**Michigan Technological University**

**EE 5750**

**Distributed Embedded Control Systems**

**Lab Experiment 04**

**Closed Loop Electronic Throttle Control**

**Submitted By:** Kirk D’Souza

Lopamundra Baruah(Lab partner)

**Instructor:** Dr. Bo Chen

**T.A.:** Ming Cheng

**Date Performed:** 2/22/2016

**Date Submitted:** 3/14/2016



1. **INTRODUCTION**

The purpose of this lab is to develop a feedback position control system for an electronic throttle control using a potentiometer to simulate the accelerator pedal position and a throttle position sensor as the feedback. The main objective of the lab is to use a discrete controller to control the throttle body opening based on the received throttle command and understand the relation between the proportional, derivative, and Integration gains and how the system is effected by manipulating the.

1. **SYSTEM INPUT/OUTPUT AND CONTROL LOGIC**

The system setup consists of a Bosch D V-E5 throttle body for ETC systems and a Woodward ECM 565-128. The connections for the same are as given below:

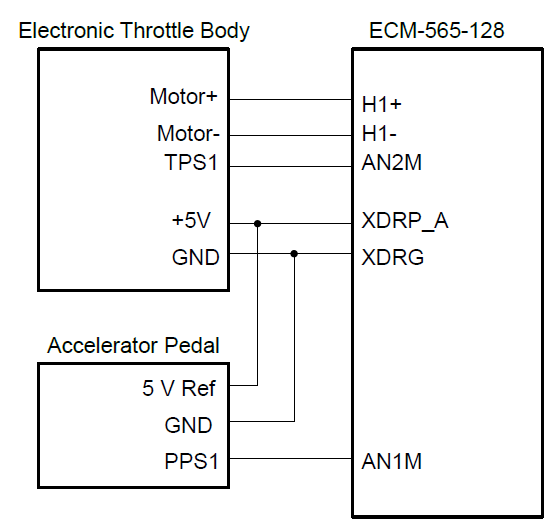


Figure 1: Circuit

The throttle body has a servo motor which is driven through a H-bridge in the ECM.

The TPS feedback which gives our error is taken from a block in the model.

The inputs for our model are the PPS Offset, PPS Gain, TPS Offset, Kp, Kd, and Ki

**Control Logic:**

* Input is taken from the Pedal Position Sensor (PPS) which is given by the user as a potentiometer setting.
* The PPS is calibrated via the gain and offset to synchronize with the TPS values
* The PID output is set to minimize the error by manipulating the Kp, Ki, and Kd values accordingly
* The output is converted into PWM signals via a Motothawk PWM block, which are used to modify the position of the throttle valve
* The electronic throttle body changes the position of the throttle valve via a DC servomotor, according to the PWM signals
* The feedback is taken from the Throttle Position Sensor (TPS) from the ETC which gives the current Throttle valve position
* This Feedback is again used for the error measurement for the PID control

1. **MODEL DEVELOPMENT**

The model is built in Simulink, Matlab using the Simulink and Motohawk blocks.

The main model calibrating the foreground model with the ECU is given below:

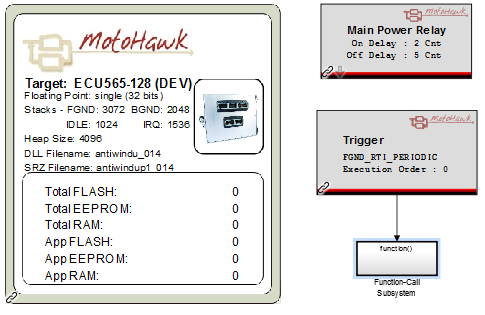


Figure3.1: Main Model

The Foreground model is given below:

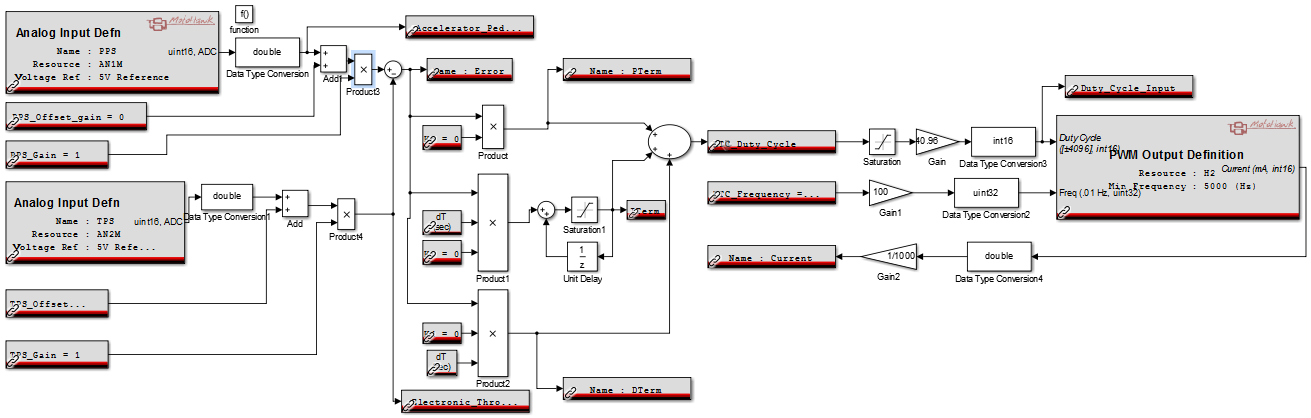


Figure3.2: Foreground PID Model

**PPS Calibration Blocks:**

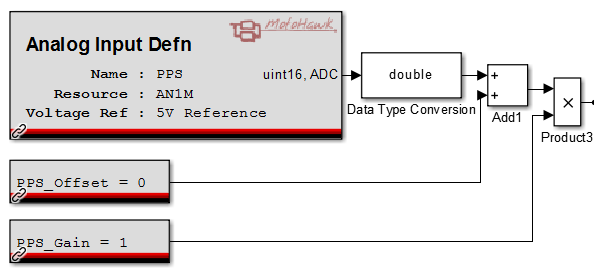


Figure3.3: PPS Blocks

Here, the Pedal position values are taken from the ECM. An Analog Input definition block is used to take those values as input for the PID controller. A data conversion block is used to convert it into the Simulink double format. Two Calibration blocks are used to help calibrate the PPS offset and gain in real time when using the Mototune software.

**TPS Calibration Blocks:**

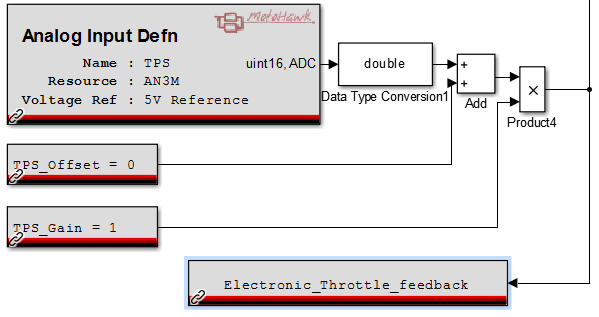


Figure3.4: TPS Blocks

Here, the Throttle position values are taken from the throttle body via the ECM. An Analog Input definition block is used to take those values as input for the PID controller. A data conversion block is used to convert it into the Simulink double format. Two Calibration blocks are used to help calibrate the TPS offset and gain in real time when using the Mototune software.

**PID Controller:**

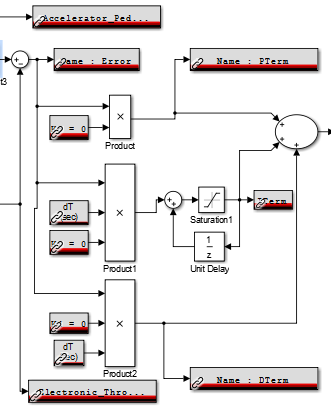


Figure3.5: PID Blocks

The PID controller takes the PPS and Throttle feedback difference (measured error) as input for the Proportional, Integral, and Derivative Controllers. All the controllers have a calibration block to set their respective gains in real time when using the Mototune software. All the gains are initially set to 0 The Sample time for the Integral and Derivative Controllers is given from the dT blocks.

A Saturation a Unit Delay block is used at the output for the integral controller.

**PWM Output Controller:**

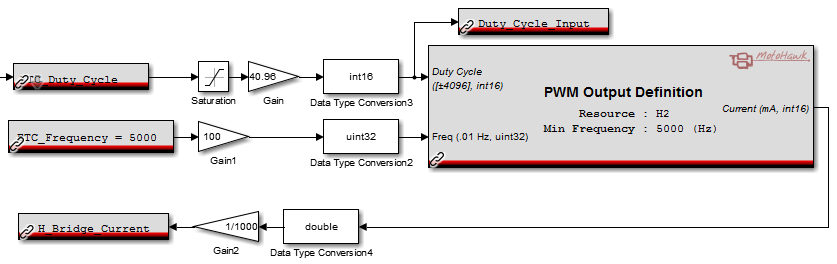


Figure3.6: PWM Output Blocks

Here, the PWM block is used to convert the PID controller output into PWM signals for the DC servo motor in the throttle body. The saturation block is used to set the minimum and maximum limits and the gain is calculated such that the duty cycle can go from 0 to 100%. Real time calibration of the Duty cycle can be done via the Override block. The H-Bridge current can also be monitored in the Mototune software.

**Additional feature:**

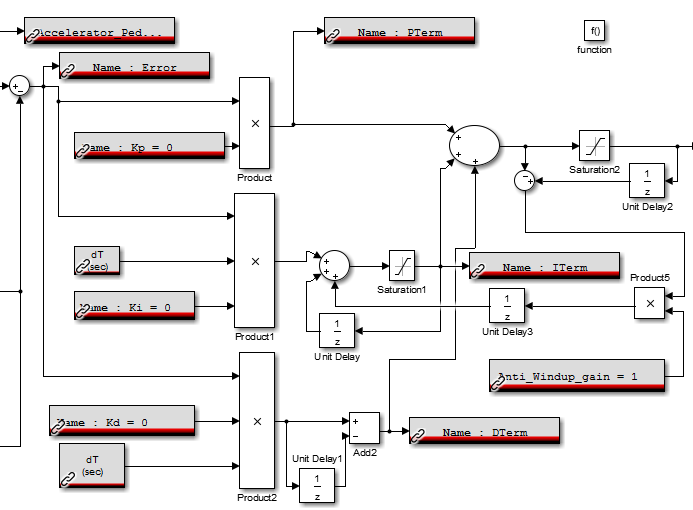


Figure3.7: PID Controller with Anti-Wind up Blocks

Here, to avoid the wind up problem when there is an input saturation, which causes the Integral controller to keep integrating the built up error, a back calculation process is utilized.

Here, the difference between the saturated value and the filter limit is taken as feedback for the Integral controller. An Anti-wind up gain is also added via a calibration block so that it can be calibrated in real time.

**Anti-Wind up Model:**

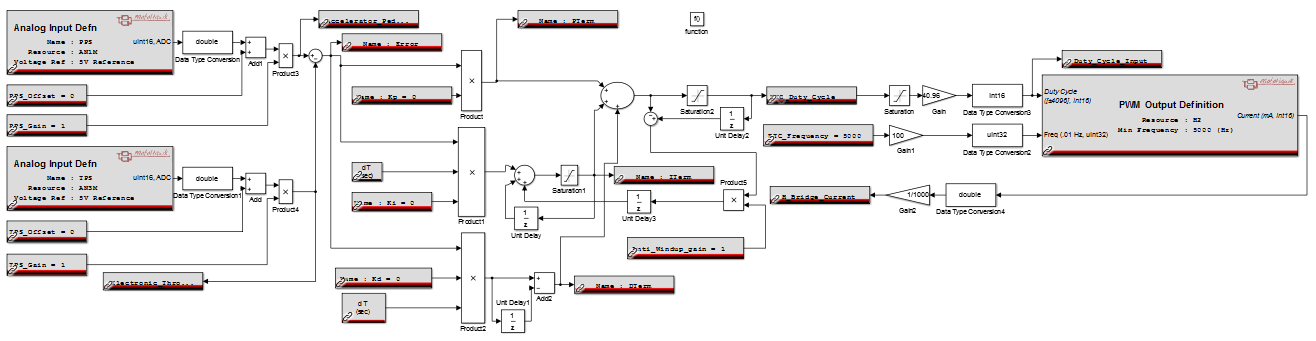


Figure3.7: PID Controller with Anti-Wind up model

1. **SIMULATION AND CALIBRATION RESULTS**

**Calibration for PPS and TPS:**

The PPS and TPS have to be in the same range so that the measured error value can be accurately obtained. Here, the DC servo motor in the Throttle body requires a duty cycle of 40% for the throttle valve to be at the maximum open position and 0% to be in a completely closed position.

To set the PPS and TPS values accordingly, a PPS gain of 0.1 and a TPS Offset of -179 is taken

**Calibrated Controller**

Taking inputs as

Kp=13

Ki=0.1

Kd=0.6

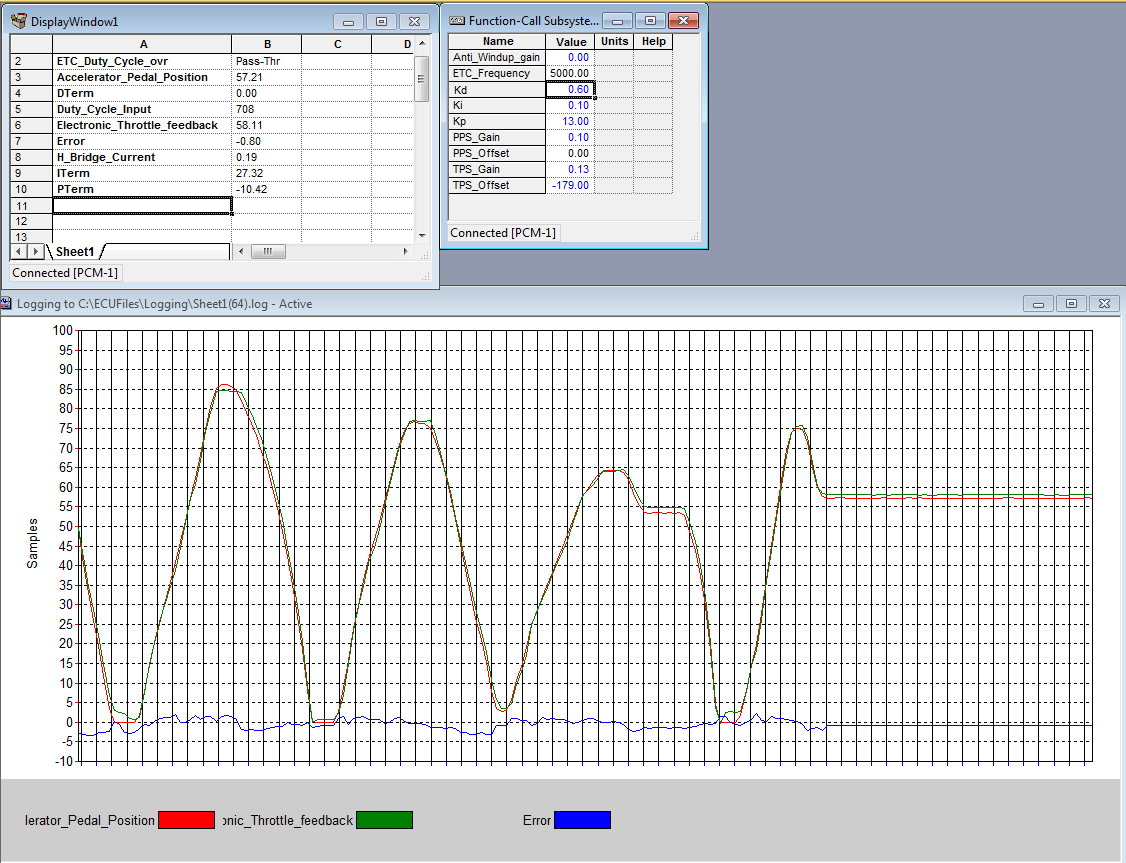


Figure 4.1: Tuned PID Controller

**Effect of Kp**

Kp = 15 from 13

Ki = 0.1

Kd = 0.6

Oscillations occur

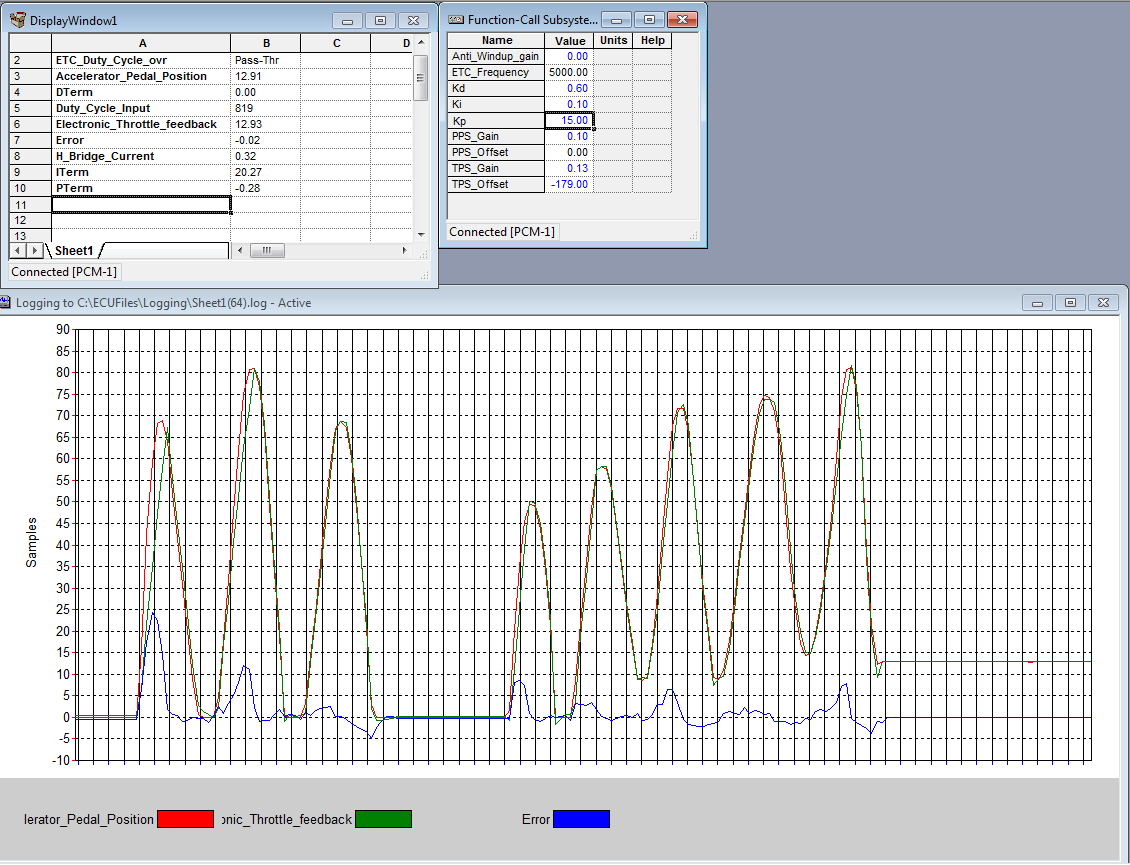


Figure 4.2: Effect of Kp

**Effect of Ki**

Kp = 13

Ki = 0.1 to 0.8

Kd = 0.6

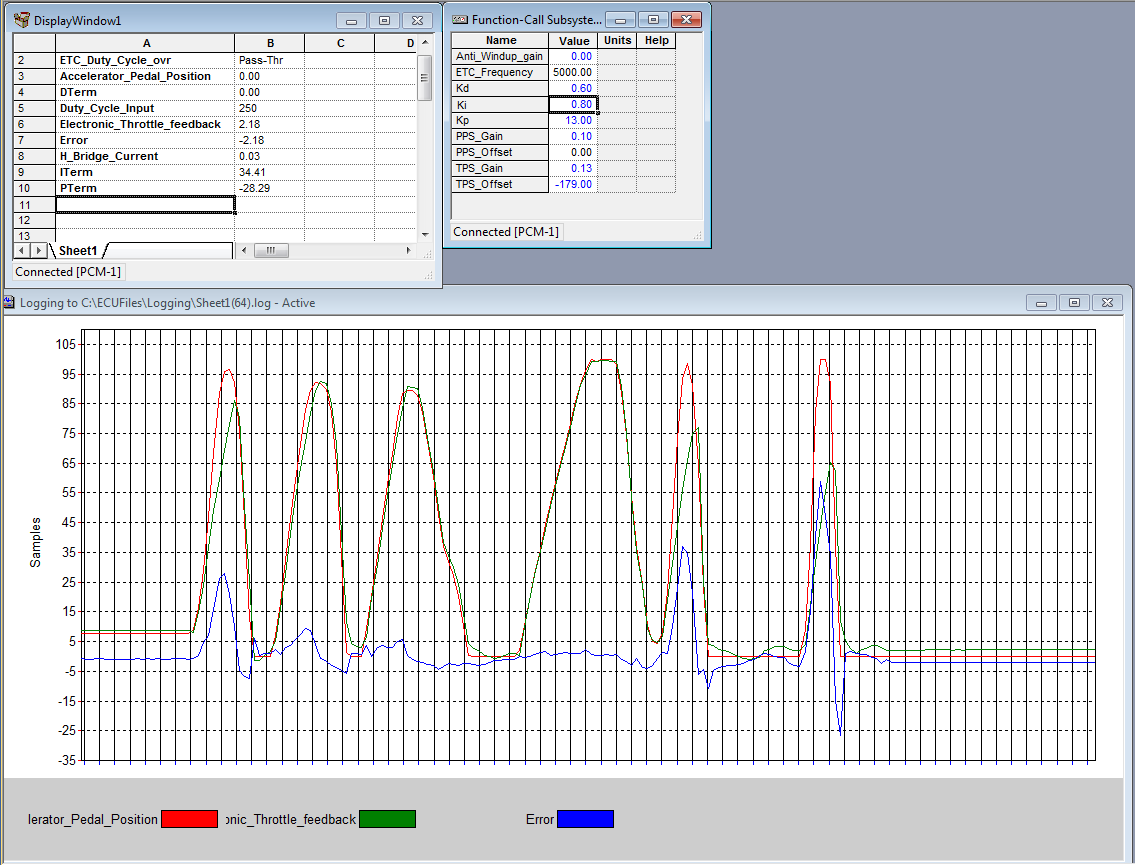


Figure 4.3: Effect of Ki

**Effect of Kd**

Kp = 13

Ki = 0.1

Kd = 0.3 from 0.6

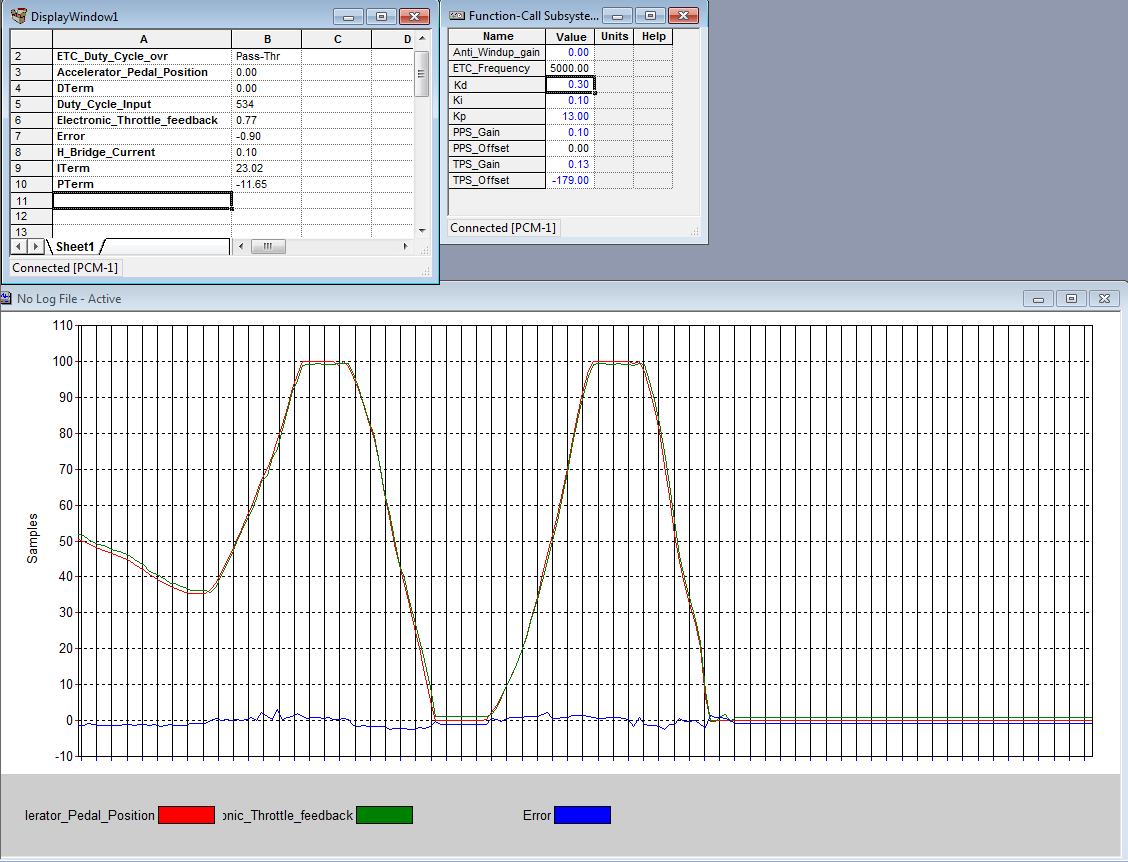


Figure 4.4: Effect of Kd

**Response of Throttle FB with change in APP**

Kp = 13

Ki = 0.1

Kd = 0.3

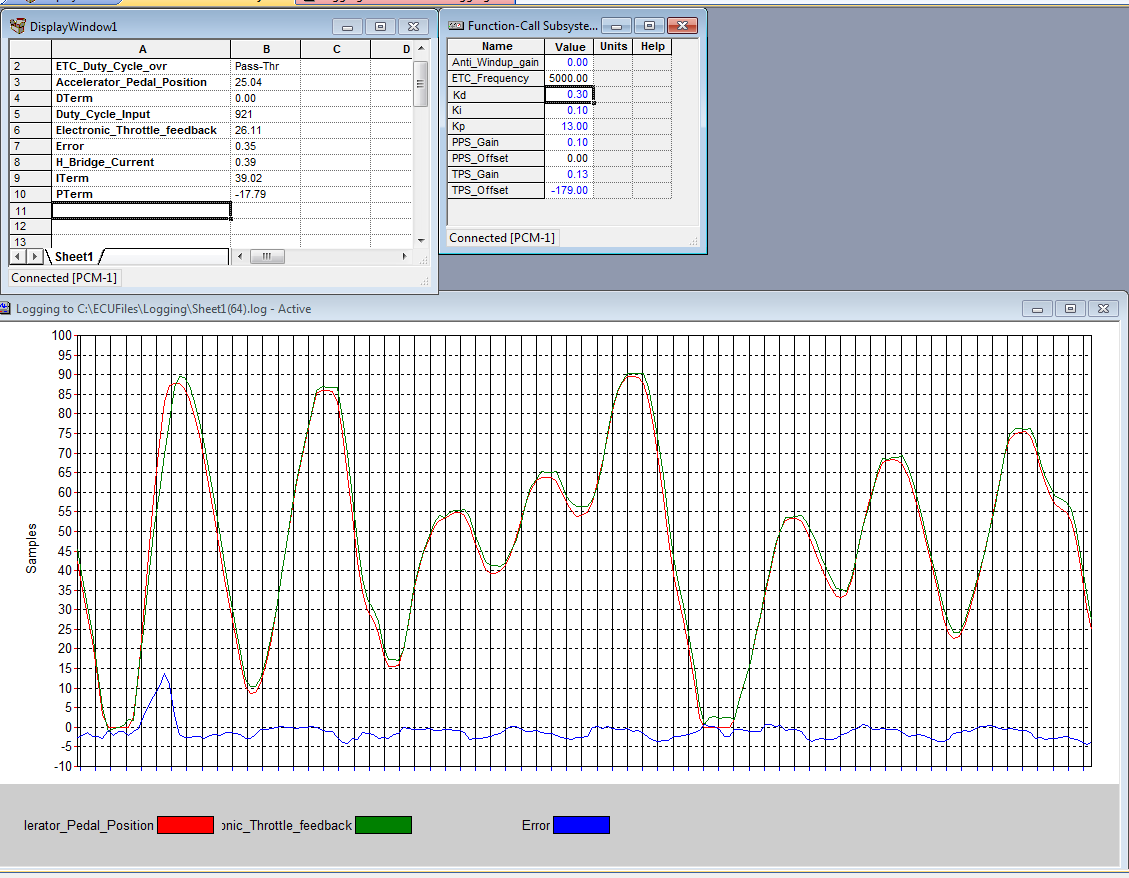


Figure 4.5: Response of Throttle Feedback with change in APP

**Effect of Anti-wind up gain**

anti-wind up gain = -0.1 from 0

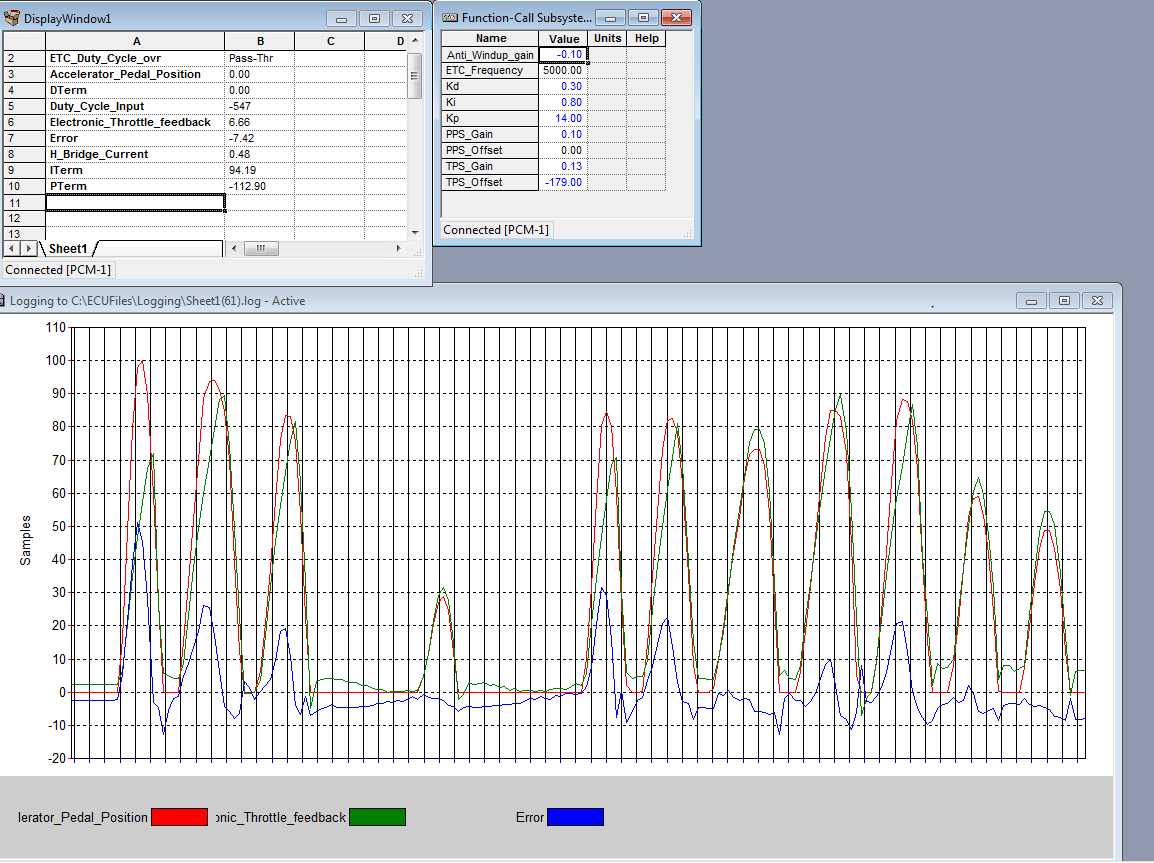


Figure 4.6: Response of Anti-Wind up model

1. **VALIDATION**

**Proportional Control:**

The graph given below is from logged data when the Kp gain in the model was increased. The outputs of the changed Kp is shown in the left side of the graph

From the graph and Figure 4.2, it can be seen that changing the proportional gain (Kp) decreases the rise time, increases overshoot.

**Integral Controller:**

The graph given below is from logged data when the Ki gain of the model was increased.

It can be seen from the graph and from Figure 4.3,that increasing the Integral gain eliminates the steady state error

**Derivative Control:**

The graph given below is from logged data when the Kd gain of the model was increased.

It can be seen from the graph and Figure 4.4, that increasing the Derivative Gain increases the rise time, decreases the overshoot

**For Tuned PID Controller:**

The graph given below is from logged data of the model with the PID Controller set with the optimum values with Kp = 13, Ki = 0.1, and Kd = 0.3.

It can be seen from the graph and from Figure 4.1 and 4.5 that the feedback response is nearly the same as the Accelerator Pedal Position and the error is very close to 0 for most of the changes in the system inputs.

1. **DISCUSSION**

**ETC:**

In ETC systems, the ECM uses electronic signals from the Throttle Position Sensor (TPS), and Accelerator Pedal Position (APP), and other sensors to control the throttle valve. When the APP is changed, the pressure applied on pedal is converted into an electric signal. The signal is sent to an ECM which also takes other outside variable into consideration. The ETC motor is duty cycle controlled by a circuit that can reverse polarity to change the direction of rotation of the motor. Since the ETC is a closed loop system, the throttle opens based on the user input, which is taken from the APP, and adjusts itself based on the feedback error, which is obtained from the TPS.

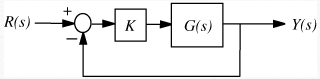


**Proportional Controller:**

A proportional Controller produces an output that is proportional to the error signal. The output is a product of the error term and a constant, which is the proportional gain Kp.

Setting a large Kp will make the system more responsive to changes in the error. However, the magnitude of the change will also be much larger. Thus, if the proportional gain set is too high, then overshoot will occur, oscillations will occur and the system can become unstable.

Setting a small Kp will make the system less responsive to changes in the error and may increase the rise time.



**Integral Controller:**

An Integral Controller produces an output that is proportional to the Integral of the error signal. The output is a product of the integral of the error term and a constant, which is the Integral Constant Ki

It is used to eliminate steady state error.

However, if the system keeps oscillating and never reaches a steady state, then adding an integral controller will make the system unstable and less responsive when responding to changes in the error signal

**Derivative Controller:**

A Derivative Controller produces an output that is proportional to the rate of change of the error signal. The output is a product of the derivative of the error term and a constant, which is the Derivative Constant Kd.

Adding a derivative controller dampens the system and improves the stability, hence it can be used to decrease overshoot and settling time. However, it has no effect on the steady state error.

**Steady State Error:**

The Steady State Error gives the difference between the constant input and constant output of a system. It is the error produced when the system response (output) has reached a steady state.

Various factors can affect the steady state error like the noise in the system input and wear and tear of the sensors in the throttle body. The noise in the system can affect the output response of the controller especially because of the Derivative term. With time, the wear and tear of the sensors used in the Throttle body may affect the readings obtained in the feedback response of the system, which will increase the Steady State Error.

The Steady State Error can be reduced by using the integral controller.

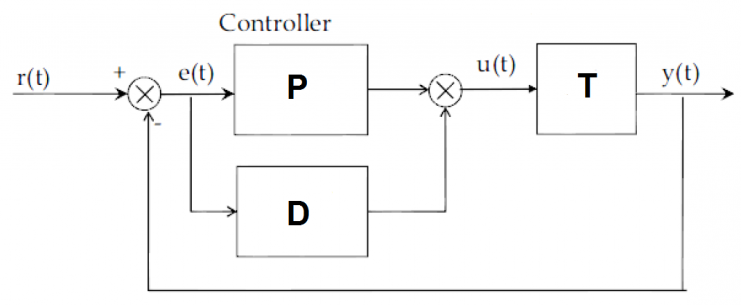
In the tuned PID controller designed for this lab, an Integral gain of 0.1 is used. This eliminates the steady state error in the system feedback as can be seen in Figure 4.1

**PD Controller:**

A PD Controller uses a combination of the Proportional and Derivative terms for controlling the output of a closed loop system.

Using a PD Controller will reduce the overshoot, settling time and improves the system damping. It can be used in systems where it is important to get a system to respond to changes in error quickly and accurately.

A basic structure for the PD Controller is given below:



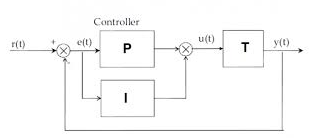
**PI Controller:**

A PI Controller uses a combination of the Proportional and Integral terms for controlling the output of a closed loop system.

Removing the Derivative term makes the system more stable during steady state when using noisy data because derivative controllers are more sensitive to the high frequency terms.

The Integral controller mitigates the offset from the Proportional Controller and also decreases the rise time. However, if the Proportional gain is too large then the overshoot may be too high.

A basic structure for the PI Controller is given below:



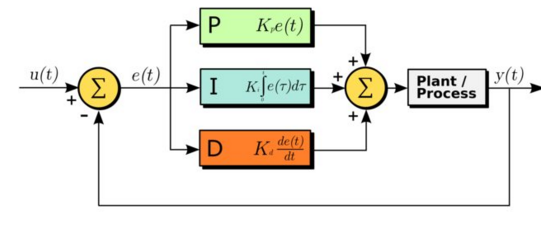
**PID Controller:**

A PID Controller uses a combination of the Proportional, Integral, and Derivative terms for controlling the output of a closed loop system.

A properly tuned PID controller with optimum values for each of Kp, Ki, and Kd gives the desired control response. Tuning the PID Controller requires proper knowledge about the effects of each term.

A problem with the PID controller is that it will amplify noise because of the derivative term and produce large changes in output

A basic structure for the PID Controller is given below:



**Anti-Windup PID Controller:**

For general PID controllers, if there is a large built up error, then the integral term will keep integrating the error even if the error input is saturating. This results in a very large settling time, and large oscillations.

To avoid this problem, an anti-windup scheme can be used. There are several methods for implementing anti-wind up such as back-calculation, clamping, etc

Here, the back–calculation method is used where in a feedback loop and saturation limits are used to modify the Integral term.

The Anti-Windup controller has no effect on the output of the system when there is no saturation

The change in system response parameters with the increase of each of Kp, Ki, and Kd gains is given in the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CL RESPONSE** | **RISE TIME** | **OVERSHOOT** | **SETTLING TIME** | **S-S ERROR** |
| **Kp** | Decreases | Increases | Small Change | Decreases |
| **Ki** | Decreases | Increases | Increases | Eliminates |
| **Kd** | Small Change | Decreases | Decreases | No Change |

**Tuning the PID Controller:**

Tuning the PID controller and minimizing the error is an important part of obtaining an optimum system response. Knowledge of the effects and system response to each of the Kp, Ki, and Kd gains is essential for obtaining optimum values. There are several processes to obtain these values.

For our model, all the Kp, Ki, and Kd gains were first set to 0. Then the Kp gain was set such that the desired rise time is obtained. The Kd gain was the increased until the overshoot was within the desired parameters and the system becomes critically damped. The Ki gain is set such that the steady state error is eliminated while keeping the overshoot and initial oscillations within the acceptable parameters.

1. **CONCLUSION**

This Lab helped us get a clear understanding of the PID controller and how it is to be designed. We learned how to calibrate the Proportional, Integral, and Derivative Gains in real time and how each of them affects the output of the controller. For optimum response, the steady state error will have to be as close to 0 as possible and the rise time, settling time, and overshoot of the system have to be kept at a minimum. Although the PID controller gives the desired output, there can be cases when the output is not as good as desired. This can be through various factors such as input saturation as was analyzed in this lab. To counter this, Anti wind up can be effectively used.

1. **REFERENCE**
2. EE 5750 Lecture and Lab notes by Dr. Bo Chen
3. Bosch D V-E 5 Throttle body for ETC Datasheet
4. Woodward ECM-0565-128-0701-C Engine Control Module (Part No. 237-1238)
5. <http://jalopnik.com/how-electronic-throttle-control-works-499966101>
6. <http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/electronic-throttle-control-systems.htm>
7. <http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID#20>
8. <https://en.wikipedia.org/wiki/Proportional_control>
9. <https://en.wikipedia.org/wiki/PID_controller#PI_controller>
10. <http://programmers.stackexchange.com/questions/214912/why-does-a-proportional-controller-have-a-steady-state-error>
11. <http://www.ni.com/white-paper/3782/en/>
12. <http://www.mathworks.com/help/simulink/examples/anti-windup-control-using-a-pid-controller.html>
13. <https://en.wikipedia.org/wiki/Integral_windup>
14. cse.lab.imtlucca.it/~bemporad/teaching/ac/pdf/AC2-09-**AntiWindup**.pdf
15. <http://iterativecode.com/blog/2012/05/10/a-better-alternative-to-position-pid-for-multirotor-control/>
16. <http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter5/ch5_1_optimizations>
17. <http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter5/ch5_1_optimizations>