7146CEM: Automotive Software Engineering - Design and Development

Coursework

Vignesh Babu Aravamuthan

[adnanv@coventry.ac.uk](mailto:adnanv@coventry.ac.uk)

Table of Contents

[1 Introduction 3](#_Toc88847287)

[2 Software Development Life Cycle 3](#_Toc88847288)

[2.1 Requirement Gathering 3](#_Toc88847289)

[2.1.1 PID controller 3](#_Toc88847290)

[2.1.2 Cruise Control 3](#_Toc88847291)

[2.1.3 Motor Speed 4](#_Toc88847292)

[2.1.4 Code 4](#_Toc88847293)

[2.2 Design 4](#_Toc88847294)

[2.2.1 PID Controller 4](#_Toc88847295)

[2.2.2 PID Tuning Algorithm 4](#_Toc88847296)

[2.2.3Cruise Control 4](#_Toc88847297)

[2.2.4 Motor Speed 5](#_Toc88847298)

[2.3 Coding 5](#_Toc88847299)

[2.4 Testing 5](#_Toc88847300)

[2.5 Validation 5](#_Toc88847301)

[3 Advantages of the used SDLC model 5](#_Toc88847302)

[4 GitHub Workflow 5](#_Toc88847303)

[5 References 5](#_Toc88847304)

[Figure 1 P Controller Implementation 4](#_Toc89191524)

[Figure 2 I Controller Implementation 5](#_Toc89191525)

[Figure 3 D controller Implementation 5](#_Toc89191526)

[Figure 4 PID Controller Implementation 6](#_Toc89191527)

[Figure 5 Cruise Control Schematic 6](#_Toc89191528)

[Figure 6 Motor Speed Schematic 7](#_Toc89191529)

[Table 1 Technical Requirements 4](#_Toc89191533)

[Table 2 Technical Requirements 4](#_Toc89191534)

[Table 3 Cruise Control Model Validation 7](#_Toc89191535)

[Table 4 Motor Speed Model Validation 8](#_Toc89191536)

Notes: make the center alignment.

# 1 Introduction

This document contains information and details regarding the workflow used to create the PID controller, tuning of PID, and generation of code for cruise control and motor speed control project. GitHub is used as a version control system for the project. It is integrated with MATLAB to easily facilitate the GitHub process.

Control systems are needed when the system deals with continuously varying parameters or external disturbances such as load, friction, wind, etc, which will affect the output of the system. To maintain the stable output of the system even in the presence of external disturbances a controller is needed to control those output variations. There are a lot of controlling algorithms available for the control system, we will use only a PID controller for cruise control and motor speed.

# 2 Software Development Life Cycle

This section gives an overall view of the software development process used to develop the cruise control and motor speed model, starting with requirements, design, development, testing, and validation.

## 2.1 Requirement Gathering

This is the first stage of the V-development cycle contains a detailed understanding of requirements and expectations for the final product.

### 2.1.1 PID controller

#### 2.1.1.1 Technical Requirements

* To design the PID controller using the following equations

Where,

* PID Controller block should contain discrete function blocks.

#### 2.1.1.2 Non-Functional Requirement

* PID controller model should be designed and made it to referenced model.

### 2.1.2 Cruise Control

#### 2.1.2.1 Function Requirement

The system should have the following functional requirement

* Speed of the car should not fluctuate with respect to the external disturbances.

#### 2.1.2.2 Technical Requirement

The system should have the following technical requirements

|  |  |
| --- | --- |
| S.No | Requirements |
| 1 | Rise time < 10s |
| 2 | Overshoot < 10% |
| 3 | Stead state error <1% |

Table 1 Technical Requirements

### 2.1.3 Motor Speed

#### 2.1.3.1 Technical Requirement

The system should have the following technical requirements

|  |  |
| --- | --- |
| S.No | Requirements |
| 1 | Rise time < 5s |
| 2 | Overshoot < 5% |
| 3 | Stead state error <1% |

Table 2 Technical Requirements

### 2.1.4 Code

Developed code for the controller should have the following requirements:

* Code should be optimized for RAM efficiency.
* To develop the code in accordance with ISO26262.
* Code should follow MISRA C Guidelines.

## 2.2 Design

### 2.2.1 PID Controller

#### 2.2.1.1 P Controller Design

P controller is design by implementing the following equation in Simulink. The following is model is made as separate subsystem to make it as a modular design.

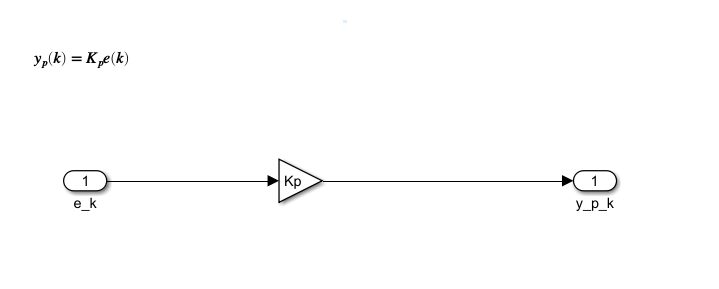


Figure 1 P Controller Implementation

#### 2.2.1.2 I Controller Design

I controller is design by implementing the following equation in Simulink. The following is model is made as separate subsystem to make it as a modular design.

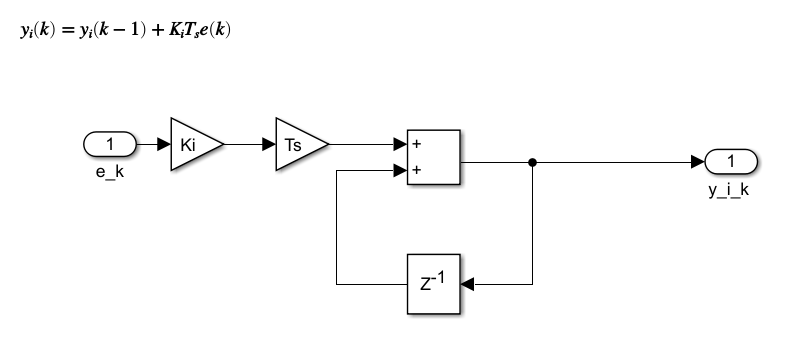


Figure 2 I Controller Implementation

#### 2.2.1.3 D Controller Design

D controller is design by implementing the following equation in Simulink. The following is model is made as separate subsystem to make it as a modular design.

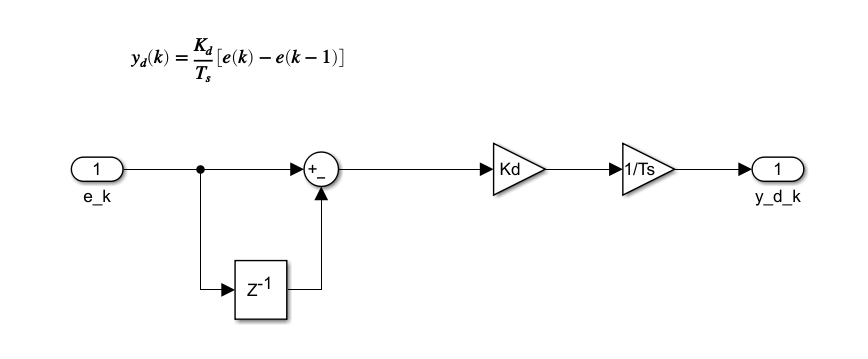


Figure 3 D controller Implementation

#### 2.2.1.4 PID Controller Design

P, I, D controllers which are designed as a separate subsystem are integrated together by implementing following equation. Sampling time of the PID Controller is set to value Ts=0.01.

This model is the converted to an atomic subsystem to make it as a referenced subsystem. Referenced subsystem are useful in case of multiple definition of the same PID controller is needed. This referenced subsystem is then used for Cruise control model and Motor speed control.

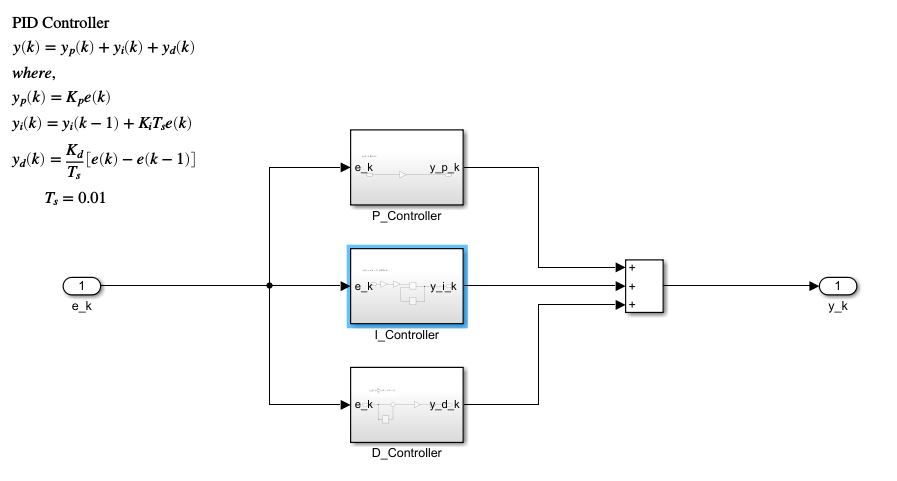


Figure 4 PID Controller Implementation

### 2.2.2 PID Tuning Algorithm

Script ‘PID\_Turning\_Script.mlx’ is created to obtain Kp, Ki and Kd values to satisfy the requirements. Script uses trial and error method to find the values. Calculation for overshoot, rise time, steady state error for the output signal is to be implemented in the script to parallelly check whether the requirements are met with corresponding Kp, Ki and Kd values.

Script will also generate a short report (refer Figure 5) of output signal and its properties such as rise time, overshoot percentage, output value at simulation stop time, steady state error percentage for the corresponding Kp, Ki, Kd values.

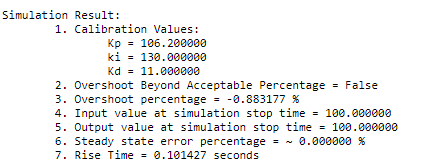


Figure 5 Simulation Report

### 2.2.3Cruise Control

#### 2.2.3.1 Analysis

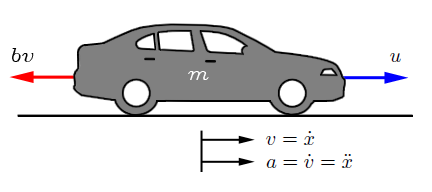


Figure 6 Cruise Control Schematic

The vehicle, of mass m, is acted on by a control force, u. The force u represents the force generated at the road/tire interface. For this simplified model we will assume that we can control this force directly and will neglect the dynamics of the powertrain, tires, etc., that go into generating the force. The resistive forces, bv, due to rolling resistance and wind drag, are assumed to vary linearly with the vehicle velocity, v, and act in the direction opposite the vehicle's motion.

#### 2.2.3.2 System equations

With these assumptions we are left with a first-order mass-damper system. Summing forces in the x-direction and applying Newton's 2nd law, we arrive at the following system equation:

#### 2.2.3.3 System parameters

Parameters of the system are assumed with following values:

* Mass of the Vehicle (m) = 1000
* Damping Coefficient (b) = 50

### 2.2.4 Motor Speed

#### 2.2.4.1 Analysis

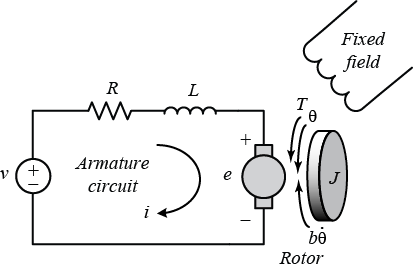


Figure 7 Motor Speed Schematic

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion. The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in the following figure.

The input of the system is the voltage source () applied to the motor's armature, while the output is the rotational speed of the shaft . The rotor and shaft are rigid. The friction torque is proportional to shaft angular velocity.

#### 2.2.4.2 System Equations

The torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field. In this example we will assume that the magnetic field is constant and, therefore, that the motor torque is proportional to only the armature current  by a constant factor  as shown in the equation below. This is referred to as an armature-controlled motor.

The back emf, , is proportional to the angular velocity of the shaft by a constant factor

The motor torque and back emf constants are equal, that is, ; therefore, we will use  to represent both the motor torque constant and the back emf constant. We can derive the following governing equations based on Newton's 2nd law and Kirchhoff's voltage law.

#### 2.2.4.3 System Parameters

Parameters of the system are assumed with following values:

* Moment of inertia of the rotor () = 0.01
* Motor viscous friction constant () = 0.1
* Electromotive force constant () = 0.01
* Motor torque constant () = 0.01
* Electric resistance () = 1
* Electric inductance () = 0.5

## 2.3 Development and Coding

Script for cruise control model is located inside ‘TASK 3/A\_CruiseControl/Scripts’ and for motor speed control it is located inside ‘TASK 3/B\_MotorSpeed/Scripts’.

### 2.3.1 Cruise Control

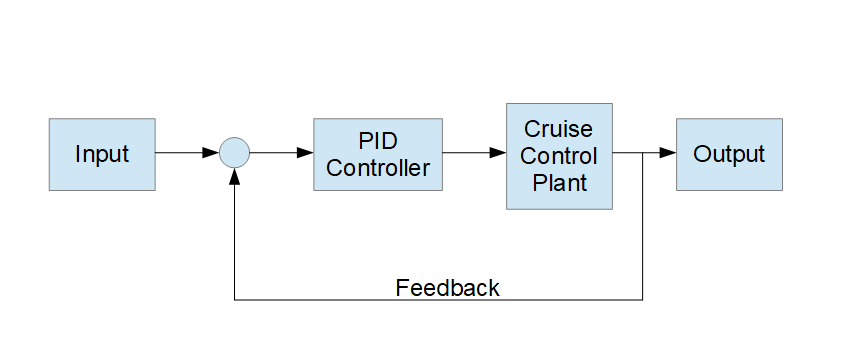


Figure 8 Cruise Control Block Diagram

### 2.3.2 Motor Speed Control

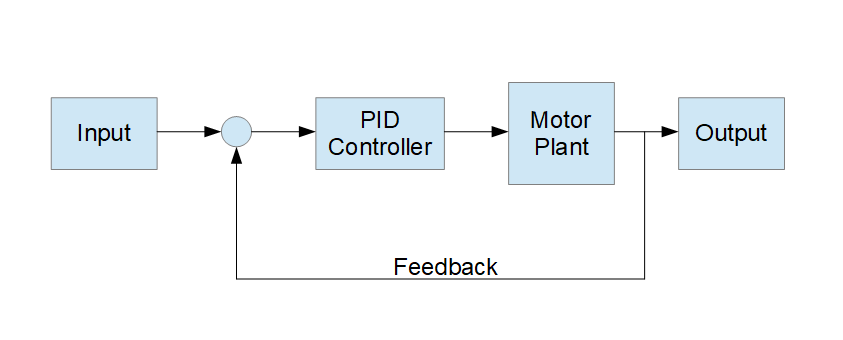


Figure 9 Motor Speed Control Block Diagram

-Autocode generation procedures, Documentation procedures used.

- code advisor report details

## 2.4 Testing

-Unit testing results, Polyspace statics analysis result. Cpp Check result

## 2.5 Validation

-Comparing requirements and output of Testing to validate.

-add risetime graphs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Requirements | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | Rise time < 10s |  |  |  |  |
| 2 | Overshoot < 10% |  |  |  |  |
| 3 | Stead state error <1% |  |  |  |  |

Table 3 Cruise Control Model Validation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Requirements | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | Rise time < 5s |  |  |  |  |
| 2 | Overshoot < 5% |  |  |  |  |
| 3 | Stead state error <1% |  |  |  |  |

Table 4 Motor Speed Model Validation

# 3 Advantages of the used SDLC model

* Mention the advantages of the SDLC model used with the development of the pid.

# 4 GitHub Workflow

* Mention the github links.
* Paste the flow chart of comits and branches as a picture

GitHub is integrated with Matlab, and all the versioning process is done within matlab itself.

# 5 References