Technical Report

William Wärn
Karl Karlsson
Arvid Linder
Carl Sjöstedt
Emil Strömblad
Claes Thunberg
Gustav Åberg

December 12, 2022

Version 0.1

LINKÖPINGS UNIVERSITET

Project Identity

Orderer: Lars Eriksson
E-mail: lars.eriksson@liu.se

Customer: Fredrik Wemmert, Aurobay

Supervisor: Robin Holmbom
E-mail: robin.holmbom@liu.se

Course Responsible: Daniel Axehill, Gustaf Hendeby
E-mail: daniel.axehill@liu.se, gustaf.hendeby@liu.se

Project Members

Name	**Responsibility**	**E-mail**
William Wärn	Project leader	Wilwa907@student.liu.se
Karl Karlsson	Documentation & Design	Kakar847@student.liu.se
Arvid Linder	Software	Arvli383@student.liu.se
Carl Sjöstedt	Test & Hardware	Carsj142@student.liu.se
Emil Strömblad	Information	Emist067@student.liu.se
Claes Thunberg	Quality	Clath750@student.liu.se
Gustav Åberg	Test & Experiment Design	Gusab106@student.liu.se

CONTENTS

DOCUMENT HISTORY

Version	Date	Changes	Done by	Reviewer
0.1		First draft	Whole group	

1 INTRODUCTION

This document is the technical report for the project *Machine Learning and Adaptive Control for Improving Servo Performance* within the project course *Automatic Control, TSRT10*. The main purpose of this document is to present the technical parts of the project as well as the performance of the system.

1.1 Parties

Customer: Fredrik Wemmert, Aurobay
Orderer: Lars Eriksson, ISY/Liu
Supervisor: Robin Holmbom, ISY/Liu
Examiner: Daniel Axehill, Liu
Project Team: Seven engineering students at Linköping University

1.2 Background

This project is based on the theory and model outlined in *Model-Based Throttle Control using Static Compensators and Pole Placement* L. Eriksson et. al.[1]. Most throttles regulators execute a calibration routine in order to determine parameters such as friction, zero point and limp home mode. These can change when the temperature of the engine changes, if the outside temperature fluctuates while driving or, if the altitude of the vehicle is changing drastically. The result of this is a sub optimal regulation which can lead to more emissions and less comfort for the driver. A more successful implementation of an adaptive regulation strategy would further improve the control of the throttle regardless of current parameters defining the characteristics of the throttle.

The throttle controls the airflow to the engine which affects the power of the engine. Therefore, the throttle is a critical part when it comes to both safety and driving experience. With a good control system it is possible to gather data in operation and identify deviations that needs to be compensated for to obtain the desired characteristics.

1.3 Aims and Purpose

The aim of this project is to improve the control of an intake throttle in an internal combustion engine. This should be done by exploring ways of identifying and updating the throttle model during a ongoing driving mission. The goal is to first explore how the throttle parameters change with time in varying conditions, and how these changes affect the performance of the throttle servo. The goal is further to implement an adaptive regulation algorithm for the throttle servo in order to improve the the control of the system when the parameters inevitably change.. The purpose is to have the possibility to handle complex engines and help the customer to gain market share and better economy.

1.4 Use

The project can find use in many different areas using servos or actuators, not just throttles. Therefore it could be of great value to the costumer. It could also be of great use in general in a lot of different disciplines for example in automotive, aeronautical or aerospace engineering. A good solution to this problem allows the engine to be run at optimal conditions at all times, increasing efficiency and limiting the environmental impact. If it is possible to find

a good solution for this specific task, this solution could definitely be adjusted and applied on other similar servo or actuator related problems as well.

1.5 Abbreviations

Table 1: Description of all abbreviations.

Abbreviation	Description
ML	Machine learning
ADC	Analog to Digital Converter
GPIO	General-Purpose Input/Output
UART	Universal Asynchronous Receiver/Transmitter
PID	Proportional Integral Derivative

1.6 Model Notation

Table 2: Notation of all parameters used in the model.

Notation	Description
T_u	Driving torque from motor
T_s	Spring torque
T_{fs}	Static friction torque
K_{fv}	Static gain regulator parameter
T_{fv}	Dynamic friction torque
T_{emp}	Back electromotive torque
T_c	Coulomb torque
J	Throttle moment of inertia
u	Control signal
e	Error value signal
θ	Throttle angle
$\theta_{lh}^{\pm}$	Angular limits of the *limp home region*
θ_{ref}	Reference throttle angle
e_θ	The difference between measured θ and θ_{ref}
$k^{\pm}$	Gradients
$m_{lh}^{\pm}$	Friction at the limits of the *limp home region*
ω	Angular velocity
α_{ref}	Throttle reference angle

2 GIVEN THROTTLE MODEL

The project is centered around the dynamics of a throttle servo, which is a physical system. The modeling and approximations of the throttle behaviour has been done in earlier works and therefore the simulation model of the throttle is taken from the course *Modelling and Control of Engines and Drivelines, TSFS09*.

To generate a reference throttle angle different drive cycles will be used. A drive cycle generates a target velocity, target acceleration and which gear to use. This will be translated into a reference angle for the throttle through the simulated model of the vehicle. Shown in Figure 1 is one example of how a drive cycle could look, in the example the EUDC drive cycle is used.

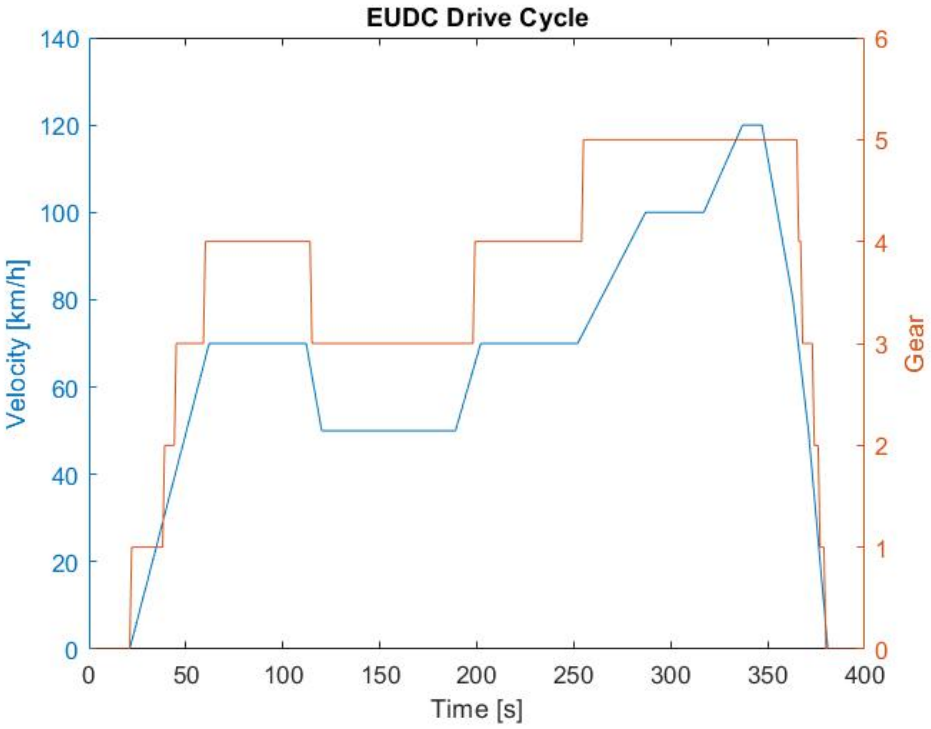


Figure 1: Example of the EUDC drive cycle .

A Simulink throttle model is needed as reference in the regulator. The throttle model in this project is based on the model outlined in *Model-Based Throttle Control using Static Compensators and Pole Placement* L. Eriksson et. al.[1]. The throttle model is approximated as outlined in Figure 2, where K_{fv} and K are static gain regulator parameters [1].

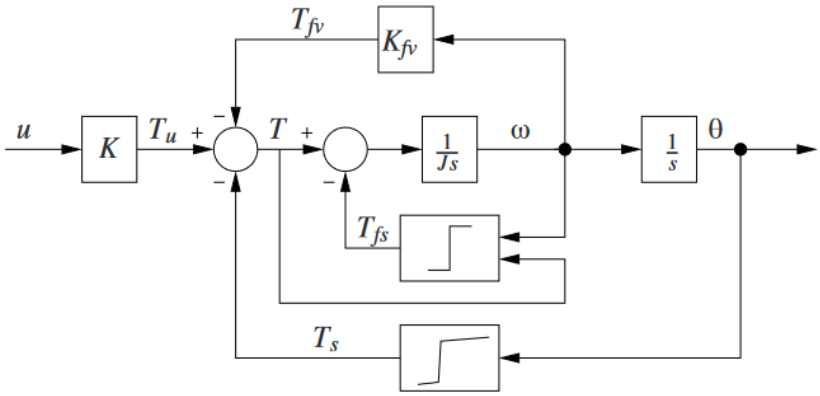


Figure 2: The throttle model as a Simulink block diagram [1].

The static gain regulator parameter K_{fv} relate to the throttle frictions as outlined in Equation (1) [1].

$$K_{fv}\omega = T_{fv} + T_{emf} \tag{1}$$

The torques acting on the throttle are the driving torque from the motor T_u, spring torque T_s, static friction torque T_{fs}, dynamic friction torque T_{fv} and back electromotive torque, T_{emp} [1].

The driving torque T_u from the motor is approximated as linearly dependent on the control signal, u.

The spring torque T_s is piecewise linearly related to the throttle angle, θ in accordance to Equation (2) where $k^{\pm}$ are different gradients, $\theta_{lh}^{\pm}$ are the angular limits of the *limp home region*, and $m_{lh}^{\pm}$ are the frictions at the limits of the *limp home region* [1]. The relation between these parameters is illustrated in Figure 3 [1].

$$T_s(\theta) = \begin{cases} m_{lh}^{+} + k^{+}(\theta - \theta_{lh}^{+}) & \text{if} \quad \theta > \theta_{lh}^{+} \\ m_{lh}^{+}(\theta - \theta_{lh})/(\theta_{lh}^{+} - \theta_{lh}) & \text{if} \quad \theta < \theta_{lh} < \theta \leq \theta_{lh}^{+} \\ m_{lh}^{-}(\theta_{lh} - \theta)/(\theta_{lh} - \theta_{lh}^{-}) & \text{if} \quad \theta < \theta_{lh}^{-} < \theta \leq \theta_{lh} \\ m_{lh}^{-} - k^{-}(\theta_{lh}^{-} - \theta) & \text{if} \quad \theta > \theta_{lh}^{+} \end{cases} \tag{2}$$

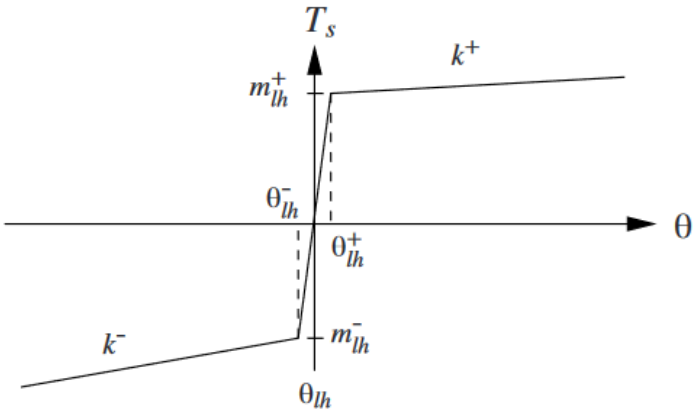


Figure 3: Friction torque, T_s as function of throttle angle, θ [1].

The static friction torque, T_{fs} in the throttle is not constant but dependent on its angular velocity, ω, coulomb torque T_c and applied torque, T in accordance to Equation (3) [1].

$$T_{fs}(T,\omega) = \begin{cases} T & \text{if } \omega{=}0 \text{ and } |T| < T_c \\ T_c sign(\omega) & \text{otherwise} \end{cases} \tag{3}$$

The dynamic friction torque, T_{fv} and back electromotive torque, T_{emp} are both linearly dependent on the throttle angular velocity, ω [1]. This model results in the relation between control signal, u and throttle angle, θ illustrated in Figure 4 [1].

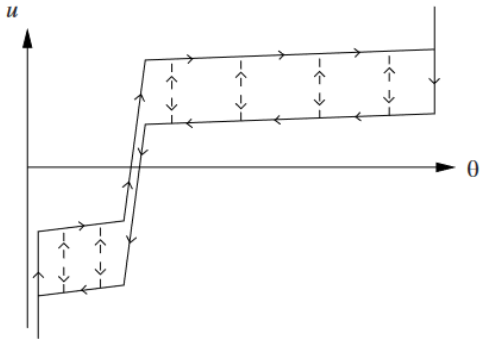


Figure 4: Control signal, u as function of throttle angle, θ [1].

2.1 Hysteresis boxes

The estimation of the nonlinearities, friction and limp home, depends on the parameters described in Figure 3, which could be determined using the hysteresis boxes. These boxes are illustrated by the principle sketch in Figure 4 above.

This principle sketch is realised by performing a slow ramp response up and down in the throttle control signal, u. Doing this results in a behaviour similar to the one illustrated in Figure 5, in this figure one can see the parameters ($A_1 - A_4$ and $B_1 - B_4$) that are to be identified by fitting a piecewise linear function.

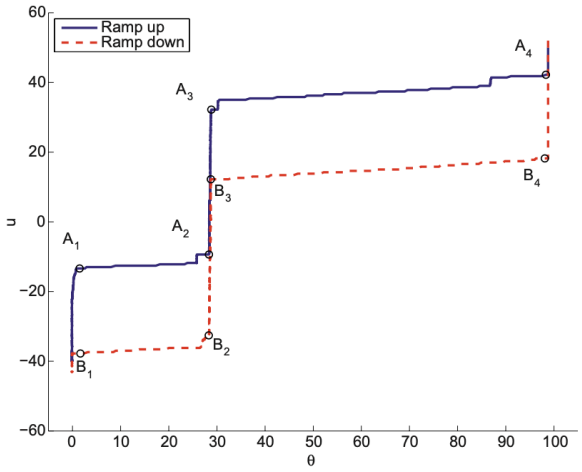


Figure 5: Parameters needed for estimations of the nonlinearities.

In the friction compensator block the coulomb friction is used to calculate the static friction, T_{fs}. Further description of how these parameters are calculated can be found in Section 4 below. The coulomb friction, T_c, is estimated as half of the distance between the curves according to the equations described in Equation (4).

$$\begin{aligned} T_c^- &= \frac{u(A_1) - u(B_1) + u(A_2) - u(B_2)}{4} \\ T_c^+ &= \frac{u(A_3) - u(B_3) + u(A_4) - u(B_4)}{4} \end{aligned} \tag{4}$$

In the limp home compensation block the parameters in Figure 3 are calculated. Further description of how these parameters are calculated can be found in Section 4 below. The equations for calculating these values are given using the equations presented in Equation (5).

$$
\begin{aligned}
\theta_{lh} &= \frac{\theta(A_2) + \theta(A_3) + \theta(B_2) + \theta(B_3)}{4} \\
\theta_{lh}^{-} &= \frac{\theta(A_2) + \theta(B_2)}{2} \\
\theta_{lh}^{+} &= \frac{\theta(A_3) + \theta(B_3)}{2} \\
m_{lh}^{-} &= \frac{u(A_2) + u(B_2)}{2} \\
m_{lh}^{+} &= \frac{u(A_3) + u(B_3)}{2} \\
k^{-} &= \frac{u(A_2) - u(A_1)}{\theta(A_2) - \theta(A_1)} \\
k^{+} &= \frac{u(A_4) - u(A_3)}{\theta(A_4) - \theta(A_3)}
\end{aligned}
\tag{5}
$$

3 HARDWARE

In this chapter all hardware used in the project is described. This includes test throttles, microcontroller actuators and sensors. The hardware parts were an essential part for being able to perform the tests later in the project, and this was thus the first system which needed to be completed. The hardware parts are connected as can be seen by the wiring scheme in Figure 6 and the different parts are described in the following sections.

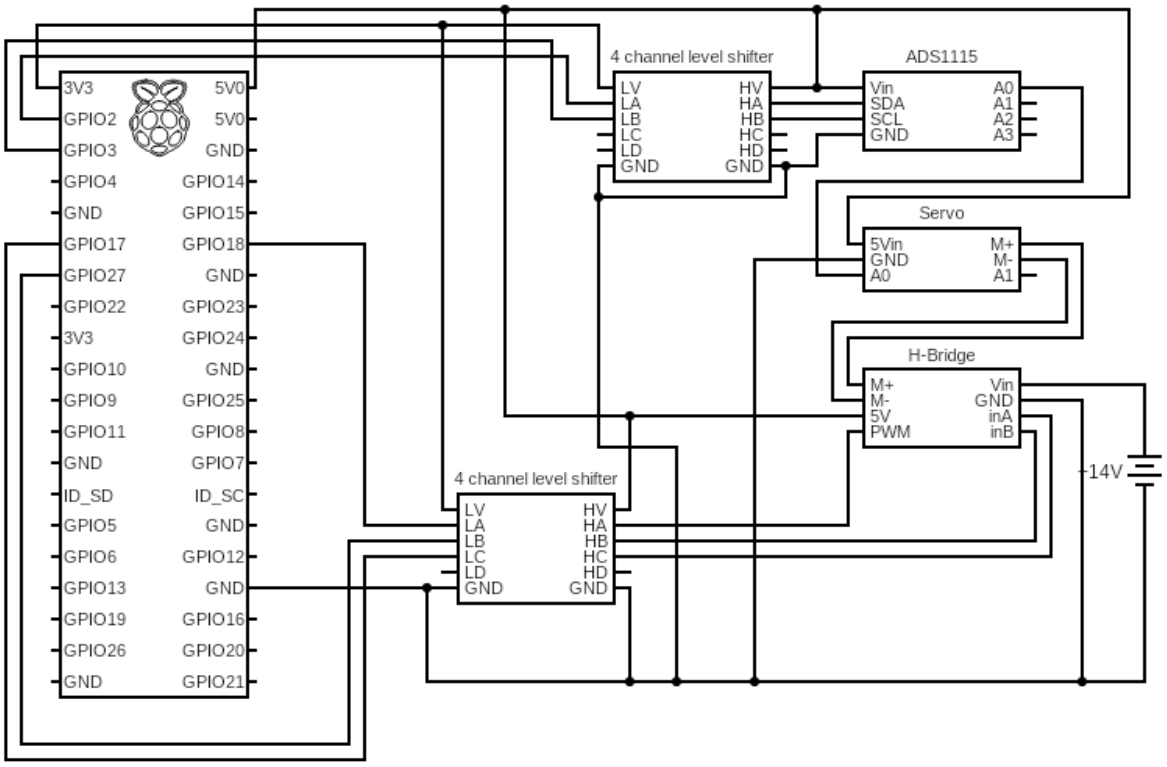


Figure 6: Schematic view of the connections between the Raspberry Pi, the throttle and the ADC.

3.1 Throttle

A main component of the project was the two throttles for measurements. The first throttle was supplied by the supervisor and it is a new, unused throttle for a Volvo VEA engine. The second throttle was provided by the customer, Aurobay, and has been used for 350,000 km in field tests.

The throttles have a motor and two potentiometers for measuring the current angle of the plate. The reason for having two sensors is to have a safety backup in real world applications. In this project we have simplified to only use one of the potentiometers since the throttle is setup in a lab environment. These potentiometers give a voltage between their respective offset voltage at fully closed up to about 5V at fully open. The offset voltage is different for the two sensors

and could differ between different throttles, hence it is a good idea to calibrate it before starting any measurements.

Each of the throttles has six connections: Two for power to the motor, one ground, one 5V power to the potentiometers and two signals from the potentiometers. The connections from the throttle can be seen in Figure 6. The motor power is connected to the H-bridge (3.2) and one of the potentiometers is connected to the ADC (3.3).

3.2 H-bridge

The H-bridge is of model AEK-MOT-2DC70S1 [2]. It can drive several motors simultaneously, but only one is needed in this project. It is capable of converting the PWM signals into a current through the motor. To work properly it needs three signals; motor clockwise, motor anticlockwise and PWM. The two first must be set opposite to each other for the motor to work. The signals to the H-bridge are delivered from the Raspberry Pi (3.4).

3.3 Analog to digital converter

The Analog to Digital Converter (hereby noted ADC) is of the model ADS1115. It is a 16-bit resolution converter with I2C communication interface [3]. For this project only one channel is needed, since only one of the two potentiometers on the throttle is used. Therefore connection A0 is connected to the throttle and the other inputs are left unconnected.

The ADC requires a stable reference voltage. The ADC is directly connected to the same voltage reference as the potentiometers to minimize differences and errors due to fluctuations and inaccuracies in voltages. The voltage for this is 5V. The ADC can only operate in discrete steps for its max voltage, and the closest above 5V is 6.144V. Also, the ADC is constructed to also handle negative voltage, why the 16-bit resolution is a bit diminished. Each step is $FS/2^{15}$ where $FS = 6.144V$ so this means the ADC has a resolution of 0.1875mV per step.

The ADC can also work in different speeds, with the fastest being 860 conversions/second. Since the control loop is set to 500Hz this is considered a resonable speed to always have new conversions ready when data will be sent to the computer.

3.3.1 *I2C for the ADC*

Specifications on how to use the I2C protocol with the ADC can be found in the datasheet [3]. The protocol is standardized, why the Raspberry Pi will have functions to handle the low level functionality. One important note is however the address selection, which is done by connecting the address pin ADDR on the ADC to either GND, VDD, SDA or SCL. It is recommended to not use SDA if possible, and there are no further requirements since no more addresses are in use. The choice was made to GND. This gives the ADC the address 1001000 in binary or 0x48 in hexadecimal.

The specific protocol to communicate and setup the ADC for this use case is here described in detail. For setup the ADC expects a write request to the configuration register followed by a write request with the data to the most significant byte and the least significant byte. The exact values can be found in the datasheet [3] and also in the code example for configuration in Listing 1.

Listing 1: Code for configuring the ADC to send continuously in 860 samples/s.

```
MSB_CONFIG_AIN0_GND = 0x40 # To use AIN0 as positive and ground as negative
MSB_CONFIG_PGA_6V = 0x00 # To use gain to 6.144V
MSB_CONFIG_MODE_CONT = 0x00 # To set the ADC in continuous mode

LSB_CONFIG_DR_860 = 0xE0 # Set speed to 860 samples/s

# Configure ADC to 475 SPS, continuous mode, single-input on channel A
configDataMSB = MSB_CONFIG_AIN0_GND + MSB_CONFIG_PGA_6V +
    ↪ MSB_CONFIG_MODE_CONT
configDataLSB = LSB_CONFIG_DR_475
configData = [CONFIG_REG, configDataMSB, configDataLSB]
# Write configuration. Should write Address, Register 0, configMSB, configLSB
pi.i2c_write_device(adc, bytes(configData))
# Set the register pointer to the conversion register
pi.i2c_write_device(adc, [CONV_REG])
```

After configuration the value in the conversion register can be read by sending a read request to the ADC. The ADC will then return the latest finished conversion.

3.4 Raspberry Pi

A Raspberry Pi 3 model B+ is used in this project. A Raspberry Pi is a small computer with general purpose input/output (GPIO) pins [4]. These GPIO pins can be configured to handle all the necessary communication between the hardware parts in the project. Some of the pins can be configured as an I2C interface to communicate with the ADC. Which they are and how they are connected to the I2C is described in Table 3. The GPIO pins can also be used to send a PWM signal, which is required to send commands to the H-bridge. PWM can be generated on all GPIO pins, so there is a free choice which one to use. GPIO18 is chosen for the PWM generation which is pin 12 on the Raspberry Pi. The H-bridge also requires two connections for motor direction, and they were chosen as GPIO17 and GPIO27 mapping to pins 11 and 13 respectively. See Figure 6 for a schematic image of the connections.

Table 3: I2C connections on Raspberry Pi.

Signal	Pin
3.3V	1
SDA1	3
SCL1	5
GND	9

The Raspberry Pi utilizes a library called *pigpio* for handling the access from Python to the GPIO pins [6]. This library includes functions for both generation of PWM signals and an implementation of the I2C protocol.

3.4.1 *Serial communication*

The Raspberry Pi uses the UART scheme to communicate with the computer through a USB port. On the Raspberry Pi the serial port used is the GPIO serial port, with TX on GPIO14 and RX on GPIO15 mapping to pins 10 and 12 respectively. The UART scheme has several parameters to set for specifying the communication and they are listed here:

- *Baudrate:* 115200 bps
- *Stopbits:* 1 bit
- *Parity bit:* None
- *Endian:* Little-endian
- *Type:* Single precision float number

The Raspberry Pi has to send the measured voltage and receive new control commands for the throttle at a speed of 500Hz to be in sync. Since the UART protocol is asynchronous some kind of timing is needed. This is implemented by letting the Raspberry Pi keep track of time and sending an angle measurement every other milliseconds (i.e. in 500Hz). The Simulink program running on the computer is also set to this update frequency to give a synchronized behaviour.

The PWM values are sent as a percentage between -100 and 100 in a 32bit float number. The measurement from the Raspberry Pi is sent as a measured voltage directly as a 32bit float number. This is in fact abundant for the angle measurement as the ADC has no more than 16 bits, but the computer can more easily work with 32 bits and it is considered not to be too much information to send. Actually one transmission consists of in total 64 bits of data which is 8 bytes. For each byte the UART protocol requires one start bit and one end bit, so 10 bits in total for each byte data sent. This means 100bits per transmission, and 500 times per second gives 50 000 bits data per second. This is far from the 115200 bits per second that should be possible to send, and thus the capacity of the UART line is not considered to be a problem.

4 CONTROL SYSTEM

The control system is composed of three major components: Limp Home compensation, Friction compensation and a modified PID controller. The following sections describe the control system in detail.

4.1 Gain scheduling

The adaptive control method gain scheduling is used for the limp home compensation and the friction compensation. The method is based on splitting of the nonlinear system into linear sub-parts at set operating points [7]. The benefit of using this compensation as an additional input to the system is to reduce the nonlinear behaviour of the system.

4.1.1 *Limp Home compensation*

The limp home compensation is used to compensate for the process gain change of moving within certain states of the system. This is a pure feed-forward component of the controller and thus does not take the current state into consideration. The limp home position is determined by two torsional springs, possibly with different characteristics. The influences of the springs are therefore separated into four piece-wise linear sections and compensated by a piece-wise linear controller output accordingly. The four partitions are visible in the 'if' and 'elseif' statements. The condition not mentioned is the case when $\theta_{ref} < \theta_{lh}^{-}$. These partitions can be considered as linear approximations of the nonlinear system in those regions. Below is the Matlab code for calculating the appropriate limp home compensation.

Listing 2: Limp Home compensation.

```
function Ts = limphomeComp(theta_ref, theta_lhplus, theta_lh, theta_lhminus,
k_plus, k_minus, m_lhplus, m_lhminus)

if (theta_ref > theta_lhplus)
    Ts = m_lhplus + k_plus*(theta_ref - theta_lhplus);
elseif ((theta_ref > theta_lh) && theta_ref <= theta_lhplus)
    Ts = m_lhplus*(theta_ref - theta_lh)/(theta_lhplus - theta_lh);
elseif ((theta_ref > theta_lhminus) && theta_ref <= theta_lh)
    Ts = m_lhminus*(theta_lh - theta_ref)/(theta_lh - theta_lhminus);
else
    Ts = m_lhminus - k_minus*(theta_lhminus - theta_ref);
end
```

The parameters in the function are determined experimentally and calculated from the spring torque in the throttle model described in Section 3.1. Correct identification of the points θ_{lh}, θ_{lh}^{+} and θ_{lh}^{-} are therefore critical for the performance of the controlled system.

4.1.2 *Friction compensation*

The friction is also modelled as piece-wise linear, hence the friction compensation is a piece-wise affine function. The function is split into six cases. Firstly, if the current error signal is less than a predetermined value θ_d, no friction compensation is performed and the PID controller is expected to compensate for the error. The following cases are split into above and below the limp home position and further into whether the absolute error position is less than

or greater than the sum of the reference and dead zone. The last case, if none of the above conditions are true, no compensation is given and the PID is expected to adjust for the error. Below is the code for the friction compensation.

Listing 3: Friction compensation.

```
function Tf = fricComp(theta, e_theta, theta_d, theta_r, Tc_plus,Tc_minus,theta_lh, ktf)
if (abs(e_theta) <= theta_d) %Deadzone
    Tf=single(0);
elseif (abs(e_theta) <= (theta_r + theta_d)) && (theta > theta_lh) %Smooth transition over lh
    Tf = single(sign(e_theta)*Tc_plus*ktf*(abs(e_theta) - theta_d)/theta_r);
elseif (abs(e_theta) > (theta_r+theta_d)) && (theta > theta_lh) %Friction over lh
    Tf=single(Tc_plus*ktf*sign(e_theta));
elseif (abs(e_theta) <= (theta_r + theta_d)) && (theta <= theta_lh) %Smooth transition under lh
    Tf = single(sign(e_theta)*Tc_minus*ktf*(abs(e_theta) - theta_d)/theta_r);
elseif (abs(e_theta) > (theta_r+theta_d)) && (theta <= theta_lh) %Friction under lh
    Tf=single(Tc_minus*ktf*sign(e_theta));
else
    Tf=single(0);
end
```

4.2 PID Controller

In addition to the limp home compensation and friction compensation, a PID controller is implemented. The PID controller compensates for any remaining errors and for uncertainties in the linearization used in the compensation blocks. The Simulink implementation of the PID controller is displayed in Figure 7. Some modifications are made to the controller to further increase the performance for this specific task, and are described in the following sections.

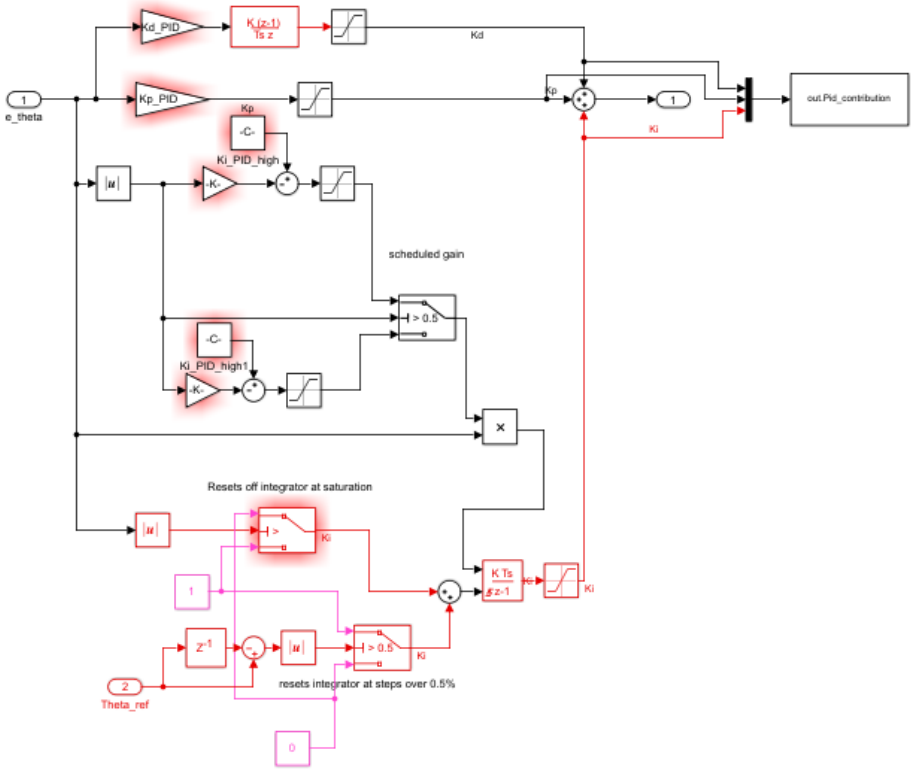


Figure 7: PID controller implementation in Simulink.

4.2.1 *Integrator*

The integrator differs from a standard PID integrator. The integrator is largely implemented as outlined in [1]. The integrator has a scheduled gain K_i which is smaller for greater errors and greater for smaller errors. The exact scheduled gain is outlined in Equation (6).

$$K_i = \begin{cases} 1 & \text{if} \quad |e_\theta| \geq 10\% \\ 11 - |e_\theta| & \text{if} \quad 10\% > |e_\theta| \geq 1\% \\ min(100, 210 - 200|e_\theta|) & \text{if} \quad 1\% > |e_\theta| \end{cases} \tag{6}$$

The integral part also resets when the reference signal (θ_{ref}) takes a step greater than 0.5% or when the error ($|e_\theta|$) is smaller than 0.125%. This is to avoid windup and oscillations around the reference.

4.2.2 *Proportional and Derivative*

The derivative gain K_d and proportional gain K_p were tuned by the same strategy as outlined in [1]. The strategy was to tune the design parameter λ and then slowly increment the derivative gain K_d until a satisfactory control behaviour was achieved. The proportional gain was calculated using Equation (7) [1].

$$\lambda = \frac{1 + K_d K_0}{K_p K_0} \tag{7}$$

The parameter K_0 is the static gain of the throttle.

5 PARAMETER IDENTIFICATION

This chapter describes the methods used to try to find the friction parameters of the throttle from recorded data.

5.1 Calibration

Before any tests could be conducted some calibration was needed to find the voltages at fully closed, fully open and limp home position. A calibration to find the static hysteresis boxes was also conducted. To find the voltages for the extreme values 10 samples were taken and averaged. This was at first done without any signal to the motor to be able to find the voltage for the limp home region. Afterwards the PWM signal was set to 25% which was considered high enough to be sure the plate would reach its end position. After 2 seconds the value was measured using again 10 averaged measurements. Finally the throttle was given commands to go back to limp home for 2 seconds and then it was given a signal of -25% to be sure it would go to the closed position. This was also sent for 2 seconds before measuring the value using ten averaged measurements.

To find an estimation of the static behaviour of the throttle, comparable to Section 2.1 a ramping was made as a part of the calibration. As the goal was to simulate a static behaviour this ramping was performed by increasing the PWM signal in small steps and measuring the resulting angle. More exactly the PWM signal was increased by 0.5% per second from 0 up to 30% and thereafter decreased down to -30%. Finally the ramp was increased back to 0%. The value 30% was chosen to be large enough to be sure the throttle would go to maximum position in all cases, but stationary not large enough to risk burning the motor. An example from the slow ramping calibration procedure can be seen in Figure 8 and can be compared to the considered model from the theory in Figure 5.

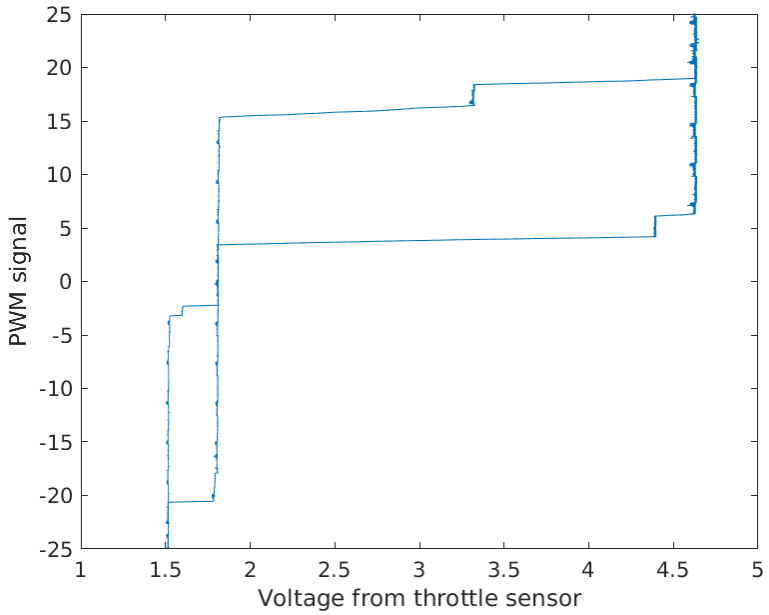


Figure 8: Plot from a slow calibration ramp to show the static behaviour of the throttle.

5.2 Data classification

While driving, the computer can continuously measure input signal to and current angle from the throttle. These two measurements can be used together to classify whether the throttle is moving or not. In a perfect world without noise or time delay this could be done by directly measure the difference in adjacent angles, and noting the throttle as moving if the value is not zero. This could then be connected to the corresponding input signal to classify a point as moving or stationary. In reality there are some more things to think about.

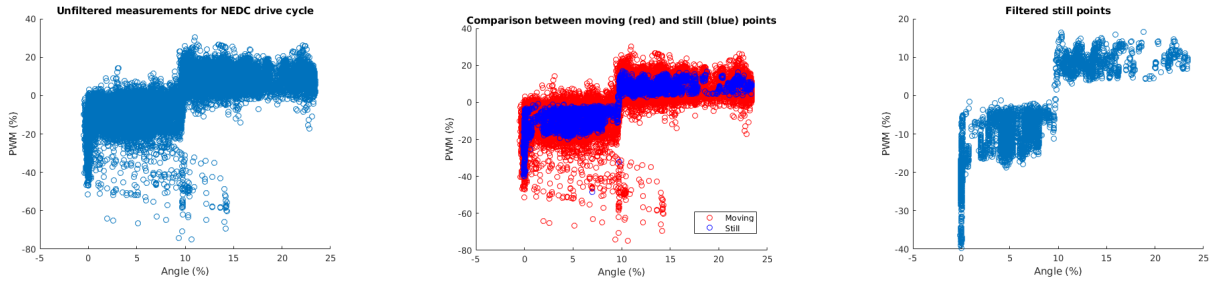
Firstly, the measurements are noisy. To solve this a threshold was introduced, and the angle measurements were also filtered through a moving mean filter. The chosen value for the threshold and the window width can be found in Table 4.

Secondly, there is a time delay from given input signal until the effect is visible in the output. This was measured in a step response experiment to be about 13 samples (corresponding to 26ms in 500Hz). To compensate for this the angle measurements were shifted 13 samples for the computation. To have some margin for errors in time delay some delays around were also checked. This results in some points being wrongly classified as moving when they are stationary, but it is still better than the other way around. The chosen values for delays to consider can be found in Table 4.

Thirdly there is also a problem with the input signal changing faster than the throttle can react, thus making some input signals wrongly classified as being stationary. To solve this a threshold for the input signal was also introduced. This threshold compares the input signal to the following input signal and discards the point if the difference is too large. The chosen value for the input threshold can be found in Table 4.

Table 4: Tuning variables for classification algorithm.

Variable	Value	Description
Angle threshold (%)	0.01	Min value to classify adjacent measurements as a moving point
Window width (samples)	5	Number of measurements to average over
Time delays (samples)	10, 11, 12,13,14, 15	Delays to consider for checking if a point is moving
Input threshold (%)	1	Limit to discard input signal as changing to fast



(a) All data points from a simulation with the NEDC cycle.

(b) Moving points (red) compared to stationary points (blue).

(c) Data points from the NEDC cycle filtered as stationary points.

Figure 9: Comparison between unfiltered and filtered data for a simulation with the NEDC drive cycle.

The result from the filtering can be seen in Figure 9. The boxes are clearly visible in the filtered data (Figure 9c) and it is clear that the algorithm is removing the points on the edges, where we know the throttle can not be stationary. This is further seen in the comparison between the stationary and moving points (Figure 9b) from where it is clear that the filtering keeps points in the middle. To conclude it seems possible to distinguish the area where the throttle can stand stationary while driving. The problem is to extrapolate these areas (especially to higher angles where there is a lack of points) and to have a satisfactory model for finding the boxes. A suggestion on a solution to the aforementioned problems will be discussed in the coming sections.

5.3 Classification using machine learning

Looking at a plot of control signal as a function of throttle angle, like in Figure 9a it is quite clear that some form pattern exists. As described in Section 2.1 an idea is that data points when the throttle is moving are outside of the hysteresis boxes while stationary data points are within. However, in Figure 10 and Figure 11 below it can be seen that is not the case. The majority of data points classified as moving are within what looks like the hysteresis boxes. Data points classified as stationary are however almost always within the boxes.

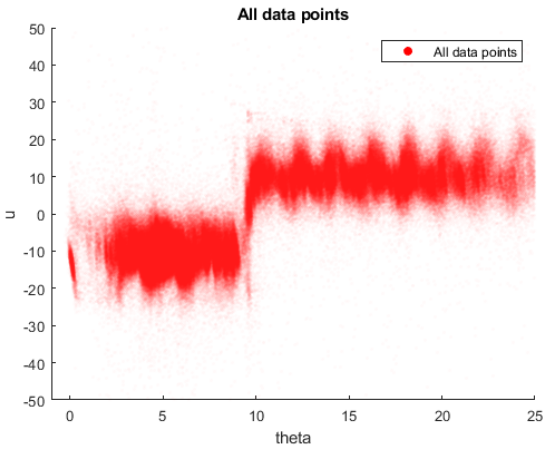


Figure 10: All data points from an Aurobay drive cycle.

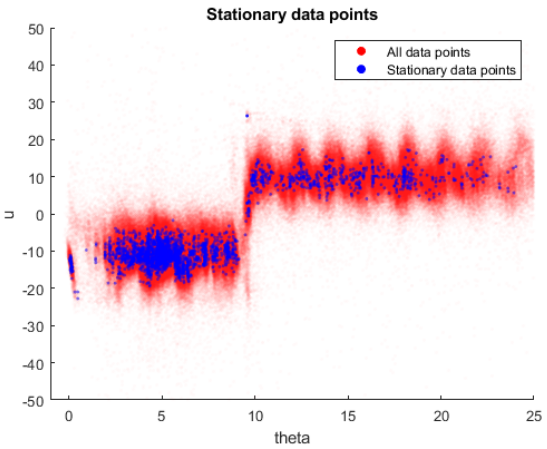


Figure 11: All data points and all points classified as stationary.

This data can be used in different machine learning classification algorithms. The features (input data) are the control signal and measured angle, the target is weather the data point is classified as moving or not.

The different classifiers tested are

- Neural Network, Multi-layer Perception (MLP)
- K-nearest neighbors
- Linear Support Vector Machine (Linear SVM)
- Radial Basis Function Support Vector Machine (RBF SVM)

- Gaussian Process
- Decision Tree
- Random Forest

5.4 Analytical approach to the parameter identification

As the moving and stationary data points have a very large overlap, classification and identification of the hysteresis boxes can be difficult. The moving data points are the most problematic since they exist both within and outside the hysteresis boxes. The stationary data points are less noisy and therefore easier to analyze. A plot where only the stationary data points are included can be seen in Figure 9.

This data is divided into three sections to be analysed and fitted into hysteresis boxes, the limp home region (around 9°), the region below and the region above. In the initiation process of the simulation, the throttle does a series of ramps to find the limp home, max and min voltage from the potentiometer, which is translated to throttle angle. Detailed description can be found in Section 2.1. Data points to be analysed as within the limp home region are selected according to Equation (9).

$$\theta_{limphome} = (V_{limphome} - V_{min}) * \frac{100}{V_{max} - V_{min}} \tag{8}$$

$$\theta_{lh-section} = [\theta_{limphome} - offset, \quad \theta_{limphome} + offset] \tag{9}$$

An offset of 1% is sufficient to include the majority of data points as seen in Figure 12.

The algorithm is very similar below and above the limp home region, only mirrored. Firstly the data points to be analysed are filtered further. When the throttle closes rapidly and hits its end position stationary data points are generated even though a large negative PWM signal is applied. These needs to be filtered and should not be considered when calculating the parameters. Simply picking points between $+0.2°$ and the limp home angle is sufficient. For the upper region data points between the limp home angle and the 99th percentile are selected to filter out any extreme outliers. These regions can thereafter be divided into segments along the x-axis. Upper and lower limits for these regions can be calculated using percentiles and lines can be fitted to these points. The resulting plot contains all necessary information to calculate the parameters for the limp home and friction compensation. In Figure 15 the smaller sections can be seen as colored points, the upper and lower limits of these sections as short red lines and the resulting thicker fitted line.

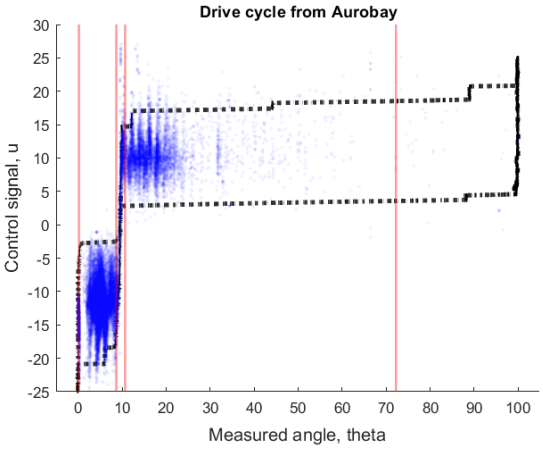


Figure 12: Boundaries dividing the three regions.

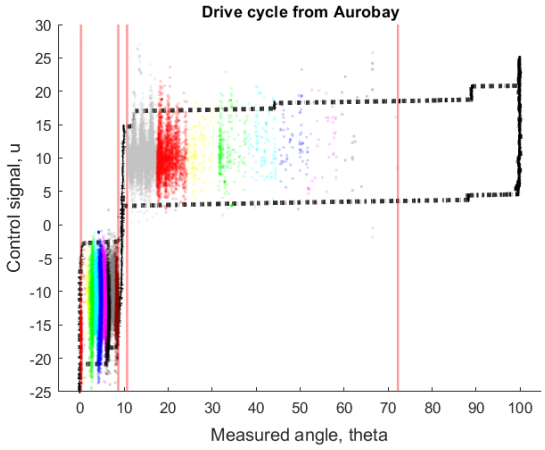


Figure 13: Each region is divided into sections.

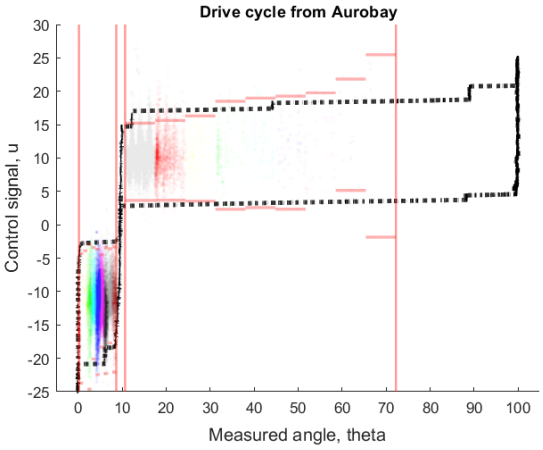


Figure 14: The upper and lower limit of each section is found as the 99th percentile.

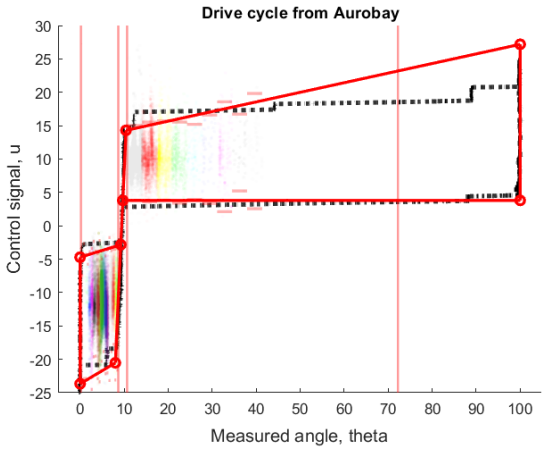


Figure 15: Lines are fitted to each series of limits.

6 RESULTS AND DISCUSSION

The project has a number of requirements on the constructed systems, and the results from testing these requirements are here shown and discussed.

6.1 Requirements on throttle regulation

The throttle control developed during the project needs to fulfill some requirements on the reference following. These requirement were provided by the supervisor and are stated in the *Requirement Specification* document.

If using the machine learning parameter estimation the requirements is partially fulfilled. The results from these step-responses can be seen in Table 5 and Table 6. If on the other hand, the parameters are estimated manually a higher rate of fulfilling the requirements is obtained. The one thing not fulfilled using this approach is the settling time. The results for these step-responses can be seen in Table 7 and Table 8. The Figures 16-19 show the step-responses these results were gathered from.

The different estimated parameters are outlined in Table 11 in Section 6.6.

Requirement	Value	Result
Settling time (±5%)	< 200ms	168ms
Overshoot	< 0.5%	1.38%
Static error	< 0.125%	0.591%
Tracking error	< 2.5%	1.28%

Table 5: Requirement for step > 50% reference signal change using the machine learning parameter estimation.

Requirement	Value	Result
Settling time (±5%)	< 100ms	118ms
Overshoot	< 0.125%	0.884%
Static error	< 0.125%	0.2865%
Tracking error	< 2.5%	1.93%

Table 6: Requirement for step < 50% reference signal change using the machine learning parameter estimation.

Requirement	Value	Result
Settling time (±5%)	< 200ms	274ms
Overshoot	< 0.5%	0.172%
Static error	< 0.125%	-0.0397%
Tracking error	< 2.5%	0.256%

Table 7: Requirement for step > 50% reference signal change using manual estimation.

Requirement	Value	Result
Settling time (±5%)	< 100ms	184ms
Overshoot	< 0.125%	0.4578%
Static error	< 0.125%	-0.0487%
Tracking error	< 2.5%	0.272%

Table 8: Requirement for step < 50% reference signal change using manual estimation.

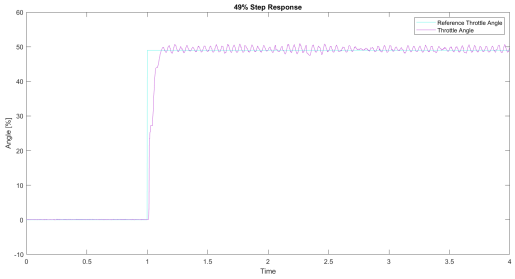


Figure 16: The 49% step response of the throttle control using machine learning estimation.

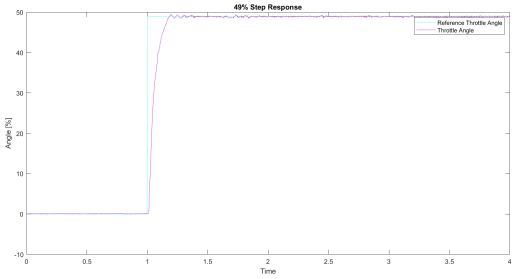


Figure 17: The 49% step response of the throttle control using manual estimation.

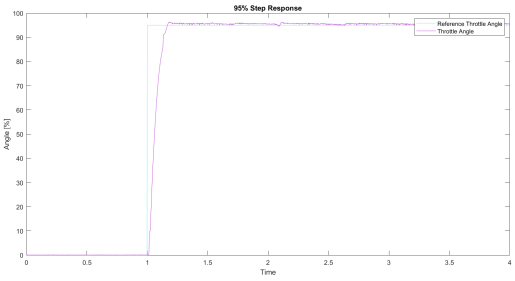


Figure 18: The 95% step response of the throttle control using machine learning estimation.

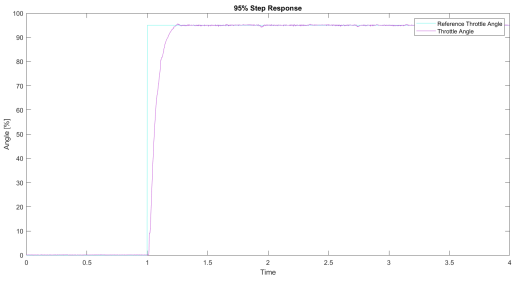


Figure 19: The 95% step response of the throttle control using manual estimation.

The performance was greater for the throttle control with manual estimation. This could be caused by the manually estimated parameters being more accurate than if estimated using ML. It could also be caused by the difference in the tuning of the controllers as the two controllers were independently tuned after respective parameter estimation.

Tables 5 to 8 show that most requirements are unfulfilled with ML parameter estimation. Some additional requirements are fulfilled with manual parameter estimation. The reason for this sub-par performance can likely be explained by a significant input delay of the controller (see Section 8.3 for further explanation). Other reasons may be errors in parameter estimations and tuning. The controller can obtain a satisfactory reference following for drive cycles despite not living up to every requirement as outlined in Section 6.2.

6.2 Drive Cycle Performance

This section presents the performance of the adaptive PID controller explained in Section 4.2 and 4 on different drive cycles. Note that the parameter estimation was done manually for all results in this section. The design parameters used to obtain the results presented in this subsection are displayed in Table 9.

The controller performance for different drive cycles can be seen in Figure 21 to Figure 23.

Table 9: Values of different design parameters used for the adaptive throttle control.

Parameter	Value
λ	0.3
K_0	2.8424
K_p	1.3061
K_d	0.04
θ_d	0.03
θ_r	0.6

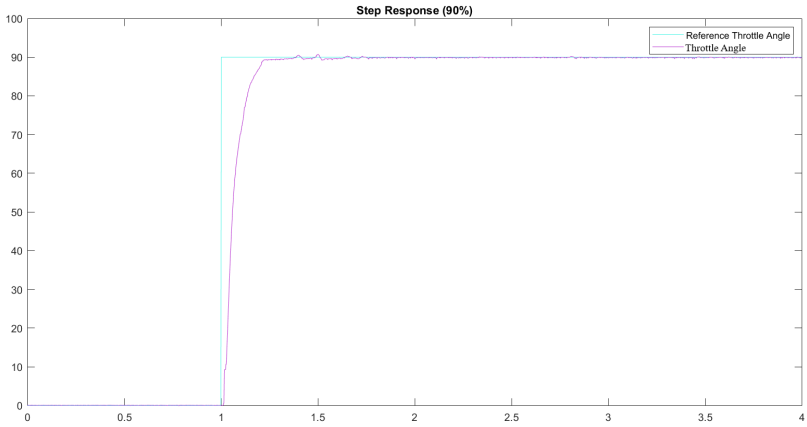
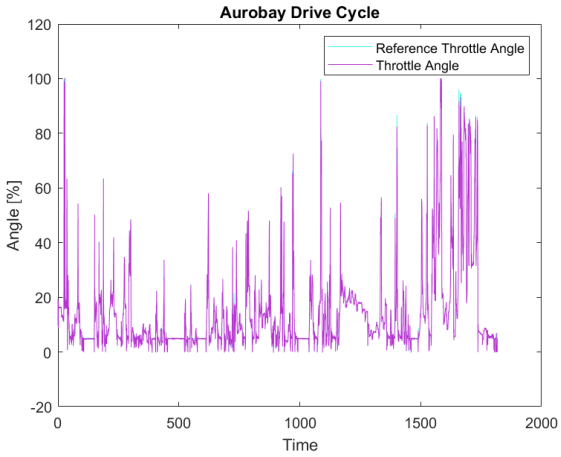
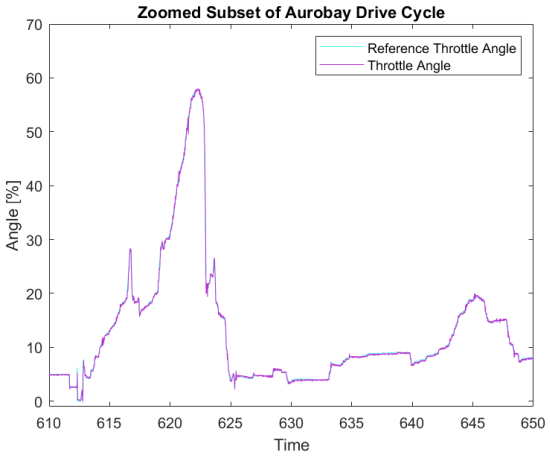


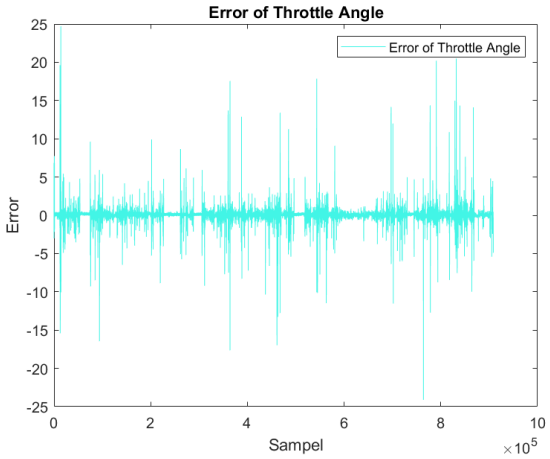
Figure 20: The performance of the adaptive throttle control with a 90% step response.



(a) The performance of the adaptive throttle control when applied to the Aurobay drive cycle.

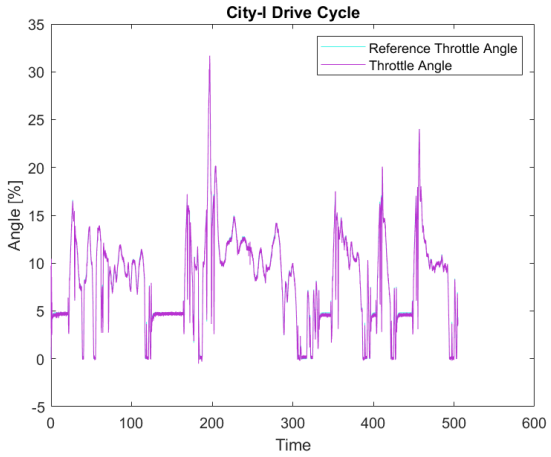


(b) The performance of the adaptive throttle control when applied to the Aurobay drive cycle. Zoomed in closer to provide better resolution.

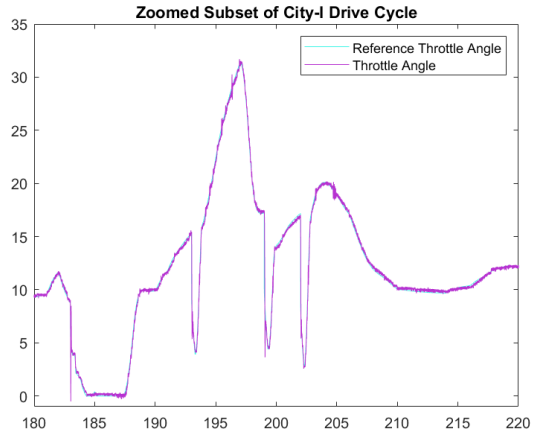


(c) The angle error at each sample during aurobay drive cycle.

Figure 21: Figures showing the reference following for the new throttle.

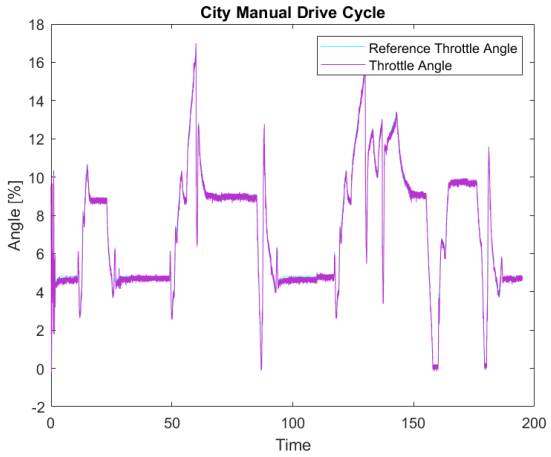


(a) The performance of the adaptive throttle control when applied to the City-I drive cycle.

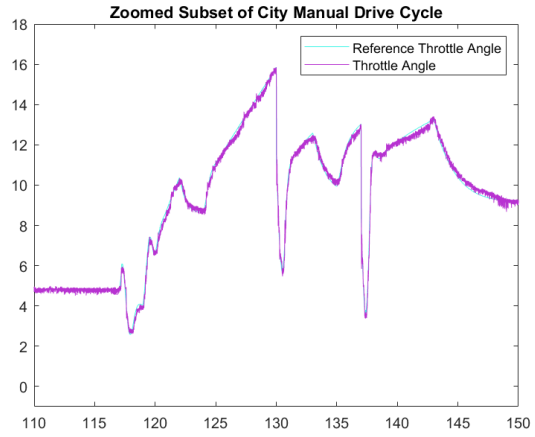


(b) The performance of the adaptive throttle control when applied to the City-I drive cycle. Zoomed in closer to provide better resolution.

Figure 22: Figures showing the reference following for the new throttle.

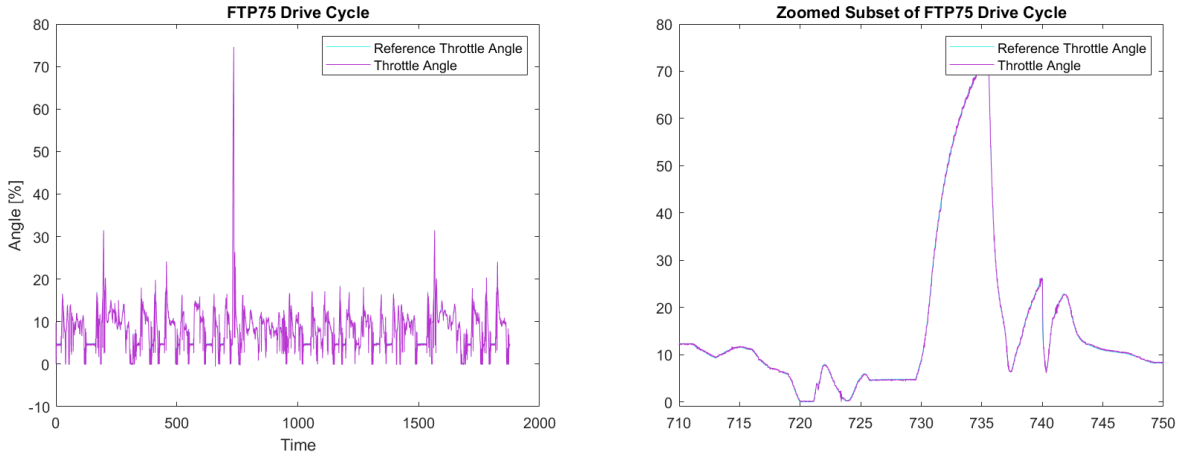


(a) The performance of the adaptive throttle control when applied to the City Manual drive cycle.



(b) The performance of the adaptive throttle control when applied to the City Manual drive cycle. Zoomed in closer to provide better resolution.

Figure 23: Figures showing the reference following for the new throttle.



(a) The performance of the adaptive throttle control when applied to the FTP75 drive cycle.

(b) The performance of the adaptive throttle control when applied to the FTP75 drive cycle. Zoomed in closer to provide better resolution.

Figure 24: Figures showing the reference following for the new throttle.

The average L2-norm of the error (the L2-norm of the error divided by amount of samples) for each drive cycle is displayed in Table 10.

Table 10: Values of different design parameters used for the adaptive throttle control.

Drive Cycle	Average L2 Norm of Error
Aurobay	$0.601 \cdot 10^{-3}$
City-I	$0.604 \cdot 10^{-3}$
City Manual	0.01
FTP75	$0.289 \cdot 10^{-3}$

These results show a reasonable reference following of the controller. Even when the controller was applied to a highly variable drive cycle such as the *Aurobay drive cycle*. The errors during the *Aurobay drive cycle* could be relatively big (see Figure 21c) but still reasonable low considering the size of the sudden spikes in the drive cycle.

6.3 Temperature dependence

To examine friction changes depending on temperature some slow ramps were conducted. The result for three different temperatures can be seen in Figure 25. There are no clear differences between the ramps, and the small changes that can be seen are considered to be caused from stick-slip friction. Also when comparing a drive cycle in different temperatures the difference in rather small, as can be seen in Figure 26.

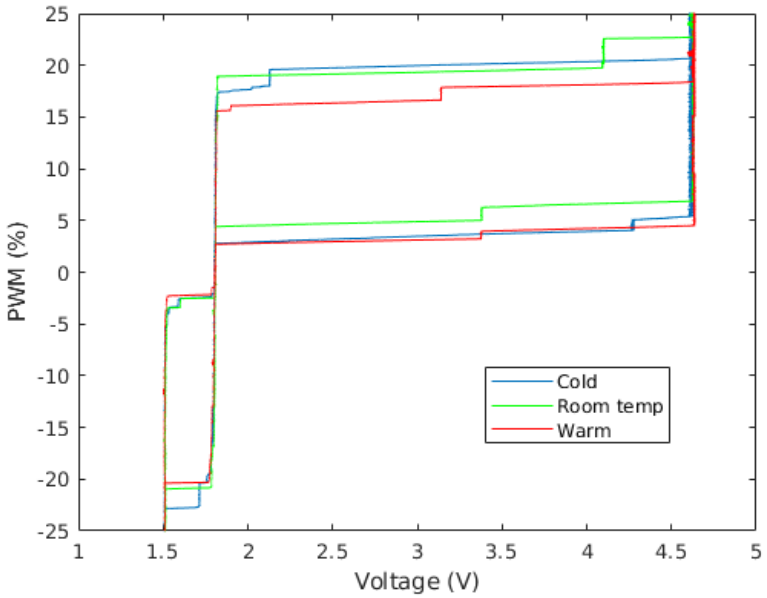


Figure 25: A comparison between slow ramps (0.5% PWM per second) for a cold, warm and room temperature throttle. The small nonlinearities in the different parts are considered to be caused by stick-slip friction.

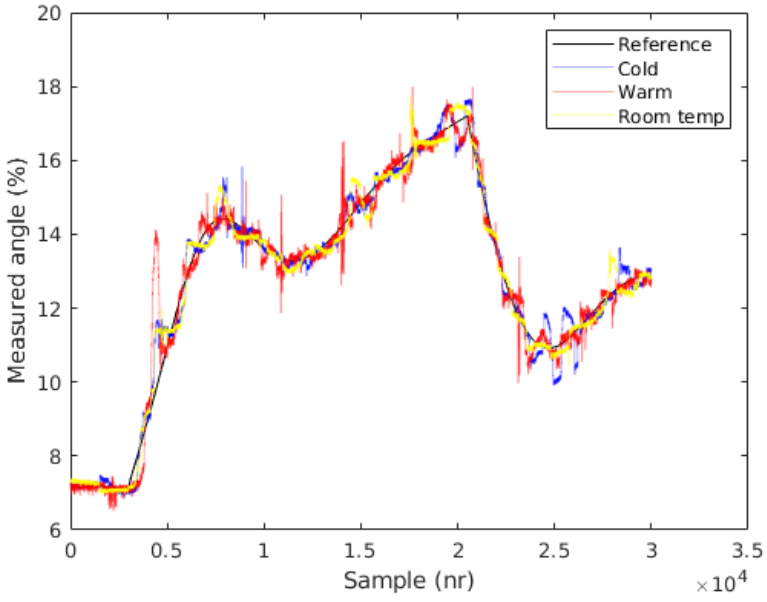


Figure 26: A slice of a test with the NEDC drive cycle for three different temperatures.

6.4 Old vs New Throttle

As mentioned in Section 3.1 both an old and new throttle is provided for comparison between them, and how the performance is affected due to how used they are. In Figure 27 the result of the reference following for the two throttles can be seen.

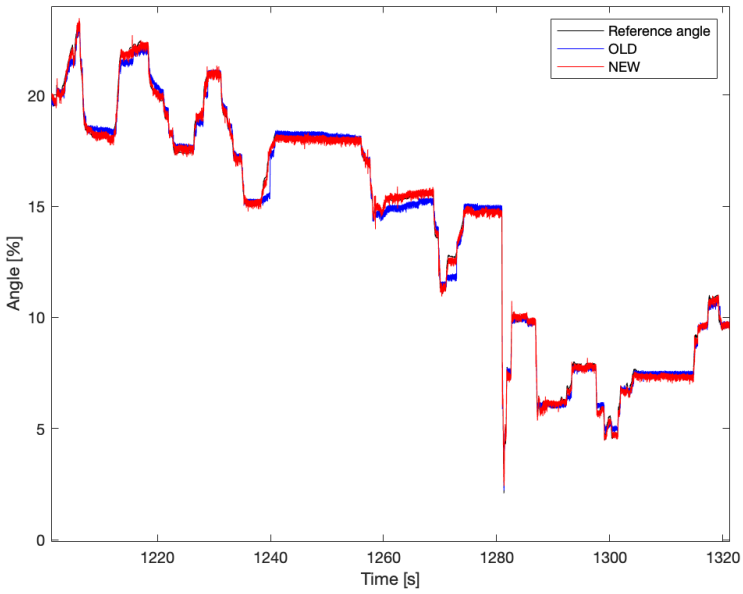


Figure 27: A slice of a test with the Aurobay drive cycle for the old and new throttle.

It is apparent in Figure 28 that the hysteresis is noticeably different for the new and the old throttle. However, this difference in hysteresis is not significant enough to make a difference during drivecycle and reference following tests.

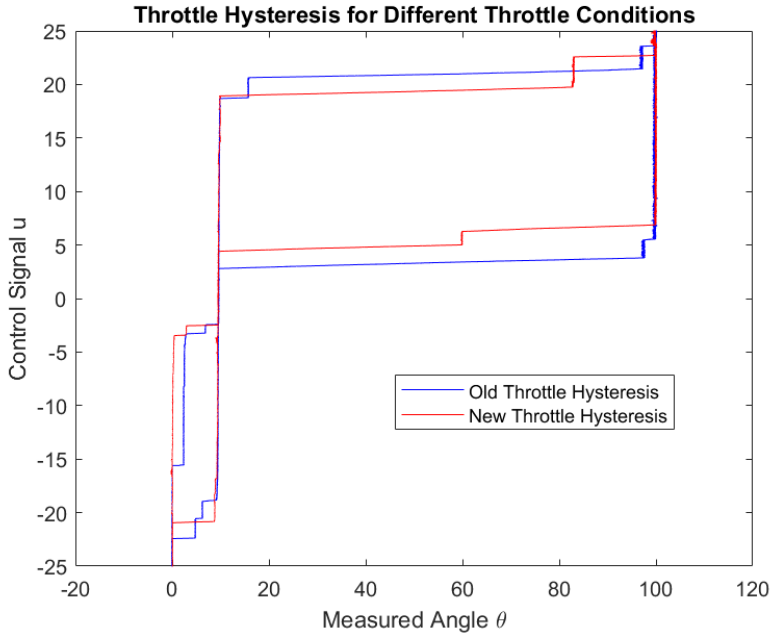


Figure 28: A comparison between slow ramps (0.5% PWM per second) for a old and new throttle. The small nonlinearities in the different parts are considered to be caused by stick-slip friction.

6.5 Classification Using Machine Learning

Seven different machine learning algorithms where tested. The goal as previously mentioned is to train a model to accurately classify and the hysteresis boxes, see Section 2.1. Recorded data from the NEDC drive cycle is used for all models and the data classification method described in Section 5.2 is used to get benchmark data for the model training and validation. The result from the initial attempt of model training can be seen in Figure 29.

Figure 29: All machine learning classification methods explored. The score of each method can be seen in the bottom right corner of each plot.

As seen from the Figure 29 above, none of the methods are capable of correctly identifying the hysteresis boxes. Nearest neighbors, Gaussian process, Linear SVM and Neural network achieve the highest score of .56. This is an extremely poor score, meaning the trained models are only slightly better than randomly guessing. This result is however expected when further analysing the data points used for the machne learning. Figure 9 displays the distribution of data points from the NEDC cycle. It is clear that there is a large overlap between data points classified as stationary and moving. Using this data for model training will most certainly fail when the goal is to create the hysteresis boxes, the data is simply too noisy. It should be mentioned that three methods where further investigated, Nearest neighbors, non-linear SVM and Neural network. However only slight improvements could be seen. It was concluded that the data used for the machine learning models is not well suited for this application.

The idea of using some form of machine learning is however not completely rejected. From the distribution of data points in any drive cycle the hysteresis boxes can clearly be seen by eye, see Figure 9. The data used for the machine

learning attempts is, as mentioned, not fitted for these types of methods. An analytical approach to the problem is discussed in the following Section 6.6.

6.6 Analytical classification for different drive cycles

Instead of using existing machine learning methods, which are very general, a more analytical approach of the data is explored. Again the drive cycle provided from Aurobay is used since it excites the system the most. The table below shows the resulting parameters from the manual selection and automatic.

Parameter	Manual	Automatic
theta_lh:	9.4	9.4191
theta_lhminus:	9.75	8.7052
theta_lhplus:	9.23	10.1330
m_lhminus:	-11.7	-10.1704
m_lhplus:	9.97	9.4575
k_minus:	0.0211	0.0134
k_plus:	0.01073	0.1788
Tc_plus:	7.2	5.0792
Tc_minus:	9	6.9660

Table 11: Parameters for friction and limp home compensation.

In the Table 11 it can be seen that most parameters are similar. Both the manual and automatic method finds the same limp home angle, the limp home region is larger from the script. The m-values and k-values are very similar. The biggest difference is in the Tc-value which are $\approx 25\%$ smaller than the values from the manual identification. Figure 30 shows the plotted hysteresis box from the script.

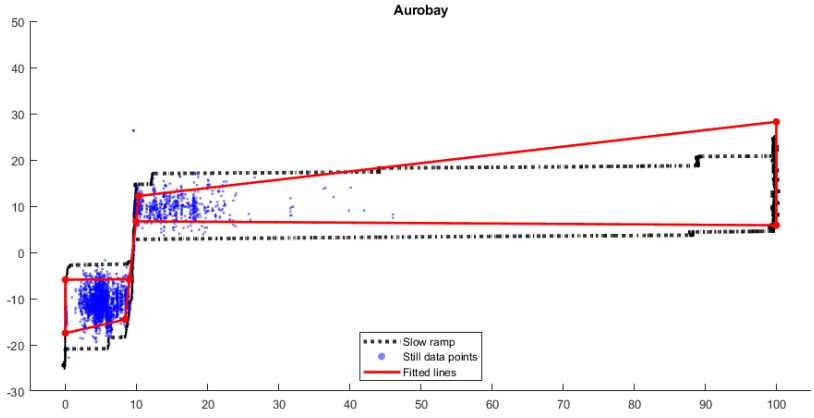


Figure 30: Resulting calculated hysteresis boxes from the Aurobay drive cycle.

6.6.1 *Performance on different drive cycles*

The script is tested on multiple different drive cycle to analyze the performance.

Table 12: Calculated friction and limp home compensation parameters from different drive cycles.

	Drive cycle			
Parameter	**Aurobay**	**City Manual**	**City I**	**FTP-75**
Average L2 Norm:	$0.60 * 10^{-3}$	$1.03 * 10^{-3}$	$0.60 * 10^{-3}$	$0.29 * 10^{-3}$
theta_lh:	9.4191	9.5102	9.5150	9.5731
theta_lhminus:	8.7052	8.7733	8.8025	8.8657
theta_lhplus:	10.1330	10.2470	10.2274	10.2805
m_lhminus:	-10.1704	-14.2663	-10.2345	-10.7170
m_lhplus:	9.4575	9.2657	9.6911	9.7275
k_minus:	0.0134	-0.7373	0.1951	0.2196
k_plus:	0.1788	0.4135	0.3005	0.5163
Tc_plus:	5.0792	2.7657	3.9800	6.2001
Tc_minus:	6.9660	0	10.5478	31.5517

As seen in the first row of Table 12 as well as Figure 31 the performance in the different drive cycles is good. The control parameters for the friction and limp home compensation varies a significant amount in the different cases. As seen in Table 12 the k-values and Tc-values vary a lot. Running the throttle on these values would result in worse performance since the PID-controller is tuned for parameters closer to those of the Aurobay drive cycle. The main problem is a lack of data. As seen in Figure 32 the amount of blue data points vary a lot between the different cycles. This is caused by two factors, primarily the length of the cycle as well as the level of excitation. The City Manual cycle is the shortest of only 200 seconds and also the least aggressive of the four which results in a poorly fitted hysteresis box. The City I cycle is better since it is more than twice as long, though it is still too slow of a cycle to generate large reference angles. This result is expected since the algorithm to calculate the hysteresis boxes uses the density of data points to fit linear approximation.

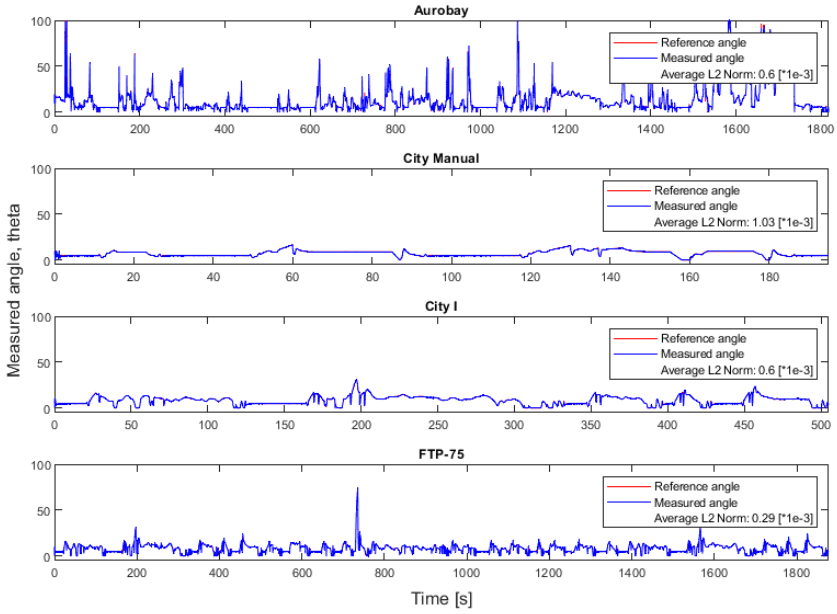


Figure 31: Reference and measured angle for different drive cycles.

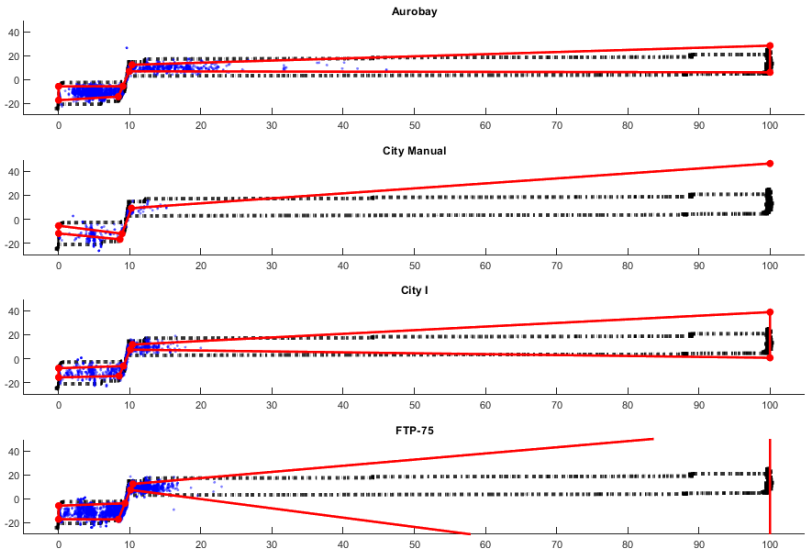


Figure 32: Calculated hysteresis boxes for different drive cycles.

6.7 Sensitivity analysis of friction and limp home compensation

A sensitivity analysis on the parameters of the friction and limp home compensation is performed. The analysis is done using the Aurobay drive cycle since it is the cycle which excites the system the most. Furthermore the analysis is done without the PID-controller enabled. If the PID-controller was enabled it would correct some of the faults caused by bad parameters which would make the sensitivity analysis less accurate. The parameters for the friction and limp home compensation can be seen in Table 12 under the Aurobay column.

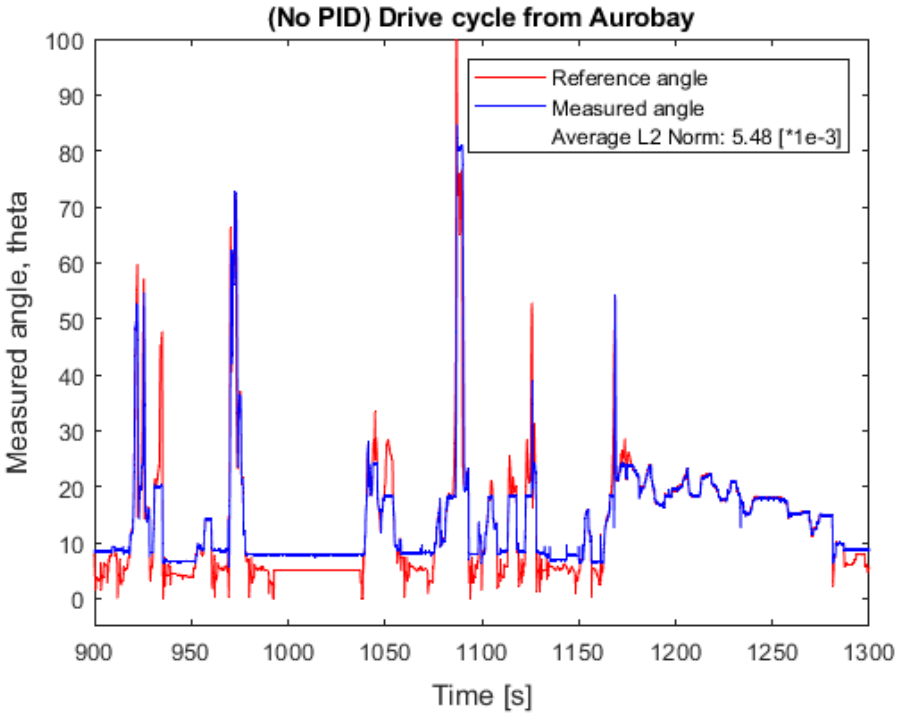


Figure 33: A section of the Aurobay drive cycle. Using friction and limp home parameters according to the method described in Section 5.4.

Without the PID-controller enabled the throttle struggles to follow the reference signal at high and low angles. Which is to be expected since it is designed to handle the non linear behaviour around the limp home region. Studying the PWM signal sent to the electrical motor of the throttle it never exceeds $\approx \pm 20\%$, which is not enough to fully open or close the throttle. In Figure 33 it can be seen that the measured signal follows the reference well slightly below and above the limp home angle while it struggles at the extreme angles. The average L2 norm of the error is $5.48 * 10^{-3}$, this value itself means little to nothing but can be used as a benchmark for comparison.

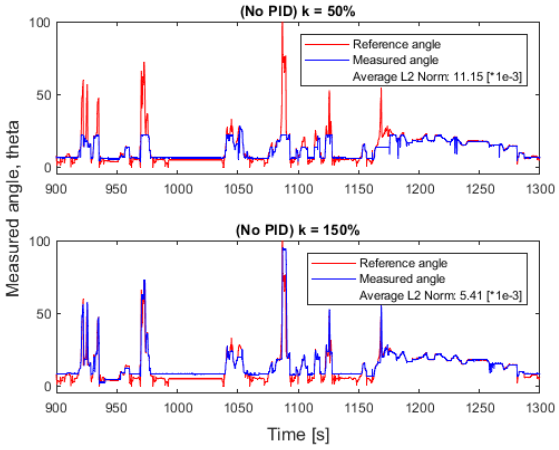


Figure 34: Varying the k_minus and k_plus parameter.

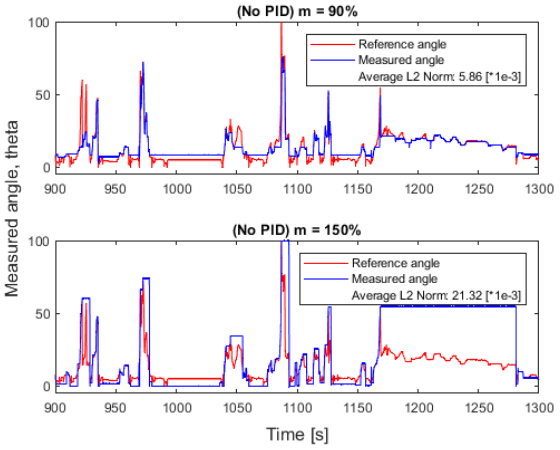


Figure 35: Varying the m_lhminus and m_lhplus parameter.

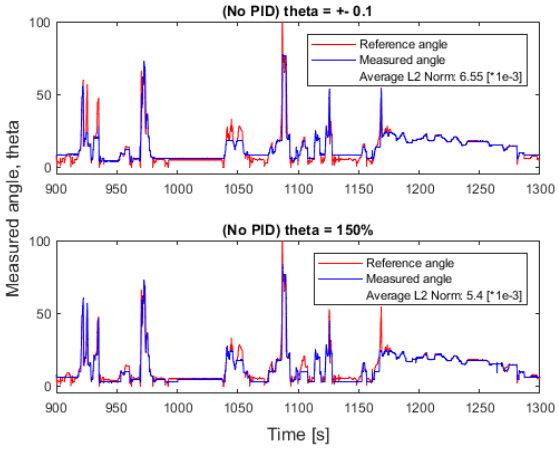


Figure 36: Varying the theta_lhminus and theta_lhplus parameter.

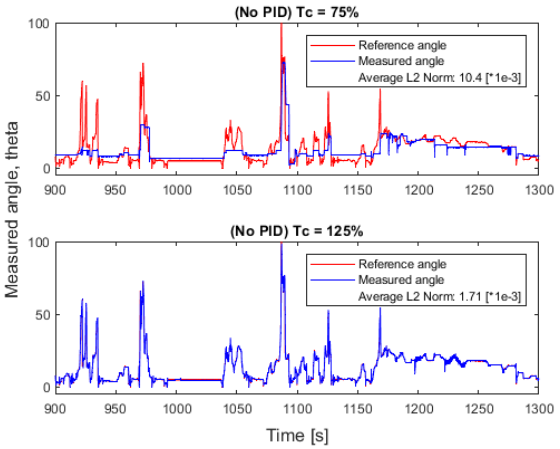


Figure 37: Varying the Tc_minus and Tc_plus parameter.

The result from the sensitivity analysis can be seen in the figures above. The value of the k parameter has a low impact on the performance of the throttle. It can be increased or decreased by 50% without major performance loss. With a smaller k-value the throttle follows small angles better while a large k-value follows large angles. The throttle is more sensitive to the m parameter, especially when decreasing it. If the m-value is too small the throttle starts to oscillate. It could only be decreased by 10% before major performance losses starts to occur. Increasing the m-value results in large control signals and poor reference following. The theta-value has a small impact on the performance. The region can be close to zero or 50% larger without large performance losses. The system is very sensitive to the value of the Tc parameter. The Tc-value is directly proportional to the control signal from the friction compensation according to Section 4.1.2. Thus varying this value has a direct impact on the performance. Decreasing it results in

smaller PWM signals, which results in worse reference following of high and low angles. While increasing it results in better performance. This is however not a perfectly fair representation of the entire system since the PID-controller is disabled. A larger Tc-value would result in a more occilative behaviour since the PID-controller is tuned to a certain Tc-value. Therefor it is not accurate to assume that a larger Tc-value results in better performance.

The result from the hysteresis box and parameter identification for the sensitivity analysis can be seen in the figures below.

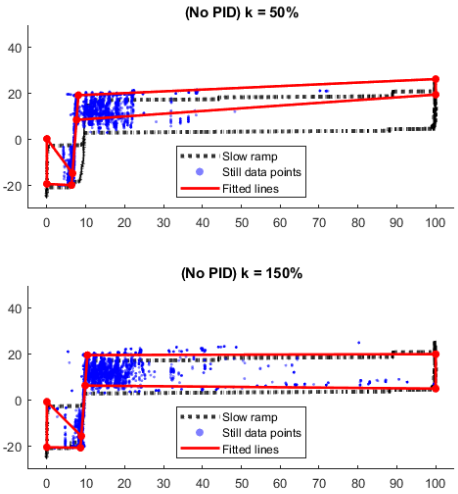


Figure 38: Hysteresis plot when varying the k_minus and k_plus parameter.

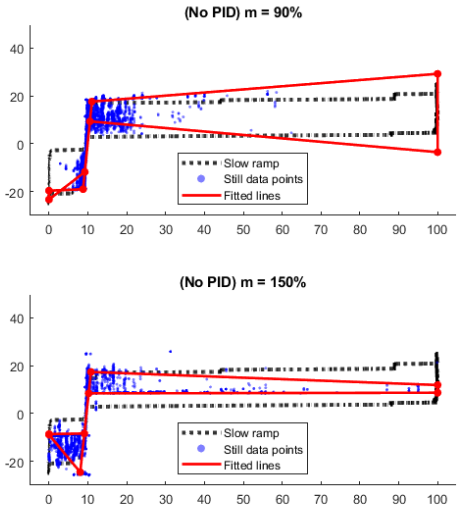


Figure 39: Hysteresis plot when varying the m_lhminus and m_lhplus parameter.

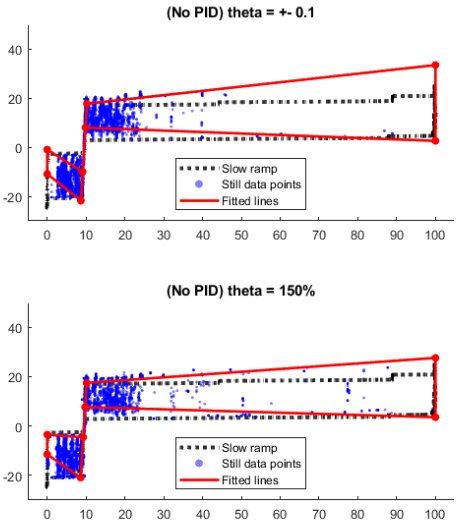


Figure 40: Hysteresis plot when varying the theta_lhminus and theta_lhplus parameter.

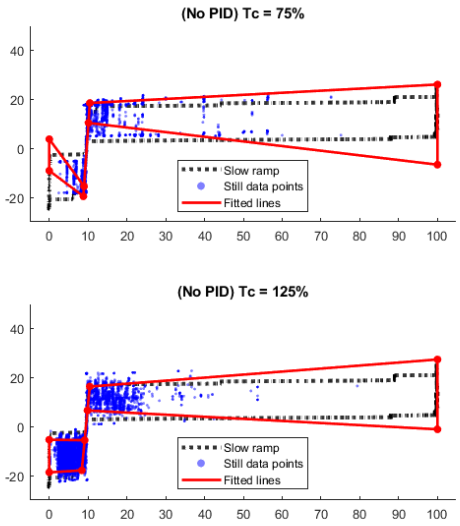


Figure 41: Hysteresis plot when varying the Tc_minus and Tc_plus parameter.

The scrips struggles due to low amount of data points. However by eye it is clear that the data points are still within the hysteresis boxes when the friction and limp home compensation parameters are varied. This means that the spread of data points are not a function of the control parameters.

7 CONCLUSIONS

7.1 Dependencies of varying temperature and condition

Different temperatures have been investigated during the project and how this affects the behaviour of the throttle. The conclusion is that there is no visible change in friction parameters on the temperature scales tested in this project. Another conclusion from this result is that it would require more gathered data to possibly get a result that shows any difference in the throttle characteristics.

Furthermore, it was investigated how the condition of the throttle affected the characteristics. This was done by evaluating the hysteresis on an old, used throttle and a new one. The conclusions from this are similar to the ones due to temperature since no significant difference is observed in reference following when plotting the drive cycle provided by Aurobay. The hysteresis is noticeably different for the new and the old throttle when conducting a slow ramp. However, this difference in hysteresis is not significant enough to make a difference during drive cycle and reference following tests.

7.2 Using machine learning algorithms

In this project several conventional classification machine learning algorithms were tested to automatically determine the parameters for the friction and limp home compensation. The result from these showed little promise, see Section 6.5. The cause of this result is most likely due to two reasons, lack of machine learning knowledge and poor training data. Visually it is clear that a pattern exists in any hysteresis plot, Figure 9 shows a good example of this. Since it can be seen by eye, it should be possible for an algorithm as well. With better filtered data and a well defined machine learning algorithm the hysteresis boxes should be possible to identify.

This approach was abandoned and an custom algorithm was developed. The algorithm defines the boxes as a function of density of data points, which is looking promising. Given enough data from a drive cycle that excites the system accurate parameters are automatically calculated.

7.3 Updating control parameters continuously

As described in Section 6.6 the parameters for friction and limp home compensation are successfully calculated using an automated script, a type of machine learning. These parameters could then be used on several different drive cycles with good reference following, see Figure 31. Therefore it could be concluded that the control parameters can be updated continuously throughout a driving mission, this is however not tested since it did not fit within the time span of the project. The result presented in Figure 32 is more worrying. From the figure it can be concluded that the machine learning algorithm does not perform well if there is a lack of data points. City Manual which is the shortest driving cycle of only 200 seconds is an example of this. The control parameters calculated from this cycle are far off from the manually selected parameters from a slow ramp. Using these parameters would most likely result in oscillatory and worse reference following for the throttle.

A solution to this problem is to do an initial ramp when the car starts and use readings from that for the first few minutes of driving. When enough data is collected the machine learning algorithm updates the parameters according to the current driving conditions, roughly five minutes seems to be sufficient in Figure 31. After this the script can

continuously run to improve the performance of the throttle. Importantly old data from the driving mission should be discarded, only roughly 10-30 minutes of running data should be used in the algorithm to account for varying conditions. Additionally some form of constraint should be implemented to limit large rapid changes of the parameters since this is not physically reasonable.

8 FUTURE WORK

8.1 Neural network model

In the considered mathematical description of the throttle it is described as a number of torques with different characteristics. We have

- $T_u = F_1(u(t-k), \dot{u}(t-k), \dot{y}(t))$ as the driving torque depending on the current PWM, the change of PWM and the angular velocity. k is the time delay from the input in the actuators.
- $T_s = F_2(y(t))$ as the spring force, only depending on the current angle.
- $T_f = F_3(\dot{y}(t))$ as the friction, with a static part and a part depending on the throttle speed.

with unknown functions F_x for all the torques. Together this can be written as

$$\ddot{y}(t) = F_1(u(t-k), \dot{u}(t-k), \dot{y}(t)) + F_2(y(t)) + F_3(\dot{y}(t)). \tag{10}$$

Now denoting the sample time T we can approximate the derivatives using finite differences as

$$\begin{cases} \dot{u}(t) \approx \frac{u(t)-u(t-h}{hT} \\ \dot{y}(t) \approx \frac{y(t)-y(t-h}{hT} \\ \ddot{y}(t) \approx \frac{y(t)-2y(t-h)+y(t-2h}{(hT)^2}. \end{cases}$$

This approximation along with Equation (10) suggest the possibility to describe the throttle at each time as some function of a finite series of delayed earlier measurements and inputs.

With the above motivation it would seem a neural network could be a good solution for estimating the model dynamically without having to know too much about the exact workings of the throttle. When tested in the project this did however not work out well. The recorded data was either not enough, or too noisy, or simply there was more preprocessing needed to train a model. With some more knowledge about neural networks, some more data and some more time this is however definitely a method worth investigating.

8.2 Testing the machine learning method continuously

The algorithm for determining the friction and limp home compensation parameters is written as a Matlab script, making it very easy to implements in the HIL-testing simply by calling upon it from the Simulink model. Due to a lack of time it was not tested during this project. As described in Section 7.3 two main problems remains to be solved before the algorithm can be fully implemented, saving relevant data and limiting rapid changes of the parameters. The data to be used in the script should continuously be replaced by fresh data and old should be discarded. From the testing done in this project 10-30 minutes of saved data seems to be a sufficient amount to ensure accuracy from the algorithm, this should however be investigated further. To ensure smooth transitions the algorithm should be restricted from changing the control parameters to much at once, such a system has not been constructed.

8.3 Handling of delays

A problem during the project was a rather extensive delay from a command sent to the Raspberry Pi to the returning response. This was seen clearly when doing steps but is also assumed to have negative impact on the general control

of the system. In order to identify where delays happen in the system some tests have been carried out, but due to lack of time it was not possible to implement all solutions before the other results were needed. Therefore the results of these test will be presented and discussed in this chapter as a possibility for a future project to further improve the throttle control.

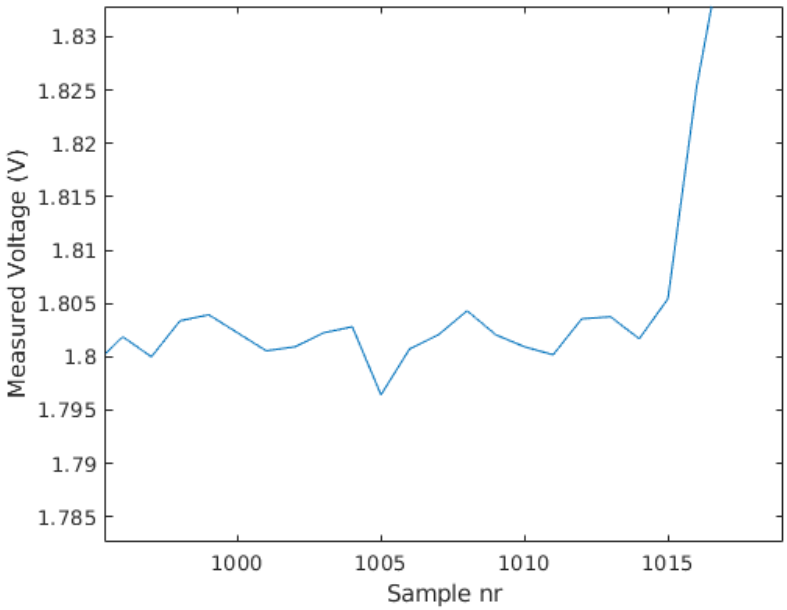


Figure 42: Typical delay for measurements, in this case with a step at sample 1000. Here the clear slope starts at about sample 1015, corresponding to a delay of about 30ms.

The problem can be seen in Figure 42 showing the response for a step at sample 1000. This delay of almost 30ms was considered way above what should be possible to achieve.

8.3.1 *Analysis of delays in the system*

A problem in monitoring the delays was that both the computer and the Raspberry Pi could be part of the problem, and to time signals and print them made the programs run slower. Therefore a method for measuring the signals directly was needed. An oscilloscope was used to be able to monitor signals in different parts of the system and thus find the delays in the connections. The oscilloscope had two channels making it possible to visually examinig delays between two different parts. The parts analyzed were:

- Delay from command from computer to PWM command from Raspberry Pi.
- Delay from PWM command to measured change in angle.
- Delay from command from computer to answer from Raspberry Pi.

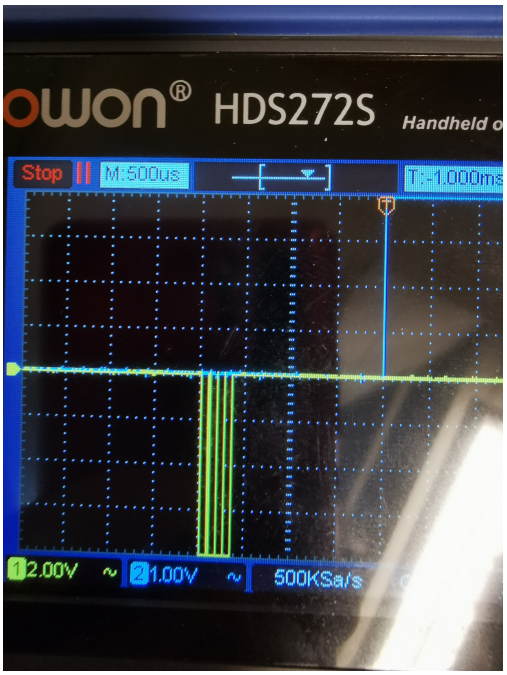


Figure 43: Serial command sent from the computer in yellow compared to when the Raspberry Pi sends a command to the motor to go clockwise (in blue). With the given time step this means a delay of about 2ms for the raspberry to process and send the command.

COMMAND FROM COMPUTER TO PWM COMMAND The result from placing one probe on the RX connection on the Raspberry and the other on the INA contact on the h-bridge can be seen in Figure 43. The command sent was to go from zero up to 100 % PWM in one step, which should directly set INA to one. A delay of about 2ms is visible in the oscilloscope. This delay can be a bit varying since the Raspberry Pi is running its update routine every 2ms.

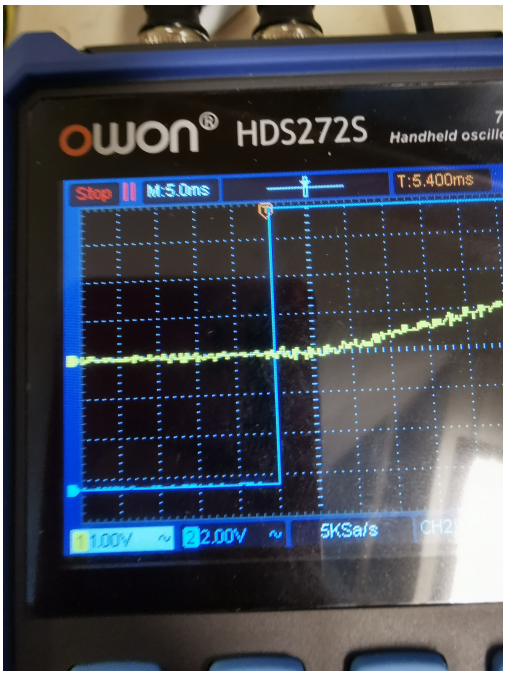


Figure 44: Comparison between measured potentiometer voltage on the throttle in yellow and a step command from zero sent from the Raspberry Pi. From visual examination one can conclude a delay of about 6ms from the command to a change in angle.

PWM COMMAND TO MEASURED CHANGE IN ANGLE When connecting one probe to INA on the h-bridge and one probe to the signal output of the throttle the result was as in Figure 44. This was for a step from 0 to 30 % and the point when the command was issued is visible as the step in INA in the figure. The delay here is the internal delay in the throttle and will be difficult to do anything about as it depends on the mechanics and construction of the throttle. This means a delay of about 6ms will be inevitable for this system.

DELAY FROM COMMAND TO ANSWER For testing the entire communication one probe was connected to the TX pin and one to the RX pin on the Raspberry Pi. Some different tests were conducted in speeds from 100 Hz to 500 Hz and the results clearly showed that the system became unsynchronized for high frequencies, as can be seen in Figure 45 This let to the belief that data can be stacked in the buffers when the Raspberry Pi or the computer is too slow, and thus a test with a direct step was conducted. This step was sent as a PWM signal of 30% directly after a setup in which the buffers are cleared. The result can be seen in Figure 46. As can be seen in the response it is considerably faster than for a step later in the sequence, for example in Figure 42. When compared to the delay in the throttle it is also not especially much more. This result concludes it should be possible to dimish the delay by always making sure the latest value is read in the UART buffers. There is however still the problem with unsynchronized signals in fast frequencies, suggesting either more optimized code or a slower update frequency is needed.

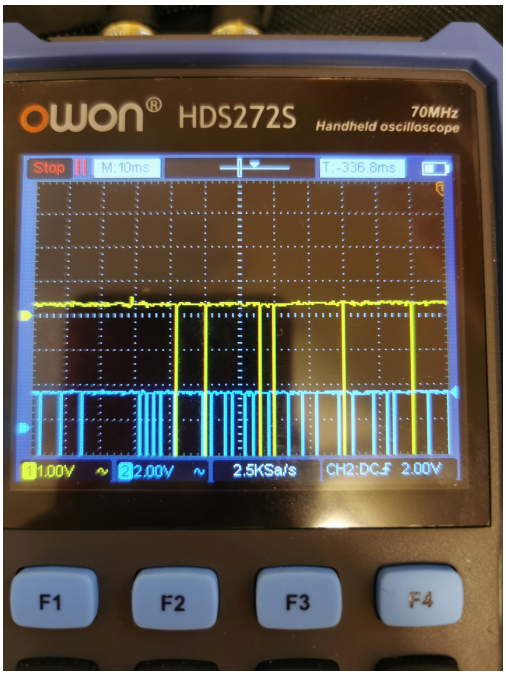


Figure 45: Commands (in blue) and responses (in yellow) for a UART speed of 500Hz between the computer and the Raspberry Pi. Notable is the unsynchronized behaviour.

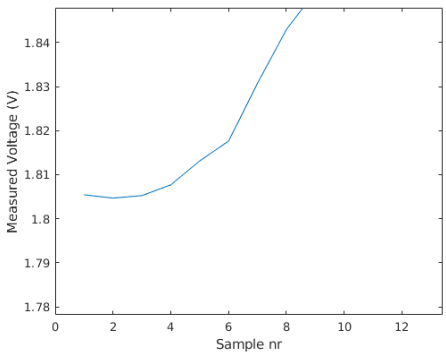


Figure 46: The measured voltage response for a step at sample 1. The resulting delay seems to be about 3 samples, corresponding to a delay of about 6ms.

8.3.2 *Analysis with different frequencies*

The nature of the delay makes it look like it comes from problems with the frequency being too high. To test this a timing was implemented at the computer to check for data at a certain frequency, and to set the value to undefined if no data was present in the buffer. This meant both the computer and the raspberry should send and receive data at a specific frequency, but now the computer is not waiting for the Raspberry to respond. This was tested at frequencies

[100, 200, 300, 400, 500] Hz with 3000 samples and the percentage undefined values was afterwards counted for each frequency. The result can be seen in Table 13 and from there the problem at 500 Hz can be seen quite clearly.

Table 13: Percentage lost values when running the communication at different frequencies.

Frequency	Percentage undefined
100	3%
200	7%
300	3%
400	10%
500	85%

8.3.3 *Conclusions from delay analysis*

For future projects this found delay should be considered early in the process and the communication should be tested for the desired speed before using it to implement the rest of the systems. We suggest either to slow down the system in total to give everything more time, but if this gives other problems caused by too slow responses we suggest to have a slower reference feedback loop and a faster loop for small compensations around this reference angle. This would give the opportunity to make more complicated calculations slowly and to have a fast PID (for example) running at about ten times this speed to actuate the reference value. Another alternative would be to use more dedicated hardware and software to be able to run the system faster.

REFERENCES

[1] *Model-Based Throttle Control using Static Compensators and Pole Placement*, A. Thomasson and L. Eriksson, IFP Energies nouvelles (2011). [article]

[2] *Automotive-grade Dual DC motor driver up to 15 A each*, ST Microelectronics [pdf]

[3] *Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converter with Internal Reference*, Texas Instruments [pdf]

[4] *Raspberry Pi 3 model B+ reduced schematics drawing*, Raspberry Pi

[5] Lars Eriksson & Lars Nielsen. *"Modeling and Control of Engines and Drivelines"*. John Wiley Sons, Incorporated, updated 4 July 2014. [Book]

[6] URL: *https://abyz.me.uk/rpi/pigpio/*

[7] *Adaptive Control Systems: Past, Present, and Future.*, S.G Anavatti, F.Santoso and M. Garratt, International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (2015). [article]