CHALMERS

Self Driving Bike Project

Implementing a Speed-Controller in a Self Driving Bike

Bachelor's Thesis in Department of Electrical Engineering

ERIC AMBY
JARED TITTUS

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

BACHELOR'S THESIS 2024

Self Driving Bike Project

Implementing a Speed-Controller in a Self Driving Bike

ERIC AMBY
JARED TITTUS

CHALMERS

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Self Driving Bike Project
Implementing a Speed-Controller in a Self Driving Bike
ERIC AMBY, JARED TITTUS

Supervisor and Examiner: Jonas Sjöberg, Professor in Electrical Engineering

Bachelor's Thesis 2024
Department of Electrical Engineering
Chalmers Universitet of Technology
SE-412 96 Göteborg
Telefon +46 31 772 1000

Cover: The green bike used in the Autobike project

Typeset in LaTeX
Gothenburg, Sweden 2024

Self Driving Bike Project
Implementing a Speed-Controller in a Self Driving Bike
ERIC AMBY, JARED TITTUS
Department of Electrical Engineering
Chalmers Universitet of Technology

Abstract

To address challenges related to slow bike response caused by friction between the ground and the wheels, our contribution focuses on developing a two-phased speed controller. The PI-controller developed in this thesis aimed to minimize both dead-time and overshoot/undershoot, while ensuring smooth transition between the start-up and rolling phase.

Additionally a simulation-based speed-controller was implemented to accelerate parameter identification for the PI-controller used on the real-time system.

A new approach has also been implemented in the simulation to ensure the bike reaches a predetermined destination at a specific time, employing a time dependent velocity. This trajectory changes the bike's speed depending on its distance to the set referenced points along the route. The time dependent velocity has only been integrated into the simulation and not tested on the real-time system.

Keywords: Autobike, self-driving, speed-controller, bike, speed, dead-time, simulation, PI-controller, LabVIEW, C.

Acknowledgements

This Bachelor's thesis was made during the spring of 2024 by two Mechatronics students from Chalmers University of Technology.

We would like to thank our supervisor and examiner Jonas Sjöberg for his assistance during the project and also for giving us the opportunity to work on the Autobikes. We would also like to thank Armin Khorsandi, student from Chalmers University of Technology, for answering questions we had concerning the Autobikes during the project. Finally we want to thank Gianmarco Pane, student from University of Naples Federico II, for all the help we got with problem solving and testing the bikes.

Eric Amby and Jared Tittus, Gothenburg, June 2024

Contents

List of Figures

List of Tables

1

Introduction

This chapter provides an overview of the project's background, our contributions, along with its boundaries and limitations.

1.1 Background

In the ever-changing world of cars, safety among humans is a top priority. As cars become more autonomous, the need to safely interact with bicycles gets even more crucial. Programmers and engineers around the world are working to develop, test and implement algorithms that could keep cyclists safe on the road. Realistic testing cases needs to be done to make sure these algorithms work well. That's why self-driving bikes are being developed. The bikes are meant to mimic real cyclists but without putting any human at risk.

The Autobike project was initiated at Chalmers University of Technology in 2017 with the aim of creating a self-driving bicycle that could closely mimic a normal bike, both balancing and following a set route. The autobike's are intended to be used for tests among autonomous cars. The projects have been developed by different students since the start and today it consists of several different bikes. There are some differences between the bikes, for example: programming language the software is based on, type of motor used or material the bike is made of. The differences aside, all bikes require fundamental functions. For example, balancing, speed-control, and following a trajectory. In this project, two platforms will be tested.

The trajectory which the platform follows consists of points with X, Y coordinates and a heading. The heading is the direction the bike has to the next point on the trajectory. For the autobikes to follow the trajectory, they are equipped with a control box mounted on the bike to communicate with a computer which controls the bike. Other than the control box, the bikes are equipped with a forward motor to accelerate, a balancing motor to ensure the autobike does not fall while running, and a GPS to be able to navigate the desired trajectory. In this thesis, the bikes will be further developed and improved with a focus on enhancing a speed-controller for the forward motor and a position controller which makes sure the bike drives a certain distance at a certain time.

1.2 Past Issues Identified

Cable management is a big part of the Autobike project. Damaged cables, loose connections and components that are unattached is not only a potential time waster but also a hazard. For instance, damaged cables might harm other bike components or even endanger the individual handling the bike. Loose connections can consume valuable time as troubleshooting various issues may be necessary when the root cause is a loose connection. To prevent this from happening, the box for the green bike (described in Section 2.1.1) needed some wire arrangement.

Previously when the bikes were given a reference speed at the start-up they did not have a quick response to overcome the friction between the ground and the wheel. Figure 1.1 illustrates the issue where at the red circle labeled 1 presents the dead time. With the speed controller having an integral action, see Section 4.1, it can be seen that the time delay causes the control signal to increase until the measured speed starts to accelerate, see circle 2 in Figure 1.1. With the increasing control signal, the risk for damaging the components due to too high current increases.

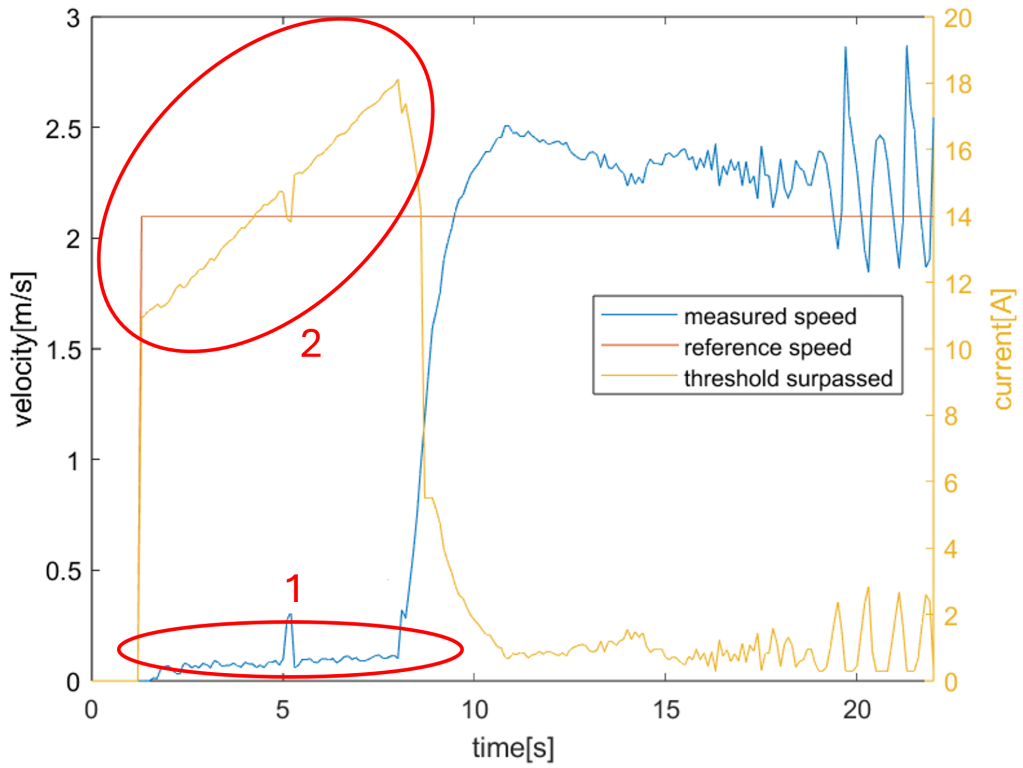


Figure 1.1: Start-up procedure. The time delay is believed to be caused by static friction in the motor.

In past tests it has been discovered that the motor which accelerates the bike can cause oscillations in speed if the current into the motor becomes too low. This can be seen in Figure 1.1 where the blue line, which is the measured speed, starts to

oscillate around $t = 20s$ where the orange line, the control signal, is around $0.3A$. The low current is a result of the overshoot when changing the velocity.

1.3 Contribution and Thesis Outline

- Section 3, work has been done to improve and secure the control box on the green bike. Worn-out cables inside the box were replaced, connections to components were re-soldered and cables of incorrect length were adjusted to prevent future breakage. Additionally a new hole was drilled in the front part of the box. This to relocate the emergency stop cable, ensuring a better protection in the event of a fall.

- Section 4.1, the speed-controller designed and developed in this thesis is a PI-controller. To speed up the tuning of the control parameters for the speed-controller used in the real-time system, a simulation-driven speed-controller has been developed.

- Section 4.2, our contribution also involves developing a speed-controller with two different phases: start-up and rolling. The start-up phase is for the bike to overcome the friction with the ground and start accelerating. The rolling phase is when the bikes is moving in a set speed. In the start-up, a method to minimize the dead-time was developed. The approach was extended to the rolling phase where the bike's overshoot/undershoot were reduced. These two phases were joined in a smooth transition.

- Section 4.3, a system has been implemented in the simulation where the bike can reach a predetermined destination at a specific time, a position controller. The velocity of the bike increases/decreases depending on whether its position is behind or ahead of the its reference point. The position controller has not been implemented on the real-time system.

- Section 4.4, to verify the disturbance compensation of the position controller, a disturbance in form of a torque is added to simulate the bike driving against head-wind or in a slope.

1.4 Accomplishments

The following has been accomplished by implementing the contributions.

- The simulation model created shows satisfying results when validation with the green bike is performed while with the red bike, a new model is required.

- By implementing a two-phased speed-controller, the time delay showed in Figure 1.1 is reduced to $0.3s$ for the two platforms tested.
- The implementation of the position controller shows satisfying results as reference speed changes depending on the current position of the bike.

1.5 Limitations

The speed-controller implemented in simulation is a simple model. This is to get a starting point of the potential parameters. The testing and validation of the real system will only be done on asphalt and is therefore not verified on other surfaces, for example grass or gravel. Only two platforms from the autobike project will be tested. The position controller will only be tested in simulation and therefore is not validated.

2

Hardware & Software Description

In this chapter the platforms which have been used in the project will be described and its differences. The chapter will also describe all the components in the control box, the motors which drives the bike, and the software programs used in the project. This information is to get a broader understanding of the functionality of the autobikes.

2.1 Platforms

For this project two platforms have been tested, which are: Green bike, and the Red bike. Both platforms are equipped with a "control box" containing the components described in Section 2.2. Beyond the control box, the platforms are equipped with a steering motor and a forward motor. These may vary depending on the platform.

2.1.1 GREEN bike

This platform refers to the bike shown in Figure 2.1. It is equipped with a portable box in which hardware lies (see Figure 3.1). The steering motor for this platform is the DCX32L motor described in Section 2.2.9 and is mounted on the steering shaft shown in Figure 2.11. To accelerate the bike, it is equipped with the Crystalyte UFO-R3525 motor described in Section 2.2.8.2.

Figure 2.1: Green Bike used in the Autobike project

2.1.2 RED bike

The red bike refers to the bike shown in Figure 2.2. This platform is equipped with a stationary control box which has been welded to the frame of the bike. The steering motor is the same model as for the green bike, i.e. the DCX32L motor (see Section 2.2.9). However, the forward motor differs from the green bike. The red bike uses the SHIMANO STEPS DU-E6010 drive motor described in Section 2.2.8.1. and is mounted between the pedals (see Figure 2.9).

Figure 2.2: Red Bike used in the Autobike project

2.2 Hardware

This section describes the hardware used on the platforms. Section 2.2.1 to 2.2.7 describes the control box which is used to communicate with the controlling computer. Section 2.2.8 & 2.2.9 describe the motors used to move and steer the bikes.

2.2.1 myRIO-1900

The myRIO-1900, seen in Fig 2.3, serves as a versatile tool for implementing various design concepts through a single reconfigurable I/O (RIO) device [7]. Equipped with MXP (Expansion Ports) and MSP (System Ports) on either side of the device. It has both analog and digital I/O, 10 analog inputs, 6 analog outputs and 40 digital I/O lines. The myRIO-1900 also has Wi-Fi capability, an onboard accelerometer and a dual-core ARM Cortex-A9 processor. Its Wi-Fi capability enables seamless integration into remote embedded applications. The programmable options for the myRIO-1900 includes both LabVIEW and C.

Figure 2.3: myRIO-1900 device

2.2.2 ESCON

To control the steering of the bike, ESCON 50/5 is used. ESCON 50/5 is a so called "PWM servo controller" and is designed by Maxon [8]. The controller is most used for speed control with both closed loop and open loop (i.e., with feedback and no feedback). As stated, ESCON 50/5 is PWM servo controller meaning it controls the motor using PWM signals (Pulse Width Modulation) which is a way to control analog devices, using a digital signal.

Figure 2.4: ESCON 50/5 device

2.2.3 Steering Motor Encoder

The `HEDS-5540#A11` is the encoder that is used for the steering motor. It is used to measure the position of the steering motor, or what angle the front wheel has compared to when the encoder was reset. The device consists of a lensed LED

source, a code wheel and an integrated circuit with detectors and output circuitry. The encoder has three outputs, two square waves in quadrature and a high true index pulse which is generated once for each time the codewheel does a full rotation [9]. The steering motor encoder used in the project can be seen in Fig 2.5.

Figure 2.5: `HEDS-5540#A11`

2.2.4 IMU

For the autobike to get measurements for whether it is tilting or moving, an Inertial Measurement Unit (IMU) is used. The specific model used is a Pmod NAV designed by Digilent. The IMU is equipped with a gyroscope, accelerometer and a magnetometer (in this project only the first two are used). The gyroscope provides the angular rate of the bike. The accelerometer provides the amount of acceleration affecting the bike [10]. Both the gyroscope and the accelerometer give measurements for three cartesian coordinates (X, Y, Z) and have the units of degree per second.

2.2.5 UART Communication

The primary function of the Universal Asynchronous Receiver/Transmitter, abbreviated as UART, is to facilitate the transmission and reception of serial data between different electronic devices[11]. These devices could be i.e. microcontrollers, displays or sensors. It transmits the data between two devices only using two wires, a transmitter (Tx) and a receiver (Rx). The Tx from device 1 is connected to the Rx on device 2 and vice versa. This is illustrated in Fig 2.6.

The UARTs transmit data asynchronously which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. The transmitting UART instead adds start and stop bits to the data being transferred. This indicates the beginning and end of the data for the receiving UART.

When the receiving device for UART detects a start bit, it reads the incoming bits at a frequency known as baud rate. This rate uses bits per second (bps) to measure the speed of the transfer data. For the communication to work between both devices the maximum allowable difference of baud rate between transmitting

and receiving UARTs is around 10%. If the difference is higher, the consequences would be loss of synchronization between the devices and ultimately communication failure.

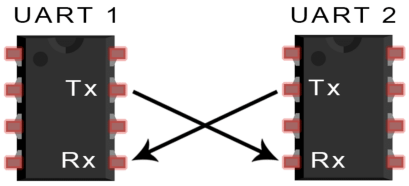


Figure 2.6: UART Communication

2.2.6 RUT955

The bike is also equipped with a router so that the myRIO will be able to communicate with the user's computer. The router used in this project is the RUT955 (Fig 2.7) which is a high-performance industrial router. It supports a range of satilite systems including GPS, GLONASS, BeiDou, Galileo, and QZSS [12]. Featuring dual SIM card slots for 4G dual SIM redundancy, the router guarantees uninterrupted connectivity even in challenging environments.

Powered by an Atheros CPU with a MIPS 74Kc architecture and running on the RutOS operating system, which is based on OpenWrt Linux OS, users can execute Linux bash commands directly on the device. Moreover, it facilitates the implementation of custom functionalities through C code and bash scripting, enhancing flexibility and adaptability.

Figure 2.7: RUT955 device

2.2.7 VESC

A VESC (Vedder Electronic Speed Controller) is a versatile device that regulates the direction and speed of electric motors in a precise way [13]. This feature makes

the VESC essential by converting electrical energy into mechanical motion.

The VESC controls current flow from the power source to the motor for speed and direction. Inside the device a microcontroller processes signals from different sources to determine the optimal motor control strategy. Advanced algorithms adjust the voltage and current to the specific needs of the desired motor. The modified electrical signals are sent to the motor and there converted into mechanical motion. The VESC utilizes Hall sensors as a feedback mechanism to guarantee the motor's performance aligns with expectations.

Figure 2.8: VESC device

2.2.8 Forward Motor

The forward motor is used to accelerate the bike. The motor can vary depending on the platform. For the green and red bike two different forward motors are used, both of them will be described in this chapter.

2.2.8.1 SHIMANO STEPS drive motor

One forward motor used in this project is the SHIMANO STEPS DU-E6010 drive motor which is located between the pedals on the red bike (Figure 2.9) [14]. The motor operates at $36V$ DC and is a brushless DC motor. It has a maximum torque of $50Nm$ with a peak power capability of $500W$. However, the nominal rated power stands at $250W$. The motor is controlled by how much current the speed controller sends as output. If the current is too low, the motor can not measure the speed which causes oscillations.

Figure 2.9: SHIMANO STEPS DU-E6010 drive motor

2.2.8.2 Crystalyte UFO motor

Another forward motor used is the Crystalyte UFO-R3525 motor [15]. This is a wheel motor meaning it accelerates the wheel without any chains, see Figure 2.10. The motor operates at $36V$ DC and has a maximum torque of $75.9Nm$ corresponding to a current of $41.20A$. The motor can have an output power peaking at 827.5W.

Figure 2.10: Crystalyte UFO-R3525 motor

2.2.9 Steering Motor

The steering motor used in this project is the DCX32L designed by Maxon [16] and is placed by the steering shaft (Figure 2.11). The motor is equipped with graphite brushes. In this project, the motor is primarily used to steer the bike to follow a trajectory, but also to balance the bike. The nominal voltage of the motor is $18V$ DC in which maximum continuous torque is $101Nm$. The maximum torque corresponds to a maximum continuous current of $5.42A$.

Figure 2.11: Steering motor for the green bike

2.3 Software

This section describes the software used in the Autobike project. LabVIEW is used to control the real-time system together with VS-code. MATLAB/Simulink is used to simulate the bike to faster troubleshoot and test new implementations.

2.3.1 LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a visual programing language which was developed by National Instruments as a workbench for controlling test instruments [1]. The programing language used for LabVIEW is called "G" and is coded in VI's (Virtual Instrument) which is the interface in where the user creates applications [2]. G is a graphical programming language in which the user connects blocks via nodes [3]. The blocks can be, for example, simple mathematical functions but also other VI's (in this case sub-VI's).

2.3.2 VS-Code

Visual Studio Code, more commonly known as VS-Code, is an opensource text editor by Microsoft. It supports a variety of programming languages, including C, C++ and Java [4]. For this project, VS-Code has been used to create C functions called by a LabVIEW VI.

2.3.3 MATLAB/Simulink

MATLAB (matrix laboratory), made by MathWorks, is a programming platform that is specifically designed to analyze and design systems. The MATLAB language is a mix of C/C++. The software makes it easy to develop algorithms, analyze data and create models/applications. MATLAB integrates with SIMULINK [5].

SIMULINK is a versatile tool for simulation and Model-Based Design. It supports i.e. continuous tests, simulation and verification of embedded systems. SIMULINK is seamlessly integrated with MATLAB and therefore makes it possible to incorporate MATLAB algorithms into models [6].

The combination of MATLAB and SIMULINK makes it possible to simulate results and analyze them before applying it to physical hardware.

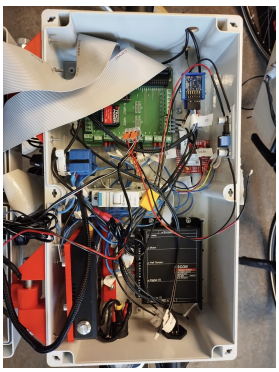
3

Improving the Control Box of the Green Bike

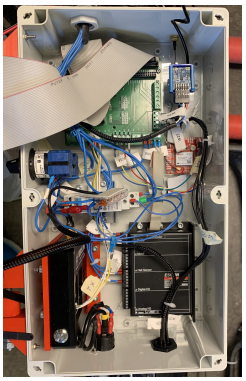
There were physical aspects of the control box for the green bike that needed to be fixed before running the bike, see Figure 3.1a. Cables inside the box had been worn-out, connection to components needed to be re-soldered and some cables were not the correct length and therefore a risk to get caught and break in the future.

Previously the hole, for the emergency stop cable, was placed on one of the long sides of the control box. This could cause problems if the bike would fall and land on that specific side. To prevent this from happening a new hole was drilled to the box, can be seen at the top of Figure 3.1b. The new hole was placed on the front of the control box and thus leading to the emergency stop cable being protected if the bike would fall.

Figure 3.2 showcases the wiring diagram for the control box used on the green bike. It gives a clear overview of all the components located inside the control box and how they are connected to each-other.



(a) The box for the green bike before



(b) The box for the green bike after

Figure 3.1: The box for the green bike before and after reconstruction

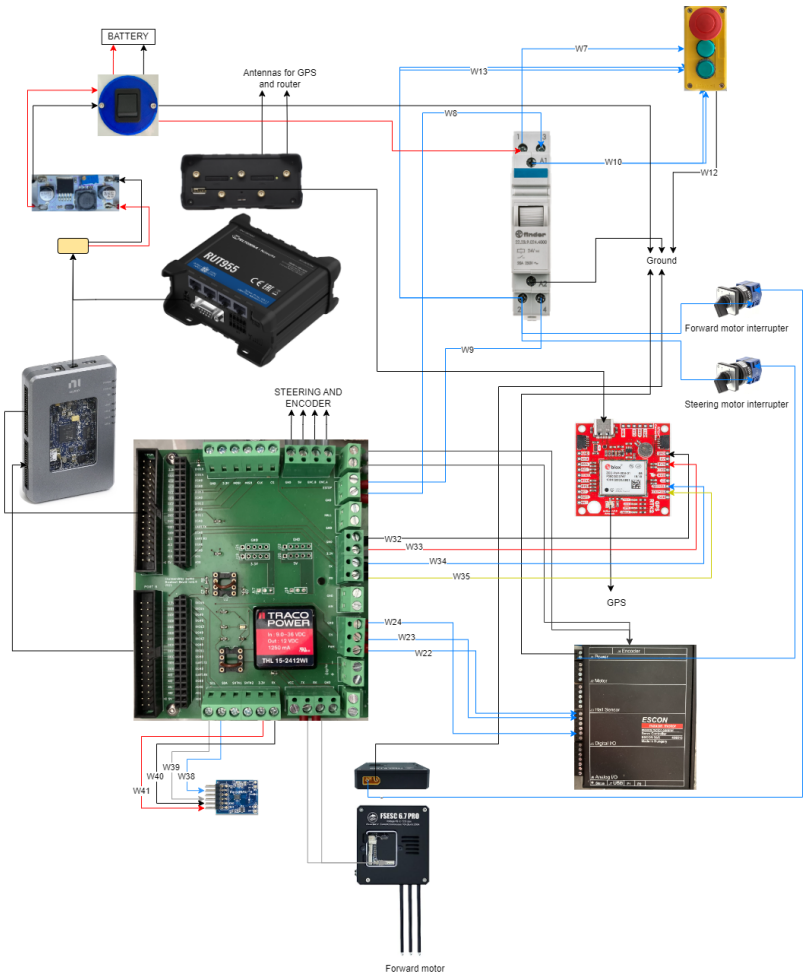


Figure 3.2: The wiring diagram for the control box

4

Theoretical Methodologies Utilized for Project Contribution

In this chapter, the theoretical methodologies used in this project will be described. The methodologies that are used are: modelling the control system for the speed-controller used in simulation, elimination of time delay at start-up procedure, and derivation of a position controller in simulation. These methodologies will contain numerical equations as well as figures to enhance the understanding.

4.1 Derivation of speed controller for the auto-bike

To control the velocity, a discrete PI controller is used. The control law of the controller is:

$$u[k] = K_p e[k] + K_i \sum_{i=1}^{k} e[i] T_s \tag{4.1}$$

$u[k]$, in Eq.(4.1), is the control signal produced by the controller after k samples, K_p is the proportional gain which determines the ratio of the output response to the error signal. K_i is the integrating gain which drives the steady-state error (difference between actual output and desired output) towards zero. $e[k]$ indicates the error between the measured speed and the reference speed after k samples. $\sum_{i=1}^{k} e[i]$ is the memory of the previous error and is referred to as the integral sum. Lastly, T_s is the sampling time [17]. The reason for using a PI-controller is to eliminate the steady state error caused by resistance and a derivative action is not desired since it is sensitive to measurement noise which causes oscillation.

The controller is placed in a closed loop together with a process consisting of the bike with an electric motor, see Figure 4.1. In Figure 4.1, the PI controller controls the motor with the control signal being a current i which results in a torque T. The gears transform the torque, T, from the motor, to the applied torque of the driving wheel, T_w. The speed sensor presented in Figure 4.1 is the VESC which measures the actual speed of the bike. The VESC is described in Section 2.2.7.

4.1.1 Model of the motor

The time constant of the motor is neglected as it is assumed to be small compared to the time constant of the bike. The model of the motor is therefore approximated

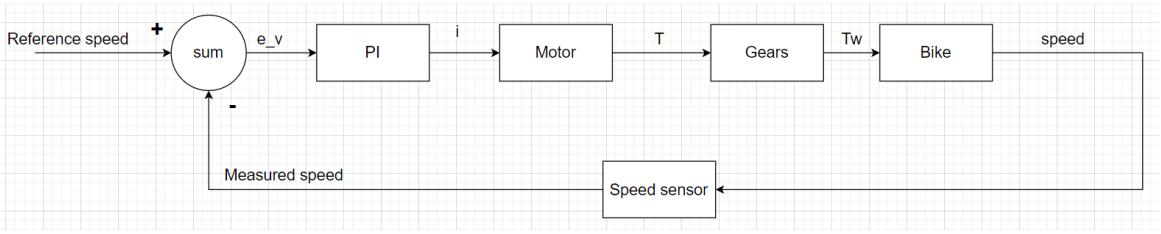


Figure 4.1: Structure of the speed controller. i indicates the current into the motor, T is the torque out of the motor, and T_w notates the applied torque of the driving wheel.

by:

$$T = i\kappa \tag{4.2}$$

with i being the current into the motor. T indicates the torque out of the motor, and κ represents the transformation constant.

4.1.2 Model of the gear

The torque out of the motor is transported via gears to the driving wheel. The equation for the torque applied on the driving wheel is given by:

$$T_w = T\epsilon\eta \tag{4.3}$$

with T_w being the torque applied to the wheel and ϵ indicating the gear ratio. The efficiency constant, η is assumed to be 1 for both platforms meaning there is no losses between the torque of the wheel and the torque of the motor.

4.1.3 Model of the bike

To derive the process of the bike powered by the applied torque of the wheel, the following dynamic equation is set forth which is extracted from [18],

$$Ma = -F_d - F_{rr} + F_t \tag{4.4}$$

In Eq.(4.4), M is the full mass of the bike plus the motor and a notates the acceleration of the bike which can be written as $\dot{v}$. F_d and F_{rr} are the drag resistance force and the rolling resistance force respectively. Lastly, F_t is the thrust force provided by the motor. Since the bike is powered by a torque, F_t is transformed to $\frac{T_w}{r}$ where r is radius of the wheel. Eq.(4.4) is therefore transformed into:

$$T_w = r(M\dot{v} + F_d + F_{rr}) \tag{4.5}$$

Since both F_d and F_{rr} are difficult to measure, and to simplify the model they are merged into F_f which is the total frictional force affecting the bike. It is also assumed that the frictional force term is proportional with the velocity of the bike, which extends Eq.(4.5) into:

$$T_w = r(M\dot{v} + F_f v) \tag{4.6}$$

By Laplace transforming Eq.(4.6) from T_w to v, the resulting transfer function for the bike becomes:

$$G(s) = \frac{V(s)}{T_w(s)} = \frac{K}{\tau s + 1} \tag{4.7}$$

with $K = \frac{1}{rF_f}$ and $\tau = \frac{M}{F_f}$.

By combining Eq.(4.2)&(4.3) with Eq.(4.7), the complete transfer function of the bike powered by an electric motor becomes:

$$\frac{V(s)}{I(s)} = \kappa\epsilon\frac{K_s}{\tau s + 1} \tag{4.8}$$

with $K_s = \epsilon\kappa K$.

4.2 Eliminating time delay at start-up

The time delay that needs to be eliminated is the time it takes for the autobike to accelerate from $0m/s$ after a reference speed has been set, see Figure 1.1, and is believed to be caused by static friction in the motor. To resolve this issue, a constant is added to the control signal, i.e. output of the PI controller. Figure 4.2 illustrates the block diagram of the speed-controller where u_f indicates the added term to eliminate the time delay.

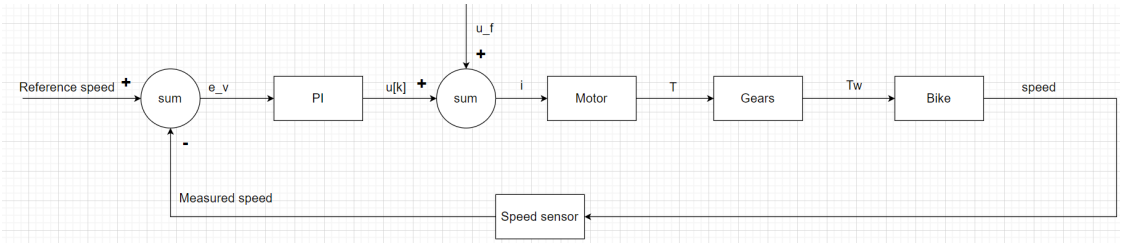


Figure 4.2: Developed structure of the speed controller with u_f added to the control signal, $u[k]$.

To select u_f, the following formula is used:

$$u_f = u_{acc} - u_0 \tag{4.9}$$

u_{acc} is the control signal needed to overcome the static friction, i.e. the control signal highlighted by the second red circle in Figure 4.3. u_0 is the control signal when the step change is made, indicating the control signal highlighted by the circle labeled 1 in Figure 4.3. However, the developed algorithm is only desired when the motor is affected by the static friction and therefore, the constant u_f should be removed, i.e. switch to the original control algorithm shown in Figure 4.1, after the measured speed surpasses a certain percentage of the reference speed. To ensure a smooth transient when the switch occurs, the integral sum in Eq.(4.1) is initialized using:

$$\sum_{i=1}^{k} e[i] = \frac{u_{ss} - K_p e[t]}{K_i T_s} \tag{4.10}$$

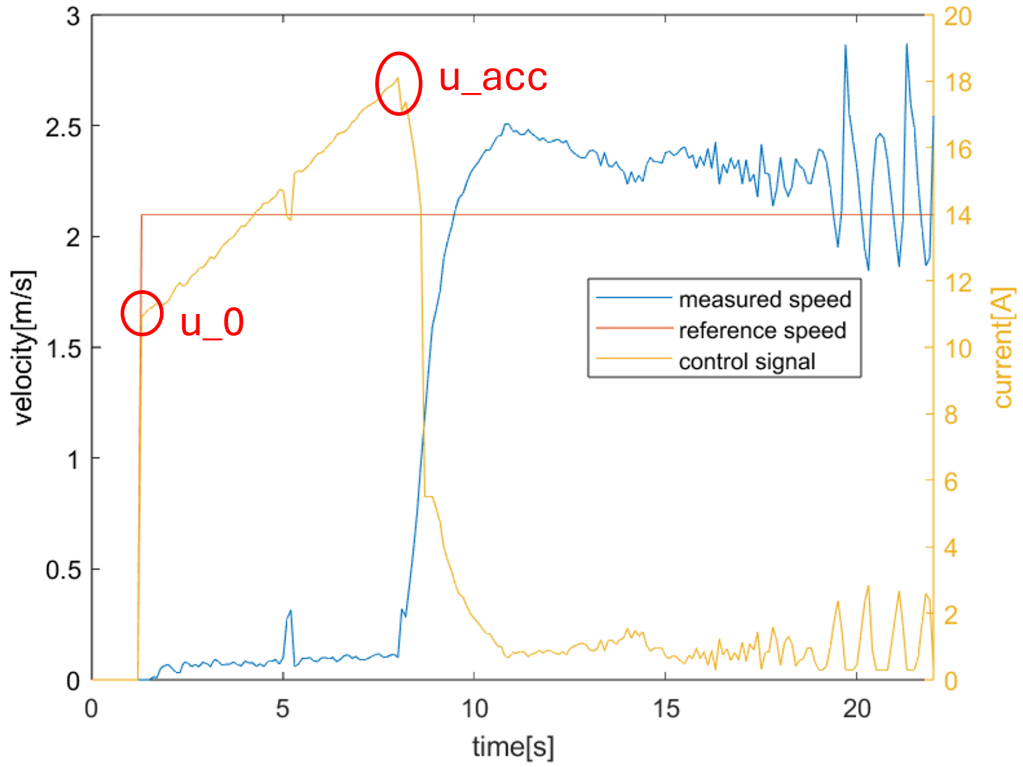


Figure 4.3: Indication where u_0 and u_{acc} are collected from.

where u_{ss} is the control signal needed to keep the measured speed at steady state. The result from Eq.(4.10) is then inserted into Eq.(4.1) which should give:

$$u[k] = u_{ss}$$

The algorithm is utilized for three iterations to ensure the control signal stays around u_{ss}.

4.3 Derivation of position controller for the autobike

As stated in Section 1.3, one of the implementations for the autobike project is a position controller to automate the bike even further. In Figure 4.1, the control structure is presented for the position controller. The inner loop is controlled by a P-controller to control the velocity of the bike. The resulting velocity of the bike is integrated to access the current position of the bike which is then compared to a reference position, P_{ref}. The reference position P_{ref} is derived from a desired velocity using the following equation:

$$P_{ref} = vt \tag{4.11}$$

where t is the current time in simulation. To control the input to the inner loop, the outer loop is equipped with an discrete PI-controller. To reduce the integrator

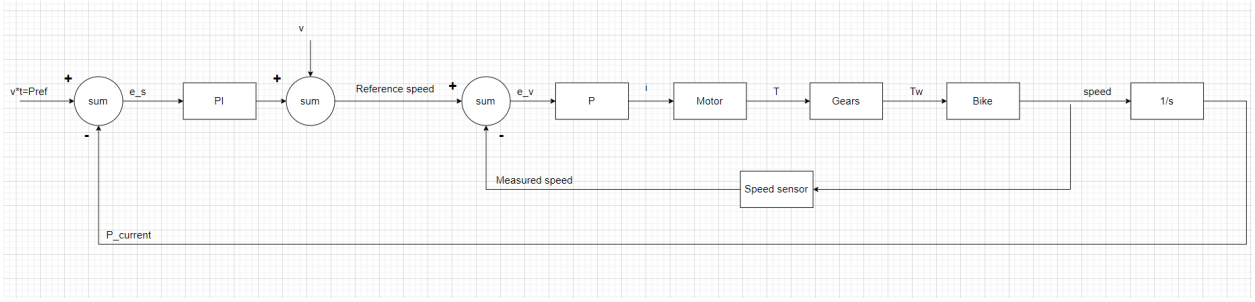


Figure 4.4: Structure of position controller.

action of the controller, the same desired velocity used in Eq.(4.11) is added to the control signal of the PI-controller.

4.4 Adding disturbance

To test the disturbance compensation of the position controller, a disturbance is added in form of a negative torque highlighted by the red circle in Figure 4.5. The physical interpretation of the disturbance is for example wind or a slope. As the figure illustrates, the disturbance is added to the torque applied to the wheel.

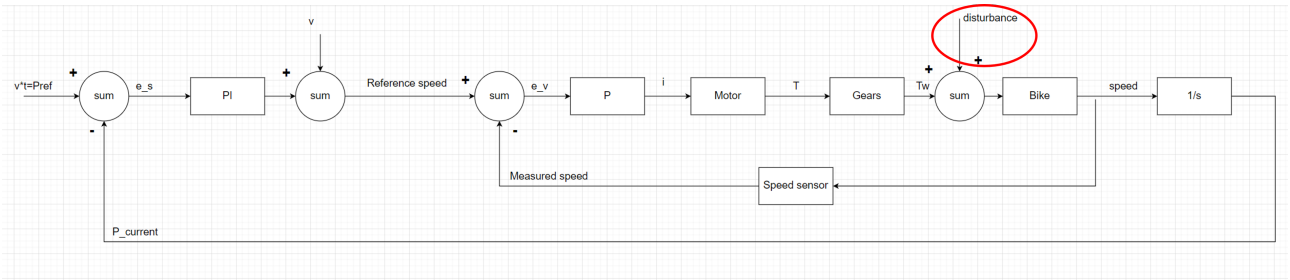


Figure 4.5: A negative torque added to simulate the bike driving in up-hill or head wind.

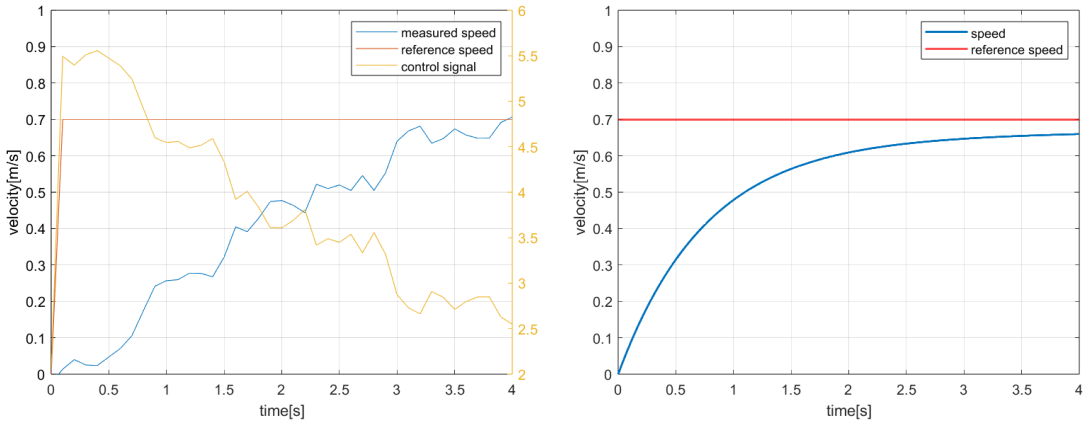
5

Finding & Validating Control Parameters

This chapter will describe how K_p and K_i are found using the simulation model introduced in Section 4.1. The chapter also contains a validation of the control parameters for both platforms. Lastly, the chapter will end with an analysis of the validation.

5.1 Finding control parameters using pole placement

In the simulation model derived in Section 4.1, two parameters are unknown, the transformation constant κ and the friction F_f. To approximate these values, a step change in reference speed of $0.7m/s$ from a non-zero velocity is made with the red bike using arbitrary control parameters, $K_p = 1.8$ and $K_i = 0.2$. Figure 5.1a shows the result.



(a) Test on real system **(b)** Mimic in simulation

Figure 5.1: A comparison of a $0.7m/s$ step change using real system and simulation

From the resulting step response, κ and F_f are approximated by imitating the response illustrated in Figure 5.1a in simulation. The result is illustrated in Figure 5.1b.

The resulting model is used to tune the speed controller. The method used is the pole placement method [21]. The discrete poles used to find suitable control parameters which should give a relative fast and non-oscillating response are:

$$p_{1,2} = 0.98 \pm 0.054j \tag{5.1}$$

By placing the poles at Eq.(5.1), the system will become stable due to $|p_{1,2}| < 1$ [22]. The system will also show a relatively fast rise time due to poles having a large real part. The damping constant, ζ, of the system is calculated by

$$\zeta = \frac{0.98}{\sqrt{0.98^2 + 0.054^2}} \tag{5.2}$$

Utilizing Eq.(5.2), ζ is calculated to become approximately 1 which indicates the system is critically damped, i.e. fast step response with, theoretically, no oscillations, [23], which is the desired properties of the speed. A reason for desiring a non-oscillating system is due to the risk of the system becoming unstable with an oscillating speed and, as stated in Section 1.2, a overshoot/undershoot when making a step change can cause a too low current into the motor which causes the motor to switch on and off. With the poles from Eq.(5.1), the resulting control parameters are calculated to become $K_p = 5.3$, $K_i = 0.5$. The parameters used in Eq.(4.8) are presented in Table 5.1.

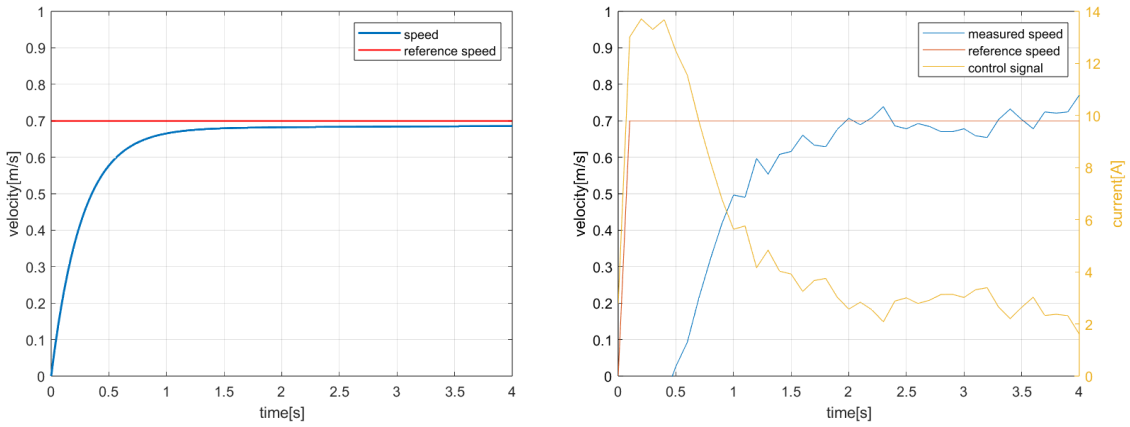
Table 5.1: Values used in Eq.(4.8) for simulating the two platforms.

Platform	M $[kg]$	r $[m]$	F_f $[Nm]$	κ	ϵ
RED	45.0	0.350	9.00	0.0770	95.0
GREEN	31.3	0.350	6.26	0.324	23.0

5.2 Validation of control parameters

With the control parameters mentioned above, the resulting response in simulation is in accordance to Figure 5.2a. The control parameters are validated on both platforms by performing an identical step change as showed in Figure 5.1. However, with these control parameters, the response for the red bike became non valid. This is due to the control parameters produced a rise time which the forward motor could not handle. The result was a pulsating current when making a step change which caused the forward motor of the red bike to turn on and off causing the velocity to oscillate. It is therefore decided to keep the parameters used to validate κ and F_f in Section 5.1 due to them being validated for the forward motor to handle. $K_p = 5.3$ and $K_i = 0.5$ are however validated on the green bike which gave the result in Figure 5.2b.

The properties being investigated from the step change are: rise time, overshoot, and undershoot. The resulting properties from using $K_p = 5.3$, $K_i = 0.5$ and $K_p = 1.8$, $K_i = 0.2$ for the green and red bike respectively are shown in Table 5.2.



(a) Step performed in simulation. **(b)** Step performed with green bike.

Figure 5.2: A step change from $0m/s$ to $0.7m/s$ using $K_p = 5.3$, $K_i = 0.5$ using the simulation model and green bike.

Table 5.2: Resulting properties from performing the $0.7m/s$ step change illustrated in Figure 5.2.

Platform	Step $[m/s]$	Rise time $[s]$	Overshoot [%]	Undershoot [%]
GREEN	0.700	0.800	–	–
RED	0.700	2.40	–	–
Simulation	0.700	0.700	–	–

5.3 Analysis of validation

In Figure 5.3, the test results from the designed controller for the green bike is analysed. The red circle in Figure 5.3 shows that the green bike when performing a step change has a time delay of $0.5s$ which is not observable with the simulation model in Figure 5.2a. This time delay has not been investigated however, it can be observed in Figure 5.2 that the rise time and the non-oscillating settlement for the green bike are close to identical to the step made in simulation. Therefore, Figure 5.2 shows that the model is effective when tuning the control parameters for the speed controller.

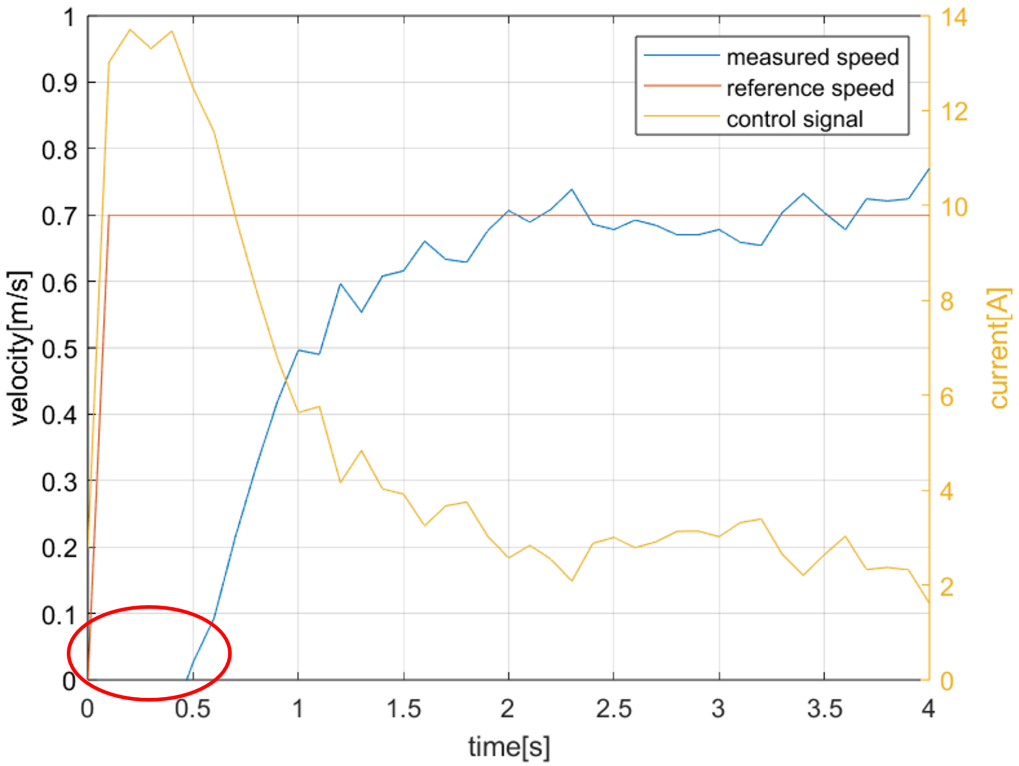


Figure 5.3: A $0.7m/s$ step change tested on the green bike using control parameters derived from pole placement. The red circle indicate noteworthy aspects from Figure 5.2.

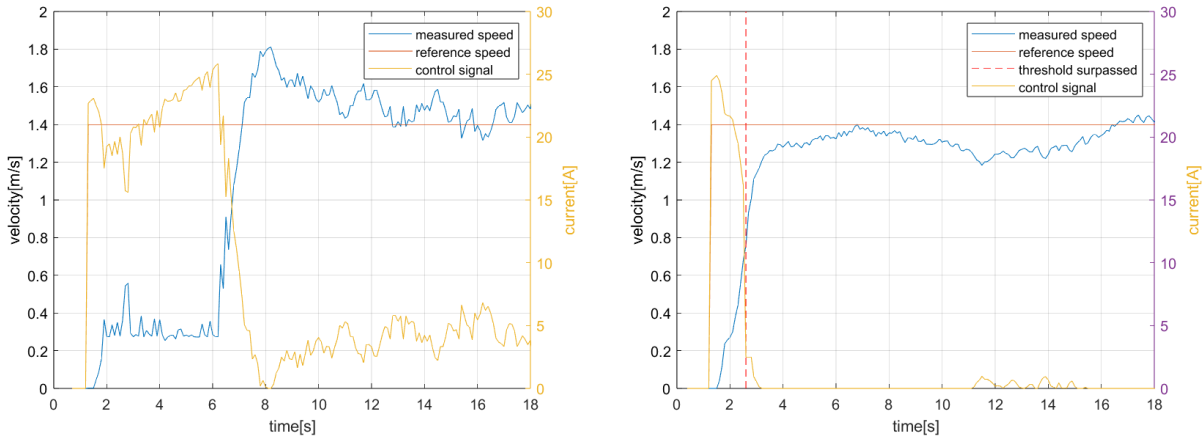
6

Elimination of Time Delay at Start-up

This chapter will show the results using the methodology described in Section 4.2. The results will be presented with figures to enhance the understanding of the methodology. The chapter will end with an analysis of the shown results.

6.1 GREEN bike

Before utilizing the method described in Section 4.2, the start-up for the green bike outside is in accordance to Figure 6.1a. The figure shows the step response from $0m/s$ to $1.4m/s$ with the original control algorithm, i.e. Figure 4.1. From Figure 5.2b, u_{ss} is read to be $2.5A$ and from Figure 6.1a, u_0 and u_{acc} are read to be $22.7A$ and $25.8A$ respectively. With u_0 and u_{acc} inserted into Eq.(4.9), the equation gives $u_f = 3.1$. The resulting step response using the developed control algorithm, i.e. Figure 4.2, is illustrated in Figure 6.1b where the vertical red dotted line indicates where the measured speed surpasses the threshold, which is 50% of the reference speed. This is also where the control algorithm switches back to the original and where the integral sum is initialized for three iterations to ensure a smooth transition.

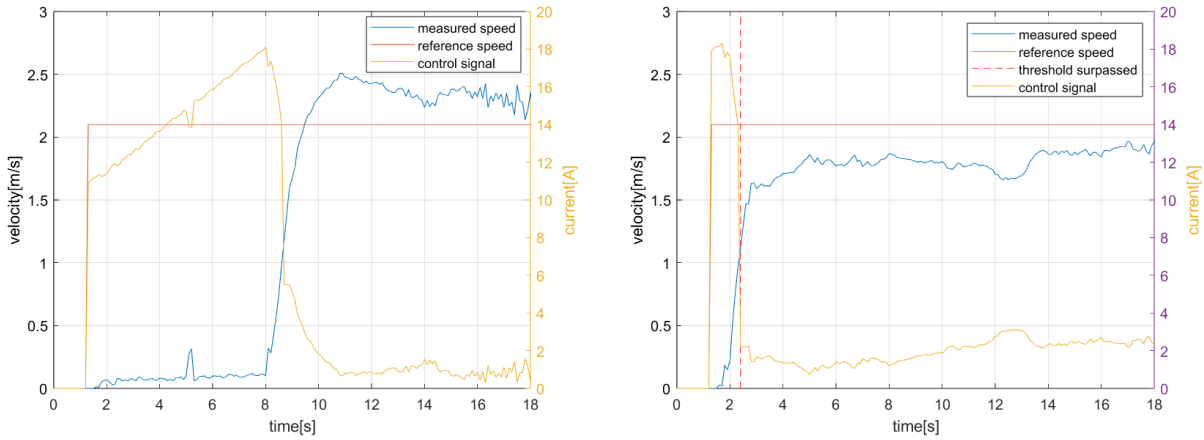


(a) $K_p = 5.3$, $K_i = 0.5$, no u_f **(b)** $K_p = 5.3$, $K_i = 0.5$, $u_f = 3.1$

Figure 6.1: Step response from $0m/s$ to $1.4m/s$ for the green bike outdoors. The tests are performed to verify the methodology described in Section 4.2.

6.2 RED bike

The test conducted using the green bike is replicated with the red one. However, unlike for the green bike, the step change performed is from $0m/s$ to $2.1m/s$. This is due to inconsistent behavior from the forward motor when making a step change from $0m/s$ to $1.4m/s$. Figure 6.2a illustrates the start-up of the bike using the original control algorithm and presents the same test as Figure 1.1&4.3. From Figure 6.2a, the following can be extracted: $u_0 = 11A$ and $u_{acc} = 18A$. u_f is calculated to become 7 by utilizing Eq.(4.9) with the measured u_{acc} and u_0. For the red bike, u_{ss} is measured to be $2.2A$ and utilized in Eq.(4.10) for three iterations. Figure 6.2b illustrates the step response using the developed algorithm with $u_f = 7$. The red dotted line in the figure indicates where the measured speed surpasses 50% of the reference speed and where the control algorithm switches back to the original.



(a) $K_p = 1.8$, $K_i = 0.2$, no u_f **(b)** $K_p = 1.8$, $K_i = 0.2$, $u_f = 7$

Figure 6.2: Step response from $0m/s$ to $2.1m/s$ for the red bike outdoors. The tests are performed to verify the methodology described in Section 4.2.

6.3 Analysis of algorithm

The developed algorithm used to eliminate the time delay at the start-up is proved to be effective for both platforms by observing Figure 6.1&6.2. Noteworthy aspects from the result are highlighted in Figure 6.3.

6.3.1 Analysis of GREEN bike

For the green bike, the time delay highlighted by the first red circle of Figure 6.3a is measured to be reduced to $0.3s$ indicating the time delay has been reduced drastically.

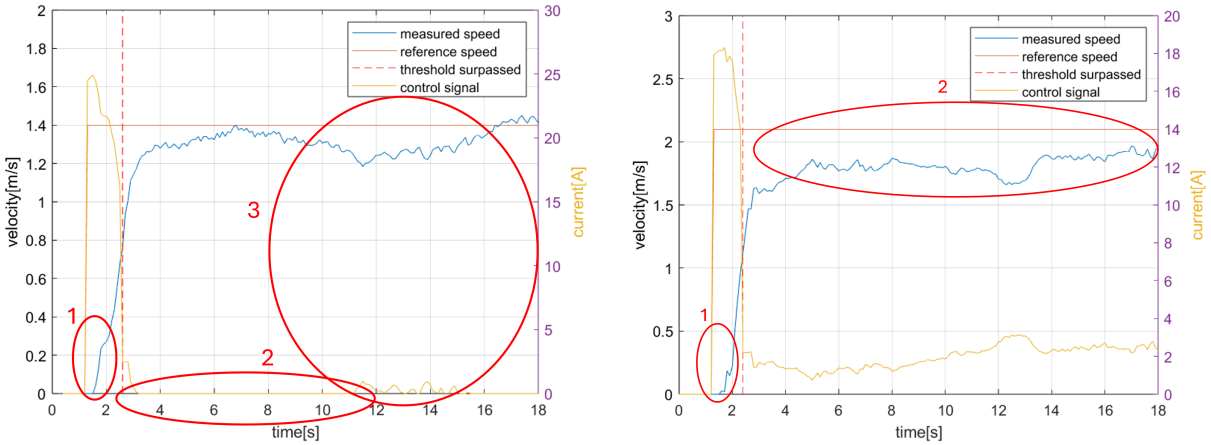
The second circle highlights that after the switch of the control algorithms is made, the control signal is held at $2.5A$ for three iterations. However, the circle also highlights that the control signal converges at $0A$. Due to the threshold being 50% and

the large K_p value, when the integral sum is initialized, the result of Eq.(4.10) will be negative and since the error becomes smaller and smaller, the term $K_p e[k]$ will be smaller and smaller leading the control signal to go towards zero. By inserting the known parameters for Eq.(4.10), the result is:

$$\sum_{i=1}^{k} e[i] = \frac{2.5 - 5.3\frac{1.4}{2}}{0.005} = -242$$

This issue causes the controller to be slow when correcting the error when the measured speed becomes lower than the reference speed, as highlighted by the third circle in Figure 6.3a. By decreasing the K_p value or increasing the threshold, the issue with a negative integral sum will be resolved.

However, by increasing the threshold, the risk for an overshoot increases and other option, to decrease K_p, would lead to a slower response.



(a) Green bike $K_p = 5.3$, $K_i = 0.5$, $u_f = 3.1$

(b) Red bike $K_p = 1.8$, $K_i = 0.2$, $u_f = 7$

Figure 6.3: Analysis of Figure 6.1b&6.2b.

6.3.2 Analysis of RED bike

The red bike also shows major improvement regarding the time delay when using the developed control algorithm. When using the original control algorithm at start-up, the time delay is measured to be $8s$ for the red bike, while by implementing the developed control algorithm, the time delay is reduced to $0.3s$ as highlighted by the first circle in Figure 6.3b.

As mentioned in Section 5.3, the control parameters for the red bike produce a slow response when making a step change. This is also observable in Figure 6.3b where an offset is highlighted by the second circle. The offset is measured to be around $0.25m/s$ and lasts for about $12s$. The slow correction of the offset is a result of the control parameters which have been noted to cause a slow response. How-

ever, since the test in Figure 6.3 is to validate the developed control algorithm, it is determined that the result is satisfying.

7

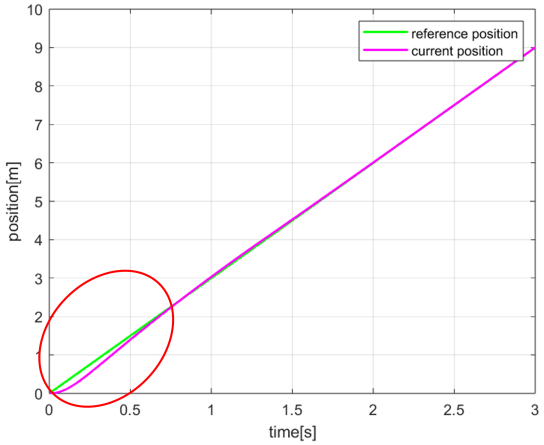
Evaluation of Position Controller In Simulation

In this chapter the position controller will be evaluated in simulation. The position controller controls the distance the bike has driven using a reference distance. It will first be tested following a ramp line without disturbances. Then, to validate the disturbance compensation of the position controller, a similar test will be performed were a disturbance is added after $5s$.

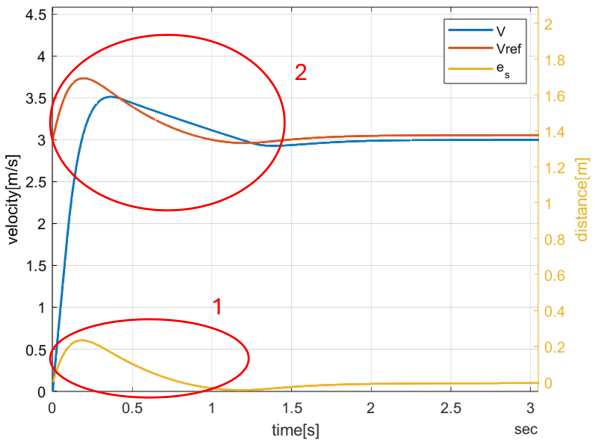
7.1 Simulation of a ramp response

By implementing the position controller, which was described in Section 4.3, into the simulation, the reference velocity of the bike changes depending on if the bike is ahead or behind the reference point. Figure 7.1 presents a simulation of the bike following a trajectory in the form of a ramp. The starting velocity used to reduce the control signal is $3m/s$. The control parameter used are for the inner loop is $K_p = 15.9$, and the outer loop uses $K_p = 3$ and $K_i = 1$. Note that the parameters are not validated to be the optimal and are only used to validate the cascade control algorithm. However, with the bandwidth of the inner loop, $11rad/s$, being faster than the bandwidth of the outer, $8.6rad/s$, the criteria for cascade control, [23], is met and the parameters can be used to test the position controller.

In Figure 7.1a, the transient of the position is illustrated when the bike starts from $0m/s$. As the red circle highlights, the position of the bike will deviate from the reference position. Figure 7.1b show the corresponding velocities, i.e. the reference speed, the speed of the autobike, and the error between the reference position and the current position. It is observable that due to the velocity of the bike starting from $0m/s$, the position of the bike is behind the reference position. This causes an error which is highlighted by the first circle. To compensate the error, v_{ref} increases to around $3.7m/s$, second circle, causing the error to decrease. After about $2s$, the error between the bike and the reference position converges leading v_{ref} to decrease. Figure 7.1b shows a steady state error after $t = 2s$ with the velocity where the reference speed exceeds $3m/s$. This is due the reference speed needs to exceed the desired velocity to compensate the resistance modeled in the simulation model and since the velocity is controlled by a P-controller, a steady state error will arises.



(a) Transient of position.

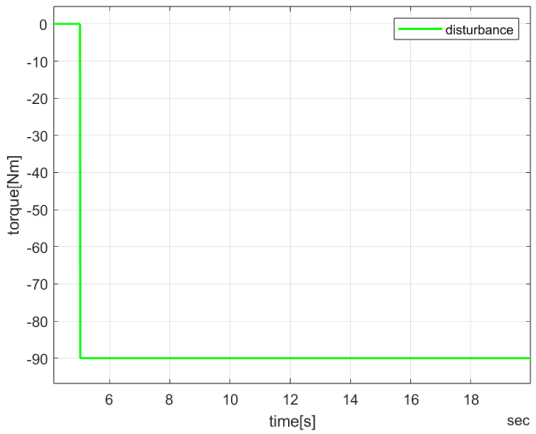


(b) Transient of velocity.

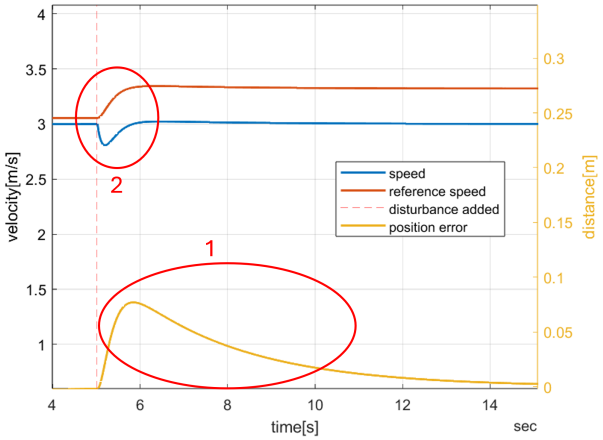
Figure 7.1: Simulation of a position controller using the structure presented in Section 4.3. The control parameters are inner loop has $K_p = 15.9$ and for the outer loop $K_p = 3$, $K_i = 1$. The reference trajectory is a ramp.

7.2 Adding a disturbance

The test to validate the disturbance compensation of the position controller is performed in the same environment as the test in Section 7.1. The disturbance added is a step disturbance in the form of a negative torque placed at $t = 5s$ and is illustrated in Figure 7.2a. The disturbance represents the bike driving in a slope or against head wind causing the velocity to decrease.



(a) Disturbance added to simulate head wind or up-hill.



(b) Transient of velocity after applied disturbance.

Figure 7.2: Simulation of the autobike affected by a disturbance at after $t = 5s$. The control parameters are inner loop has $K_p = 15.9$ and for the outer loop $K_p = 3$, $K_i = 1$. Red dotted line indicates when the disturbance is added.

In Figure 7.2, the correlating velocities of the bike is illustrated. Due to the disturbance added after $t = 5s$, the velocity of the bike will reduce, which is highlighted by the blue line in the second circle. This causes the position error to increase, which is highlighted by the first circle. To correct the error, the reference speed increases, highlighted by the red line in the second circle, causing the velocity to surpass $3m/s$. This leads to the position error to converge slowly, after around $t = 14s$, indicating that the autobike has caught up to the reference position.

8
Conclusion

In this chapter a short summary will be given of the content of this project. The chapter will also present the accomplishments from implementing the contributions described in Section 1.3. Lastly, the chapter will end with examples of what can be done in the future regarding the autobike project.

8.1 What has been done

In this project, a simulation model of a speed-controller have been created and used to tune the control parameters for both platforms. The model has been validated to be effective when tuning the control parameters for both platforms.

By implementing the developed control algorithm, the time delay at start-up is reduced to $0.3m/s$ for both platforms meaning the algorithm to prevent time delay when starting from $0m/s$ is found effective. The parameter u_f for the developed control algorithm should be for the green and red bike, $u_f = 3.1$ and $u_f = 7$ respectively when using the control parameters discovered in Section 5.1 and making a step change from $0m/s$ to $1.4m/s$ and from $0m/s$ to $2.1m/s$ with the green and red bike respectively.

The implementation of a position controller enables the autobike to follow a predetermined reference velocity. The controller has also been validated by adding a disturbance and found effective to increase the disturbance compensation of the position controller. The position controller enables the autobike to automatically change the reference velocity depending on the current position relative to the reference position.

8.2 Future work

This section will provide examples of prospective enhancements to the project and outline the validation requirements identified in this report.

- Derive a new model for the red bike to use in simulations. The current process used is only suitable when simulating the speed-controller for the green bike. To find the transfer function of the red bike, a open loop step change is suggested to be made.
- Implement the position controller on the real system. Currently, the position controller is only implemented in the simulation.

- As illustrated in Figure 1.1, when the current becomes too low the measured speed start to oscillate. To prevent this phenomena, a breaking torque can be applied at all times to restrict the control signal from becoming to low. However, a downside to implementing a constant friction can potential be that the control signal becomes too large. This can cause damage in the electrical components.

Bibliography

[1] electronics-notes, "What is LABVIEW", *electronics-notes*, [Online]. Available: URL, Accessed on: 2024-04-06.

[2] electronics-notes, "LABVIEW VIs: Virtual Instruments", *electronics-notes*, [Online]. Available: URL, Accessed on 2024-04-06.

[3] National Instruments, "Programming in G", *National Instruments*, [Online]. Available: URL, Accessed on: 2024-04-06.

[4] A. Mustafeez, "What is Visual Studio Code?," *educate*, [Online]. Available: URL, Accessed on: 2024-04-06.

[5] MathWorks, "What is MATLAB?," *MathWorks*, [Online]. Available: URL, Accessed on: 2024-04-06.

[6] Dr I. F. Mear, *An Introduction to Using Simulink*. 5th ed, Oxford, England: University of Oxford, 2018. [Online]. Avialable: URL, Accessed on: 2024-04-06.

[7] National Instruments, "myRIO-1900," *National Instruments*. [Online]. Available: URL, Accessed on: 2024-04-06.

[8] Maxon Motor, "ESCON Servo Controller",*Maxon Motor*. [Online]. Available: URL, Accessed on: 2024-04-06.

[9] AVAGO TECHNOLOGIES, "Quick Assemnbly Two and Three Channel Optical Encoders", *AVAGO TECHNOLOGIES*. [Online]. Available: URL, Accessed on: 2024-04-06.

[10] VECTORNAV, "What is an Intertial Measurment Unit", *VECTORNAV*. [Online]. Available: URL, Accessed on: 2024-04-06.

[11] S. Campbell, "BASICS OF UART COMMUNICATION", *Circuit Basics*. [Online]. Avaible: URL, Accessed on: 2024-04-06.

[12] Teltonika Networks, "RUT955 - Industrial Cellular Router," Teltonika Networks. [Online]. Available: URL, Accessed on: 2024-05-05.

[13] ONE STOP BOARD SHOP, "Intro To VESC", *ONE STOP BOARD SHOP*. [Online]. Available: URL, Accessed on: 2024-04-06.

[14] SHIMANO, "SHIMANO STEPS Drive Unit for Coaster Brake", *SHIMANO*. [Online]. Available: URL, Accessed on: 2024-04-06.

[15] Crystalyte, "Crystalyte UFO Motor Series", *Crystalyte*. [Online]. Available: URL, Accessed on: 2024-04-06.

[16] Maxon Motor, "Online shop for high precise drive system by maxon", *Maxon Motor*. [Online]. Available: URL, Accessed on: 2024-04-06.

[17] B. Thomas, “PI-control,” in *Modern Control Engineering*: 5th ed., B. Magnusson, Ed. Stockholm, Sweden: Libra, 2016, ch. 4.6, pp. 64.

[18] C. Abagnale, M, Cardone, P. Iodice, S. Strano, M. Terzo, G. Vorraro, "A dynamic model for the performance and environmental analysis of an innovative

e-bike", 69th Conference of the Italian Thermal Machines Engineering Association, Milan, Italy, 2014, pp. 618-627. [Online]. Available: URL, Accessed on: 2024-02-14.

[19] K. J. Åström, R. M. Murray, "Kalman Filtering", in *Feedback Systems: An Introduction for Scientists and Engineers*, 2.10b ed., USA: Princeton University Press, 2009, ch. 7.4, pp. 215-219 [Online]. Available: URL, Accessed on: 2024-04-20.

[20] B. P. Gaizka, "Trajectory control with Kalman Filter-based State Estimation for bicycle," Chalmers University of Technology, Gothenburg, Sweden, 2023. [Online]. Available: URL, Accessed on: 2024-05-16.

[21] B. Thomas, “Pole Placing Method”, in *Modern Control Engineering*: 5th ed., B. Magnusson, Ed. Stockholm, Sweden: Libra, 2016, ch. 19.2, pp. 358-362.

[22] B. Thomas, “Where should the poles be placed?”, in *Modern Control Engineering*: 5th ed., B. Magnusson, Ed. Stockholm, Sweden: Libra, 2016, ch. 19.3, pp. 362-366.

[23] B. Thomas,*Modern Control Engineering*: 5th ed., B. Magnusson, Ed. Stockholm, Sweden: Libra, 2016.

A

Simulink Model of Speed Controller

This appendix will showcase and describe the Simulink model of the implemented speed controller.

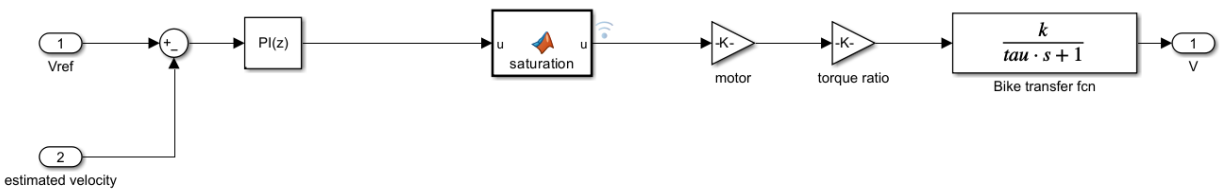


Figure A.1: Simulink model of the speed controller.

Figure A.1 presents the Simulink model of the speed controller used to tune the control parameters in simulation. The block labeled "Bike+Motor transfer function", is the transfer function derived in the end of Section 4.1. The middle block labeled "saturation" is used to make sure the control signal cannot become negative or become too high. The block contains the following code.

```
function u = saturation(u)
if u <= 0
    u = 0;
elseif u >= 30
    u = 30;
end
```

where the parameter "u" indicates the control signal.

B

Code for Eliminating Time Delay

This appendix will illustrate the code used to apply and remove the enhancement and initialize the integral sum. The code is written in VS Code using the C language.

```
1  // Makes sure enhancement is only enabled when the
   reference speed over zero is.
2  // And disables the enhancement when measured speed
   surpasses threshold.
3  if (reference_speed <= 0 || (measured_speed > threshold
   * reference_speed) || enhancement_on = 0; )
4      {
5      flag = 0;
6      if (measured_speed > threshold * reference_speed)
7      {
8          enhancement_on = 0;
9      }
10 }
11 else
12 {
13     flag = 1;
14 }
15 // Initiate integral sum for three iterations.
16 if ((measured_speed > threshold * reference_speed) &&
17 (counter >= 0 && counter <= 2))
18 {
19     integral_sum = (u_ss - Kp * error) / (Ki *
          Tasking_period_s);
20     counter++;
21 }
22 // Formula for PI controller + enhancement.
23 *control_signal = Kp * error + (Ki * integral_sum) *
   Tasking_period_s + u_f * flag;
```

The parameter "u_ss" on line 19 is the control signal at steady state and "u_f" notates the enhancement used to eliminate time delay at start-up.

C

MATLAB Implementations for Time Trajectory

This Appendix will present the solution to simulate the position controller in Simulink.

Figure C.1 shows the corresponding structure to Figure 4.4 in Simulink.

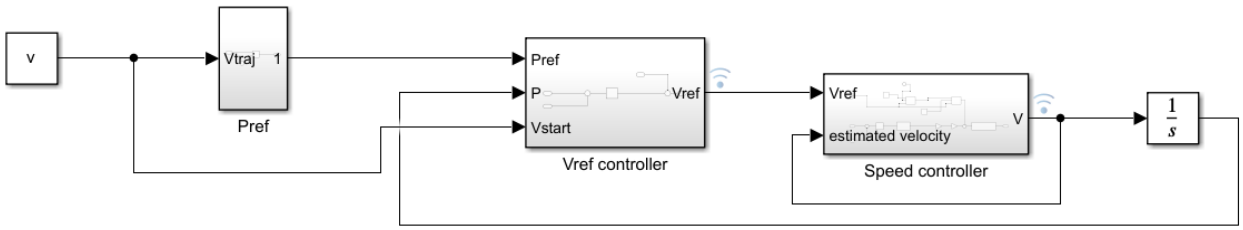


Figure C.1: Simulink model of the position controller.

The block labeled "Pref" multiples a desired velocity to the the current time to get a position, which is the output of the block.

The block labeled "Vref controller" is the outer loop of the position controller and contains the structure according to Figure C.2. The block labeled "Vstart" is the same velocity used to calculate the reference position, i.e. Eq.(4.11).

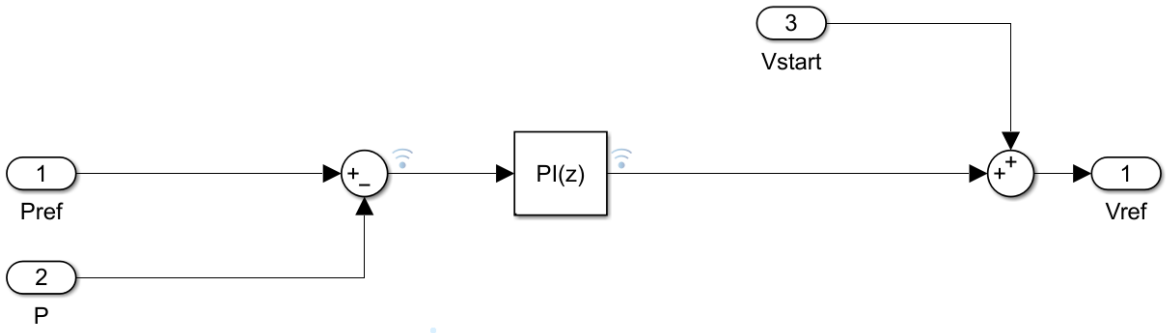


Figure C.2: Simulink model of the block labeled "Vref controller".

The block labeled "Speed controller" is the inner loop presented in Figure 4.4, and is showcased in Figure C.3. Excluding the structure highlighted by the second circle, the control loop is similar to the one presented in Figure A.1 with the exception of the controller being a P-controller, circle 1. The structure highlighted by the second circle is the disturbance added at $t = 5s$ in Section 7.2.

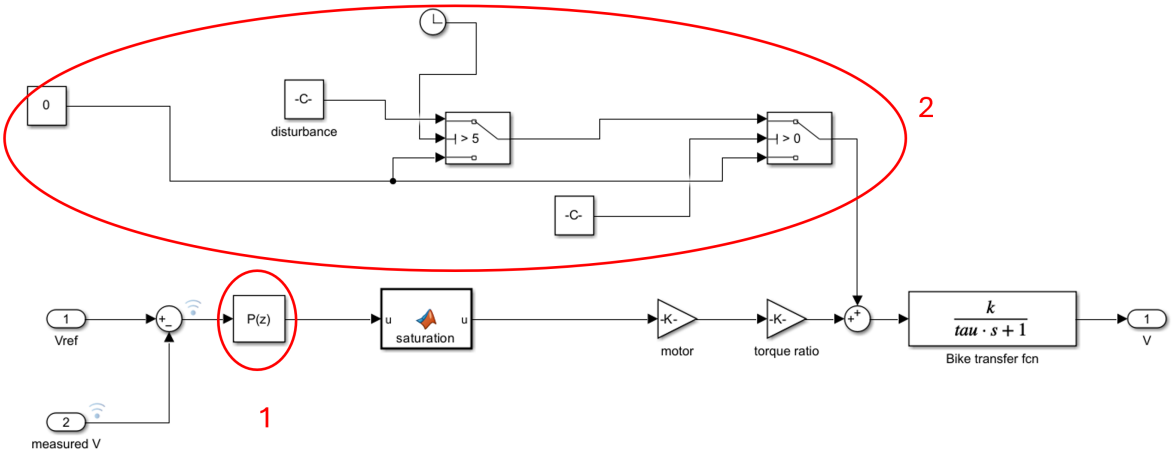


Figure C.3: Simulink model of the block labeled "Speed controller".

The output of block is then integrated and compared to "Pref" in the block labeled "Vref controller".

INSTITUTIONEN FÖR något ämne
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS