Industrial Automation Project 1

Kedar More

Abstract

This report will continue from the previous project that is Project 0. It will walk through the process of switching the main controller which was Arduino Nano and it was changed to Arduino Due. The parts to work on will be getting the Due started and uploading the code or logic via Matlab Simulink and then build a controller to make the system stable. For that to happen the Due must accept the raw data from the potentiometer and send signal to the BLDC motor accordingly. The final goal is to make the arm hover at a specified angle with little or no overshoot after external excitation.

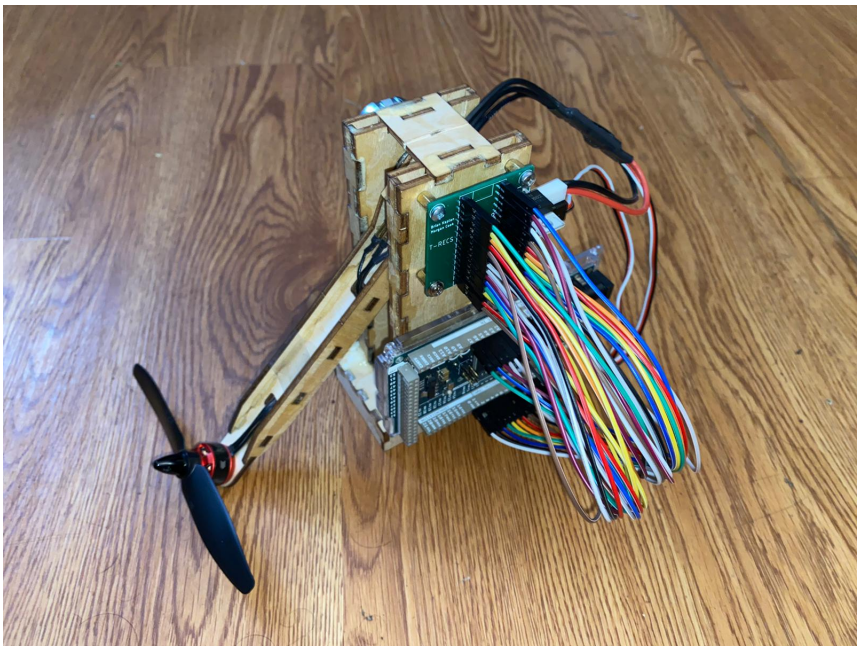


Figure 1: TRECS Assembly[3]

1 Introduction of Components

1. Arduino Due

   While swapping out the Nano with Due the same pins were connected with the male female connectors to the PCB.

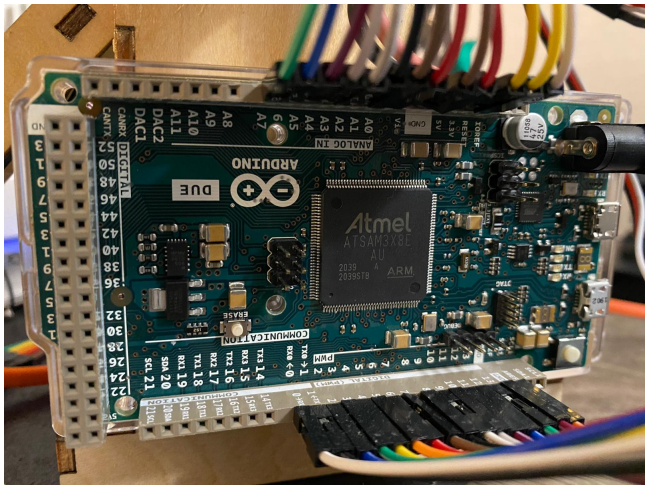


Figure 2: Arduino Due

The basic specs we need to look for in the Due is the Digital to Analog conversion resolution which is a 12 bit conversion. This corresponds to the voltage levels which vary from 0 to 4095. From the documentation we also get to know that the actual voltage maps from 0.55 V to 2.75 V. Thus the smallest change which can be observed is $0.537 \cdot 10^{-3}$V.[1]

2. ESC

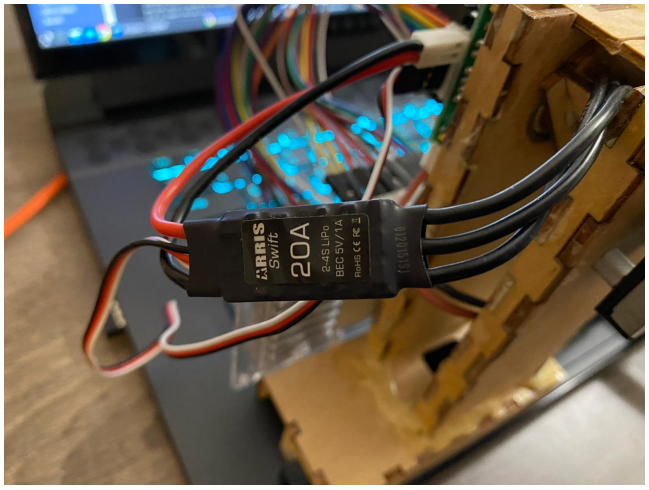


Figure 3: ESC

The ESC is a controller which is used to run the BLDC motor. It needs an Arming and Calibration sequence to initialise the motor and then the inputs can be given in terms of microseconds which are converted to angular velocity.[2]

2 Program Due via Simulink

Setting up Simulink and uploading Blink Program:

1. Install the two Add-ons for Arduino on Simulink

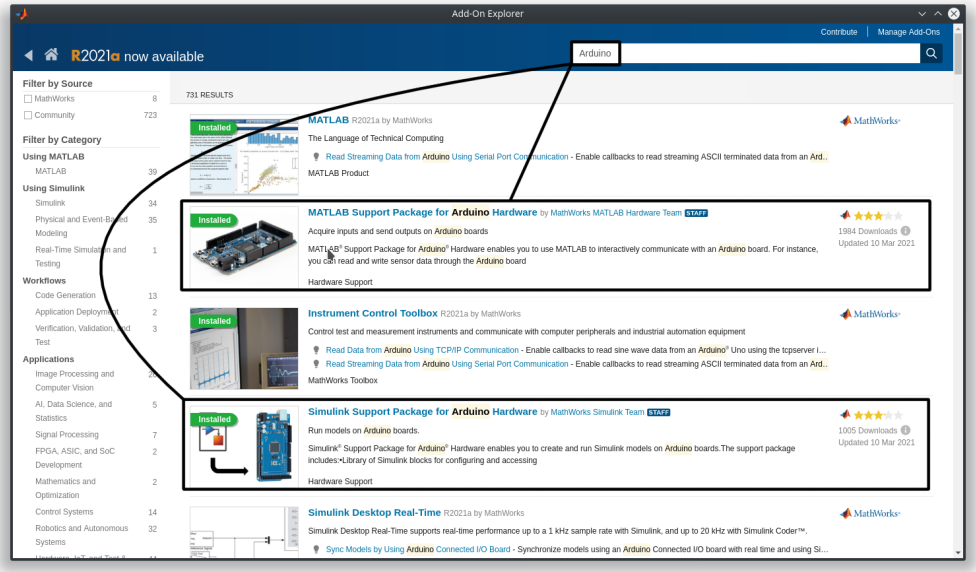


Figure 4: Add-ons in Matlab

2. Setup Arduino Due to be programed

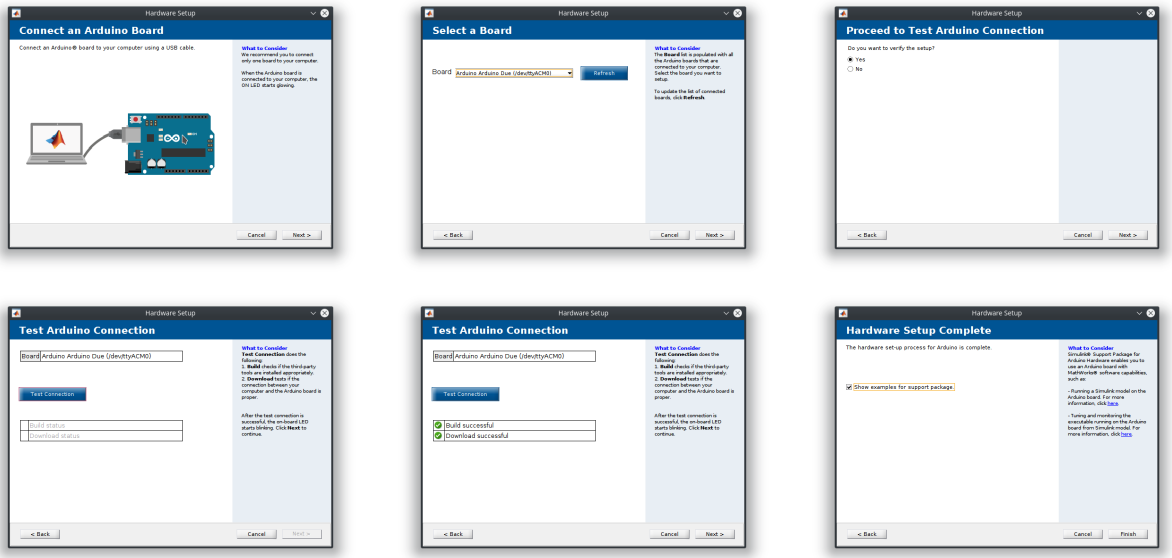


Figure 5: Arduino Due Setup

For initial connection we need to setup the board to build and run the code from the device.

3. Blink Program

   This is the basic program to test all the connections and the setup for the Due.

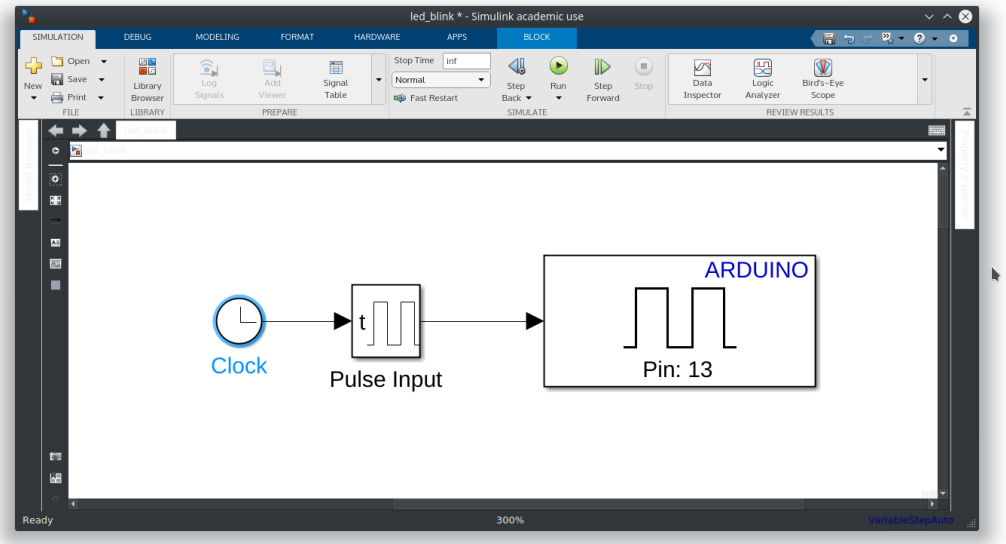
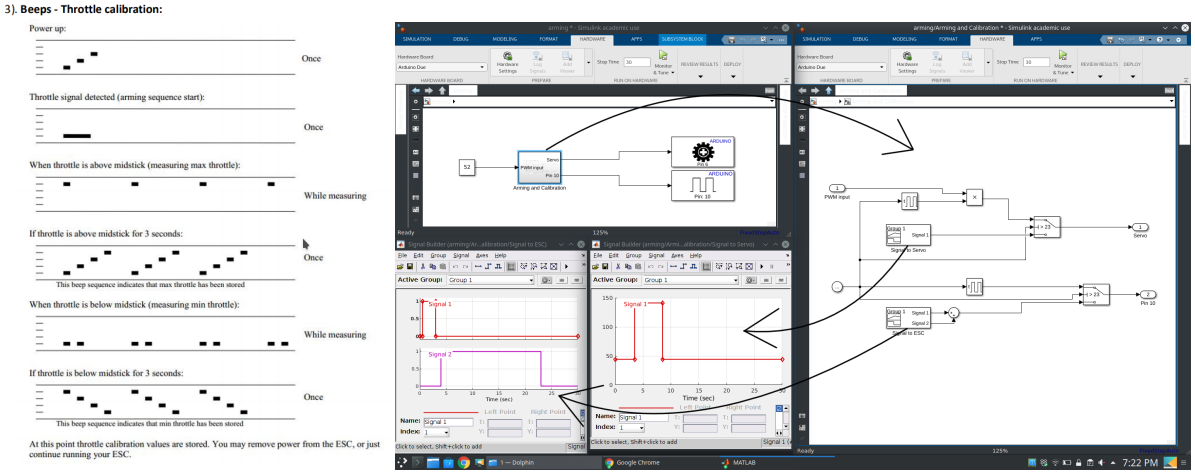


Figure 6: Blink Program

Arming and Calibration:

Looking at the data sheet for the ESC there is a certain sequence of pulses which are needed to be input as the servo motor input and the ESC pin input. The sequence being:

1. Power up the ESC and servo to full throttle value for 1 second to ARM the motor.
2. To start the calibration sequence ramp up the signal to highest in 3 seconds, here a peculiar sound is heard which increases in frequency as the input is increased.
3. the signal is then measured for a certain amount of time ($>$ 1 second) where the signal is constant.
4. The signal is the ramped down as it was ramped up. This time the sound heard decreases in frequency.
5. After these steps the motor can be given a constant signal in microseconds according to the speed you want.



(a) Datasheet[2] (b) Implementation

Figure 7: Arming Sequence

Here the microseconds values 1000 to 2000 corresponds to 0° and 180° respectively. But by observation the ESC values correspond to 544 to 2400. Hence by calculating the values for angle inputs for 1000 and 2000:

for 1000 microseconds

$$\frac{x-0}{1000-544} = \frac{180-0}{2400-544}$$

$$x = 44.224°$$

for 2000 microseconds

$$\frac{x-0}{2000-544} = \frac{180-0}{2400-544}$$

$$x = 141.207°$$

3 Linearzed Model

We have obtained the physical model of the system in the previous project. The basic equation is:

$$P = \frac{J}{L}\ddot{\theta} + d\dot{\theta} + mg\cos\theta$$

$$\theta'' + \frac{d}{mL^2}\theta' + \frac{g}{L}\cos\theta = \frac{PL}{J}$$

$$A = \frac{d}{mL^2}$$

$$B = \frac{g}{L}$$

$$C = \frac{PL}{J}$$

After performing certain experiments like drop test and hover test we go the values for the lumped parameters.

$$\theta'' + 37.001\theta' - 70.707\cos\theta = 3.834 * 10^{-3}u^2$$

To obtain the transfer function we use Laplace Transform:
Transfer Function:
Taking the Laplace transform of $\theta'' + A\theta' + B\cos\theta = C$
Linearizing about $\theta = 0$ we get $\cos\theta = 1$

$$s^2\mathcal{L}(\theta) + As\mathcal{L}(\theta) + B\frac{1}{s}\mathcal{L}(\theta) = 3.834 * 10^{-3}\frac{2}{s^3}\mathcal{L}(u)$$

$$\frac{\mathcal{L}(\theta)}{\mathcal{L}(u)} = \frac{3.834 * 10^{-3}\frac{2}{s^3}}{s^2 + As + B\frac{1}{s}}$$

$$\boxed{\frac{\mathcal{L}(\theta)}{\mathcal{L}(u)} = G = \frac{7.668 * 10^{-3}}{s^5 + 37.001s^4 + 70.707s^2}}$$

This is the theoretical equation of the system but in this project we are using the physical system itself to implement the controller.

4 Implementing Feedback Loop

For the feedback loop we will need an output measurement from the T-RECS. This will be taken in form of the angle of the arm. In my case the arm would go from -47° to 38° with respect to the horizontal. These values were measured by an external device. The analog values from the A0 pin at those points are 2600 and 4100. To convert them a map function block was implemented.

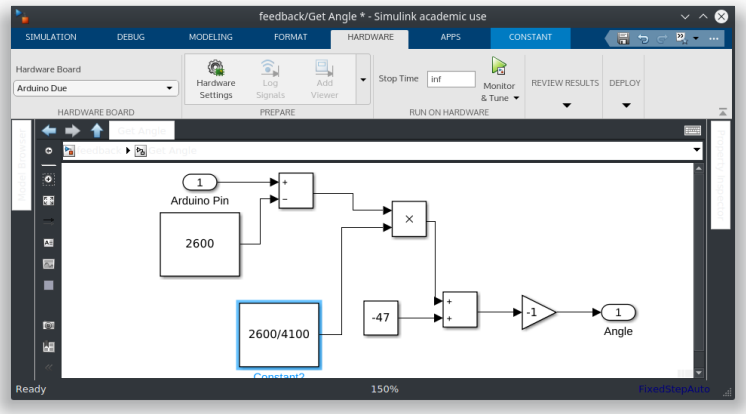


Figure 8: Converting raw Analog Value to Angle

We don't have the transfer function for the physical system so I had to tune the PID according to the output observations.

I tested the system with 2 major controllers P, PD. For the P controller there is too much variation with a small external disturbance. This is expected there is no derivative term to damp the output. Even after adding the derivative term the results didn't change much. By changing the Kd to a higher value the system response started becoming slow. The expected stage was reached but the jitters still remained.

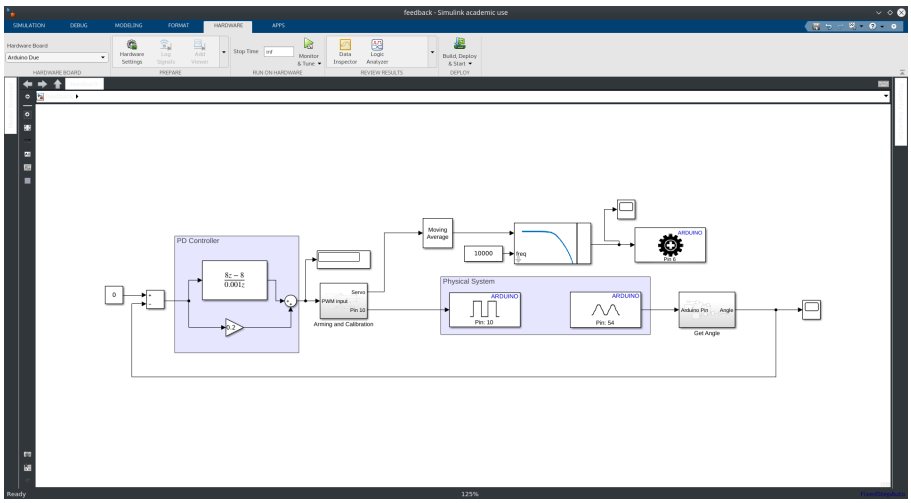
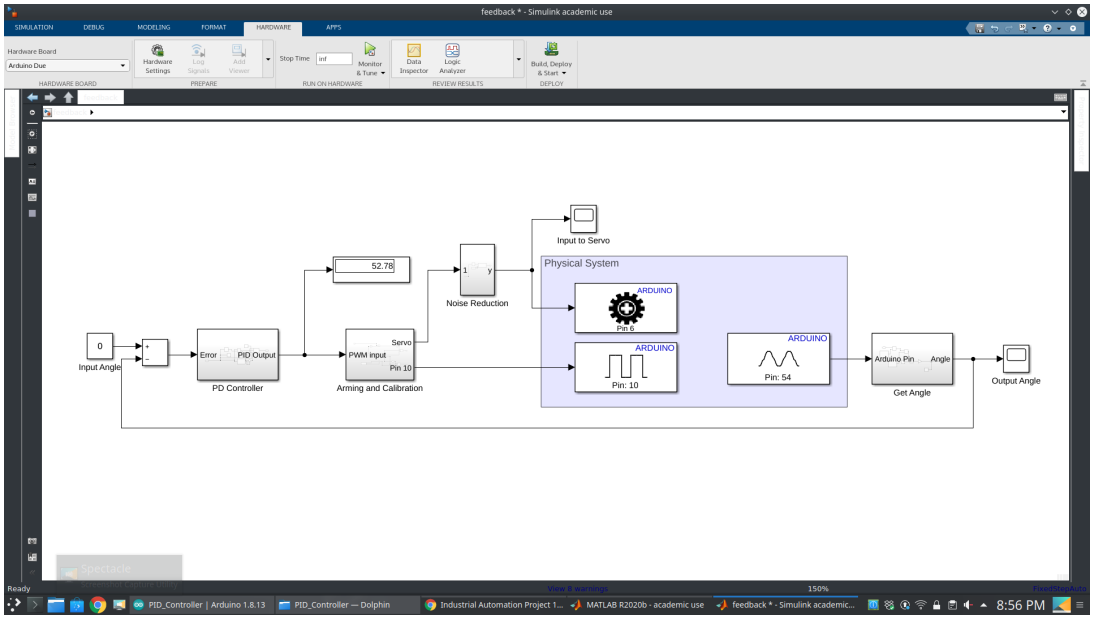


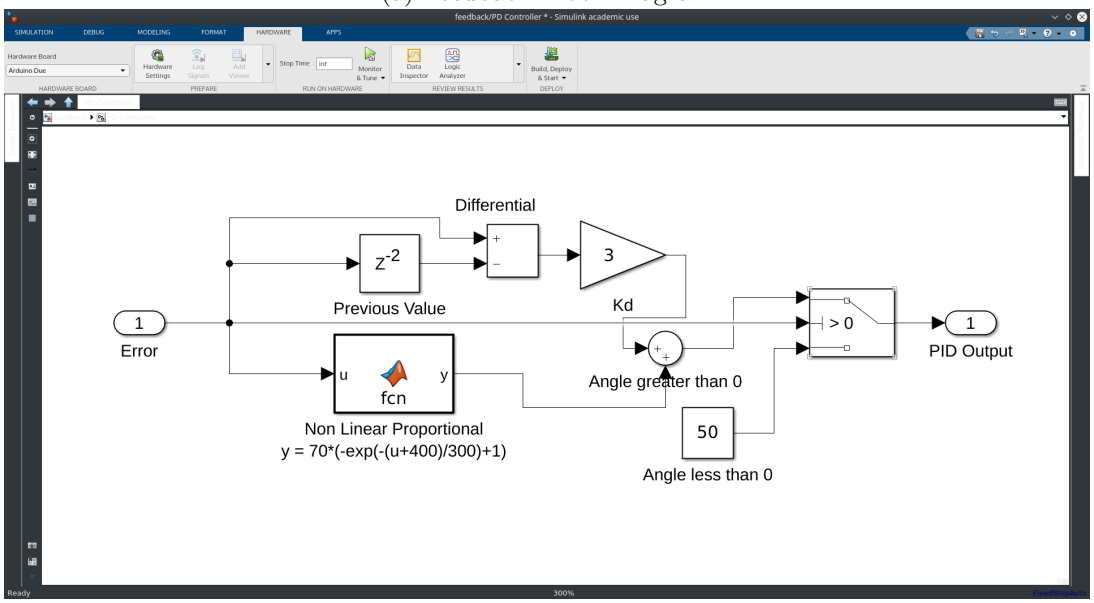
Figure 9: Initial Attempt with PD controller

The system is itself non linear so by using a linear controller won't do the job. Hence I tried using a non linear Proportional Controller.[4] The controller equation was designed such that at higher values of error the change in input is not much and the input suddenly decreases as we approach the input state. This is useful in term of an angular change. We need a higher input to go from a higher error to a steady state. And as we approach it the input should start remaining constant. This can not be achieved by a derivative term as it

depend on the change in output or the velocity term. The input should remain constant in spite of the approach velocity. But even then the system was too fast to be handled by the controller. Hence I tried adding a derivative term with Kd=3.



(a) Feedback Block Diagram



(b) Controller Block Diagram

Figure 10: Improved Controller

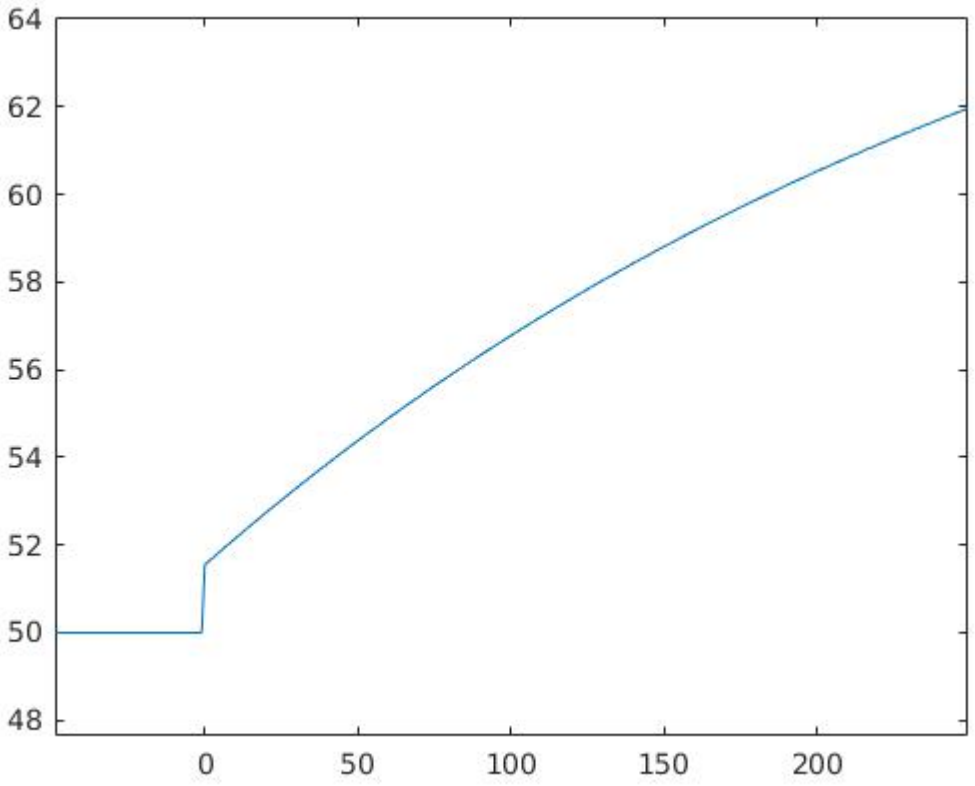


Figure 11: Non linear P controller

Explanation of the Improved Controller:

1. This is a (non linear P)D controller. Here the proportional term is not a straight line but a curve of form.
$$y = -70(e^{-(u+400)/300} + 1)$$

2. The delay block is used to implement he derivative term where the input value is given after the delay of one sampling rate. That value is used to subtract with the current value and multiply the difference with the Kd value.

3. Below the error of zero i.e. negative angle, if we keep the input as zero the arm drop very fast and becomes unstable. Hence an input of 50 is sufficient to fall down with a slow pace.

4. Reduction in noise was also a difficult task. I used a moving average and a low pass filter to achieve the task.These blocks are needed to be added as a filter before the servo input. This will reduce the jitters in the input signal and also smooth it.

5. To limit the current input going to high into the motor a saturation block is used after calculating the input value for the servo.

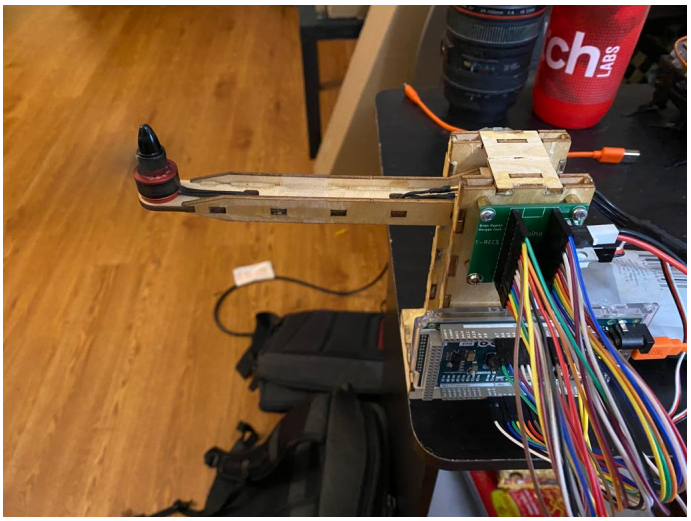
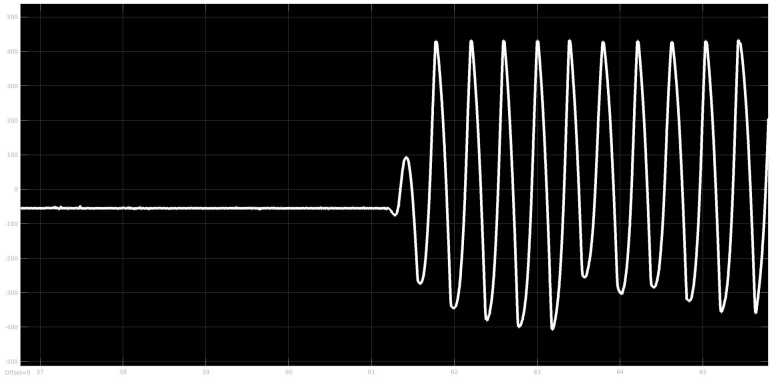
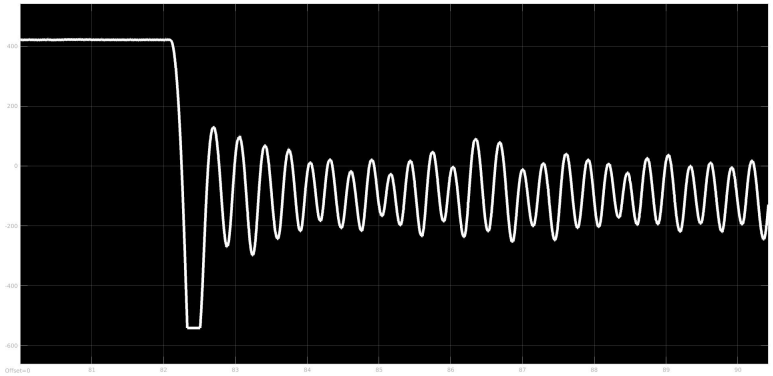


Figure 12: System with the improved controller

5 Observations

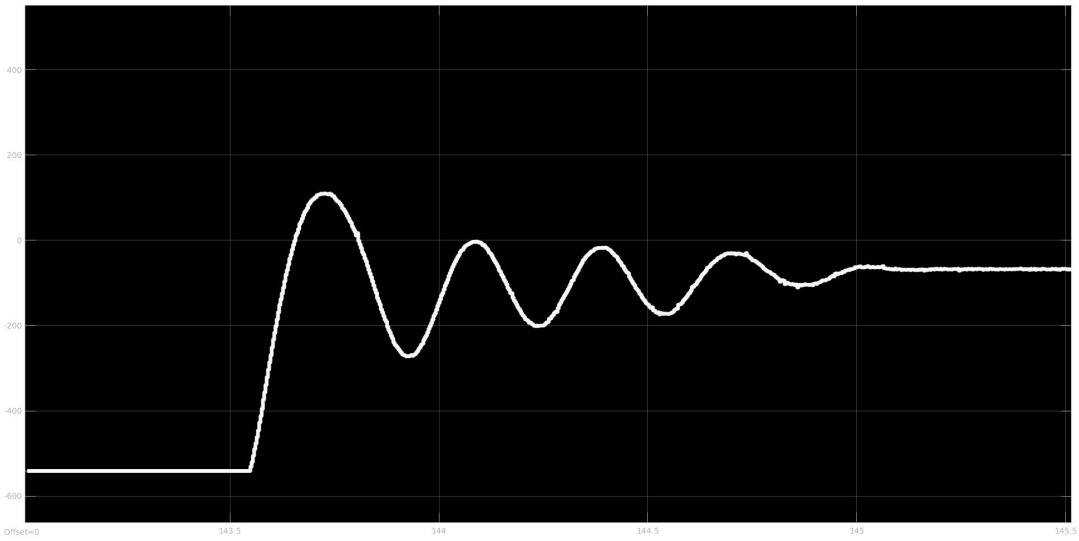


(a) Proportional

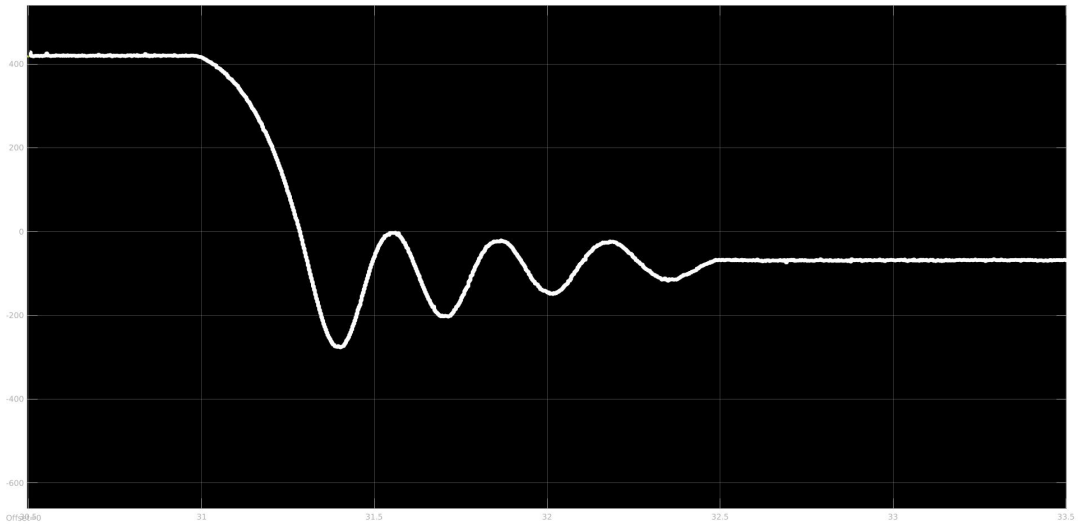


(b) Proportional Derivative

Figure 13: Step Response of Traditional Controller



(a) Positive



(b) Negative

Figure 14: Step Response of Modified Controller

As observed in the scope the PD controller in very unstable and it does not approach the steady state after an external disturbance or an impulse is given to the system. The improved PD controller makes the system stable about a given input with the following attributes:

1. Rise Time = 0.1 sec
2. Settling Time = 1 sec
3. Percent Overshoot = 8.33%

To obtain the above parameters the negative impulse was selected as it was using the controller output and not the constant value.

6 Results

In this report a controller was designed to stabilise the T-RECS system at 0° angle i.e. horizontal. Any non linear systems includidng this one requires a non linear controller to stabilise it. While working with a real physical system there are certain attributes which do not reflect from the theory. We have to experiment with those attributes to entirely identify the system. These points were instrumental in designing a controller and modifying the regular PID into something different.

References

[1] *Arduino Due — Arduino Official Store*. https://store.arduino.cc/usa/due. (Accessed on 04/18/2021).

[2] *ARRIS Swift Series 20A 2-4S BEC 5V/1A Brushless ESC for RC Multi-rotor*. https://www.rc-wing.com/arris-swift-20a-2-3s-blheli-brushless-esc.html. (Accessed on 04/18/2021).

[3] *Downloads — Tangibles That Teach*. https://www.tangiblesthatteach.com/downloads. (Accessed on 02/20/2021).

[4] Yangming Xu, John M Hollerbach, and Donghai Ma. "A nonlinear PD controller for force and contact transient control". In: *IEEE Control Systems Magazine* 15.1 (1995), pp. 15–21.