Design Specification

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

December 12, 2022

Version 0.2

Project Identity

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

Customer: 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	Reviewed
0.1	2022-09-16	First draft	Whole group	Robin Holmbom
0.2	2022-10-	Revisions as suggested by reviewer	Whole group	

1 SYSTEM OVERVIEW

This section describes the system in an overall manner with some background information together with the aims and goals of the project.

1.1 Aims and Goals

The purpose of the project is to improve the control of an intake throttle in an internal combustion engine. This is to be done by exploring ways of identifying the throttle parameters in real time. One possible way of solving this could be machine learning. This could be utilized in order to find a general model for learning the throttle parameters.

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 final product should also provide the same driveability as a normal throttle. A car equipped with this throttle controller should not require any special skills or adaptation of driving style. In short, it should be like driving a normal car.

For the group it is relevant to practice how to work in projects and how to structure a project within a group. The goal for the project group is further to gain knowledge within relevant technical areas such as machine learning and control theory.

1.2 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, Aurobay. 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.

1.3 Model Notation

Table 1: Notation of all parameters.

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*
$k^{\pm}$	Gradients
$m_{lh}^{\pm}$	Friction at the limits of the *limp home region*
ω	Angular velocity
α_{ref}	Throttle reference angle

1.4 Background Information

The design specification is a mandatory part of the project course Automatic Control, TSRT10.

1.4.1 *Limp Home Region*

Limp home region refers to the throttle angle, θ that the throttle has when the control signal, u is close to zero.

1.4.2 *PID Regulator*

A pid regulator have the structure outlined in Equation (1).

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{1}$$

Where $u(t)$ is the control signal and $e(t)$ is the error value ($e(t) = r(t) - y(t)$). K_P, K_I and K_D are all constants.

2 SIMULATED VEHICLE MODEL

This chapter will discuss the implementation of the Matlab/Simulink vehicle model that is to be used in this project. The model is taken from the course TSFS09. There will be no major changes to the model in regards of functionality, however some changes to adapt the model to this project might be necessary. An overview is given in Figure 1.

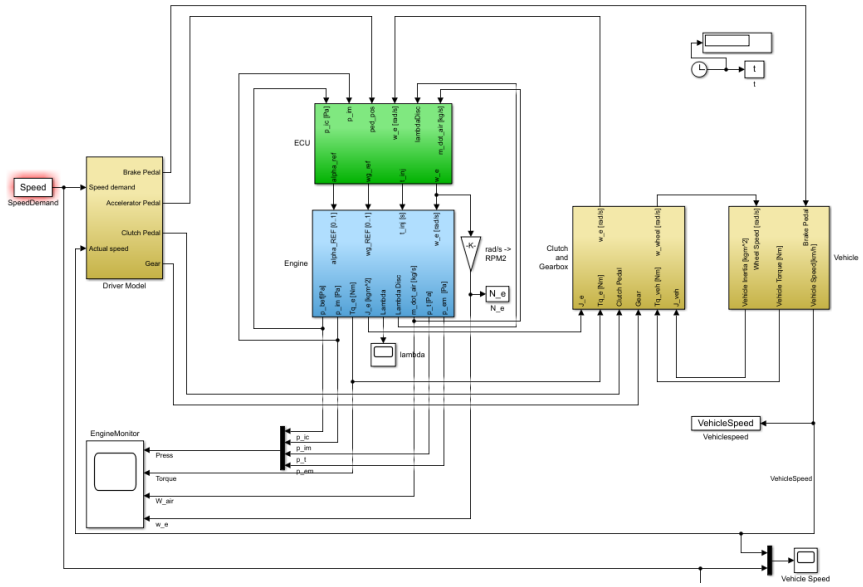

Figure 1: An overview of the full system in the Simulink environment in its unaltered form.

2.1 System Overview

2.2 Throttle

The main component of this project will be the throttle. The simulation model of the throttle is taken from the course TSFS09 where it was modeled with a fuel mass flow, $\dot{m}_{at}$, described as

$$\dot{m}_{at} = \frac{p_{bef,thr}}{\sqrt{R_{air} \cdot T_{bef,thr}}} \cdot A_{eff}(\alpha_{th}) \cdot \Psi(\Pi) \tag{2}$$

where R_{air} is the air gas constant, $T_{bef,thr}$ is the temperature before the throttle, $p_{bef,thr}$ is the pressure before the throttle. The effective area for the throttle is modeled with a polynomial depending on the throttle angle α_{th}.

$$A_{eff} = a_0 + a_1\alpha_{th} + a_2\alpha_{th}^2 \tag{3}$$

The parameter Ψ determines the fluid velocity and is calculated according to Equation (4) [8].

$$\Psi(\Pi) = \sqrt{\frac{2\gamma}{\gamma - 1}\left(\Pi_{lim}^{\frac{2}{\gamma}} - \Pi_{lim}^{\frac{\gamma-1}{\gamma}}\right)} \tag{4}$$

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 [1]. K_{fv} and K are static gain regulator parameters [1].

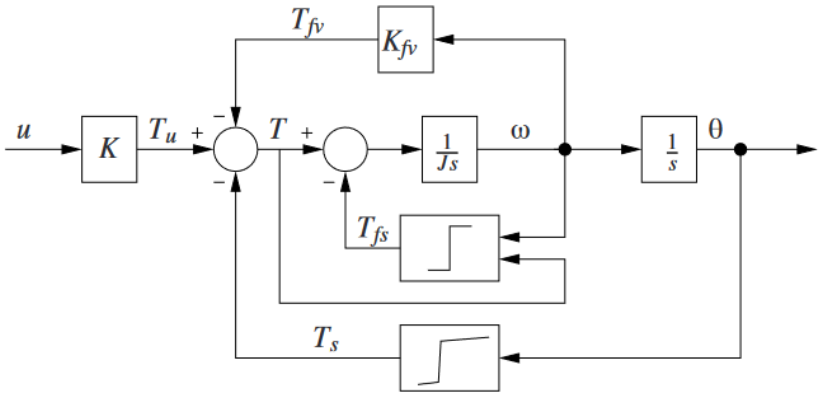

Figure 2: The throttle model as a Simulink block diagram [1]

The static gain regulator parameter K_{fv} relate to the throttle frictions as outlined in equation (5) [1].

$$K_{fv}\omega = T_{fv} + T_{emf} \tag{5}$$

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 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 (6) where $k^{\pm}$ are different gradients, $\theta_{lh}^{\pm}$ is the angular limits of the *limp home region*, and $m_{lh}^{\pm}$ is the frictions at the limits of the *limp home region* [1]. The relation is illustrated in Figure 3 [1].

$$T_s(\theta) = \begin{cases} m_{lh}^{+} + k^{+}(\theta - \theta_{lh}^{+}) & \text{if} \quad \theta > \theta_{lh}^{+} \\ m_{lh}^{+}(\theta - \theta_{lh})/(\theta_{lh}^{+} - \theta_{lh}) & \text{if} \quad \theta < \theta_{lh} < \theta \leq \theta_{lh}^{+} \\ m_{lh}^{-}(\theta_{lh} - \theta)/(\theta_{lh} - \theta_{lh}^{-}) & \text{if} \quad \theta < \theta_{lh}^{-} < \theta \leq \theta_{lh} \\ m_{lh}^{-} - k^{-}(\theta_{lh}^{-} - \theta) & \text{if} \quad \theta > \theta_{lh}^{+} \end{cases} \tag{6}$$

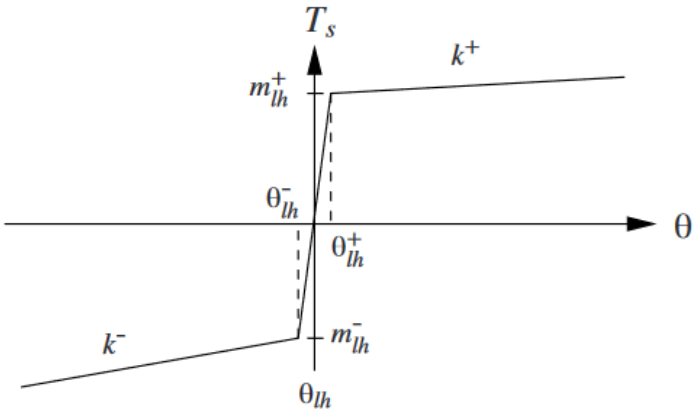

Figure 3: Friction torque, T_s as function of throttle angle, θ [1].

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 (7) [1].

$$T_{fs}(T,\omega) = \begin{cases} T & \text{if } \omega{=}0 \text{ and } |T| < T_c \\ T_c sign(\omega) & \text{otherwise} \end{cases} \tag{7}$$

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].

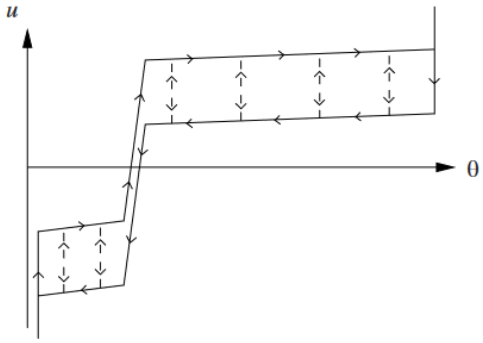

Figure 4: Control signal, u as function of throttle angle, θ [1].

2.3 Vehicle Model

This subsystem (see Figure 5) is a part of the full system and gives the vehicle an inertia and weight. Giving the vehicle dynamics further helps with simulating how a vehicle behaves in the real world.

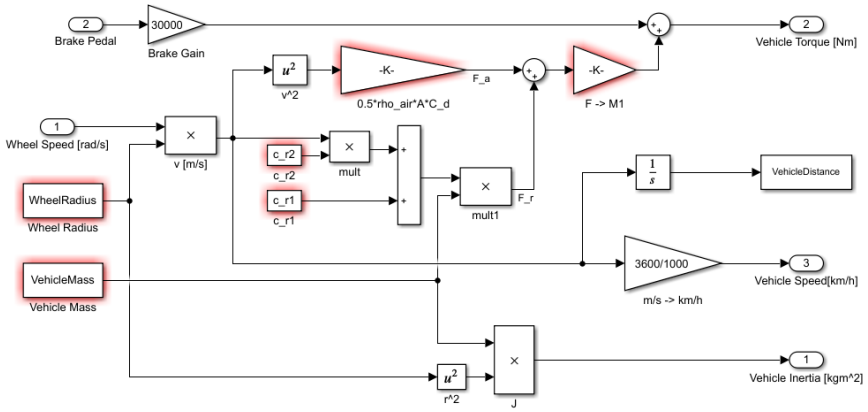

Figure 5: An overview of the vehicle subsystem.

2.4 Drive Cycles and Driver Model

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 value for the throttle through the simulated model of the vehicle. Shown in Figure 6 is one example of how a drive cycle could look, in the example the EUDC drive cycle is used.

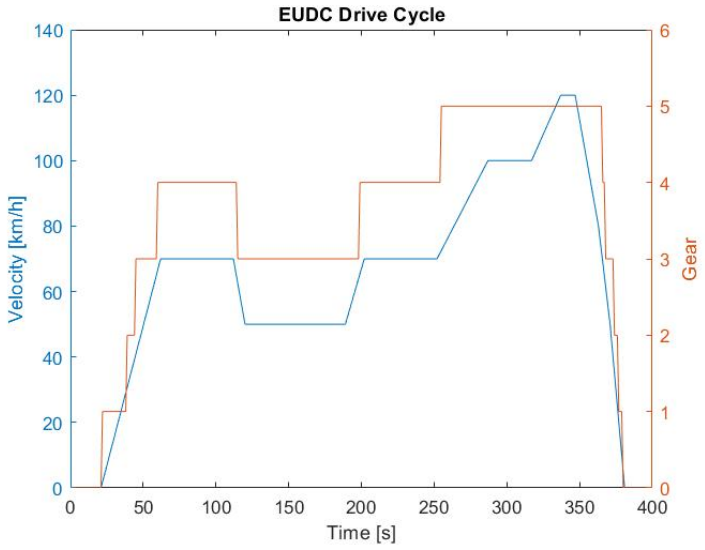

Figure 6: EUDC drive cycle is one of the drive cycles to be used in the simulation.

The driver model (see Figure 7) translates the demanded speed and the actual speed of the simulated vehicle into pedal actions. The most important signal is the reference value for the acceleration pedal position. This will later be translated into throttle angle.

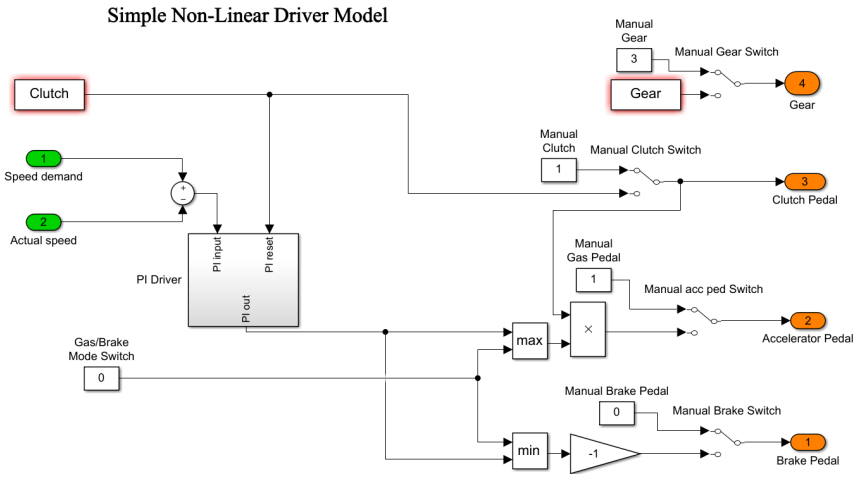

Figure 7: Simulink schematic of the driver.

2.5 Clutch and gearbox

To simulate the cut off in power when disengaging the clutch a subsystem is implemented to give the characteristics. Further dynamics such as inertia to/from the engine is also handled in this subsystem (see Figure 8).

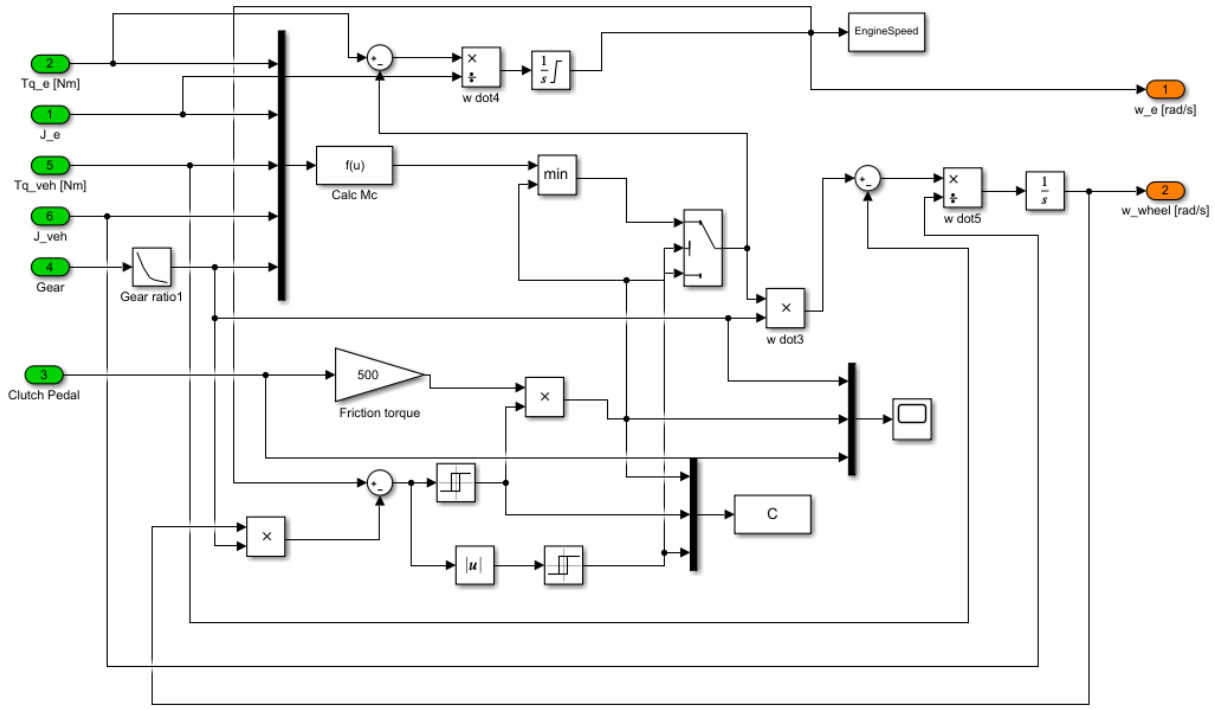

Figure 8: Simulink schematic of Clutch and gearbox.

2.6 ECU Model

The electronic control unit, ECU, acts as the computer for the simulated engine converting driver inputs to engine control signals. This is a vital component for this project since it is responsible for converting the pedal position from the driver to a reference angle, α_{ref}, for the throttle. This is done using the pedal position, current engine speed and pressures in the engine. This controller is based on both feed forward using a model of the air flow through the cylinders and a feedback loop with a regular PID-controller.

Further actions that the ECU handles are the boost controller if a turbocharger is present in the engine and the injection time for each cylinder to minimize knock and also keep the λ as close to 1 as possible (see Figure 9).

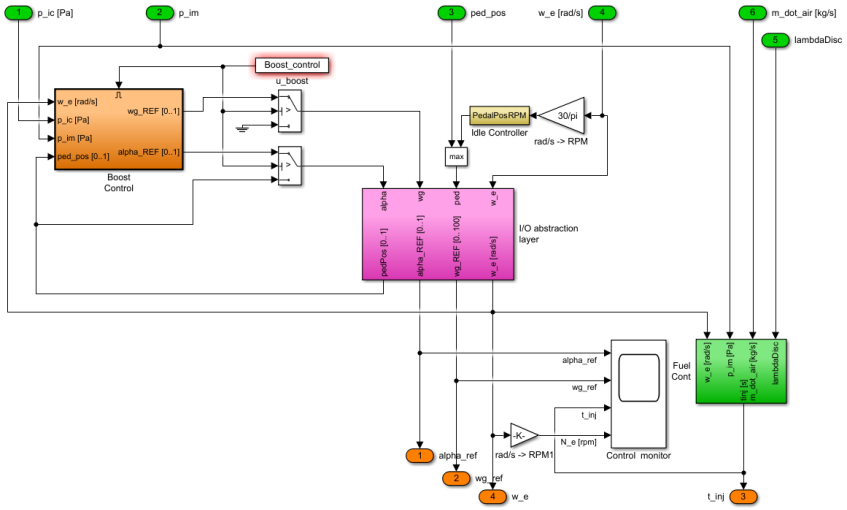

Figure 9: Simulink schematic of ECU.

2.7 Engine Model

Engine model refers to the examining engine model in the course TSFS09 [1] (see Figure 10). This is a model of an engine with some simplifications and linearization to optimize between performance and computing time. The model is constructed in the Matlab extension Simulink.

Within the model there are a few subsystem that provides models for components in the engine. For this project the significant part is to be spent on increasing the precision of the throttle and therefore the other subsystems are of lesser importance. The main function of the model will for this project to give a desired α which is to be applied to the physically real throttle.

1 The engine Simulink model is provided by *Vehicular Systems Institution* at Linköpings University.

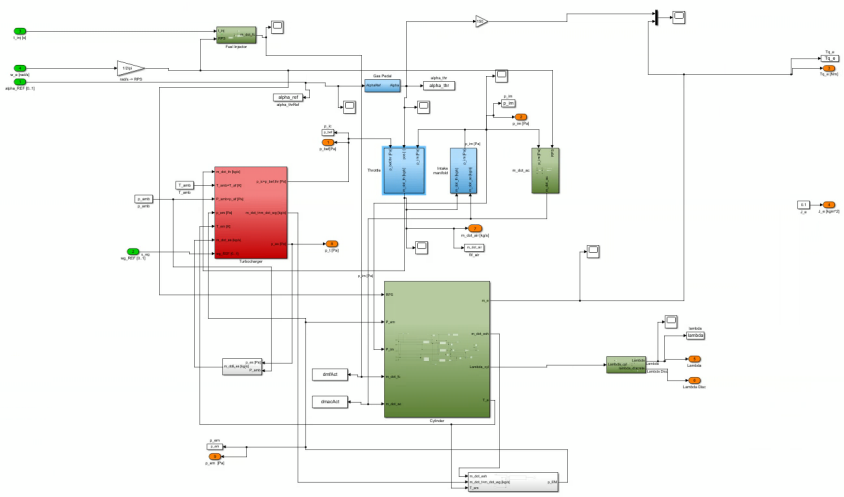

Figure 10: An overview of the complete engine block done in Matlab/Simulink. The model includes fuel injector, gas pedal, throttle, intake manifold, cylinder, turbocharger and lambda sensor.

3 HARDWARE

In this chapter all hardware for the project is described. This includes test throttles, microcontroller actuators and sensors.

3.1 Throttle

The main components of the project will be two throttles. The first throttle is supplied by the supervisor and it is a throttle from a Volvo VEA engine (exact model is not yet determined). The second thottle is provided by the customer, Aurobay, and has been used 350,000 km in field tests. Therefore neither of the throttles are to be designed by the group.

The throttles have a motor and two potentiometers for measuring the current angle. The reason for having two sensors is to have a backup in case something goes wrong with measurements. These potetntiometers 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, why is is a good idea to measure and 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.

3.2 H-bridge

To transform the PWM signals from the Raspberry Pi to a motor current we will use the H-bridge of model AEK-MOT-2DC70S1 [6]. It can drive several motors simultaneously, but only one will be necessary for this project. To

work properly it will need three signals; motor clockwise, motor anticlockwise and PWM. The two first must be set opposite to each other for the motor to work.

3.3 Analog to digital converter

The Analog to Digital Converter (hereby noted ADC) to be used in the project is the model ADS1115. It is a 16-bit resolution converter with I2C communication interface [5]. For this project only onechannel is needed, and they will be connected to oneof the two potentiometers on the throttle. The I2C interface will be connected to the Raspberry Pi.

Important for the ADC is to have a stable reference voltage. The primary idea is to drive it from the 5 V voltage regulator, which is the same as the voltage for the potentiometers reading the throttle angle. One possible problem with this setup it that the ADC only uses 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 will be diminished. Each step is $FS/2^{15}$ where $FS = 6.144V$ so this means a resolution of 0.1875mV per step ideally.

The ADC can also work in different speeds, with the fastest being 860 conversions/second. There is however more noise when the ADC works faster, why one idea is to use the slower 475 conversion/second instead. If the loop is to run in 500 Hz this is actually a bit to slow, but it is considered not to be a problem as it is almost as fast as the control loop. The ADC speed will be decided after recording data at both speeds and comparing their respective variance.

3.3.1 *I2C for the ADC*

Specifications on how to use the I2C protocol with the ADC can be found in the datasheet [5]. 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 not 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.

3.4 Raspberry Pi

Raspberry Pi is a small computer with general purpose input/output (GPIO) pins [7]. Some of these pins can be configured as an I2C interface to communicate with the ADC. How to connect the pins to the I2C is described in Table 2. 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 as GPIO27 mapping to pins 11 and 13 respectivly.

Table 2: I2C connections on Raspberry Pi.

Signal	Pin
3.3V	1
SDA1	3
SCL1	5
GND	9

The Raspberry Pi also runs a Linux operational system and can be connected to WiFi. The Raspberry Pi can either be connected via USB or via Wifi to any computer running Matlab. These features make it ideal for using as a connection between the physical actuators and the software simulation. The version used in this project is Raspberry Pi model 3 B+.

3.4.1 *Serial communication*

The Raspberry Pi will communicate with Matlab using a serial UART scheme. 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

Further it is planned to send commands as a PWM percentage numbered from -100 to 100 from Matlab to the Raspberry and to send angles from the Raspberry Pi as a percentage from 0 to 100 back to Matlab. One problem that can arise is if there is to much data to send, in which case this has to be re-evaluated.

4 DYNAMIC PARAMETER IDENTIFICATION

The control system shall be able to dynamically identify necessary parameters to make a complete plot of the hysteresis as seen in Figure 4 with a limited amount of data.
Measured data points are determined to be in one of the six regions outlined in Figure 11 depending on the current angle θ and whether the throttle is currently ramping up or down. Then the parameters in Equation (6) can be found from the measured data points depending on what regions the measured data points are in. The hysteresis will then be updated every time a new parameter estimation is made. Important to note that only measurement points where the throttle angle θ change are relevant as the point otherwise is "within" the hysteresis and can thus not be mapped to a region in Figure 11.

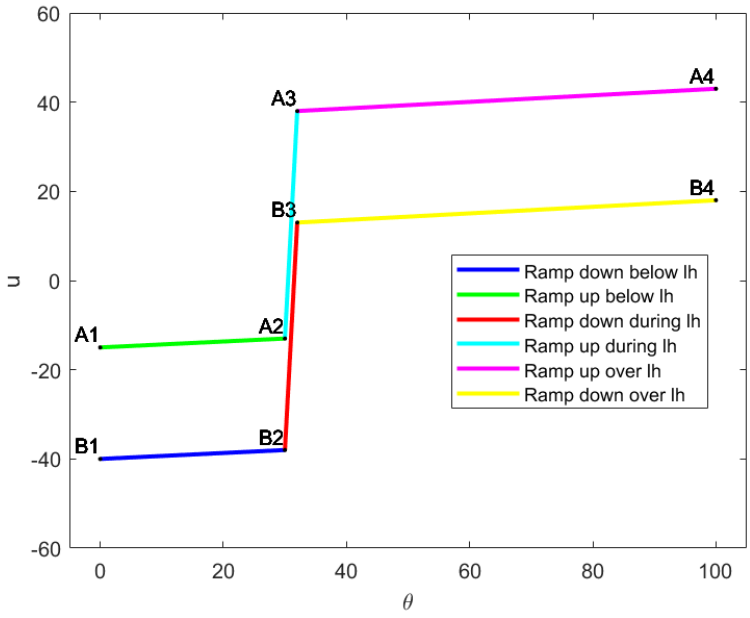

Figure 11: Typical hysteresis for the throttle [1].

Data points in the *"Ramp up over lh"* and *"Ramp down over lh"* can be used to determine k^+ (see Equation (6) and Figure 3) which equals the gradients of the regions (that should be roughly equal). Measurement data from these regions can also be used to find the points $A3$ and $B3$ (see Figure 11) which can be found by finding the extrapolated values of the *"Ramp up over lh"* and *"Ramp down over lh"* regions at the angle θ_{lh}^+ (see Equation (6) and Figure 3). The points $A3$ and $B3$ can then in turn be used to determine m_{lh}^+ see Equation (6) and Figure 3) by using Equation (8).

$$m_{lh}^+ = \frac{A3 + B3}{2} \tag{8}$$

Data points in the *"Ramp up below lh"* and *"Ramp down below lh"* can be used to determine k^- (see Equation (6) and Figure 3) which equals the gradients of the regions (that should be roughly equal). Measurement data from these regions can also be used to find the points $A2$ and $B2$ (see Figure 11) which can be found by finding the extrapolated values of the *"Ramp up below lh"* and *"Ramp down below lh"* regions at the angle θ_{lh}^- (see Equation (6) and Figure 3). The points $A2$ and $B2$ can then in turn be used to determine m_{lh}^- (see Equation (6) and Figure 4) by using equation (9).

$$m_{lh}^- = \frac{A2 + B2}{2} \tag{9}$$

Data points in the *"Ramp up during lh"* and *"Ramp down during lh"* can be used to determine the angles θ_{lh}^+ and θ_{lh}^- (see Equation (6) and Figure 4) by finding the points where these regions intersect the other regions (see Figure 11). The angle θ_{lh} can then be found by using Equation (10).

$$\theta_{lh} = \frac{\theta_{lh}^+ + \theta_{lh}^-}{2} \tag{10}$$

5 CONTROL SYSTEM

In the following section the control problem is stated and some initial ideas on how to approach the problem are disclosed.

5.1 General Design of the Control System

Simulink will handle the computational part of the control system. A reference angle is fed from the motor simulation to the control system. The control system is also fed the actual angle from the physical throttle. From these values the control system will produce a desired PWM signal from -100 to 100. Feedback from the potentiometers is sent through the Raspberry pi to the Simulink controller system at a rate of 500 Hz.

5.2 Friction Compensation

Compensating to the throttle friction outlined in Equation (7) can be problematic as it could make the control very sensitive around the reference angle [1]. Therefore it might be necessary to add a *deadzone* when the angle is close to the reference angle [1]. It is also important to make sure the friction overcomes the coulomb friction T_c outlined in Equation (7). Therefore a modified version of Equation (7) is proposed to be used as friction reference as outlined in Equation (11) where e_θ is the angle error and k_{tf} is a constant slightly above 1 to make sure that the friction overcome the coulomb friction [1].

$$\tilde{T}_f = \begin{cases} 0 \; if \; |e_\theta| \leq \theta_d \\ k_{tf} \cdot T_c \frac{\theta - \theta_d}{\theta_r} sgn(e_\theta) \; if \; \theta_d < |e_\theta| \leq \theta_r + \theta_d \\ k_{tf} \cdot T_c \; if \; |e_\theta| > \theta_r + \theta_d \end{cases} \tag{11}$$

Equation (11) result in the friction compensation outlined in Figure 12[1].

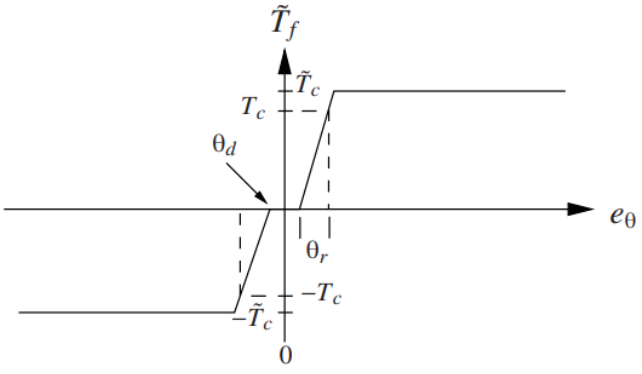

Figure 12: The friction compensator outlined in equation (11) [1]. $\tilde{T_c} = k_{tf} \cdot T_c$

An alternative or supplement to this may be filtering through a low-pass filter to remove oscillations [10].

5.3 Control Design Methods

An online adaptive control algorithm is required to address the problem of varying parameters during operation. One way to deal with varying parameters is to implement a self tuning controller which continuously estimates the plant parameters and adjusts the control law based on current estimates. A self tuning regulator can utilize direct adaptation of controller parameters or indirect adaptation by separating the parameter estimation and the controller design. A schematic overview of the adaptive control algorithm is presented in Figure 13.

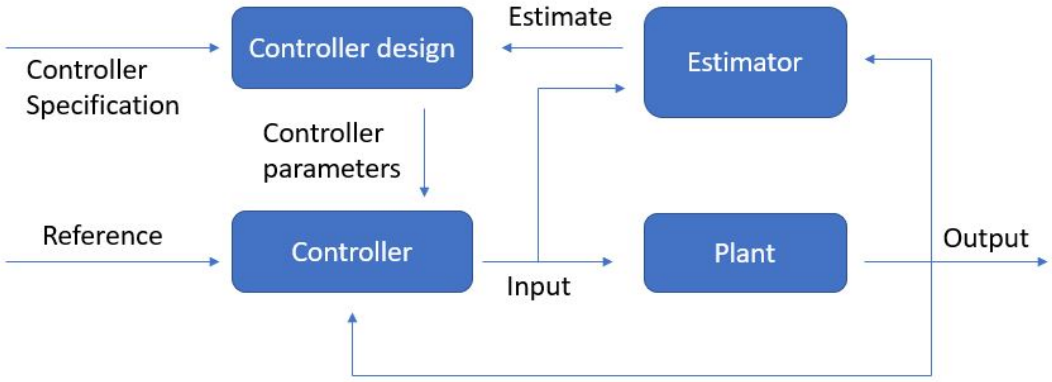

Figure 13: An overview of the general idea of a self tuning regulator.

Another solution to handle uncertainties in system parameters is to implement a model reference adaptive control (MRAC) system. In this controller the closed-loop behaviour of the plant is determined by the reference model and the adjustment mechanism adapts the controller parameters to match the output of the plant to the model. The adjustment mechanism can be chosen, and two common methods are Lyapunov stability and the MIT rule using gradient descent. A schematic overview of the MRAC algorithm is presented in Figure 14.

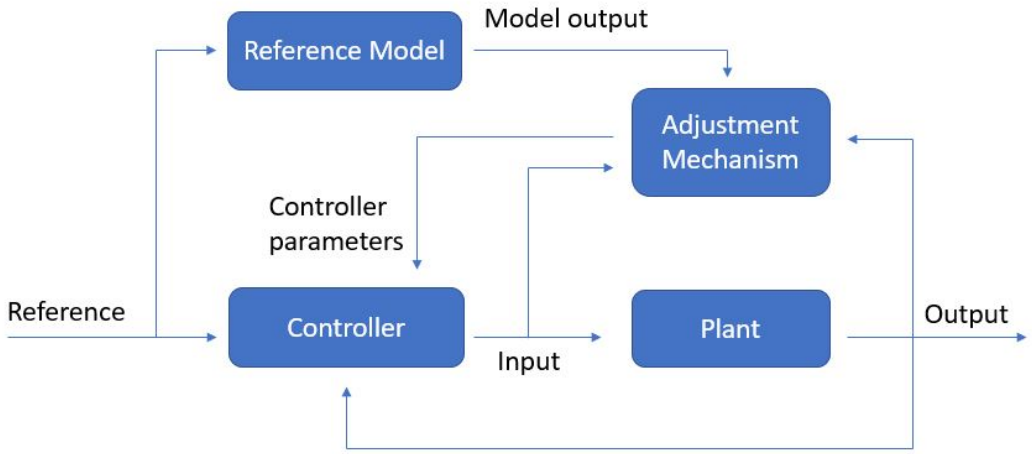

Figure 14: An overview of the general idea of an adaptive controller using MRAC.

5.4 Anti Windup

The hysteresis (see figure 4) provide a challenge for integrators in the control system. This is because the coulomb friction (T_C) hinders the throttle from moving unless the signal is strong enough. Thus the integrator might windup when there is a small continuous angle error (e_θ) as the throttle does not move during an extended amount of time. This windup might result in overshoots.
A solution to this is to implement anti-windup to the integrator. One example of anti-windup is to cap the integrator to a minimum and maximum value [10].

5.5 Bump-less Transfer

The controller will most likely need to use different values on the control parameters depending on if the angle is above or below the *limp home region* (see section 1.4.1) or not.
This will require the controller to implement a bump-less transfer when the control parameters change values.One way to achieve bump-less transfer with new *pid* parameters K_{Pnew}, K_{Inew} and K_{Dnew} (see section 1.4.2) is to adapt the integral part of the *pid* to keep the control signal continous with the previous control signal generated with the old *pid* parameters K_{Pold}, K_{Iold} and K_{Dold} [11]. This principle is demonstrated in equation (12) where the *pid* parameters change at the time t_s.

$$u(t) = \begin{cases} K_{Pold}e(t) + K_{Iold}\int_0^t e(\tau)d\tau + K_{Dold}\frac{de(t)}{dt} & t < t_s \\ K_{Pnew}e(t) + K_{Inew}\left(\int_0^t e(\tau)d\tau - K_{compensator}\right) + K_{Dnew}\frac{de(t)}{dt} & t \geq t_s \end{cases} \tag{12}$$

The constant $K_{compensator}$ are the adapted in accordance to equation (13) to ensure a bump-less transfer.

$$K_{compensator} = \frac{1}{K_{Inew}}\left((K_{Pold} - K_{Pnew})\,e(t_s) + (K_{Iold} - K_{Inew})\int_0^{t_s} e(\tau)d\tau + (K_{Dold} - K_{Dnew})\,\frac{de(t_s)}{dt}\right) \tag{13}$$

6 DYNAMIC UPDATING OF FRICTION PARAMETERS

The goal of this project is to improve the throttle regulator by using dynamic values on the friction models outlined in Equations (6) and (7).
Parameter estimation in closed-loop systems introduces convergence difficulties and loss of identifiablilty if low order feedback is used. In low order feedback systems the regressor matrix must be linearly independent to avoid loss of identifiability. It is also possible to use higher order feedback to avoid loss of identifiablility. To avoid unwanted behaviour of the controller, the controller parameters should be adapted gradually.

Machine learning methods will be used however it is still unclear to what extent. But these methods will be explored when more data and knowledge about the system has been acquired.

6.1 Machine Learning with Nonlinear Kalman Filter

One method of handling the dynamic updating of friction parameters is with a nonlinear Kalman filter. It is however not certain that a Kalman filter will be able to handle the nonlinearities in the system.

6.2 Support Vector Machine

Support vector machines is a method for classifying in machine learning. It works by finding borders between inputs and their respective class. In our case this could be used to classify if at a specific angle and signal the throttle is moving or not. By recording several of these points and process them the support vector machine should be able to give the hysterisis model of the throttle as in figure 11.

6.3 Neural Network for Finding Model

As a final step one could try to give the entire problem to a well designed neural network. By having enough data it is possible the network will be able to find the specific parameters continuously from data while driving.

A neural network utilizes "Nodes" that only sends data to the next node if the data it receives is above its "threshold" in accordance to Equation (14) wherein w are weights, x are nodes data is received from, b is a bias and $f(x)$ is the sent data [9].

$$f(x) = \begin{cases} 1 \; if \; \sum_{k=1}^{m} w_k x_k \; + \; b \geq 0 \\ 0 \; if \; \sum_{k=1}^{m} w_k x_k \; + \; b \leq 0 \end{cases} \tag{14}$$

The network are then built up in a structure with "input layer" nodes which are the nodes with known inputs and which sens data to the "multiple hidden layers" who in turn send the final data to an "output layer" with the sought data (see Figure 15)[9].

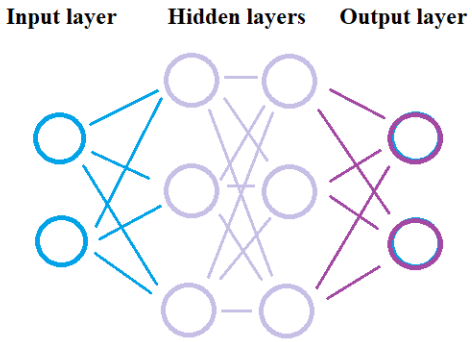

Figure 15: An overview of how neural networks operate.

6.3.1 *Recurrent Neural network*

To reduce the convergence time of the adaptive controller a recurrent neural networkc(RNN) could be implemented for an improved initial guess of the system parameters and thereby controller parameters. This guess could consider the current run time of the engine, outside factors and the value the parameters most likely converge to for this specific system. A RNN has some disadvantages though. Training them takes a lot of time and it has problems with vanishing gradients. The time aspect might prove problematic since the throttle should update its parameters in real time. Vanishing gradients can cause problems with long data sequences.

6.3.2 *Artificial Neural network*

Another possibility to implement machine learning in this project is to use an artificial neural network (ANN) for parameter estimation or to implement an ANN as a controller. To use an ANN as a controller there is a need for extra steps to verify the safety of the system. One way to ensure the safe operating of the system is to implement a Simplex Architecture. Another potential drawback with using an ANN as a controller is the fact that they require a lot of computational power.

7 IMPLEMENTATION

7.1 Simulation Environment

The simulations environment used will be Simulink. Simulink is used since the vehicle model is already implemented and it is a very powerful software for control related tasks. It is also a software which all members in the group have extensive experience with.

The throttle should be able to run during simulations as "Hardware In the Loop" (HIL). This means that the throttle is actuated as if it was in the engine when it in reality isn't. Instead the engine and all the other connected components are simulated by the other subsystems in the project.

7.2 Regulator Implementation

The new regulator will be implemented into Simulink and first be able to run on a Raspberry Pi. However if this software is to used in the engine it also needs to be able to be implemented into the ECU.

8 FUTURE WORK

8.1 Throttle

8.2 Control system

8.3 Dynamic updating of friction parameters

REFERENCES

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

[2] *Adjustable and fixed low drop positive voltage regulator TO220*, ST. [pdf] https://www.elfa.se/Web/Downloads/_t/ds/ST_LD1117_eng_tds.pdf

[3] *3 A low drop positive voltage regulator: adjustable and fixed TO220*, ST. [pdf] https://www.elfa.se/Web/Downloads/_t/ds/LD1085_eng_tds.pdf

[4] *N-Channel Logic Level Enhancement Mode Field Effect Transistor*, FAIRCHILD SEMICONDUCTOR. [pdf] https://www.elfa.se/Web/Downloads/_t/ds/BSS138_eng_tds.pdf

[5] *Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converter with Internal Reference*, Texas Instruments [pdf] https://www.elfa.se/Web/Downloads/_t/ds/1085%20Gain%20Amplifier_eng_tds.pdf

[6] *Automotive-grade Dual DC motor driver up to 15 A each*, ST Microelectronics [pdf] https://www.elfa.se/Web/Downloads/_t/ds/AEK-MOT-2DC70S1_eng_tds.pdf

[7] *Raspberry Pi 3 model B+ reduced schematics drawing* Raspberry Pi https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-reduced-schematics.pdf

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

[9] "*What are Neural Networks"*. IBM Cloud Education, updated 17 august 2020. [Website]

[10] "*Industriell reglerteknik Kurskompendium"*. Institutionen för systemteknik, Linköpings universitet, 8 january 2014. [Book]

[11] "*Bumpless Transfer for Adaptive Switching Controls"*, S-Y. Cheong and M. Safonov, Department of Electrical Engineering - Systems University of Southern California, 11 juli 2008. [Article]