these data as much as possible, a Kalman filter is used in the state estimator. After passing the data through the Kalman filter, the outputs are not as noisy as before.
- Balancing controller: The balancing controller is a PD controller, which controls the steering angle. In this block, a maximum steering angle is also defined in order to stop steering after the maximum allowed angle. The structure of the balancing controller is shown in Figure 8 below.

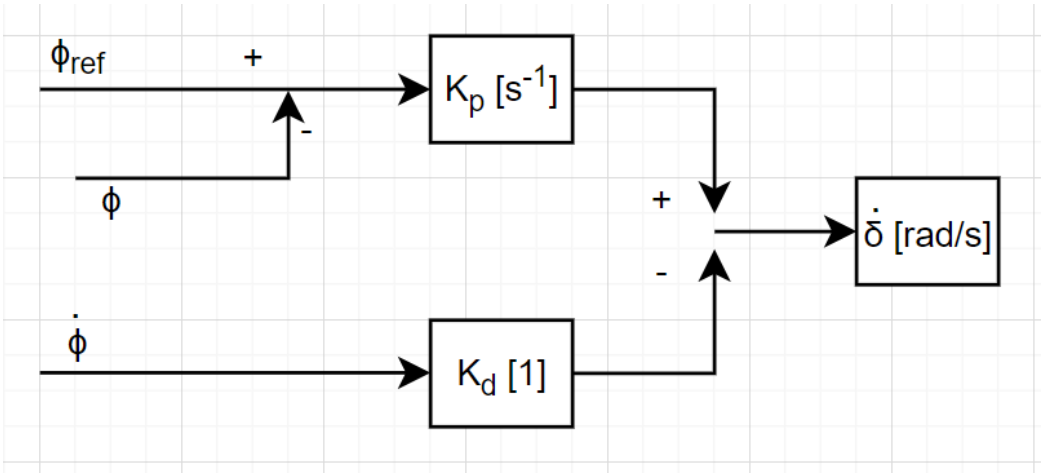


Figure 8: PD balancing controller structure.

- Forward controller: Receives an angular speed from the user and, by communicating with the VESC forward motor controller, runs the front wheel of the scooter.
- Trajectory controller: The trajectory controller is not used in this project as the scooter has not been stable. Therefore, it was disabled in LabVIEW.
- Logging and visualization: Displays all the measurements on the front panel of the main.VI, as well as saving all data on a .csv file for further data analysis purposes when necessary.

After running main.VI, it can be stopped either by using the stop button on the front panel or the emergency stop button on the control box.

### 2.4.3 Differences between LabVIEW codes

In this section, the differences between the LabVIEW code that was given from MDH and the LabVIEW code used in this project are presented. While the functionality of the codes is almost the same. The differences encompass:

- There were no configuration or calibration VIs, which made the process of writing the necessary settings in the main VI difficult. In addition, some important parameters were missing, which could lead to further problems.
- The code had no state estimator in order to reduce the noise of the measurements.
- The code was not standardized, which was important to have a standardized LabVIEW code in all different self-driving projects at Chalmers.

Because of these reasons the code changed to the one used throughout the project.

# 3 Testing

The testing of the scooter was divided into two parts: component testing and system testing, where component testing was conducted first to validate that the separate parts of the e-scooter worked as intended.

## 3.1 Component Tests

The component tests were designed to evaluate specific components separately to see if they behaved as expected. The components that were of most interest were the IMU, steering motor, GPS, router, and driving motor.

- **IMU**: When connected, it could be tested by simply reading the accelerations and gyro along the X, Y, and Z axes on the software, LabVIEW, when standing totally straight.
- **Forward motor**: The forward motor's accuracy was tested by placing a stopwatch beside it and filming it with a slow-motion camera. A default gear ratio of 250:1 was given to the software as an input. By doing this, the wheel revolutions could be counted and compared to the RPM input in the computer, with the purpose of validating the driving motor gear ratio and maximum velocity.
- **Router**: The router validation test was done by connecting to it with a computer through WiFi and uploading the C files to MyRio to see if that was possible.
- **GPS**: The GPS was validated by connecting it to its own software, u-center, and checking if the GPS values were being updated at the correct rate.

## 3.2 System tests

The system tests main purpose was to validate simulations that had been conducted. The tests were also used to validate that the components react as expected when connected to the system.

- **Steering motor encoder:** The first system test was conducted on the steering motor's encoder to validate the encoder's performance. The scooter is standing still, and the front wheel is turned manually while the steering motor is disabled. The encoder data is read from LabVIEW while the test is performed.
- **Steering motor:** The second system test was done in the same way as the encoder test, but the difference was that the steering motor was enabled and effectively rotated the front wheel. The configuration was

done in parallel with the test to achieve a valid performance from the motor.

- **Finding correct Kp and Kd:** At this stage, various control values from the range that showed a stable and good result in simulation were tested, and the performance and reaction of the steering motor were seen by watching the scooter's front wheel reaction to find a good Kp and Kd that worked on the scooter as well as in the simulation.

- **Balancing test without Forward motor:** After the steering motor's behavior while standing still was validated, the next test was done by walking at a rapid speed with the scooter and letting the steering motor do the balancing. This test was done with Kp = 1 and Kd = 3, with a speed of approximately 2 m/s. After jogging with the scooter for a few seconds, the steering motor was enabled while the forward motor was turned off. This was used to observe the behavior of the e-scooter and determine whether it responded satisfactorily to the control values used. If this test showed a stable result, the next test could be performed.

- **Balancing test with forward motor:** Afterwards, the same test was performed, but this time the drive motor was turned on and the velocity was stepwise increased to 2 m/s. The balancing and steering of the scooter were done manually at this stage. After the speed reached 2 m/s, one of the members tried to keep the scooter straight without any roll or steering offset, while the steering motor was enabled to let the scooter try to balance itself. This was the actual balance test. This test was also performed with Kp = 1 and Kd = 3.

- **Trajectory test:** This test was supposed to be done by creating a line or a specific trajectory for the scooter and seeing if the scooter could follow the path while balancing. This test was not performed as the e-scooter did not manage to completely balance.

# 4 Result

In the following chapter, the results of the simulations and tests are presented.

## 4.1 Simulation Results

The aspects discussed in section 2.2 were simulated.

### Stable, Minimum Velocity for the E-scooter and Bike:

In this test, the simulation is executed, increasing the constant velocity of the scooter and bicycle until the estimated- and true paths are not oscillating, but are instead for stable and linear (details of which can be found in Appendix C.1).

Doing this, the e-scooter was found to be stable at speeds above 1.7 m/s, which can be seen in Figure 9. The e-scooter's actual movement is oscillating when v = 1.7 m/s, but when v is increased to 1.8 m/s, it is following the reference path smoothly. "Ref" is the reference path, and "True" is the true path of the e-scooter. "Estimate" is the position that the simulation estimates.

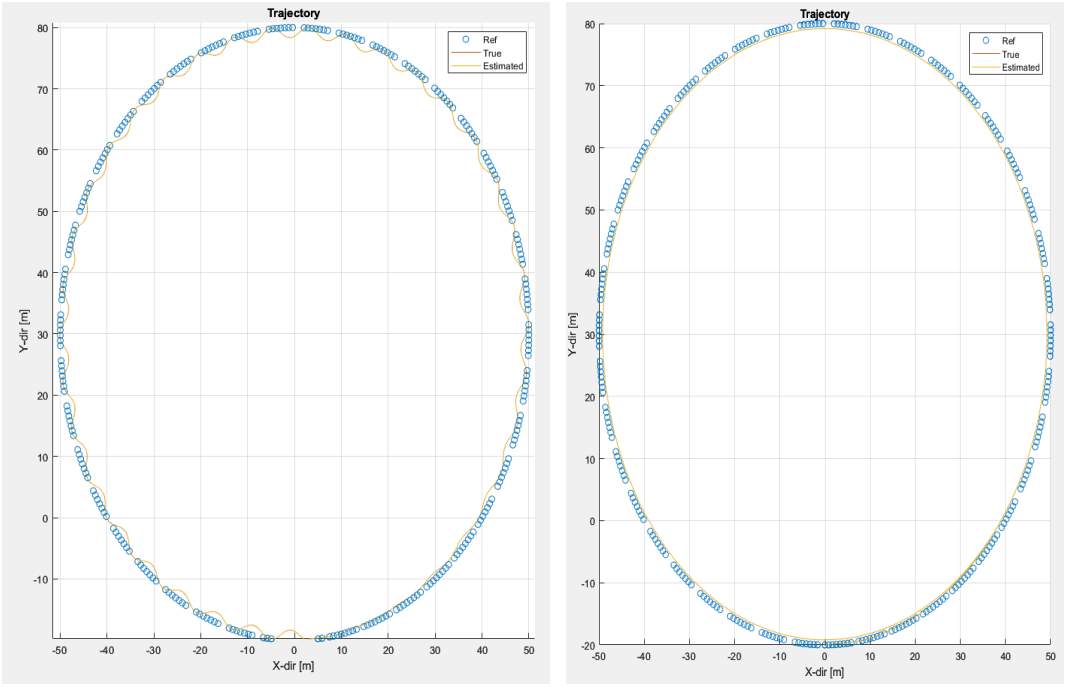


Figure 9: The simulated trajectories. v = 1.7 m/s in the left plot, and v = 1.8 m/s in the right.

Comparing the images, it can be seen that the *Estimated path* (the yellow line in the plot) is not oscillating at the right unlike the left. This is the threshold where testing should be significantly easier, as the scooter is more stable.

The bicycle was simulated with the same conditions and speed but did not balance until 2.0 m/s; see Figure 10.

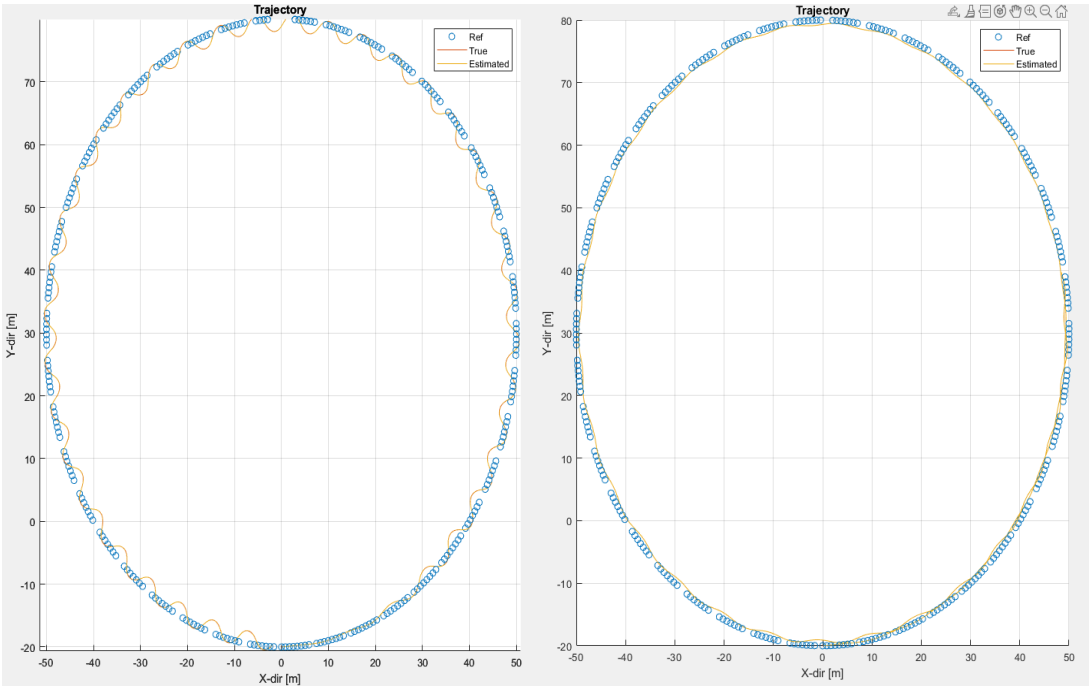


Figure 10: The simulations of the bike with the same conditions as the scooter. v = 1.9 m/s in the left plot and v = 2.0 m/s in the right.

Once again, it can be seen that the right plots *Estimated path* oscillates significantly less.

### 4.1.1 Controller Values

Checking controller values is about finding a reasonable starting point and then testing different values with the actual scooter. (Details of the simulation are in Appendix C.2).

Visualizing the difference in the derivative $D$ value:

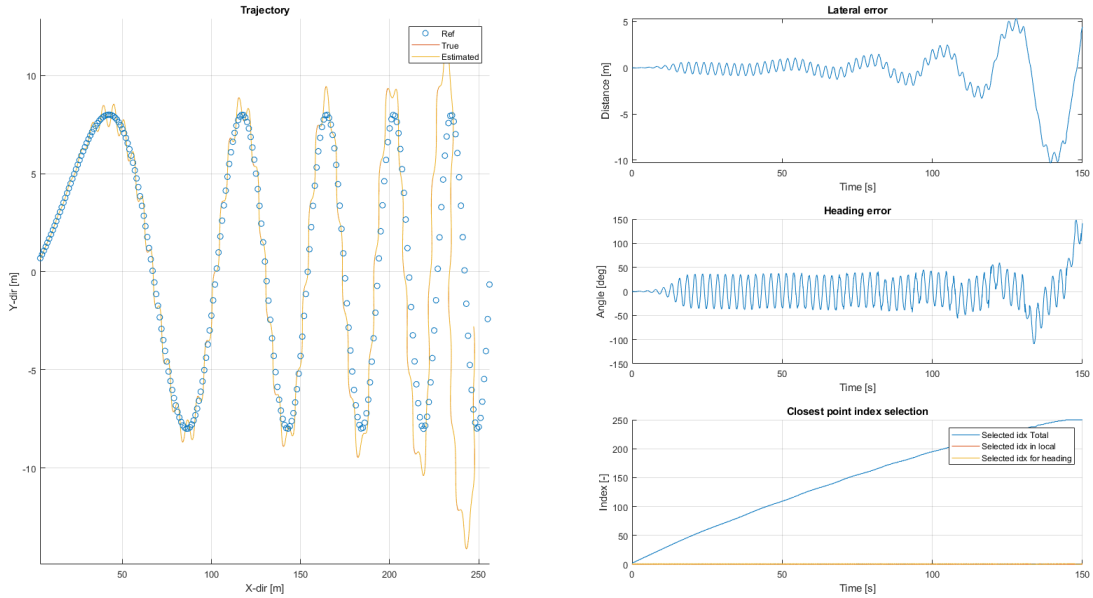


Figure 11: D = 1.2

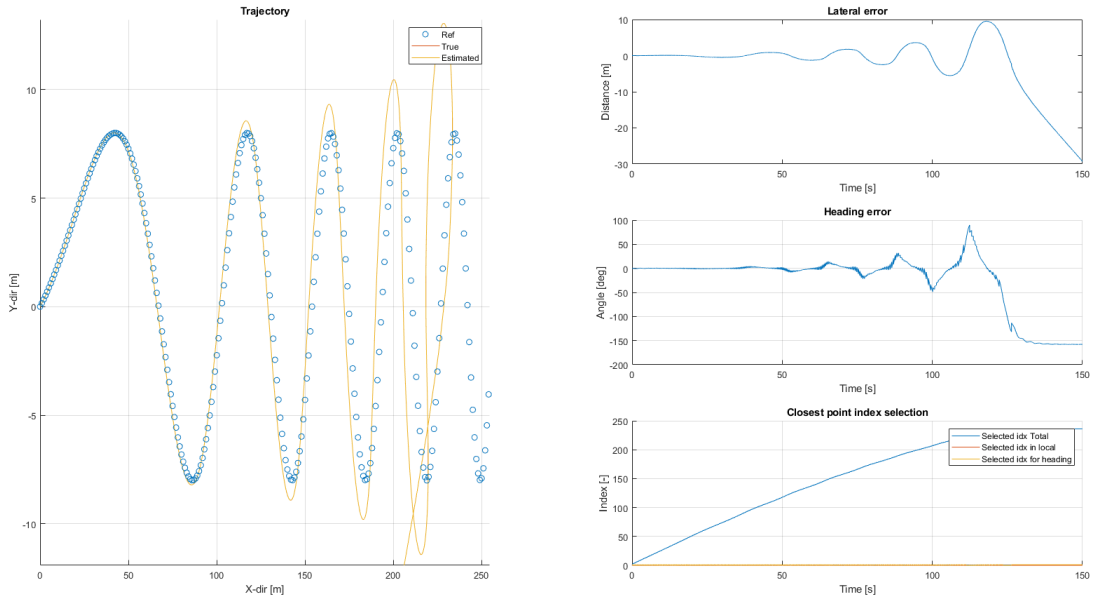


Figure 12: D = 1.4

The difference can be seen in how well the scooter oscillates; otherwise, it follows the trajectory, with a P value of 1.3 in both the simulated examples shown above.

For the P value, it is not as clear as to how high of a value it needs to be.

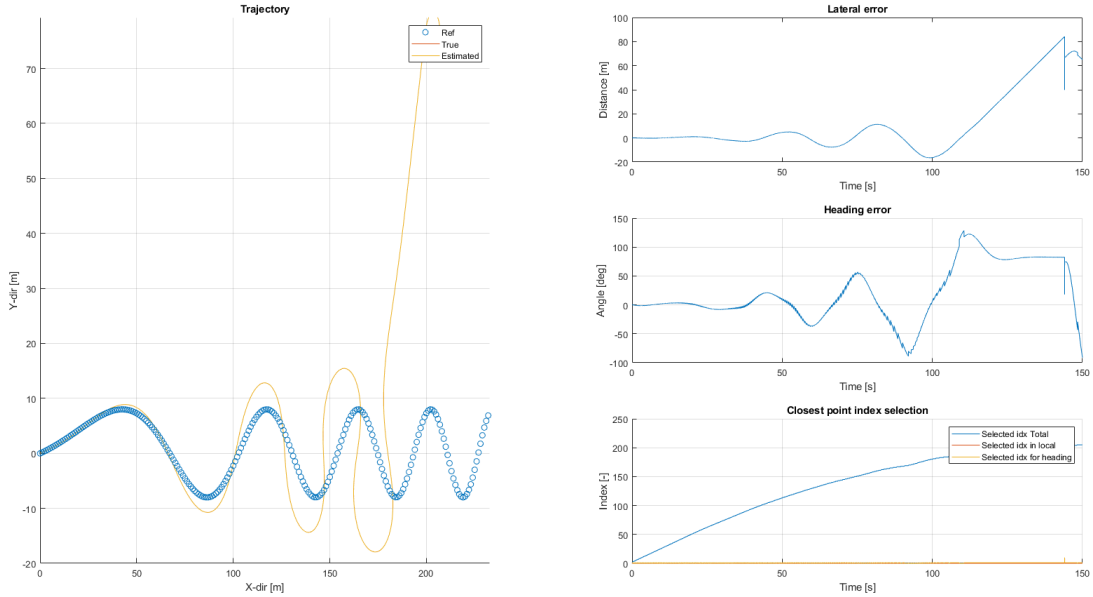


Figure 13: P = 0.7

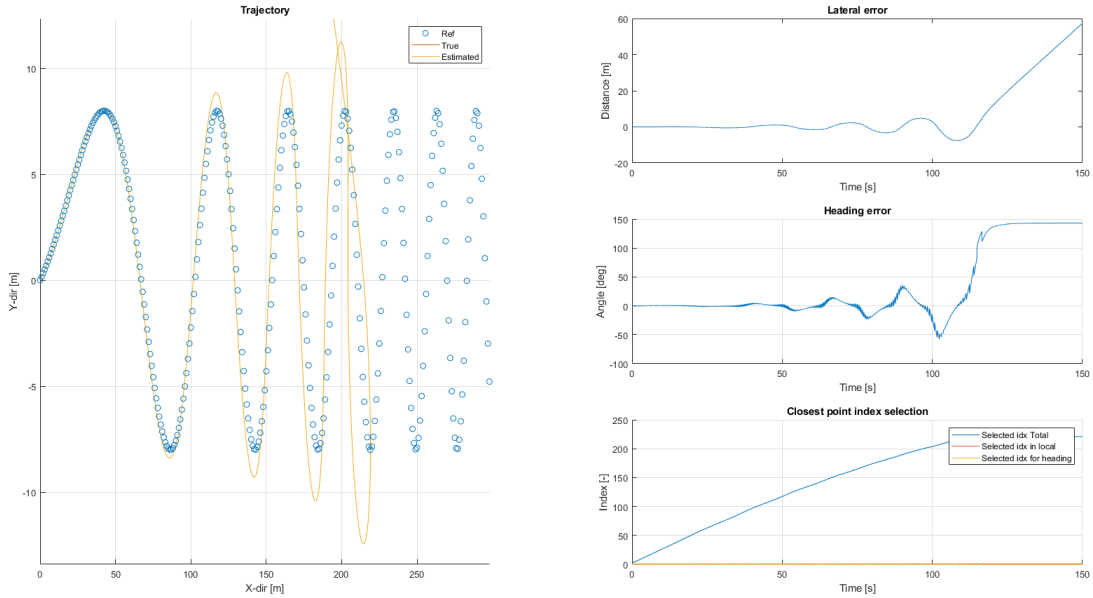


Figure 14: P = 10

Based on these results it seems likely that the P value should be as high as possible, but it is actually appropriate to be on the interval of 1 - 2. Like the bike's P value of 1.3.

The D value of the controller is determined to be higher than 1.4, while the P needs to be high enough to balance reasonably well and follow a future trajectory. This is achieved with a P value higher than 1.

### 4.1.2 Altering the Mass and Position of the Center of Mass

These were the alterations that were simulated:

- $COM_{pos(x,y)} = (0.44\ ;\ 0.31)$, $m_{tot} = 28$ kg, 30 cm above CoM.
- $COM_{pos(x,y)} = (0.44\ ;\ 0.34)$, $m_{tot} = 33$ kg, 30 cm above CoM.
- $COM_{pos(x,y)} = (0.44\ ;\ 0.44)$, $m_{tot} = 33$ kg, 50 cm above CoM.

Using the same controller values over "ascending_sine" trajectory these were the following results:

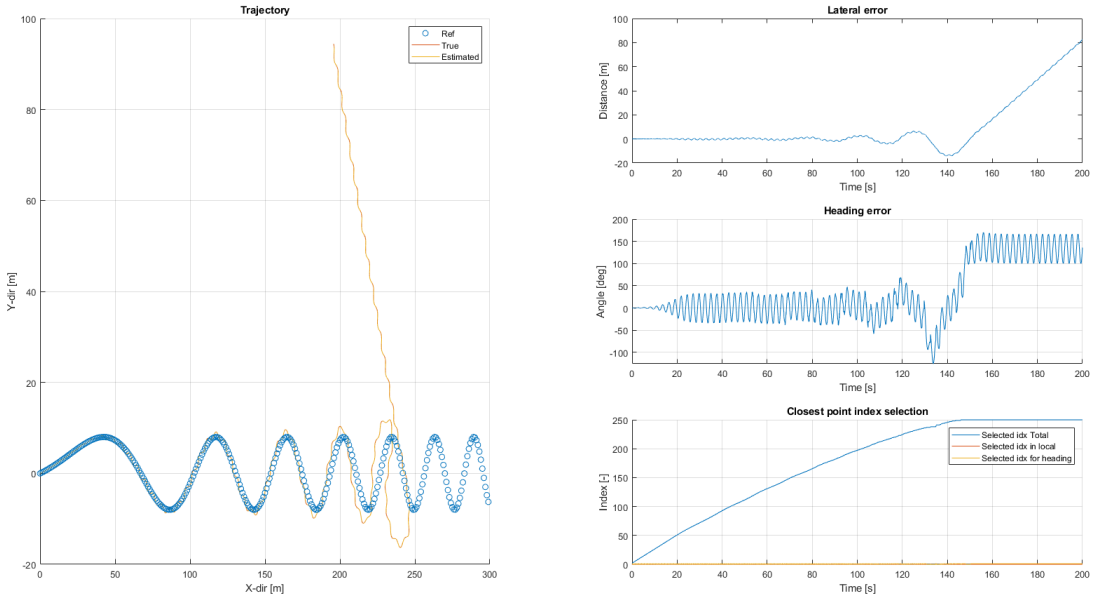


Figure 15: 10 kg elevated 30 cm, D = 1.3

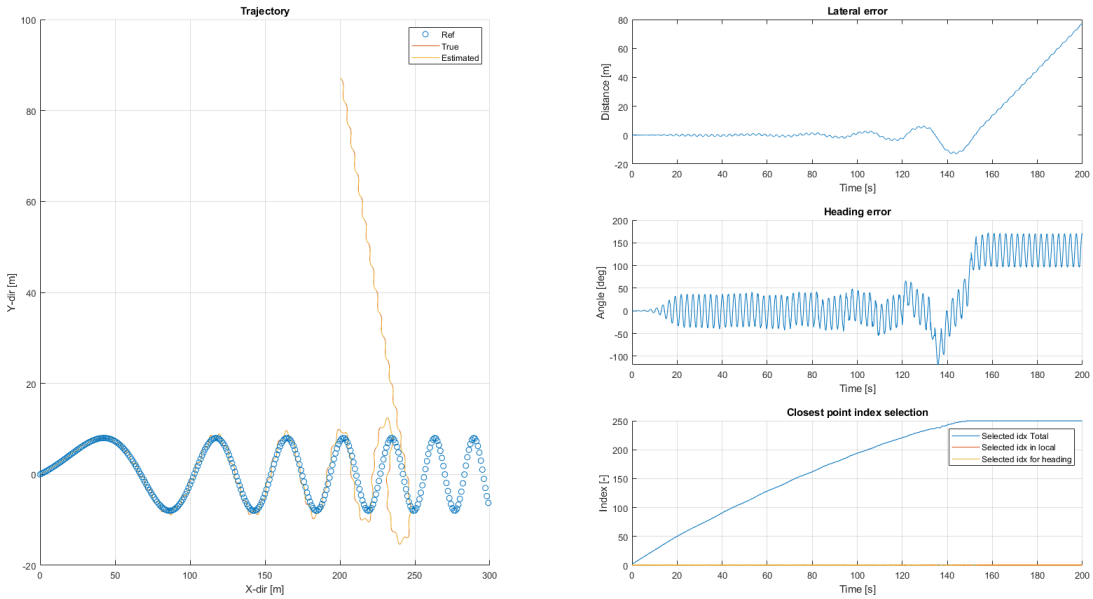


Figure 16: 15 kg elevated 30 cm, D = 1.3

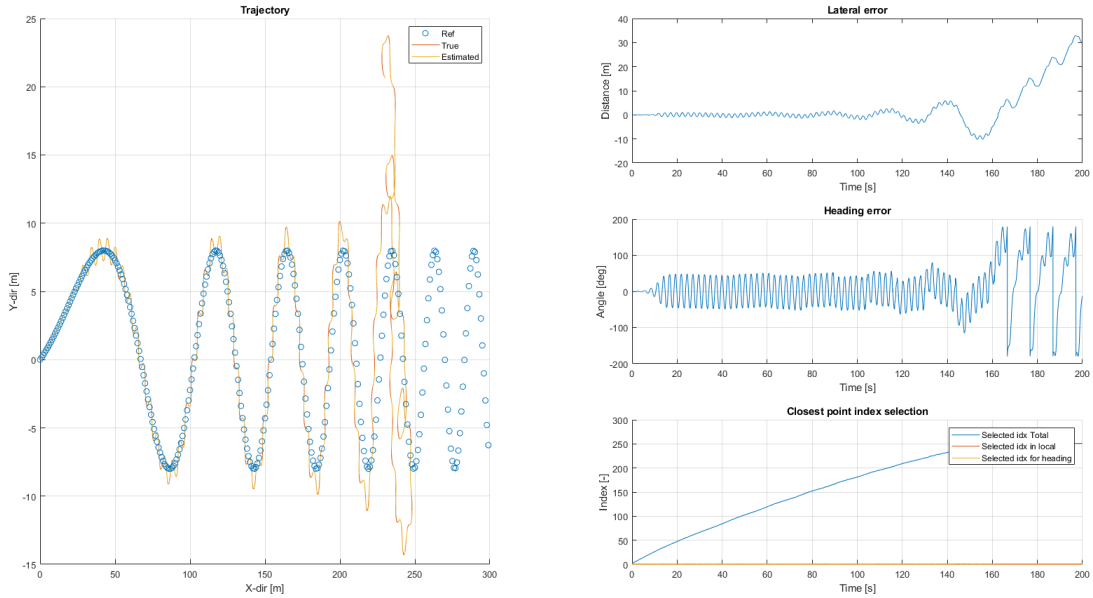


Figure 17: 10 kg elevated 50 cm, D = 1.3

What can be seen is that the higher located CoM does act has higher errors stray further from the path but oscillates less in its movement.
None of these simulations are done using suitable controller values that should balance the scooter.

If the scooter alterations balance there isn't any noticeable difference in simulation.

## 4.2 Test Result

### 4.2.1 Component test results

- **IMU:** The result of the IMU sensor test 3.1 is explained in this section. As it can be seen in figure 18, the values from the sensor were as they should have been. The gyroscope data in x, y and z direction after calibration of the IMU are in its initial position very close to zero. The acceleration in z-direction is 9.70 which is very close to the value of gravity (9.81). The acceleration in x direction is also very close to zero and the acceleration in y-direction required manual calibration to become as close as possible to zero, which was directly done in LabVIEW. Therefore, the functionality of the IMU sensor was validated.

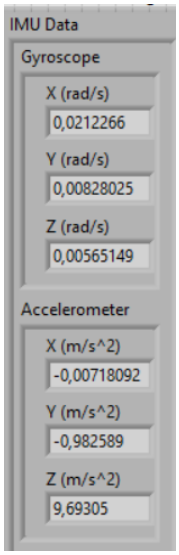


Figure 18: The IMU data on LabVIEW

- **Forward motor:** By performing this test, described in section 3.1, the correct gear ratio of the motor was found to be 16:1, and the maximum velocity of the motor was validated to be approximately 22 kilometers per hour (56 radians per second).
- **Router:** The router needs three to four minutes after starting its power to be connectable. After connecting to it, the process of uploading and downloading files, and also running the e-scooter via LabVIEW could be done.
- **GPS:** The GPS responds with data at a sampling rate of every 0.1 seconds. The GPS was not used in this project as the purpose was to balance the e-scooter but its performance is dependent on the environment.

### 4.2.2 System tests results

- **Steering motor encoder:** The encoder's test 3.2 was successful, as the counts from the encoder were read in LabVIEW. What was seen was that the encoder responded to the steering motor movements, and after resetting the encoder's count in the initial position (Straight standing still), a maximum turn to the right and a maximum turn to the left showed the same counts but with different signs. This process required the configuration of the encoder in ESCON Studio.
- **Steering motor:** The steering motor was also configured in ESCON Studio. The measured speed of the motor was compared to the steer rate

input and should have matched. As it is shown in Figure 19, the measured speed was not following the input at all, but after configuring the motor in ESCON Studio, the motor's measured speed was almost following the input. As ESCON Studio is built to configure Maxon motors and required some data from the motor that was not found in the datasheet of the bought steering motor, a small offset was visible even after the configuration. It was acceptable as the offset was small enough to not affect the performance of the motor.

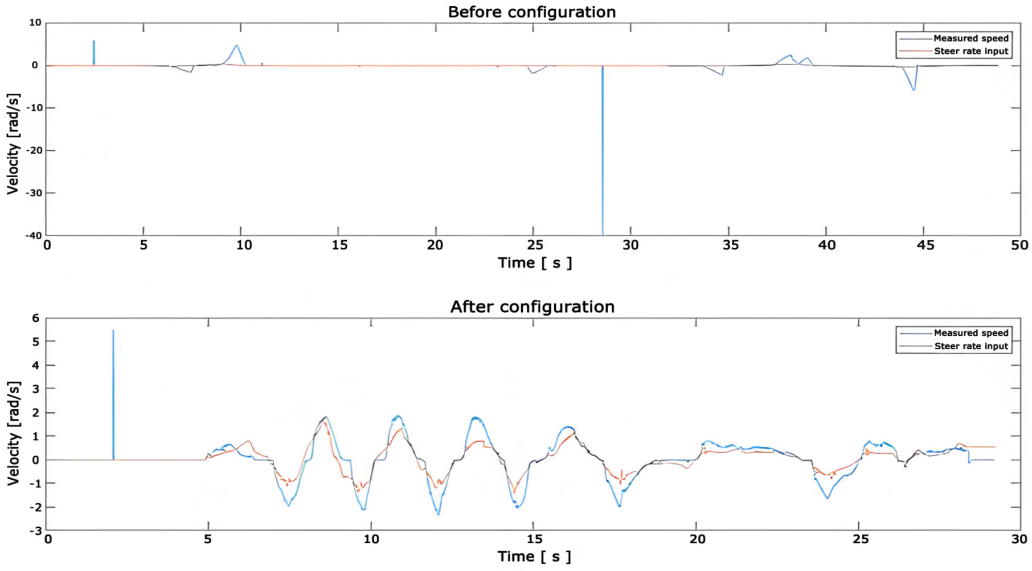


Figure 19: Measured speed vs steer rate input

- **Finding correct Kp and Kd:** As the simulation results for the controller values had shown that a range of Kp and Kd values could be used in real life, it was necessary to do experiments on the scooter with different Kp and Kd values to see the reaction of the steering according to the roll given. After running this experiment on various controller parameters, the wheel's reaction to slow and fast roll changes seemed to be sharper at a Kp = 1 and Kd = 3, which were chosen to go forward within the other tests. This test was filmed with the chosen controller parameters, Kp = 1 and Kd = 3, and can be seen here [23].

- **Balancing test without Forward motor:** In this test, 3.2, which was done three times the scooter was stable with a speed of around 2 meters per second (jogging speed) and was able to keep its balance. As the velocity increased more than that, the scooter started oscillating which then resulted in becoming totally unstable. This happened due to two main reasons.

One was the low positioning of the center of mass of the scooter, and the The second issue was that the scooter was propelled by a person, and when reaching higher speeds, this could cause unsteady movements and lead to instability. However, the result of this test with a velocity around 2 meters per second provided necessary conviction to proceed to the next test, which was the actual balancing test. A discussion about this test can be found in Section 5.2.

- **Balancing test with Forward motor:** The result of the actual balancing, test 3.2, was that the scooter was on the edge of being stable and unstable, as two of these four times, the scooter could keep the balance before stopping the test due to obstacles along the way. In the other two tests, the scooter was unstable, and the test was stopped to not let the scooter fall and crash. A video from two of the tests, one in which the scooter is able to keep its balance, and the second one in which the scooter falls immediately after being left alone, can be seen here [24].

  From the video of the these tests, the two times that the e-scooter could keep its balance, it was left alone in a very stabilized state that the wheel did not need to do any aggressive rotational motions. A discussion about the balancing tests result can be found in section 5.2.

# 5 Discussion

In the following section the results are discussed based on experience gained from this project.

## 5.1 Simulation

The simulation results in Figure 9 and 10 in section 4.1 show that both the scooter and the bicycle requires less steering with a higher velocity. Notably the scooter has a lower stable velocity than the bicycle. This is in line with equation 12 from the mathematical model in section 2.1.2 which predicts that a broader wheelbase $b$ demands a higher velocity for stability.

The simulations in section 4.1.2, where the scooter's physical parameters (specifically the height and mass of the scooter's centre of mass) were varied, indicate that raising and increasing the mass of the scooter's centre of mass would enhance its stability. By raising the scooter's centre of mass its oscillations became wider when unstable and this gives the scooter more time to react and rectify the error. An increased mass makes the scooter more resilient to external disturbances due to a raised inertia. Based on this, hardware modifications on the actual scooter could be implemented to facilitate balancing the scooter.

## 5.2 Tests

In both tests it was made apparent that the balancing of the scooter is very sensitive to its initial conditions. If the scooter was released with a roll angle offset, it was unable to recover and balance and instead, it immediately fell. If the roll angle was very close to zero, the scooter was able to keep its balance before the tests were stopped due to external factors such as obstacles. Consistently giving the scooter the same or an ideal roll angle proved difficult and dependent on the environment. Improving the test environment and utilizing a roll angle offset-free rig would enhance the tests by prolonging them, increasing the amount of data collected, and facilitating test result comparison.

## 5.3 Simulation vs Testing

The results from the simulation showed that with a velocity higher than the minimum velocity and the range of controller parameters which worked in simulation, the scooter should be able to balance without oscillating. In the actual tests discussed in section 5.2 there was no trace of oscillation, and the scooter could correct small changes in roll angle. However, due to the scooter reacting fast with large changes in steering to correct the roll angle this led to an overcompensation. This was more prevalent if the scooter started at an offset in roll angle, at high

velocities or effects from the environment. This did not occur in the simulations and changes to the hardware, such as adding a weight on top of the scooter and tuning the control values are necessary to achieve balancing as in the simulations.

# 6 Conclusion

After the component tests in section 4.2.1 and the system tests in 4.2.2 it can be concluded that the modifications made to the scooters steering assembly and steering motor allows for testing. The addition of the router and GPS mentioned section 2.3.3 was made for future development of the scooter to enable remote control and development of path tracking compatibility.

In the tests the e-scooter showed signs of being able to balance itself for a limited time with velocities and control values resulting from simulations in section 4.1. It is not able to recover large disruptions of its roll angle but can manage smaller roll offset. This makes the scooter sensitive for an imperfect environment with uneven ground and wind that the simulations did not account for.

Because of the scooter's low centre of mass in comparison to a bicycle it has proved to be more difficult to balance. The controller can be adjusted and improve the e-scooter's ability to balance. For the balance and stability to be satisfactory for any applications it might be required to raise its centre of mass.

# 7 Future Work

The work done in this bachelor's thesis is a step along the way to creating a self-driving e-scooter. Following are things that can be improved and done to further develop this project and possibly help future groups working on it.

## 7.1 Hardware

These are suggestions of future work done to the hardware:

- **Improving controller box-mounts:** The original solution is not strong enough to keep the controller box still during a crash, only during driving. Either the same sort of solution gets redesigned to hold on tighter and work better, or a new solution has to be found that should still provide the flexibility of the existing one.
- **Remote emergency stop:** The current emergency is placed on the controller box, onto the scooter, making it hard to access during testing in case of an emergency. It was also something Mälardalen mentioned as future work. This isn't as much of a problem anymore since the inclusion of WiFi allows the scooter to be remotely shut off through the computer. In case that fails however, it would be good to have another safety measure.

### 7.1.1 Movable CoM

As discussed, due to the importance of the CoM position in the mathematical model and the balancing of the e-scooter, the scooter might not be able to balance due to how low the CoM is on the scooter. To add a structure where more weight could be distributed higher above the scooter to heighten its CoM, would possibly aid it in balancing.

## 7.2 Software

On the software side, there is the necessary inclusion of sensor fusion and path-tracking for the future of the scooter. An obvious addition for the development of this project, and one that the bikes are working on currently.

# References


[1] F. Duartea and C. Ratti, "The impact of autonomous vehicles on cities: A review," Jul 2018. https://www.tandfonline.com/doi/epdf/10.1080/10630732.2018.1493883?needAccess=true&role=button.

[2] W. Huang, K. Wang, Y. Lv, and F. Zhu, "Autonomous vehicles testing methods review," Nov 2016. Publisher: IEEE, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7795548.

[3] B. Botello, R. Buehler, S. Hankey, A. Mondschein, and Z. Jiang, "Planning for walking and cycling in an autonomous-vehicle future," May 2019. https://reader.elsevier.com/reader/sd/pii/S2590198219300120?token=64CBE8DE6FA7F8C57836B13F0EB9F2896DDE28EA8E3C949958290C719472F6C0\F85EA6A072A47C9CBA47C5E67C7F6ED3&originRegion=eu-west-1&originCreation=20230514143737.

[4] J. Keane, "E-scooter survey says users 'drastically' reducing car use in european cities," Aug 2022. Publisher: **Forbes**, https://www.forbes.com/sites/jonathankeane/2022/08/25/e-scooter-survey-says-users-drastically-reducing-car-use-in-european-cities/?sh=78dcbcb46e54.

[5] O. Andersson and T. Djärv, "Electric scooter accidents leading to emergency department visits: influence of alcohol and outcomes in stockholm, sweden," Apr 2023. Publisher: Scientific reports, https://www.nature.com/articles/s41598-023-32857-1.

[6] J. Tark, "Micromobility products-related deaths, injuries, and hazard patterns: 2017–2021," Sep 2022. https://www.cpsc.gov/s3fs-public/Micromobility-Products-Related-Deaths-Injuries-and-Hazard-Patterns-2017-2021.pdf?VersionId=ZwIbrSm70AOuwb4de8hlVrn63Jx_SB.e.

[7] "2021 mercedes s-class - intelligent drive," sep 2020. https://www.youtube.com/watch?v=fajRI44z6Q4.

[8] E. Andersson and H. Hultergård, "Programming a self driving bike," jun 2020. https://www.youtube.com/watch?v=fajRI44z6Q4.

[9] P. Lidholm, M. Elyes Hamila, L. Bravenius, P. Gustavsson, T. Pascal Mugisha, and J. Nylander Nordström, "Project e-scooter — dva490 project course — mälardalen's university," Jan 2023. https://drive.google.com/drive/folders/10wYbklwVM8mSFPb2D5gnWJC-BijGMt1L.

[10] "Mi electric scooter pro 2 nordic edition," 2023. Publisher: **Mi** https://mistore.se/collections/pro-2/products/mi-electric-scooter-pro2-nordic-edition-1?variant=39483742847149.

[11] "myrio-1900," 2021. Publisher: **National Instruments** https://www.ni.com/sv-se/shop/hardware/products/myrio-student-embedded-device.html.

[12] "Digilent pmod nav: 9-axis imu plus barometer expansion module." Publisher: **RS online** https://se.rs-online.com/web/p/sensor-development-tools/1346482.

[13] J. De Freitas, A. Censi, B. Walker Smith, L. Di Lillo, S. E. Anthony, and E. Frazzoli, "From driverless dilemmas to more practical commonsense tests for automated vehicles," Oct 2021. Publisher: https://www.pnas.org/doi/epdf/10.1073/pnas.2010202118.

[14] K. Pandey, "Risks posed by self-driving vehicles," Apr 2022. Publisher: **Jumpstart** https://www.jumpstartmag.com/risks-posed-by-self-driving-vehicles/.

[15] C. Savoye, "Modelling and path tracking control of an autonomous bicycle — masters thesis eenx30 — chalmers university of technology," Jun 2019. https://odr.chalmers.se/server/api/core/bitstreams/19e5aaa9-a128-4cb5-9774-418ead0e00a6/content.

[16] G. Wen and J. Sjöberg, "Lateral control of a self-driving bike — chalmers university of technology," Nov 2022. https://ieeexplore.ieee.org/abstract/document/9986548.

[17] K. J. Åström, R. Klein, and A. Lennartsson, "Bicycle dynamics and control — lund's university," 2005. https://lucris.lub.lu.se/ws/portalfiles/portal/4692388/625565.pdf.

[18] "Personvåg digital," 2023. https://www.clasohlson.com/se/Personvåg-digital/p/44-3815.

[19] "Spur gear motor 12vdc 0.53a 56rpm 4.22w," 2023. Publisher: **Transmotec** https://transmotec.com/product/SD3039-12-100-FEC/.

[20] "Luxorparts variabel spänningsregulator switchad," 2023. Publisher: **Kjell Company** https://www.kjell.com/se/produkter/el-verktyg/utvecklingskit/arduino/stromforsorjning/luxorparts-variabel-spanningsregulator-switchad-p87049.

[21] "Zed-f9p," 2023. Publisher: **ublox** https://www.u-blox.com/en/product/zed-f9p-module.

[22] "Rut955," 2023. Publisher: **Teltonika** https://teltonika-networks.com/products/routers/rut955.

[23] "Steering testing to find kp and kd," Jun 2023. https://www.youtube.com/shorts/WnfRUJxXwUs.

[24] "Balancing tests," Jun 2023. https://www.youtube.com/shorts/UGUm3fHd7o0.

[25] "Escon 50/5, 4-q servocontroller for dc/ec motors, 5/15 a, 10 - 50 vdc," 2023. Publisher: **Maxon group** https://www.maxongroup.com/maxon/view/product/control/4-Q-Servokontroller/409510#:~:text=The%20ESCON%2050%2F5%20is,up%20to%20approximately%20250%20Watts.

[26] "Vesc 6 75v," 2023. Publisher: **Trampa boards** https://trampaboards.com/vesc-6-75--p-33322.html.

# A Appendix - CAD Drawings

## A.1 Drawing motor mount

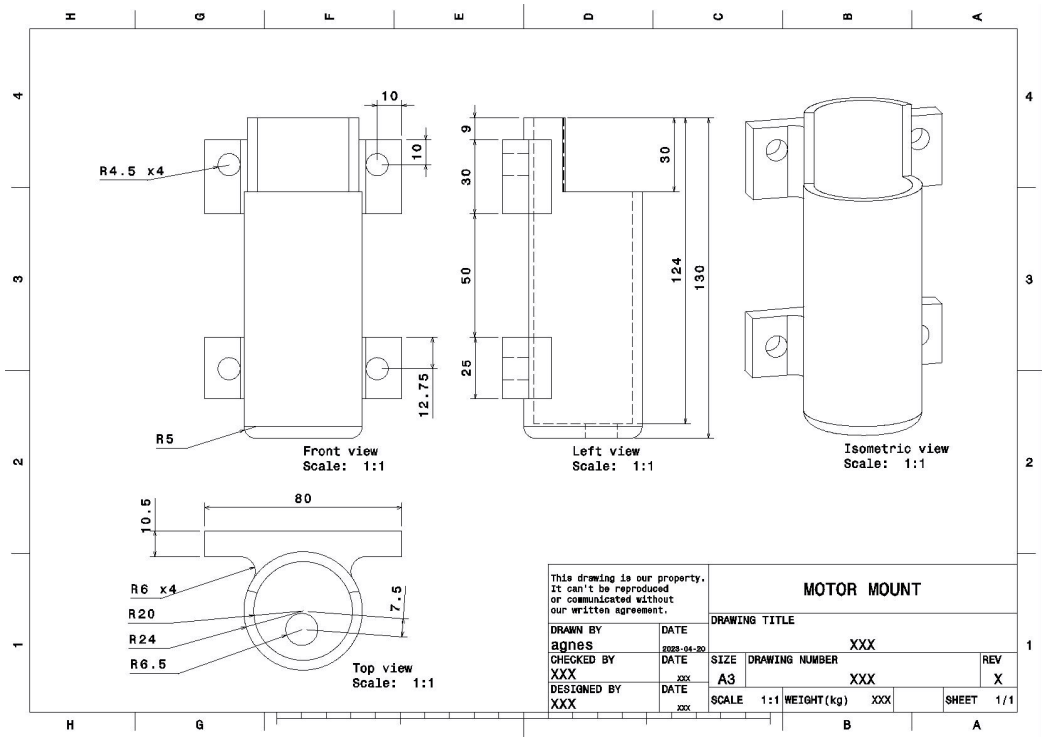


Figure 20: Drawing of the motor mount

## A.2 Drawing plate

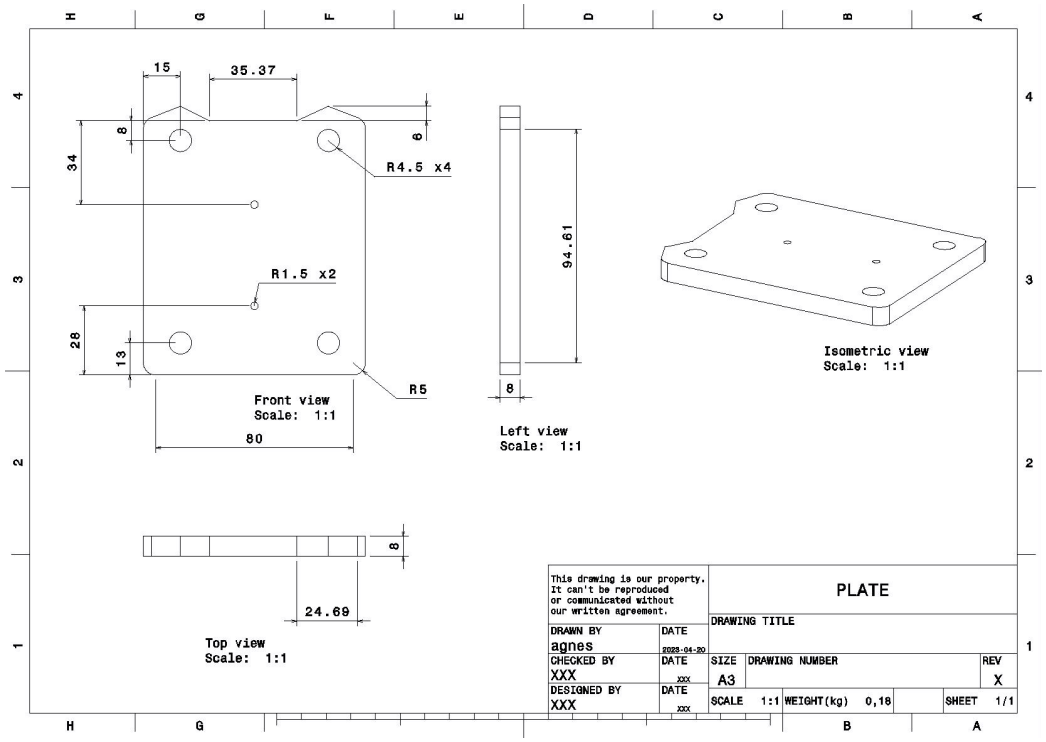


Figure 21: Drawing of the plate

## A.3 Drawing box mount 1

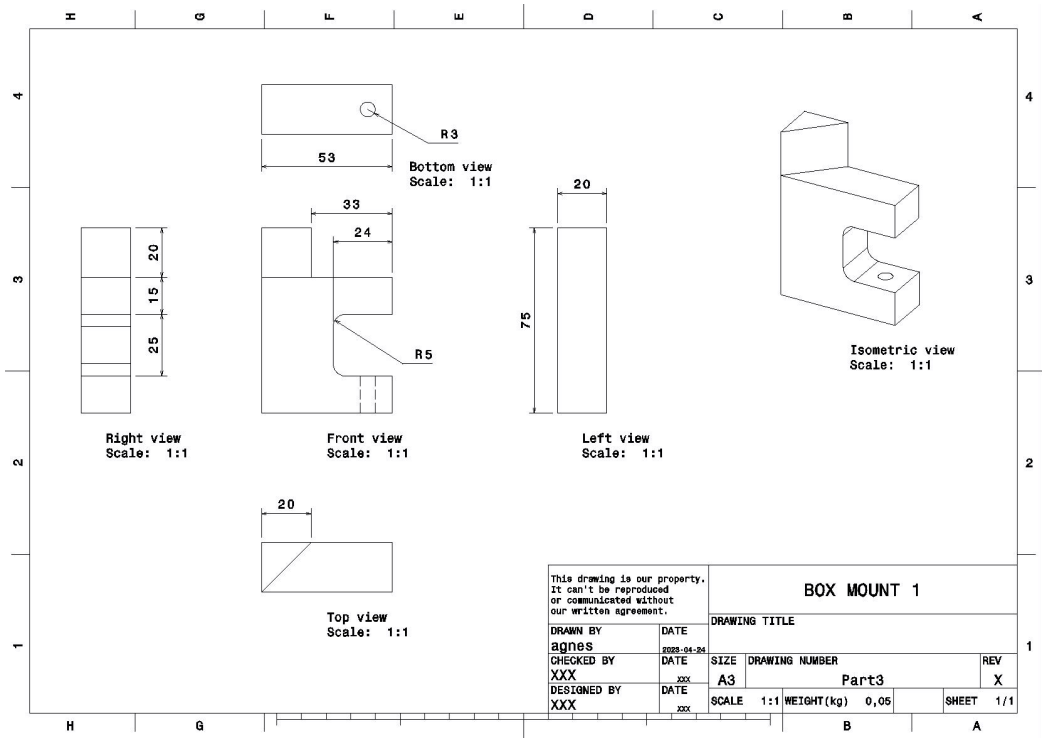


Figure 22: Drawing of box mount 1

## A.4 Drawing box mount 2

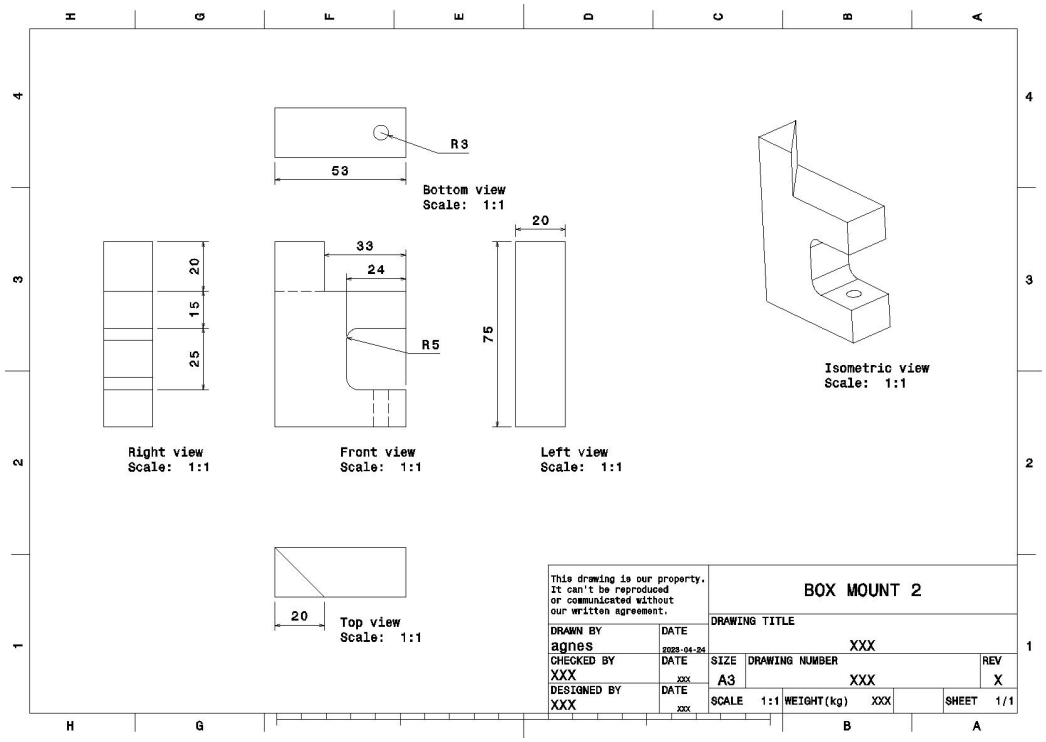


Figure 23: Drawing of box mount 2

# B Appendix - Component List

| Components | Model | Characteristics | Description |
|---|---|---|---|
| Scooter | Mi Electric scooter pro-2[10] | - | - |
| - Wheel motor | BLDC JYX36300 | Poles: 3, Nominal power: 300W, Max power 600W, Nominal voltage: 36V | Brushless DC motor with three poles built in the front wheel. Included in with scooter. |
| - Battery | NED-1004-K | Nominal voltage: 36V, Max voltage: 42V, Capacity: 446Wh | 10-cell Lithium-ion battery. Included with scooter. |
| Steering | - | - | - |
| - Motor | SD-3039[19] | Nominal voltage: 12V, Nominal current: ¡ 530mA | DC motor with built in gearbox and encoder |
| >> Encoder | - | 14 counts per revolution, 2 channels A and B | - |
| >> Gearbox | - | Gear ratio: 100:1, Nominal speed: 56 RPM, Nominal torque: 294 mNm | - |
| - Mount | custom built | - | - |
| >> Plate | - | - | Acrylic plate laser cut after specifications. |
| >> Screw | - | M8, about 20 cm | - |
| >> Holder | - | ABS-plastic | 3D-printed. |
| Control box | - | - | - |
| - PDB | Luxorparts variabel spännings-regulator LM2596S[20] | Input: 3-40 V, Output 1.25-35V | Power distribution board or buck converter. Conerts battery's voltage down to 12 DCV for the MyRio. |

| Components | Model | Characteristics | Description |
|---|---|---|---|
| - MyRio | MyRio - 1900[11] | Input voltage: 8-16V | Reads sensors, performs calculations for the controller and sends instructions to the motors via the servo controllers. |
| - Breakout board | Custom built | 34 pin connectors connected to MyRio, distributes to screw terminals. | Custom built by a previous project at Chalmers. Connects the output of the MyRio to the rest of the control box. |
| - IMU | Pmod NAV 9[12] | Max sample rate: 952 Hz, Accelerometer, Gyroscope, Magnetometer | Main sensor that measures the scooters movement. |
| - Steering motor contoller | Escon 50/5[25] | Power: 250 W, Operating voltage: 10-50 V | PWM servo controller for DC and EC motors. |
| - Forward motor controller | VESC 6 75 V[26] | Operating voltage: 14-63 V, Max voltage: 75 V, Operating current: 60 A, Max Current 120 A | PWM servo controller for BLDC and DC motors |
| - GPS | ZED-F9P GNSS[21] | | - |
| - Router | RUT955 Industrail Cellular Router[22] | - | - |

# C Appendix - Simulations:

These are the parameters of the e-scooter and the "red" bicycle used in the simulations. Simulation results are written in the Result section 4.1.

```
elseif strcmp(bike,'scooter')
    % e-scooter
real parameters and positions on bike
    Parameters.inertia_front = 0.039;  %[kg.m^2] inertia of the front wheel
    Parameters.r_wheel = 0.1079;       % radius of the wheel
    Parameters.h = 0.23;   % height of center of mass [m]
    Parameters.lr = 0.4401;            % distance from rear wheel to frame's center of mass [m]
    Parameters.lf = 0.88-0.4401;       % distance from front wheel to frame's center of mass [m]
    Parameters.c = 0.03;               % length between front wheel contact point and the extention of the fork axis [m]
    Parameters.m = 18.2;               % Bike mas [kg]
    Parameters.lambda = deg2rad(78.7);   % angle of the fork axis [deg]
    Parameters.IMU_height = 0.23;      % IMU height [m]
    Parameters.IMU_x = 0.0;           % x Position of the IMU measured from rear wheel (parallel to bike) [m]
    Parameters.IMU_roll = 0;           % Orientation offset in roll (degrees)
    Parameters.IMU_pitch = 0;          % Orientation offset in pitch (degrees)
    Parameters.IMU_yaw = 0;            % Orientation offset in yaw (degrees)
    Parameters.Xgps = 0.0;             % Actual GPS position offset X (measured from middlepoint position parralel to bike heading) [m]
    Parameters.Ygps = 0.0;             % Actual GPS position offset Y (measured from middlepoint position perpendicular to bike heading) [m]
    Parameters.Hgps = 0.0;             % GPS position height (measured from the groud to the GPS)   [m]
```

Figure 24: Scooter Parameters

```
if strcmp(bike,'red')
    % Red bike
real parameters and positions on bike
    Parameters.inertia_front = 0.245;  %[kg.m^2] inertia of the front wheel
    Parameters.r_wheel = 0.311;        % radius of the wheel
    Parameters.h = 0.2085 + Parameters.r_wheel;   % height of center of mass [m]
    Parameters.lr = 0.4964;            % distance from rear wheel to frame's center of mass [m]
    Parameters.lf = 1.095-0.4964;      % distance from front wheel to frame's center of mass [m]
    Parameters.c = 0.06;               % length between front wheel contact point and the extention of the fork axis [m]
    Parameters.m = 45;                 % Bike mas [kg]
    Parameters.lambda = deg2rad(70);   % angle of the fork axis [deg]
    Parameters.IMU_height = 0.615;     % IMU height [m]
    Parameters.IMU_x = 0.0;           % x Position of the IMU measured from rear wheel (parallel to bike) [m]
    Parameters.IMU_roll = 0;           % Orientation offset in roll (degrees)
    Parameters.IMU_pitch = 0;          % Orientation offset in pitch (degrees)
    Parameters.IMU_yaw = 0;            % Orientation offset in yaw (degrees)
    Parameters.Xgps = 0.0;             % Actual GPS position offset X (measured from middlepoint position parralel to bike heading) [m]
    Parameters.Ygps = 0.0;             % Actual GPS position offset Y (measured from middlepoint position perpendicular to bike heading) [m]
    Parameters.Hgps = 0.0;             % GPS position height (measured from the groud to the GPS)   [m]
```

Figure 25: Bicycle Parameters

The IMU_x, _roll etc, are not necessary in this early stage of development. This applies both for the e-scooter and the bike. As the simulation program used is from the bicycle group the parameters for bicycles were already written, meanwhile the scooter parameters were measured as discussed in Mathematical Model 2.1

## C.1 Altering Velocity $v$

Simulating the e-scooter and bicycle over varying velocities. These simulations were made with the "circle" path-trajectory as it is a test that could be done with physical e-scooter too (given a new referenced, constant lean angle not equal to 0). Controller values:

**Outer loop – Roll Tracking**

P balancing outer = 1.3

I balancing outer = 0.0

D balancing outer = 0.0

**Inner loop – Balancing**

P balancing inner = 3;

I balancing inner = 0;

D balancing inner = 0;

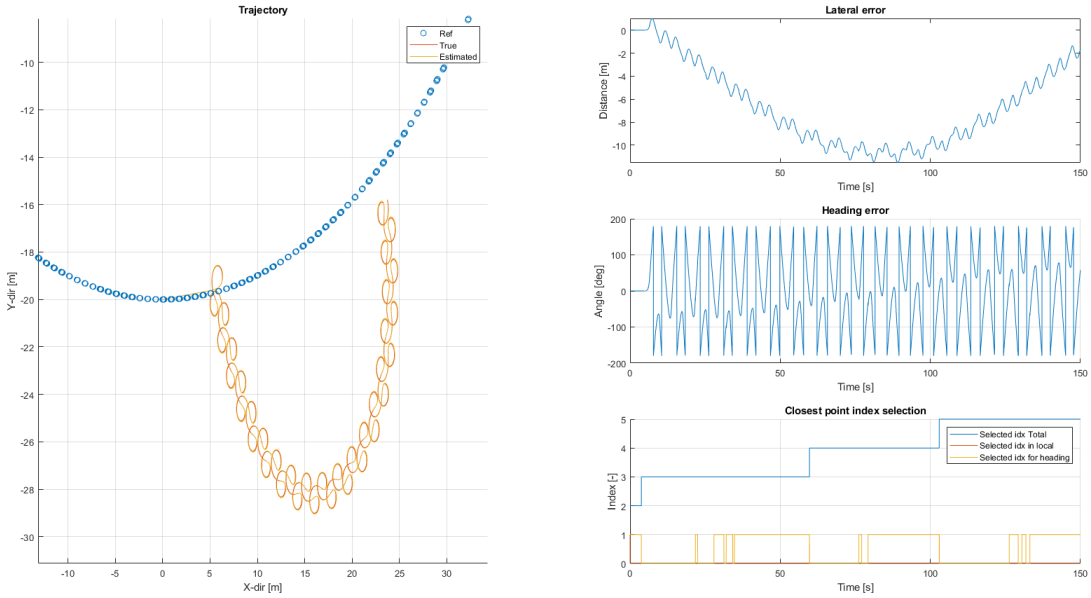


Figure 26: The scooter with speed 1.0 m/s

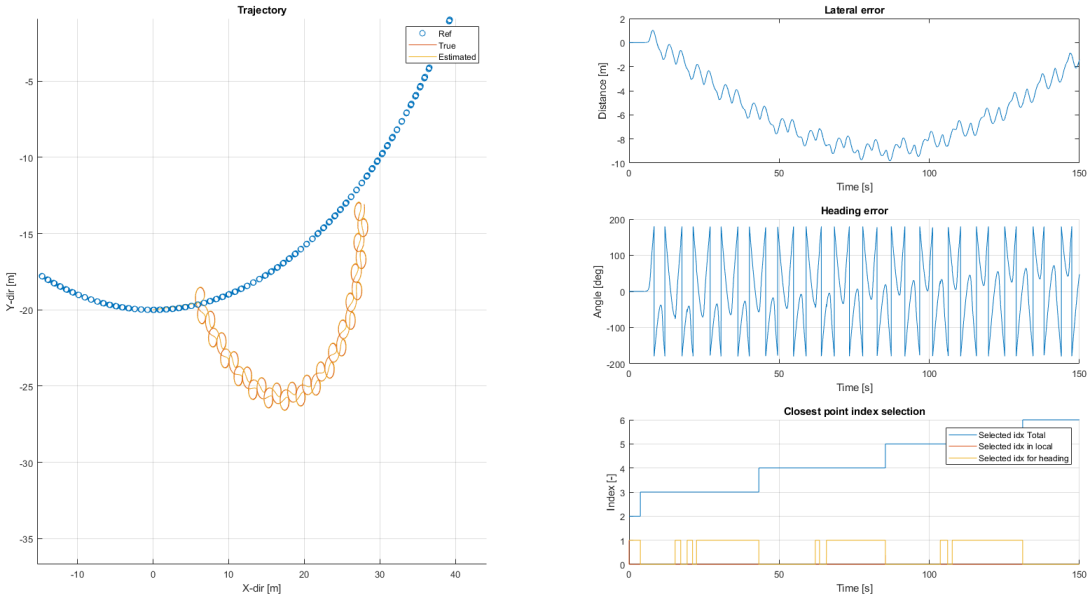


Figure 27: The bike with speed 1.0 m/s

The trajectory looks very similar between the scooter and the bicycle as this is whenever the vehicle makes a “simulated fall”. The differences can be seen clearest in ”Closest point index selection” window on the bottom right.

Increasing the speed until stable, results in the following graphs:

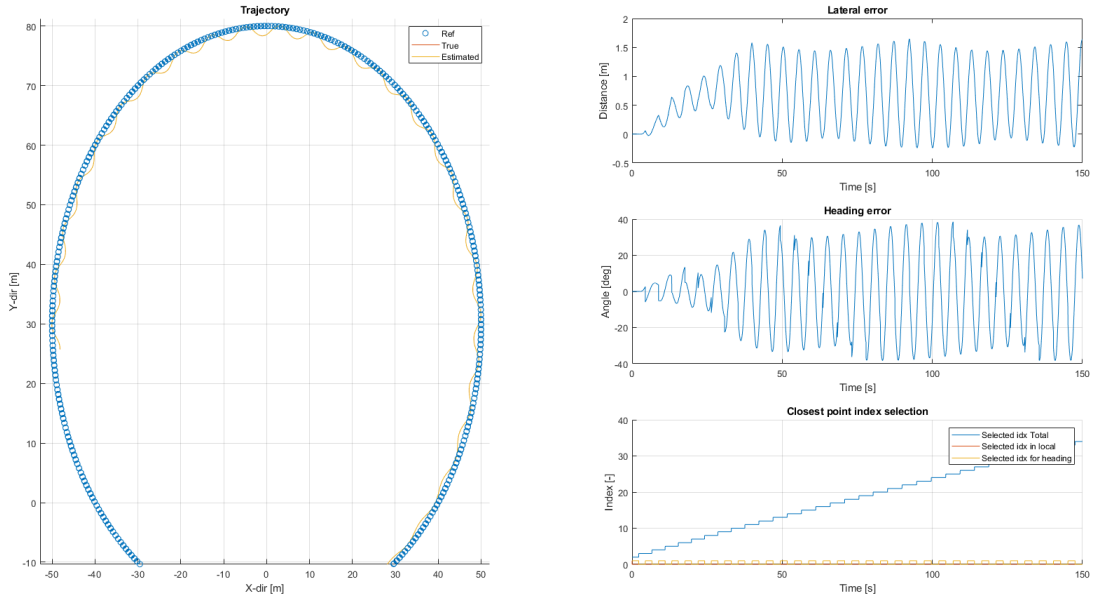


Figure 28: The scooter with speed 1.7 m/s

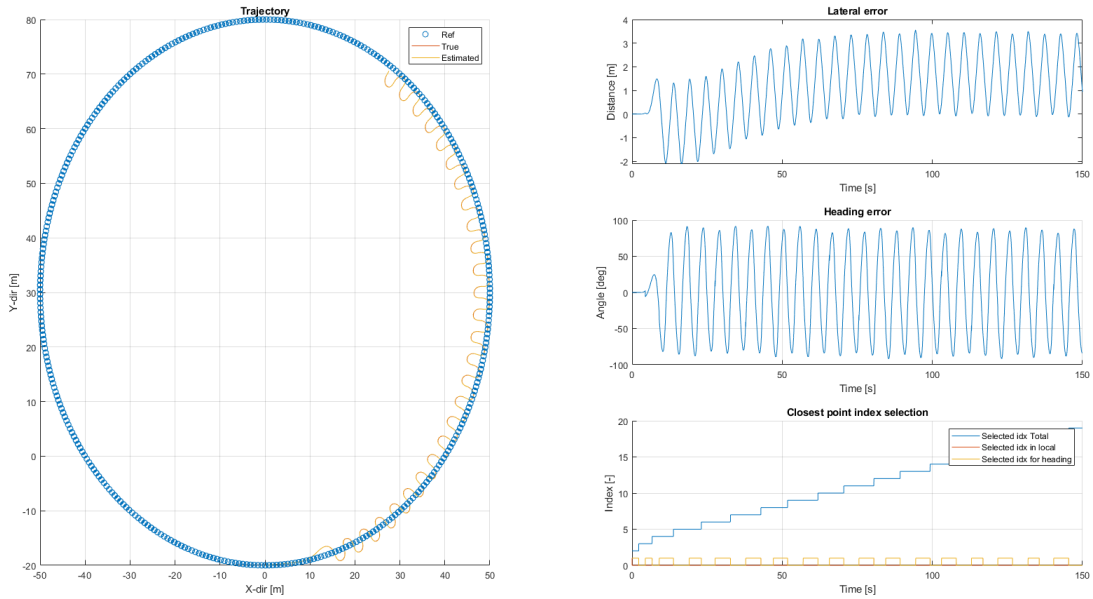


Figure 29: The bike with speed 1.7 m/s

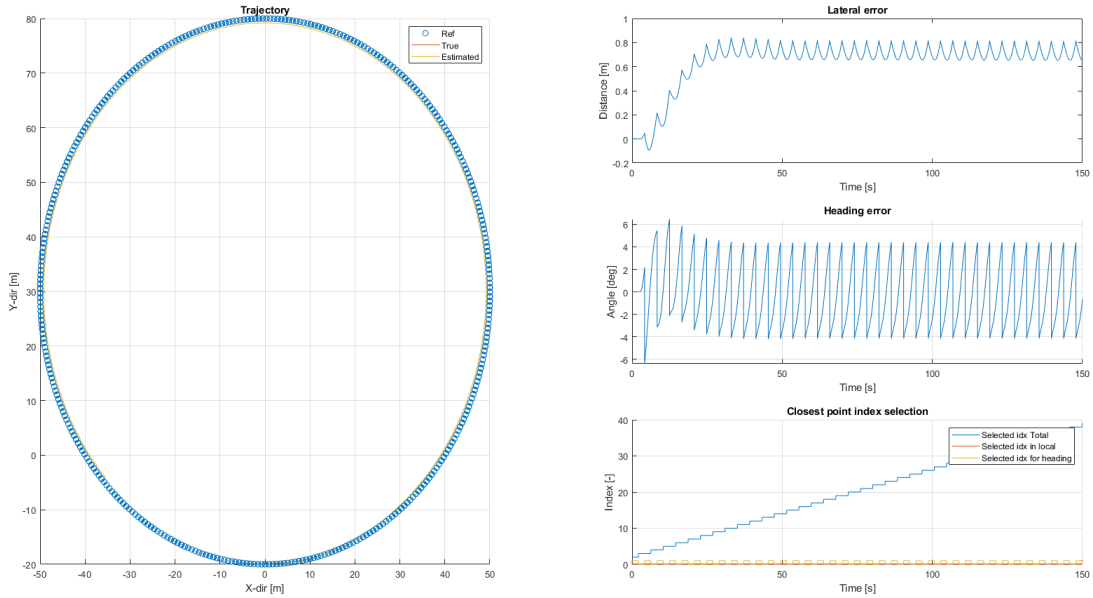


Figure 30: The scooter with speed 1.8 m/s

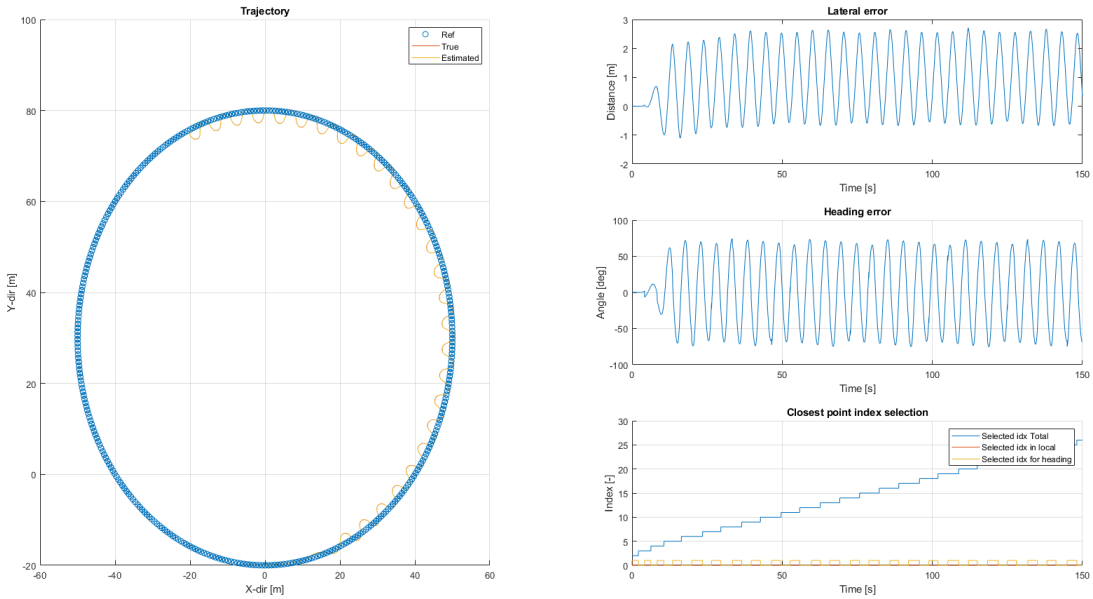


Figure 31: The bike with speed 1.8 m/s

The scooter becomes stable when v $>$ 1.8 m/s. However, the bicycle is still unstable.

The reason for this is because of the difference in wheel base as discussed in section 2.1.2.

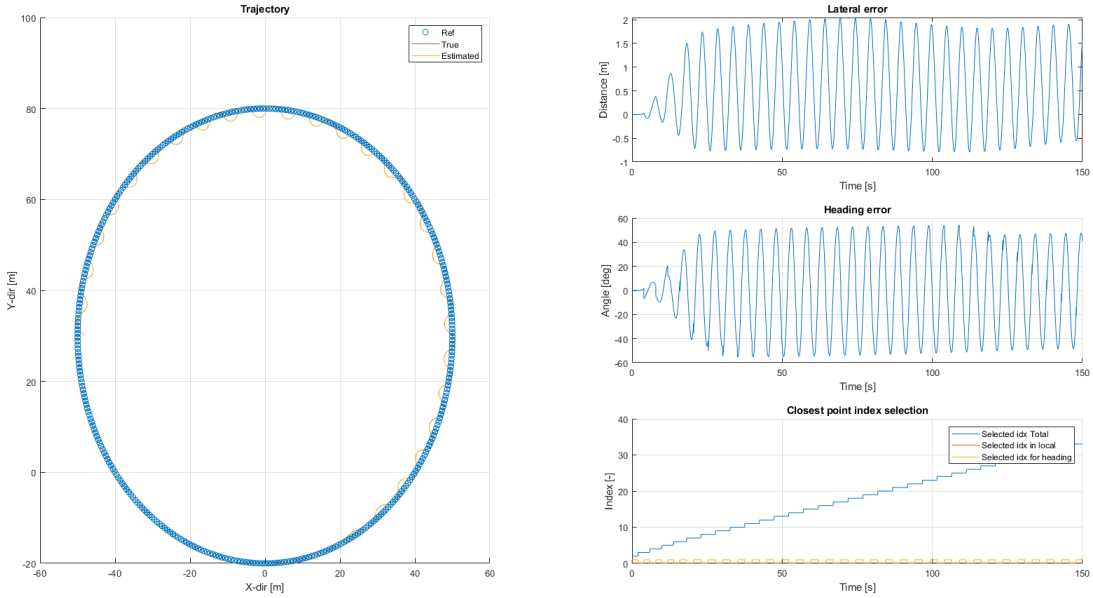


Figure 32: The bike with speed 1.9 m/s

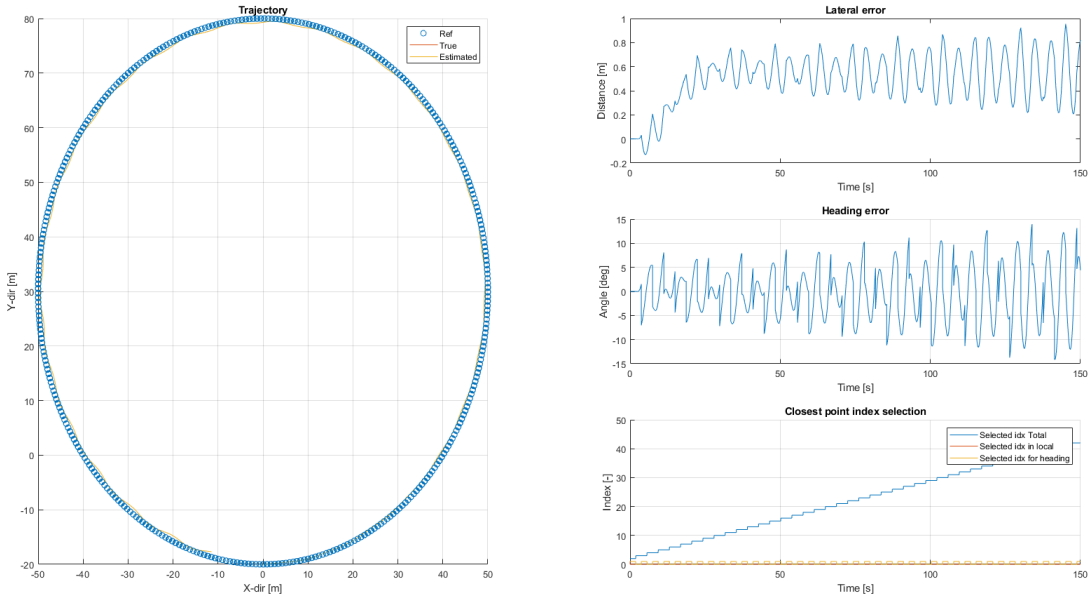


Figure 33: The bike with speed 2.0 m/s

Here, compared to the e-scooter, the bicycle becomes stable at v $>$ 2.0 m/s, a consequence of the larger wheelbase.

If the velocity is too high, the vehicles cannot follow the trajectory curves.

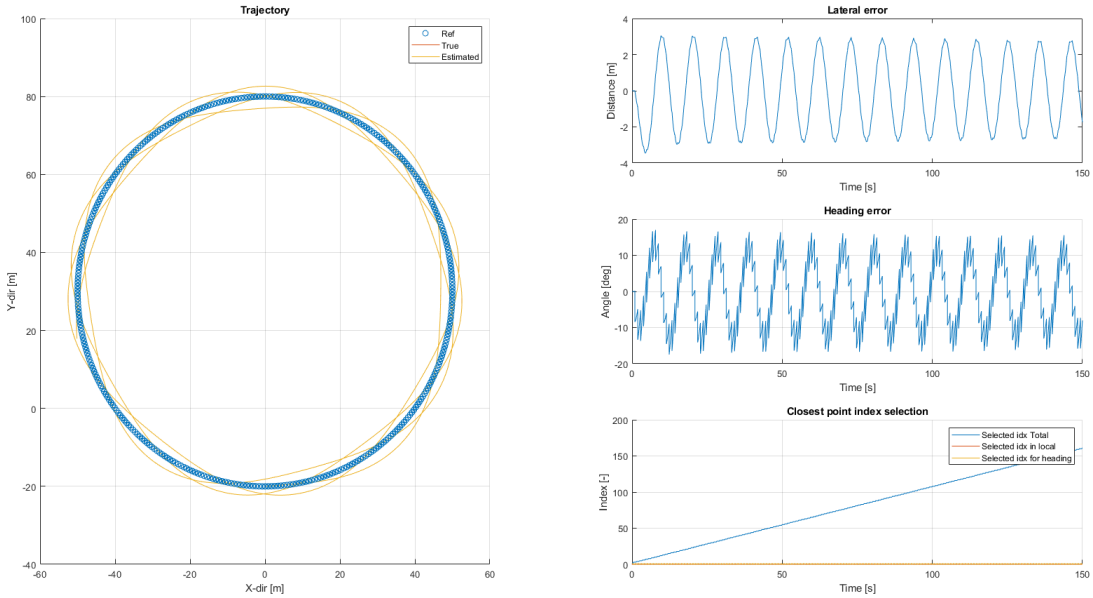


Figure 34: The scooter with speed 8.0 m/s

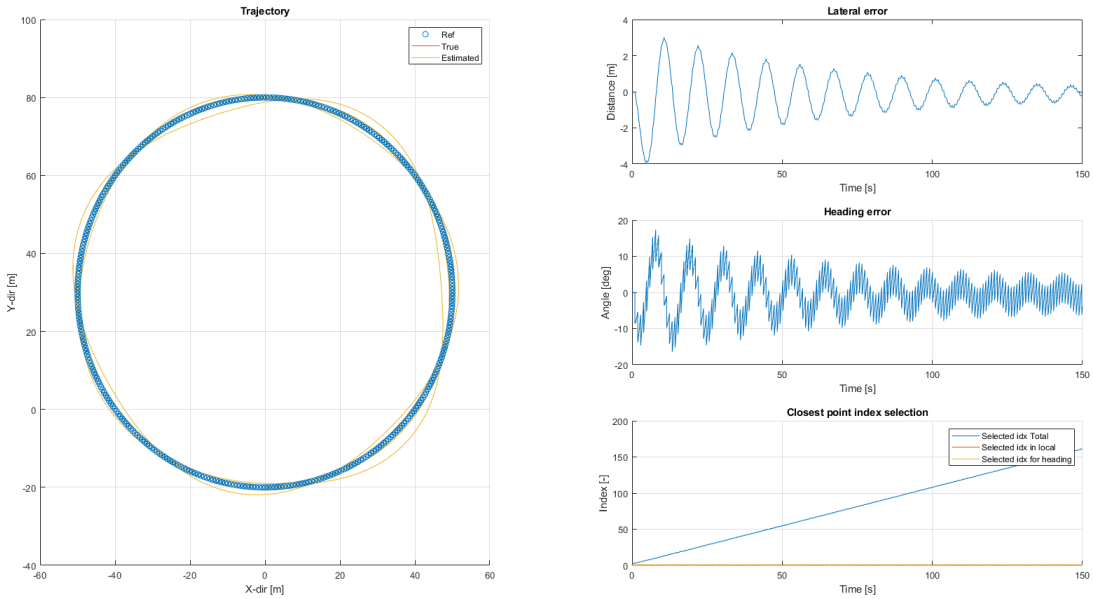


Figure 35: The bike with speed 8.0 m/s

Depending on the test and trajectory the max speed will be different.

## C.2 Altering Controller Values

Simulating the e-scooter's behavior when altering the controller values. What is of interest is the range of controller values where the scooter works properly. The velocity will be held at a consistent realistic speed of 2.5 m/s and the test is done in "ascending sin" path trajectory.

### C.2.1 Altering the D value.

Altering the D value of the controller, keeping P at a constant 1.3:
First, testing increasing D constants:

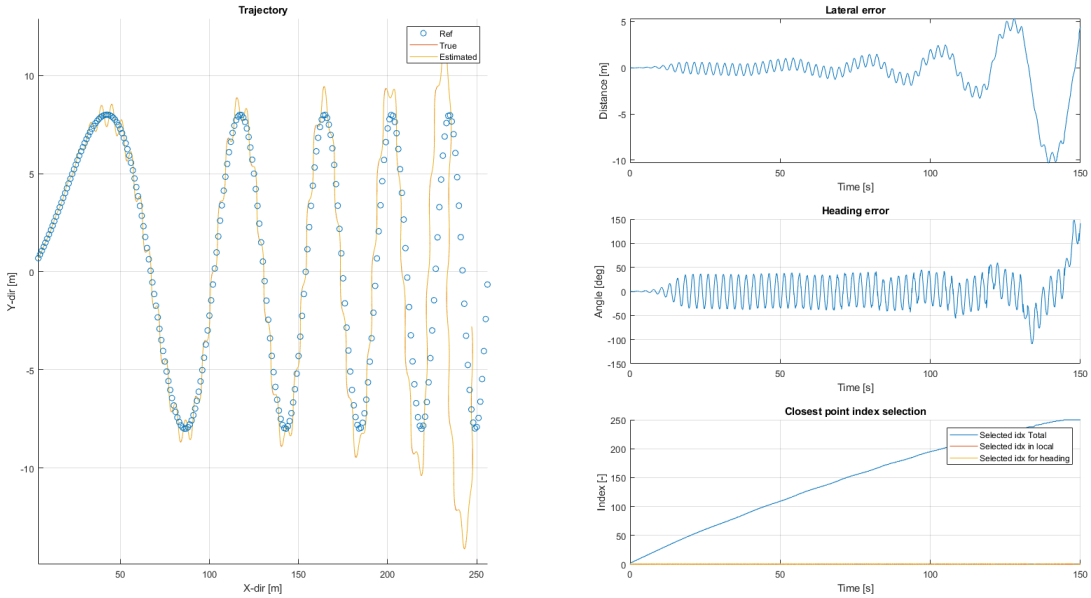


Figure 36: D = 1.2

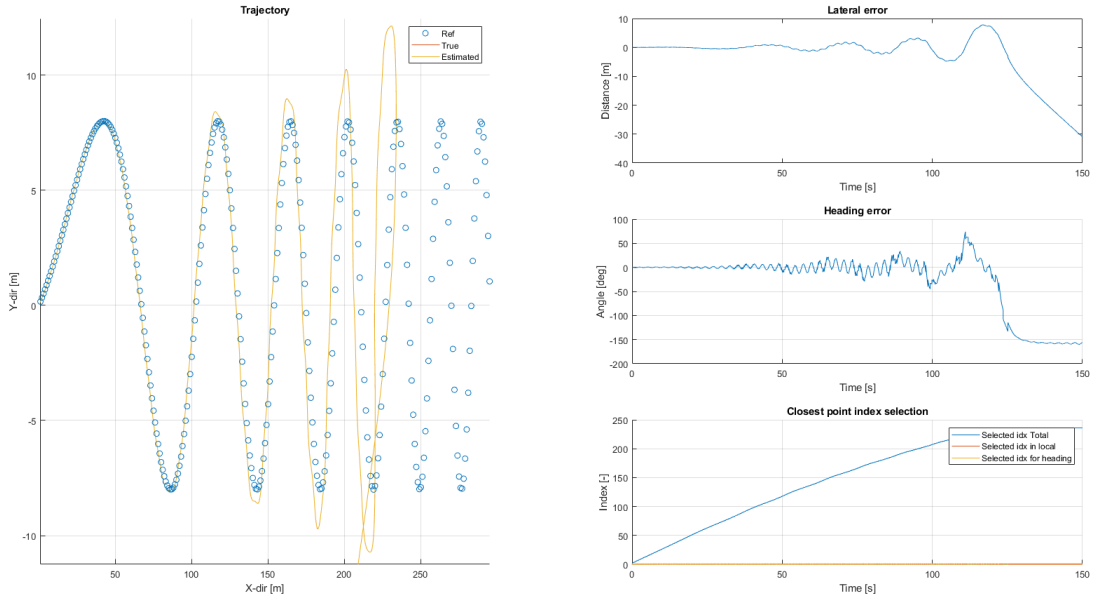


Figure 37: D = 1.3

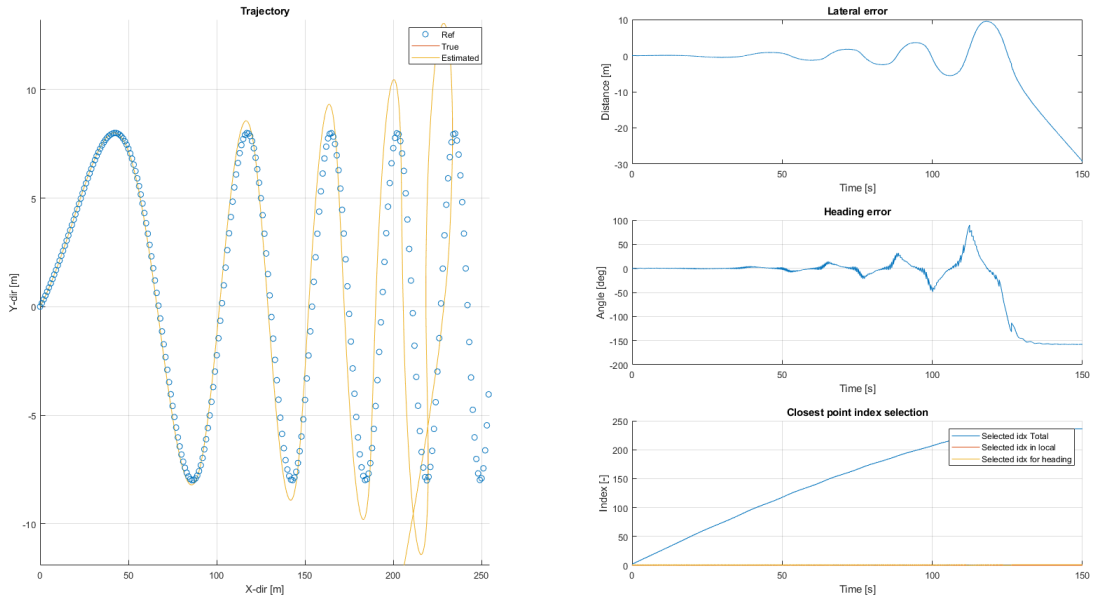


Figure 38: D = 1.4

The balance controller should at least have a D > 1.4, as seen in the results above. Even then, the roll angle shows that 1.4 is a bit low:

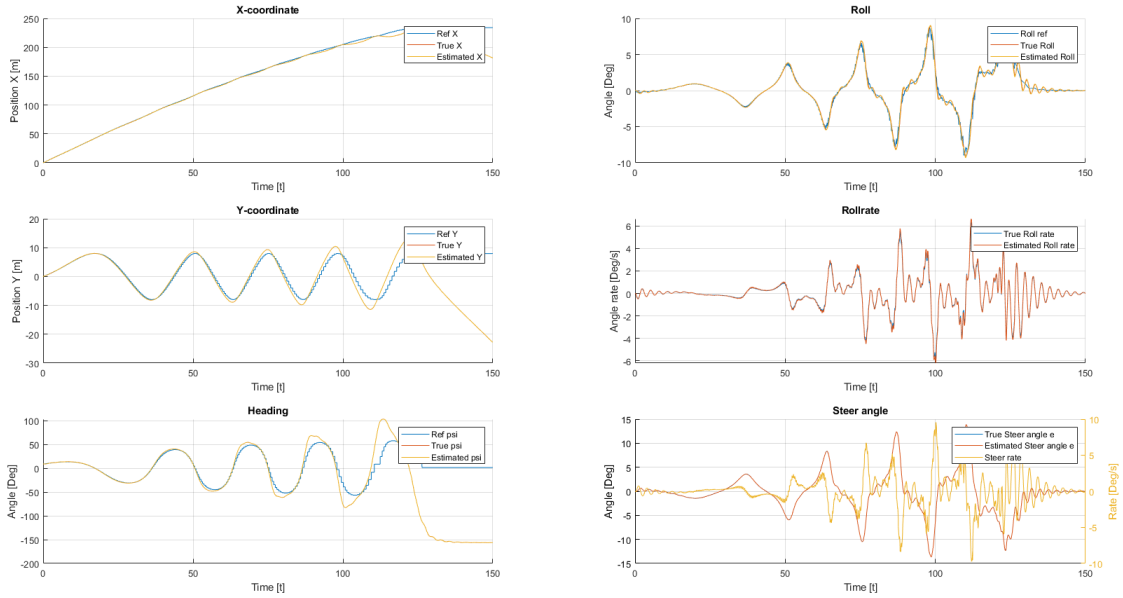


Figure 39: The errors of the coordinate's and the display of angles.

Increasing the D value of the balance controller impacts the rate of which the scooter can react, making it faster. In real life there are outer factors making it necessary for quicker reactions. Hence, the D value could be higher to make it faster if necessary.

Increasing the D values too high will result in too much of a reaction from altering paths, making this particular test fail.

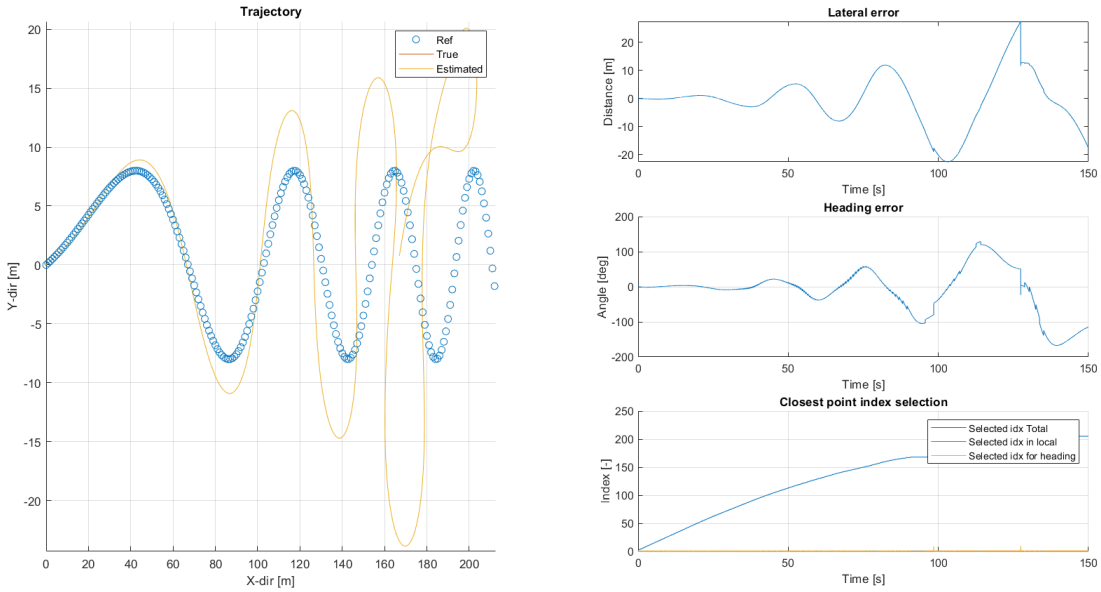


Figure 40: D = 4.5

### C.2.2 Altering the P value

Altering the P value of the controller, keeping the D at a constant value 3 results in the following graphs:

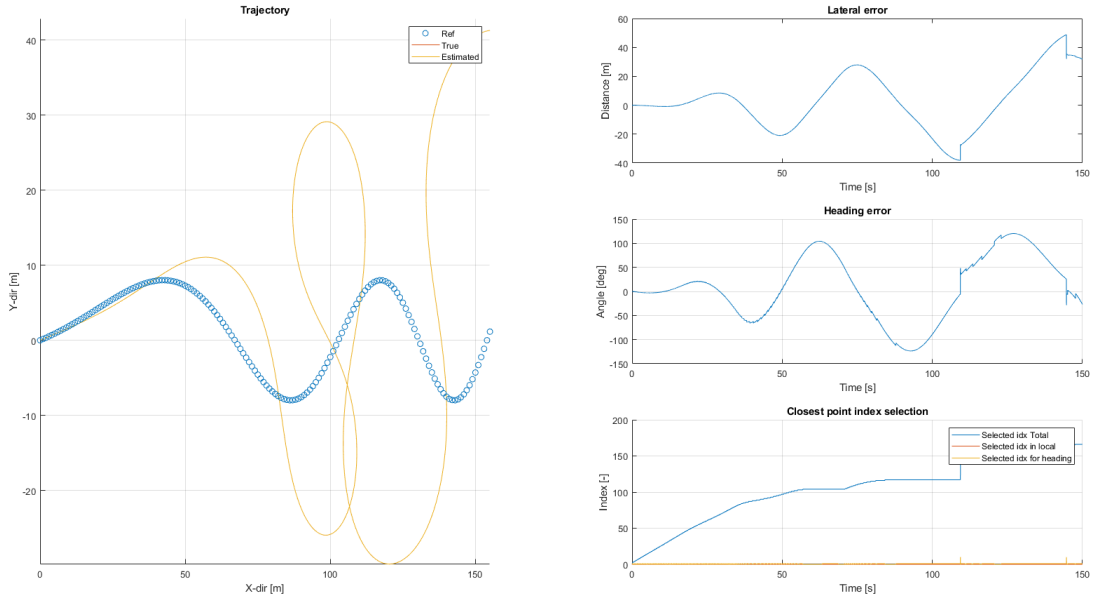


Figure 41: P = 0.1

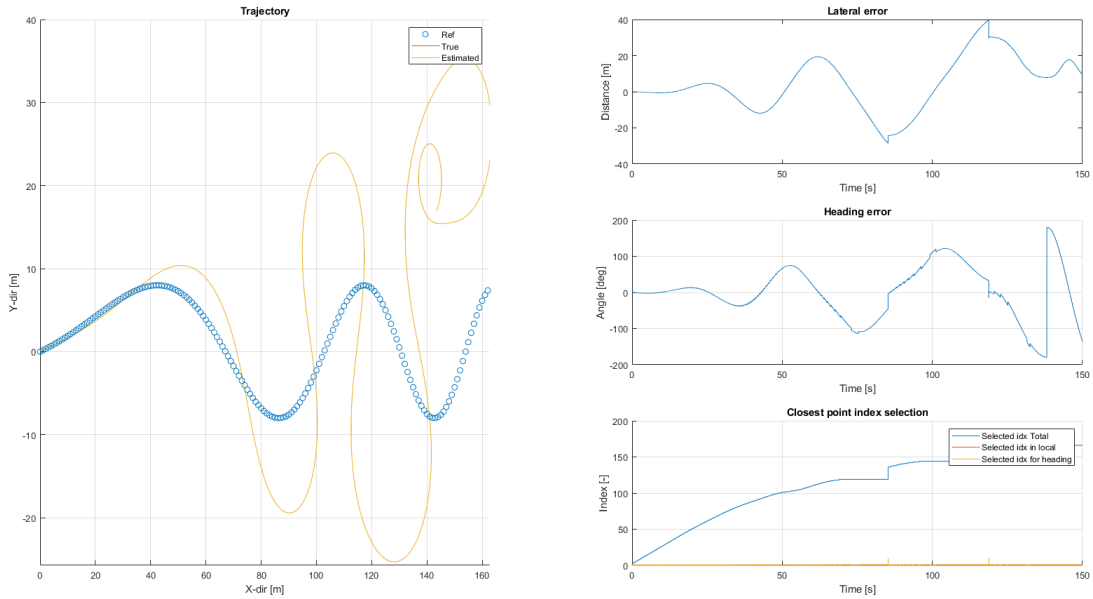


Figure 42: P = 0.2

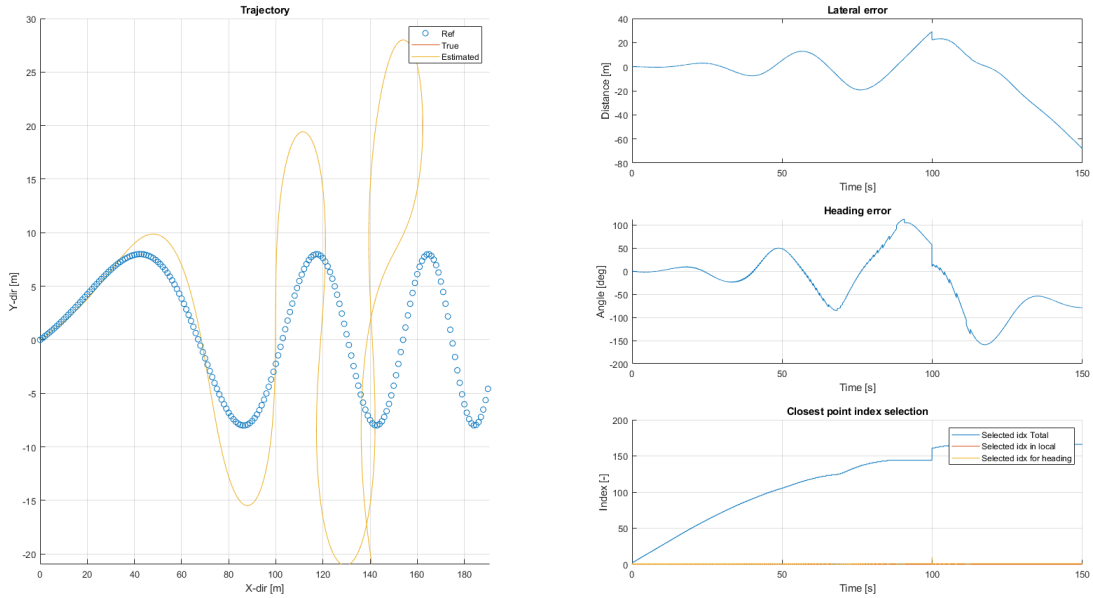


Figure 43: P = 0.3

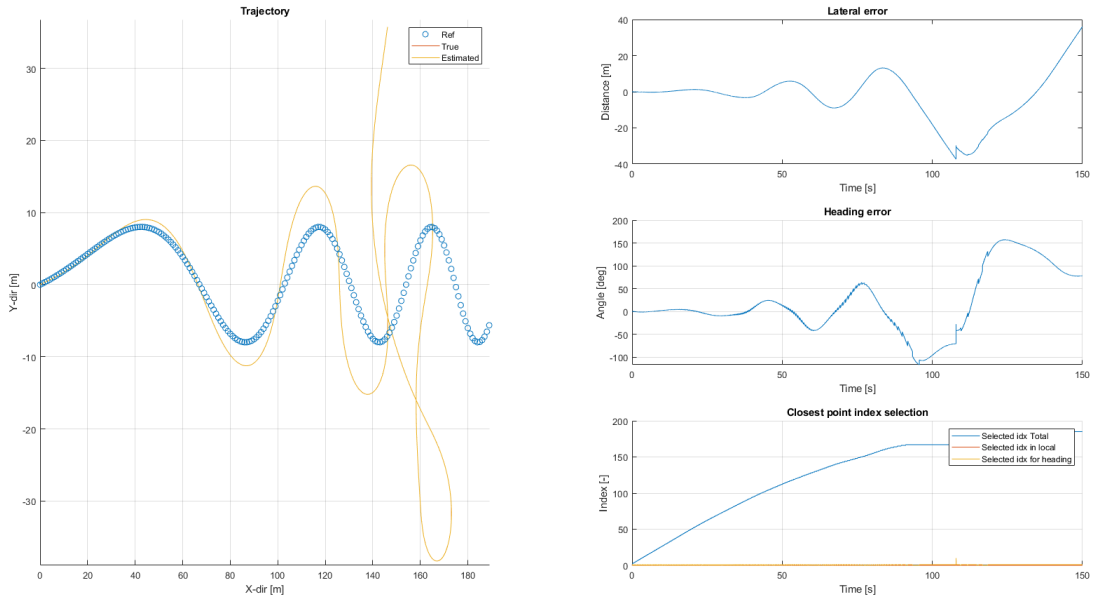


Figure 44: P = 0.6

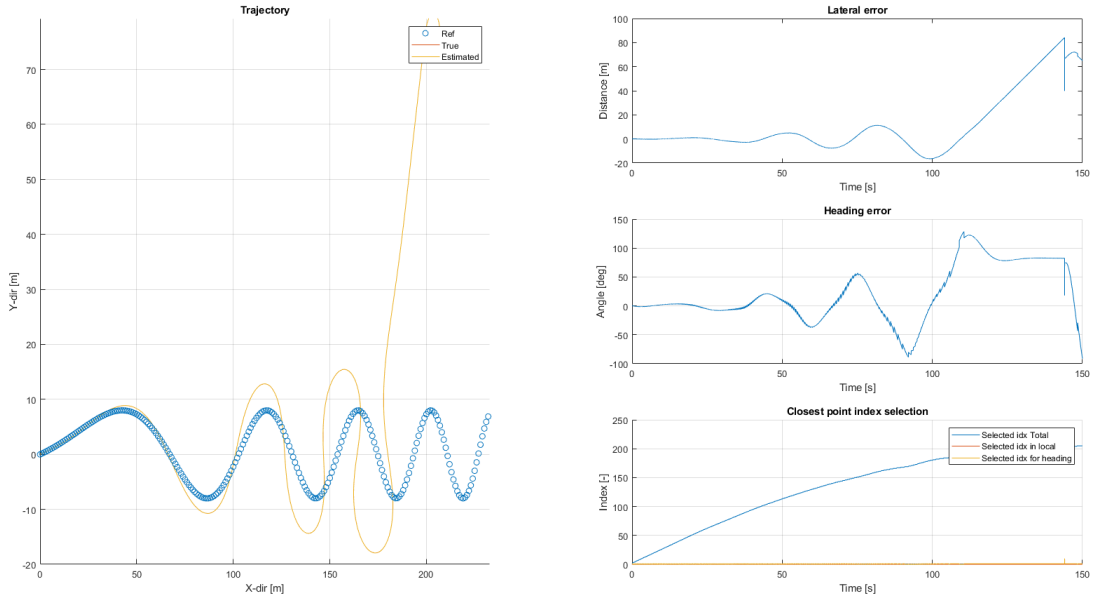


Figure 45: P = 0.7

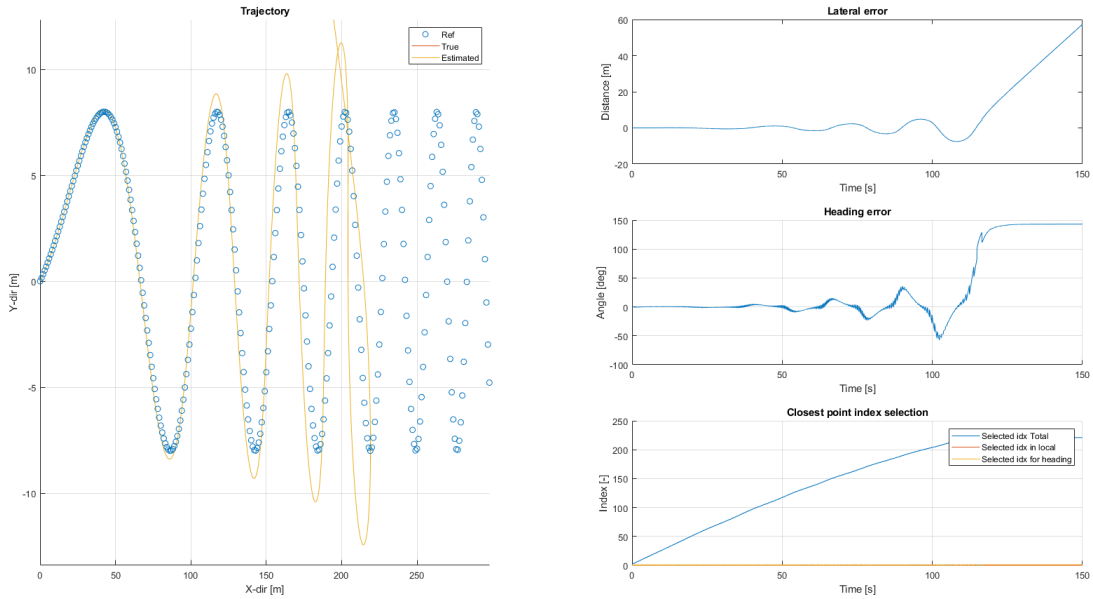


Figure 46: P = 10

Increasing the P value makes the simulation react better. D = 3 means that this is quite a reasonable controller for the scooter even if 10 is an unreasonably high P value. Anything over 1 should be good, with a fitting D value to it. Considering the bike has values P = 1.3, D = 3, the e-scooter will also have similar values.

## C.3 Altering the Scooter CoM

For simulating the center of mass, the "ascending_sine" test was chosen, like with the controller values. Running at the same speed of 2.5 m/s, the different models of altered CoM get put through different controller values to see how they impact the controller.

The different alterations to the scooter are as follows:

- The base scooter for reference with the same parameters.
- One alteration with 10 kg suspended 30 cm above the scooter bases's CoM.
- One alteration with 15 kg suspended to the same height.
- Another with 15 kg but suspended 50 cm above the scooter base's CoM.

All of the alterations keep the center off mass on the same $x_{pos}$. By keeping the same center of mass position, the distance from the rear wheel to the CoM, $a$, will remain the same, and by putting these values in the formula 13 gives us the following calculation. This example is calculated for 10 kg, 30 cm above scooter's center of mass:

$$(\mathrm{a}, \mathrm{h}_{CoM}) = \sum_{i=1}^{5} \frac{m_i(x_i, y_i)}{m_{scooter} + m_{weight}} = \frac{10(0.44; 0.15 + 0.3) + 12.5(0.41; 0, 15) + 0.9(0.7516; 0.7632) + 0.7(0}{18 + 10} \tag{16}$$

$$(\mathrm{a}, \mathrm{h}_{CoM}) = \sum_{i=1}^{5} \frac{m_i(x_i, y_i)}{m_{scooter} + m_{weight}} =$$

$$= \frac{10(0.44; 0.15 + 0.3) + 12.5(0.41; 0, 15) + 0.9(0.7516; 0.7632) + 0.7(0.6832; 1.1065)3.7(0.42; 0.23)}{18 + 10} =$$

$$= (0.44; 0.31)$$

This is done for all the alterations providing these results:

(0.44 ; 0.34), $\mathrm{m}_{tot}$ = 32 kg, 30 cm above CoM.

(0.44 ; 0.44), $\mathrm{m}_{tot}$ = 32 kg, 50 cm above CoM.

### C.3.1 Controller Values:

Keeping the P of a constant 1.3 changing the D value and analyzing the difference between alterations and setups for the scooter.

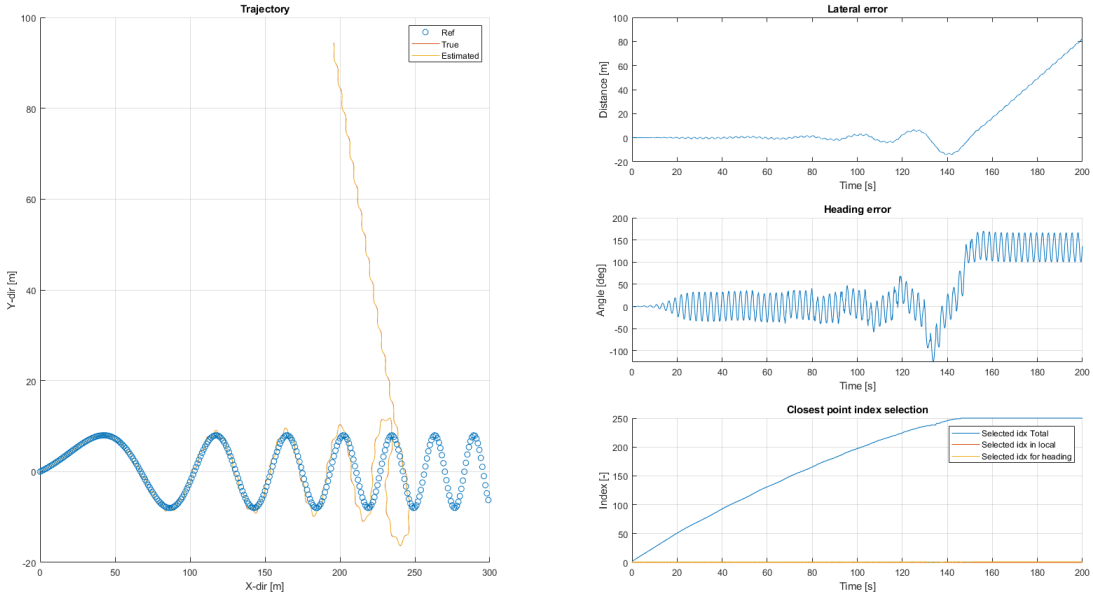


Figure 47: 10 kg elevated 30 cm, D = 1.3

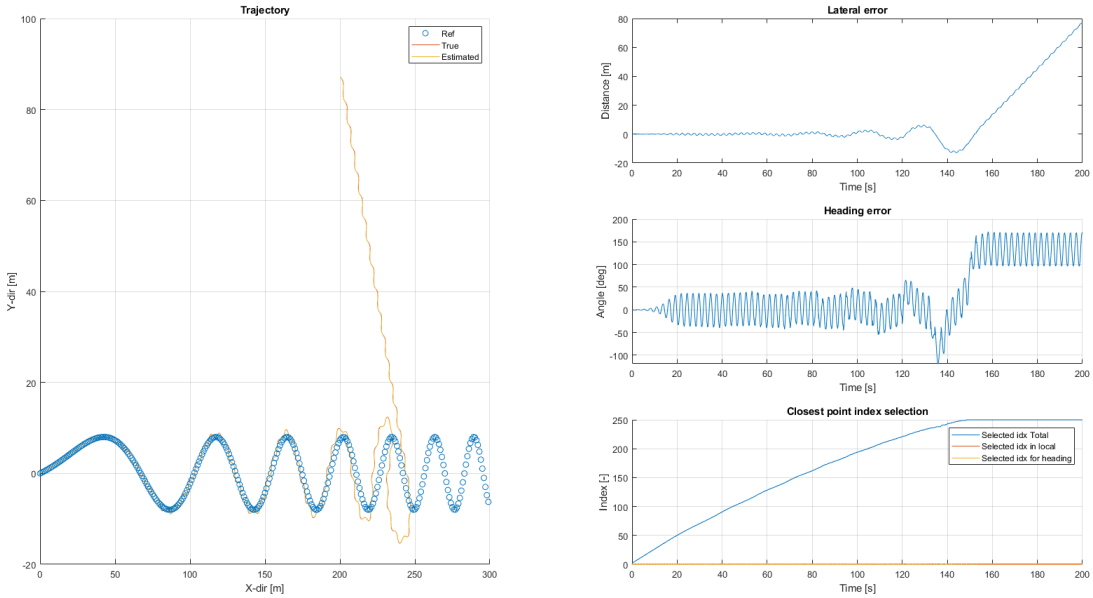


Figure 48: 15 kg elevated 30 cm, D = 1.3

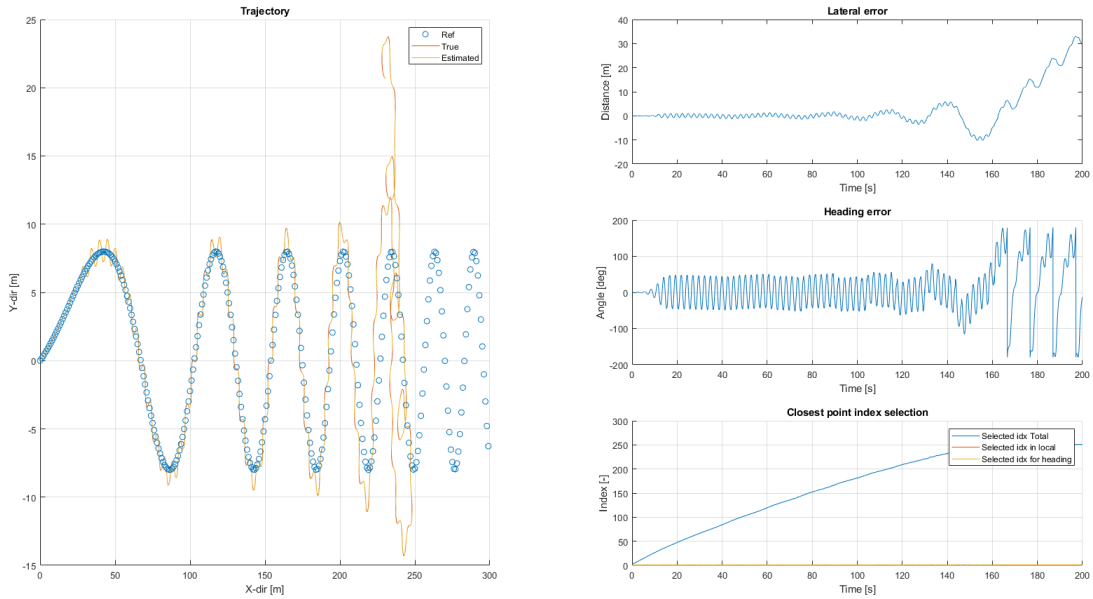


Figure 49: 10 kg elevated 50 cm, D = 1.3

With the increased mass and higher CoM there's also a higher oscillation. These values for D are not suitable for balancing as they're too low, but the importance of the controller is showcased when the different scooter alterations are used with them. As can be seen, unstable systems with a larger mass and a higher CoM act less stable and, hence, require a controller model that truly suits them. Once the controller is good enough, the alterations act the same way as the normal scooter.

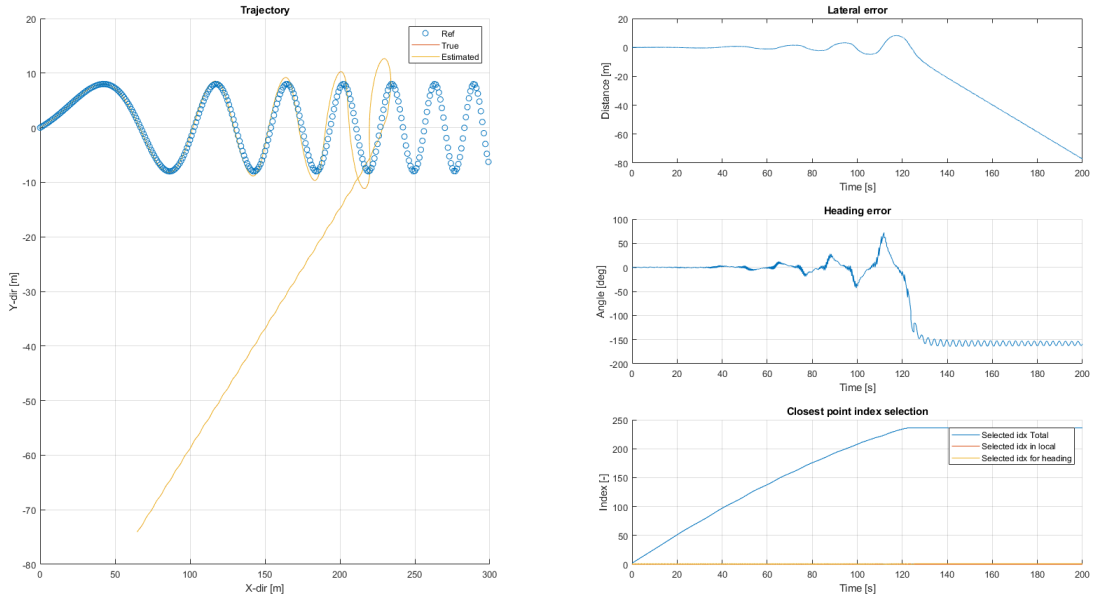


Figure 50: 10 kg elevated 30 cm, D = 1.4

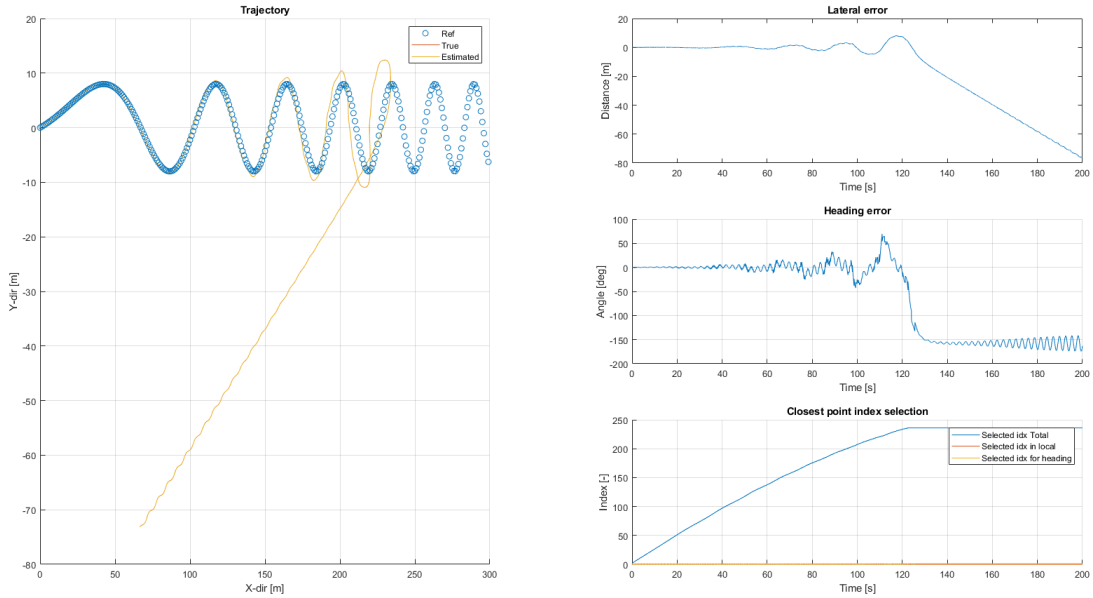


Figure 51: 15 kg elevated 30 cm, D = 1.4

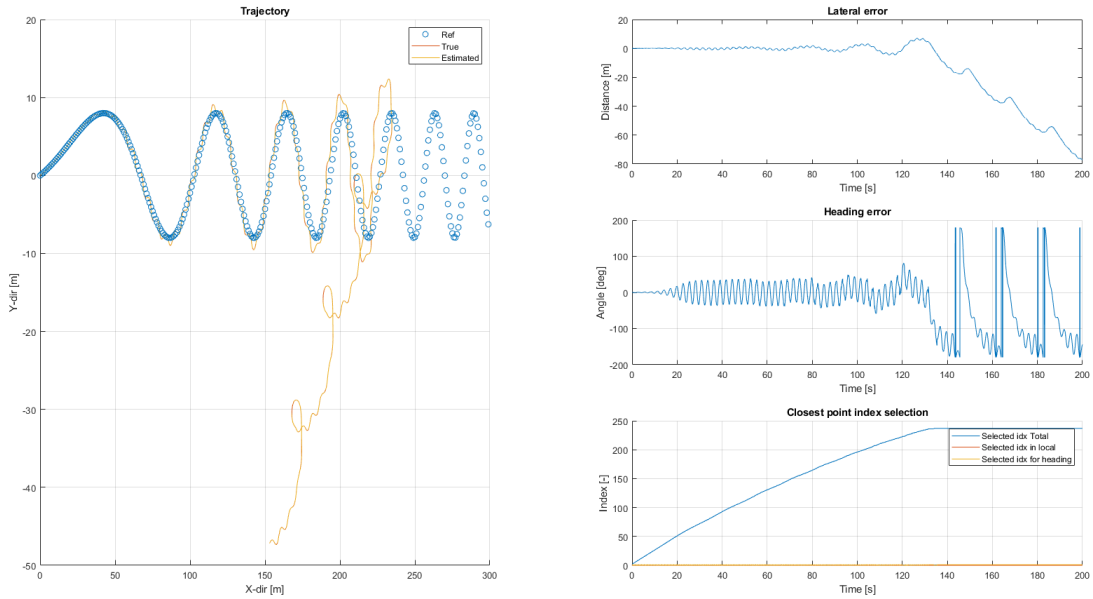


Figure 52: 10 kg elevated 50 cm, D = 1.4

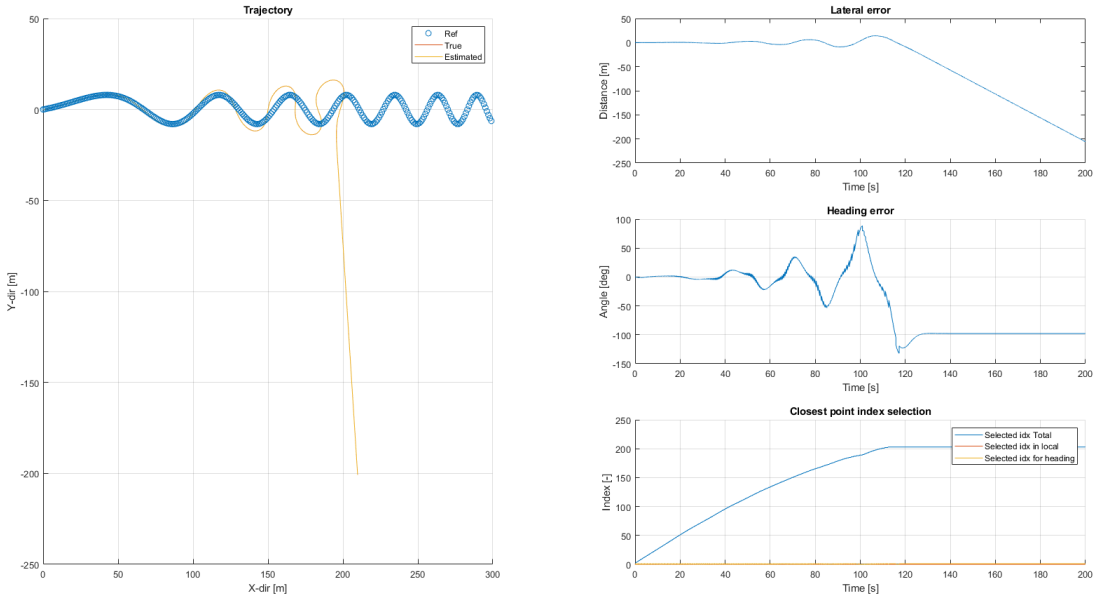


Figure 53: 10 kg elevated 30 cm, D = 3

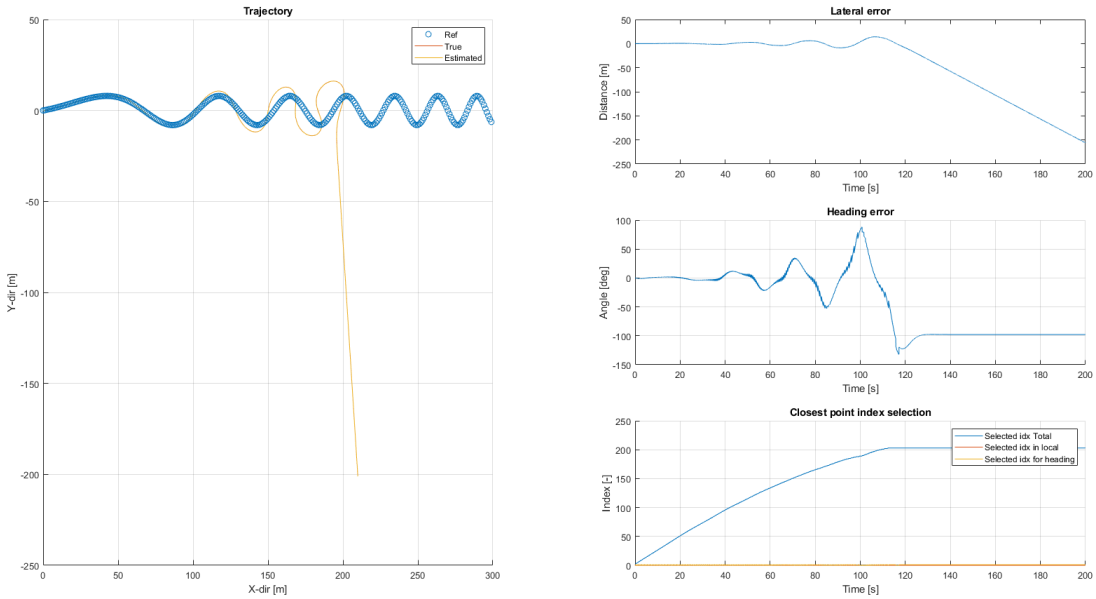


Figure 54: 15 kg elevated 30 cm, D = 3

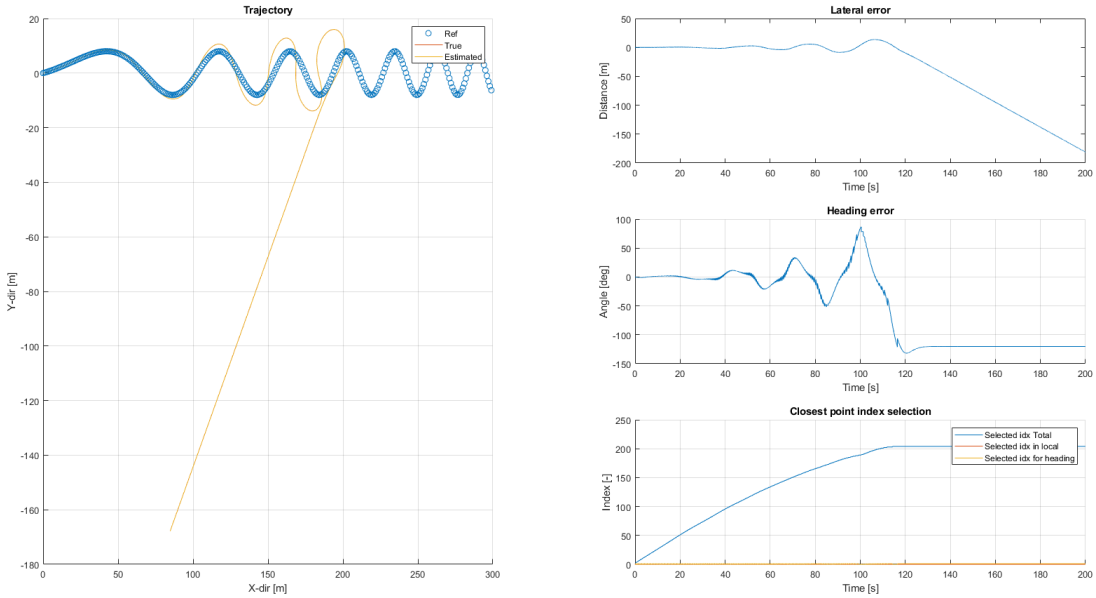


Figure 55: 10 kg elevated 50 cm, D = 3

Of course, there are differences, but once the controller is good the difference is slight.

Increasing the P value to 2 has the following effect:

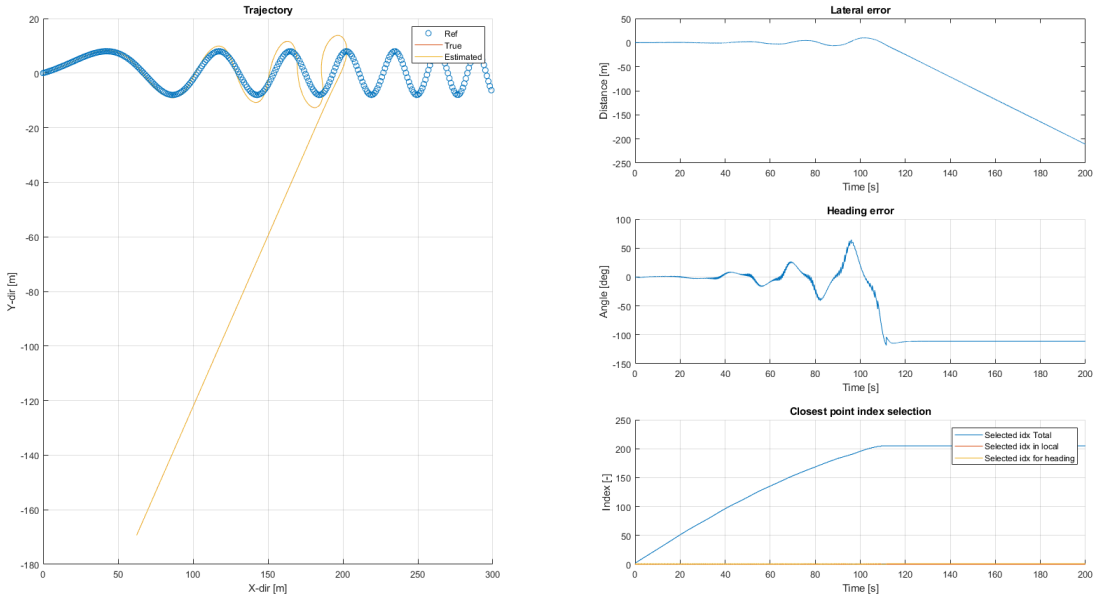


Figure 56: 10 kg elevated 30 cm, D = 3, P = 2

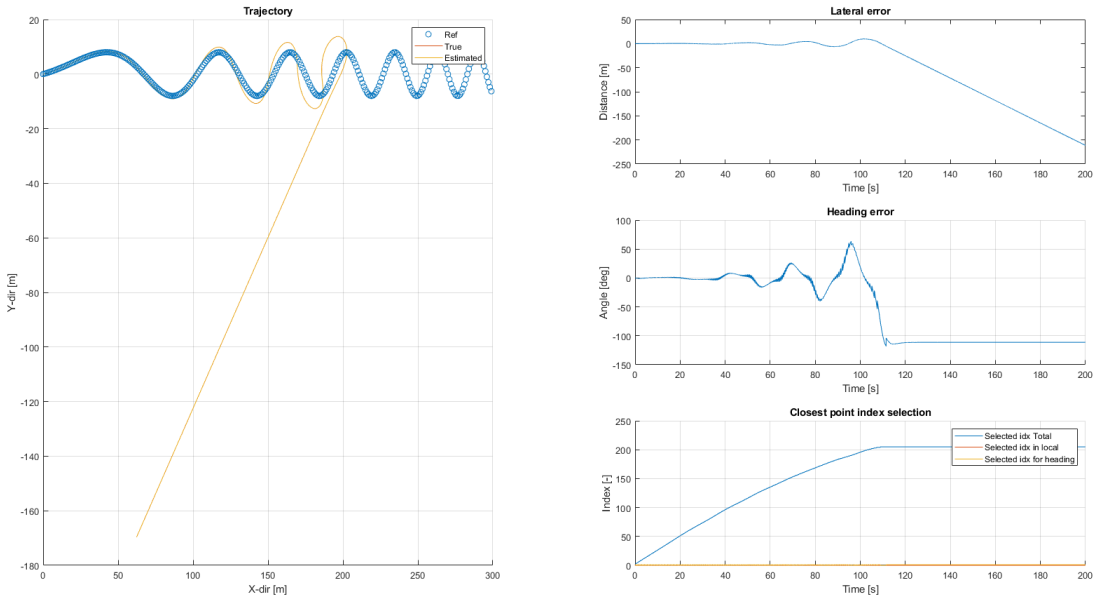


Figure 57: 15 kg elevated 30 cm, D = 3

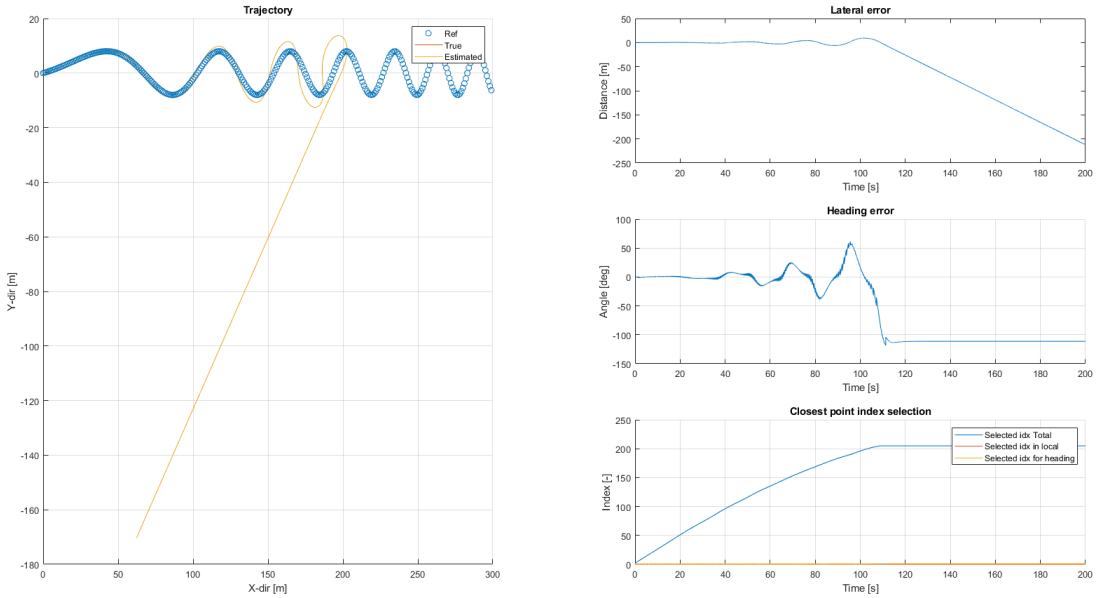


Figure 58: 10 kg elevated 50 cm, D = 3, P = 2

Once again, suitable controller values don't change much, and the result is aligned with the results from increasing the P values when analyzing the controller values.

## C.4 Simulation for the Testing

When suitable controller values and a suitable speed was achieved, the exact setup was simulated that the test will utilize as well. The circle trajectory is a great test to see whether the scooter can balance and is possible to reenact with the scooters current setup.

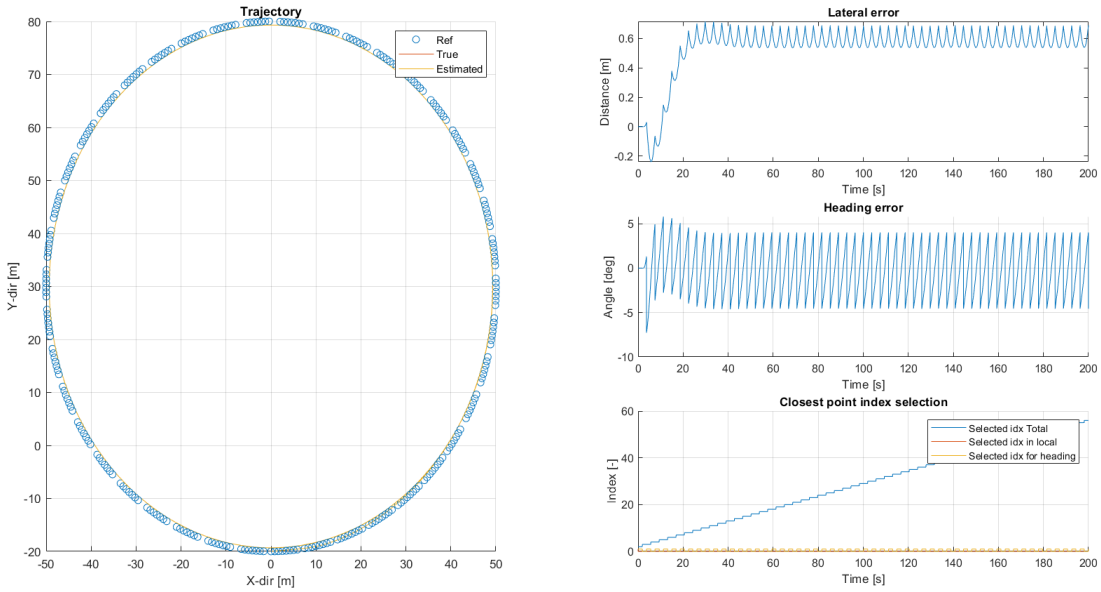


Figure 59

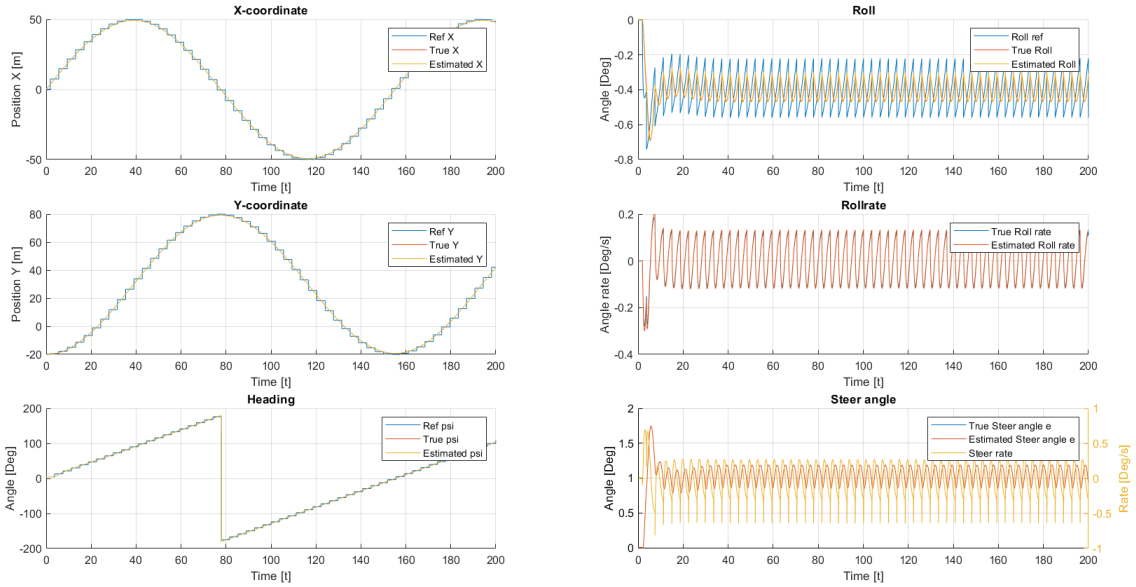


Figure 60

# D Appendix - LabVIEW:

Here, the overview, functionality and explanation of every subVI in main.VI is given, starting with Emergency Stop.

## D.1 Emergency Stop:

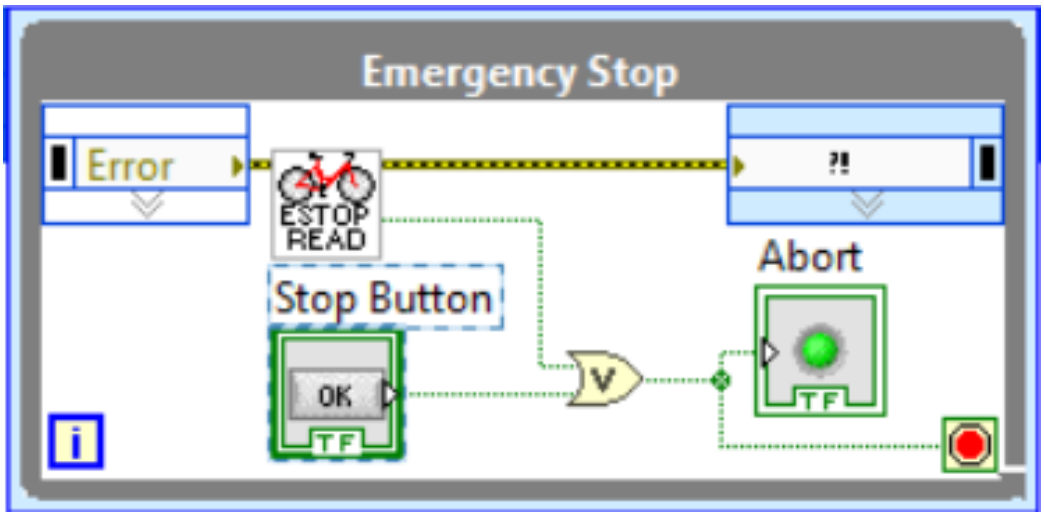


Figure 61: An overview of Emergency stop subVI in main.VI

As can be seen in Figure 61, the working procedure of emergency stop is very simple. A boolean control which is a Stop button is created and connected to the aborting system of main.VI. At the same time, any error in the system will also shut down the program by stopping the loop. These two ways of turning off the program are created in emergency stop section.

## D.2 IMU:

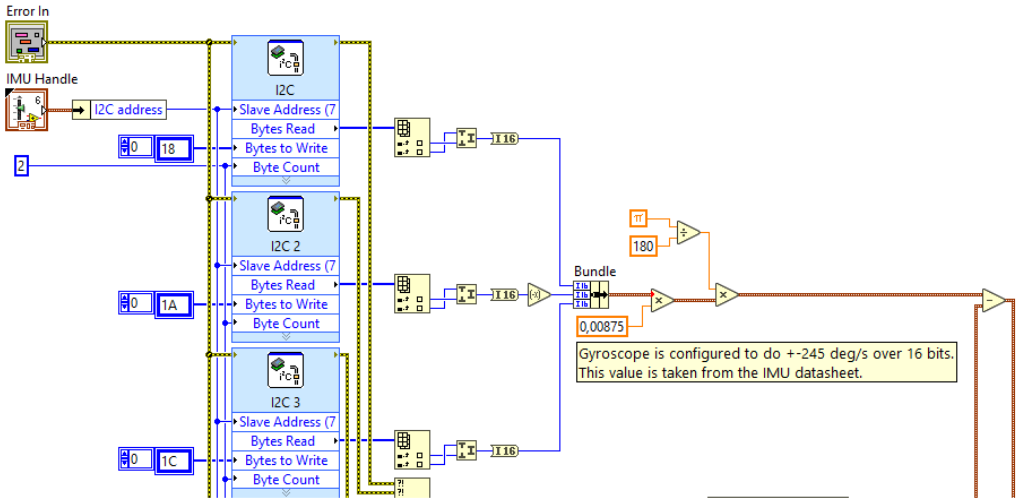


Figure 62: IMU.VI showing how the IMU is connected to LabVIEW

Figure 62 above is showing how the gyroscope data is taken from the IMU and given to the LabVIEW program. As can be seen, three I2C ports are used. The I2C communication protocol is often used in electronic devices, including IMUs, to link components. Here, three I2Cs are used for the x,y and z directions of the gyroscope, and according to the IMU data sheet and in order to convert the values to radians per second, some mathematical operations are done on the data bundle. The same procedure is done for the accelerometer of the IMU.

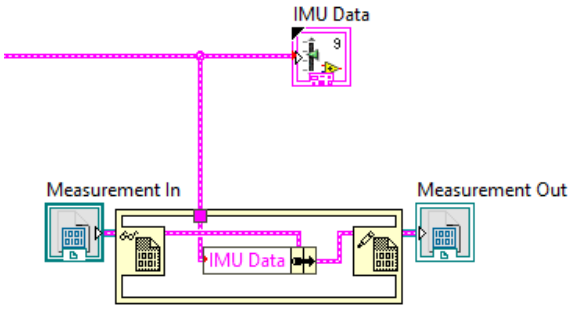


Figure 63

Figure 63 show that the gyroscope and accelerometer data are displayed in an indicator bundle, called IMU Data, and the same data is written to the measurement file.

## D.3 Steering Motor Encoder:

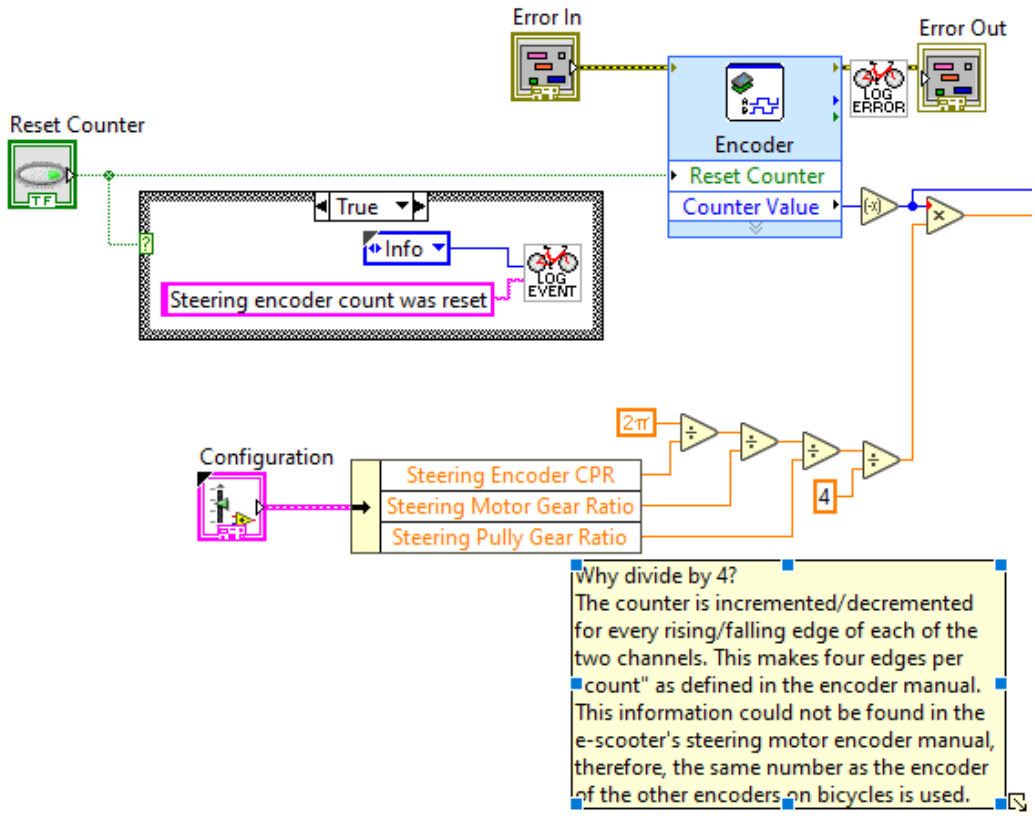


Figure 64

From Figure 64, it is visible that based on the configurations of the encoder, gear box and the steering pulley, an angle in radians is created which is then multiplied by the counter value coming from the encoder. The steering motor encoder's counter value can be reset using a boolean control, called Reset steering motor Encoder, found in the front panel of main.VI. What that does is simply reseting the encoder counter value to zero, which makes the steering angle value also zero.

In the end, the values are sent to an indicator bundle to visualize the encoder data on the main.VI's front panel, as well as writing the data in the measurement file. As this final procedure is repeated in every subVI, adding a picture for visualization could be unnecessary. Figure 62 can be seen as reference.

## D.4 GNSS:

The correct GPS configuration such as baud rate first needs to be written to the GNSS Handle which then gives the configurations and data out to the GNSS read. This process is done in GNSS OPEN.VI. Afterwards, these data is collected into an array, being separated for the required mathematical operations to be done on each of them.

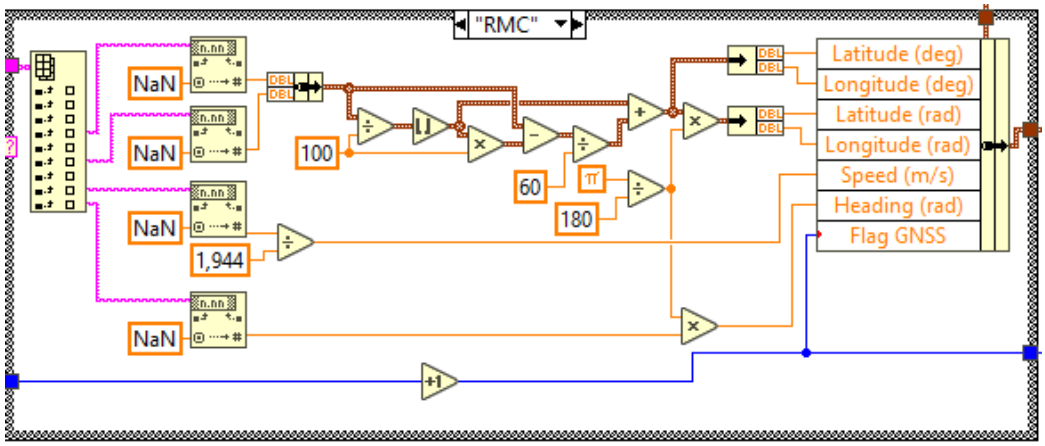


Figure 65: Mathematical operations on the GPS data

From Figure 65, it is visible that the latitude, in both degrees and radians, longitude, in both degrees and radians, GPS speed in meters per second and Heading angle in radians are converted to the correct unit by mathematical operations.

These outputs, are collected in GNSS Data. As this final procedure is repeated in every subVI, adding a picture for visualization could be unnecessary. See Figure 62 as reference.

## D.5 State Estimator kalman:

In the state estimator, all the needed data collected from the GNSS, Configuration.VI, Balancing controller and matrices coming from the startuplabview.m, which is a Matlab code written for configurations of LabVIEW, are put into the C-code of the kalmanfilter in order to reduce the noise of them. The outputs are the X in meter, Y in meter, etc. The outputs can be seen in Figure 66 below.

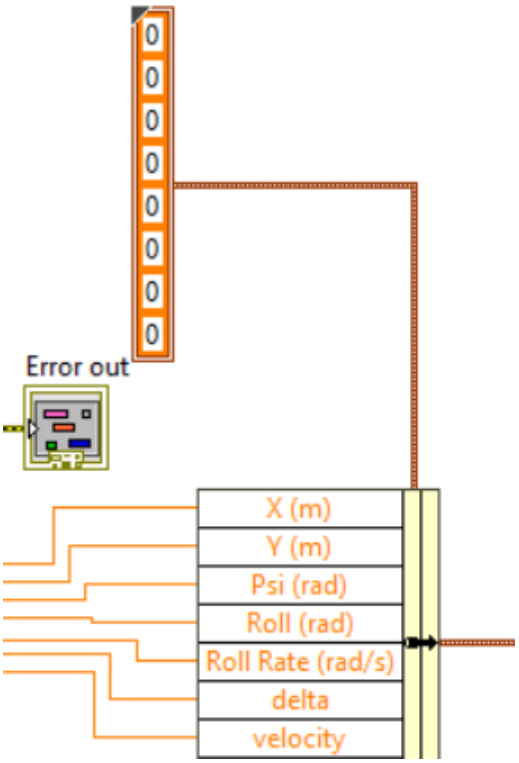


Figure 66: The outputs of the state estimator kalman

In the beginning, the filter used in the state estimator was a complementary filter which was then changed to a kalman filter. The complementary filter filtered the measurements to get a roll angle estimate. The Kalman filter was introduced to add position feedback as well (x, y, psi). The Kalman filter also improved the roll angle and roll rate estimate as the estimation with kalman filter was based on both the model and measurements.

## D.6 Balancing controller:

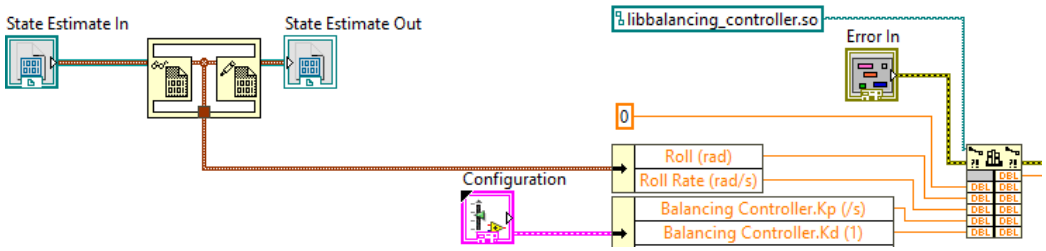


Figure 67: How the balancing controller works

As Figure 67 shows, the required data go into the balancing controller, which is written in C-code, as input values. The shape of the PD-controller that is used as the balancing controller is shown in Figure 8 .

In the balancing Controller.VI, an algorithm is also created to have a defined maximum allowed steering angle in both directions, left and right. This is done based on the steering angle from the encoder data and a predefined numeric control for the maximum allowed steering angle, as can be seen in Figure 68. If the maximum steering angle is hit, the loop is stopped and therefore, the steering is stopped.

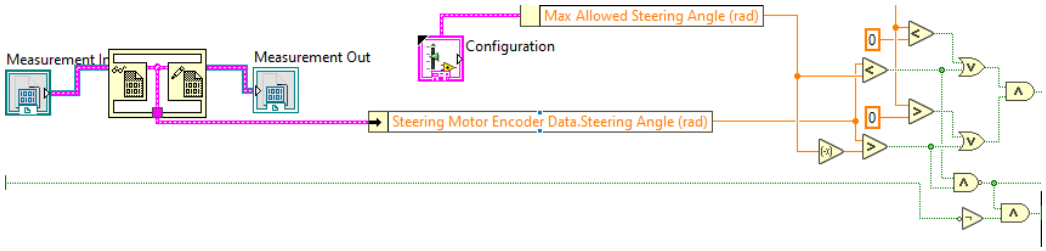


Figure 68: How the balancing controller works

## D.7 Forward or Drive Controller:

A numeric control was created and connected as an input to to the Drive Controller.VI to write the desired angular speed in radians per second and to adjust the velocity of the e-scooter. The correct configurations of the drive motor was given to the Drive Motor TXRX.VI. There, Using a C-code, the communication between the software and the VESC, the drive motor controller, was built and the angular speed in radians per second, input current in Amperes, motor current in Amperes and MOSFET, and temperature in celsius degrees were outputs of the VI.