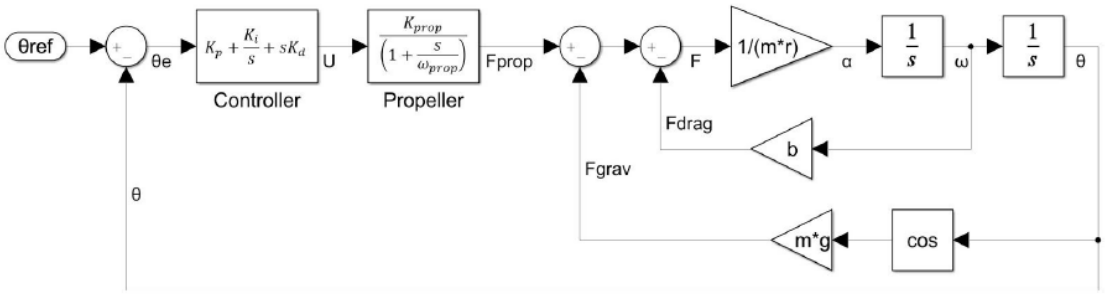
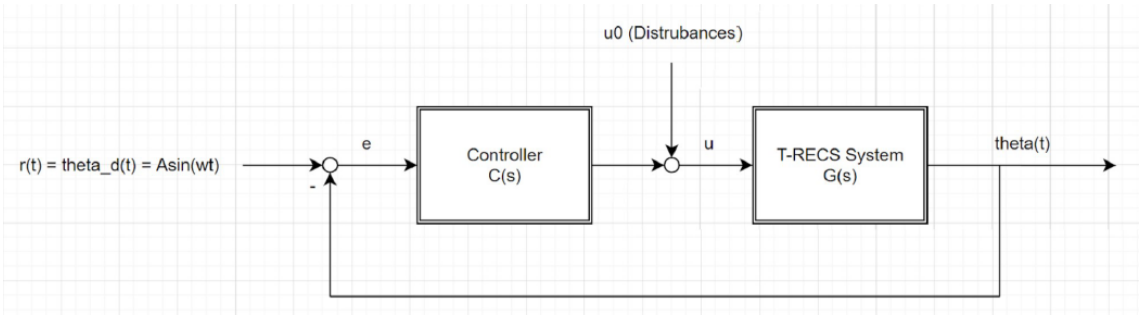
Industrial Automation - Project 1

Model Deployment and Control of T-RECS

Arpit Savarkar





1 Abstract

The objective of this project is to design a linear feedback controller around a linearized model of the T-RECS. The process of of model creation and control simulation starts from control loop modelling in Simulink (Matlab) and porting to Arduino Due. The overall goal is to design controller and simulation and porting from Arduino Nano to Arduino Due.

2 Simulink Setup For Arduino

Simulink blocks for configuring and accessing Arduino inputs and outputs work by writing and reading blocks to communicate directly with a ThingSpeak™ channel of control algorithms running on to run on Arduino board. Simulink Coder™ lets us access the C code generated from Simulink and trace it back to the original model and external interrupt blocks lets us trigger downstream function-call subsystems. The simulink model was tested for successful wiring by blinking LED on Digital Pin 13 on Due.

Figure 1: Simulink Arduino Communication of PWM with Digital Out

2.1 Arduino Due

The Arduino Due is the Arduino board based on a 32-bit ARM core microcontroller. With 54 digital input/output pins , 12 analog inputs, 2 DAC pins [1]. It is the primary board which is used in this project to communicate with the T-RECS. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 800 mA. This regulator also provides the power supply to the SAM3X microcontroller.

The role of the programming port is an important parameter when using Arduino Due with Simulink. The Due's programming port (the one closest to the DC power jack) is used. Opening and closing the Programming port connected at 1200bps triggers a "hard erase" procedure of the SAM3X chip. This is the recommended port for programming the Due. It is more reliable than the "soft erase" that occurs on the Native port, and it should work even if the main MCU has crashed.

2.2 Simulink Interfacing with Arduino Due

2.2.1 Software Install steps should be followed as below.

1. The libraries needed to be installed are "Simulink Support Package for Arduino Hardware" and "MATLAB Support Package for Arduino Hardware".
2. Connect the Due board to host, over USB Cable with figure as in 4.
3. Install the necessary drivers for Arduino Due, and select the supported ports.
4. Simulink model settings should be set as follows :

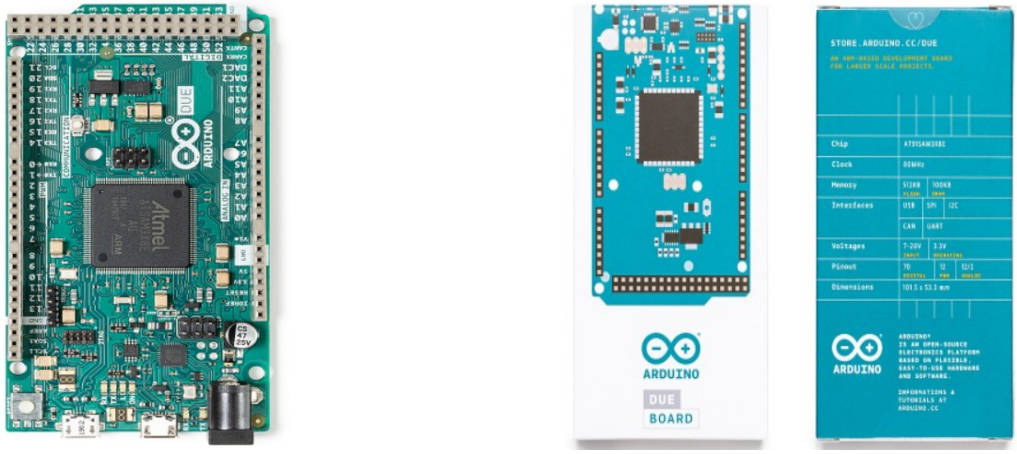


Figure 2: Arduino Due Board

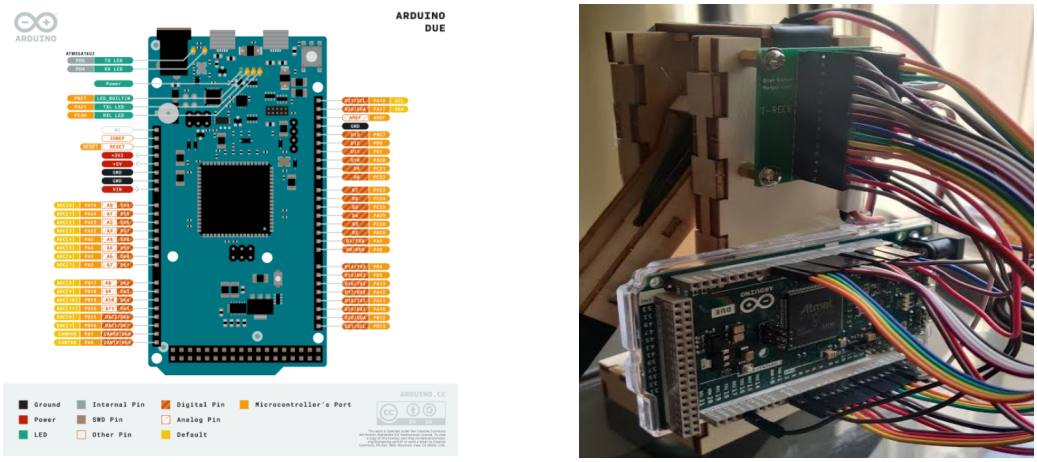


Figure 3: Wiring Arduino Due board to T-RECS

- Fixed setup size of 0.001 max solver size.
- Single simulation output should be unblocked.
- Hardware Implementation should be Arduino Due, and Device Vendor: Arm Compatible and Device type should be ARM Cortex.

2.2.2 Test Software Installation and Hardware

To test for successful wiring and Driver updates on Windows/Linux, the potentiometer connected to the PCB Board as per figure 7. The potentiometer is an Analog to Digital (A2D Pin) value which on the Due is 12bits. Running from 0 to 4096 level which corresponds to 0 to 3.3 V, indicating more layers available than in Arduino Nano . Upon movement of the arm it can be observed that the display in figure 7 varies from 3940 which corresponds to -39 degrees from horizontal and 0 V input to 2400 which corresponds to 15 degrees above horizontal. Which forms one of the block in the simulink feedback loop forming the interpolation before feeding into the TRECS. See fcn block in figure 14

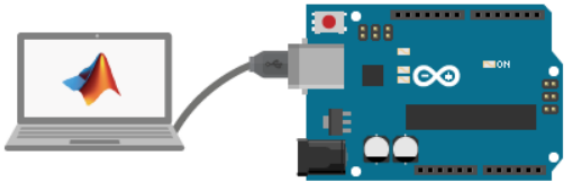
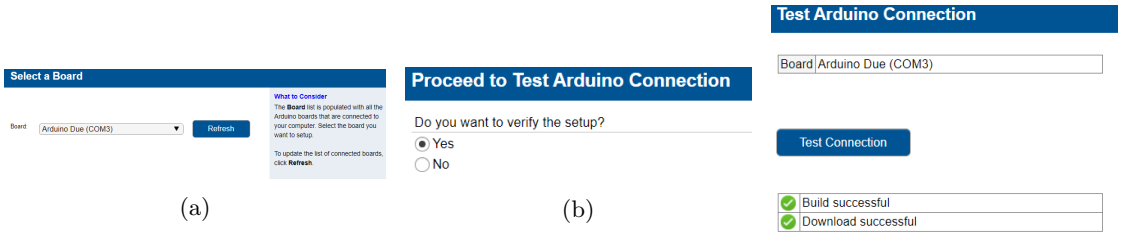


Figure 4: USB Connection



(a) (b) (c)

Figure 5: Simulink Setup to Arduino

2.2.3 Calibration Routine

For successful calibration routine sequential timed period of power up and down sequence as in figure 6. as in discussion with Dr Shalom, calibration range throttle PWM is from 1000us to 2000us correspond to 0° and 180°, and the difference between minimum and maximum throttle.For Simulink calculations are as follows. These inputs are given as servo angle input to Servo Write block in simulink. The entire calibration routine takes around 16-18 seconds, which includes some buffer considering hardware and Simulink interfacing.

$$\begin{aligned} min_{angle} &= \frac{180 * (1000 - 544)}{2400 - 544} \\ \implies min\ angle &= 44.22^\circ \end{aligned} \tag{1}$$

$$\begin{aligned} max_{angle} &= \frac{180 * (1000 - 544)}{2400 - 544} \\ \implies max\ angle &= 141.21^\circ \end{aligned} \tag{2}$$

Calibration of T-RECS on Simulink

For calibration on the simulink, a comparison of the arduino code with the simulink as seen in figure 9.

2.3 Simple Approach to System Identification

Equations of Motion

$$\begin{aligned} F_g &= mg\cos\theta \\ F_d &= b\dot{\theta} \\ F &= F_p - F_g - F_d \end{aligned} \tag{3}$$

As we know,

$$T = Fr = L\dot{\dot{\theta}} \tag{4}$$

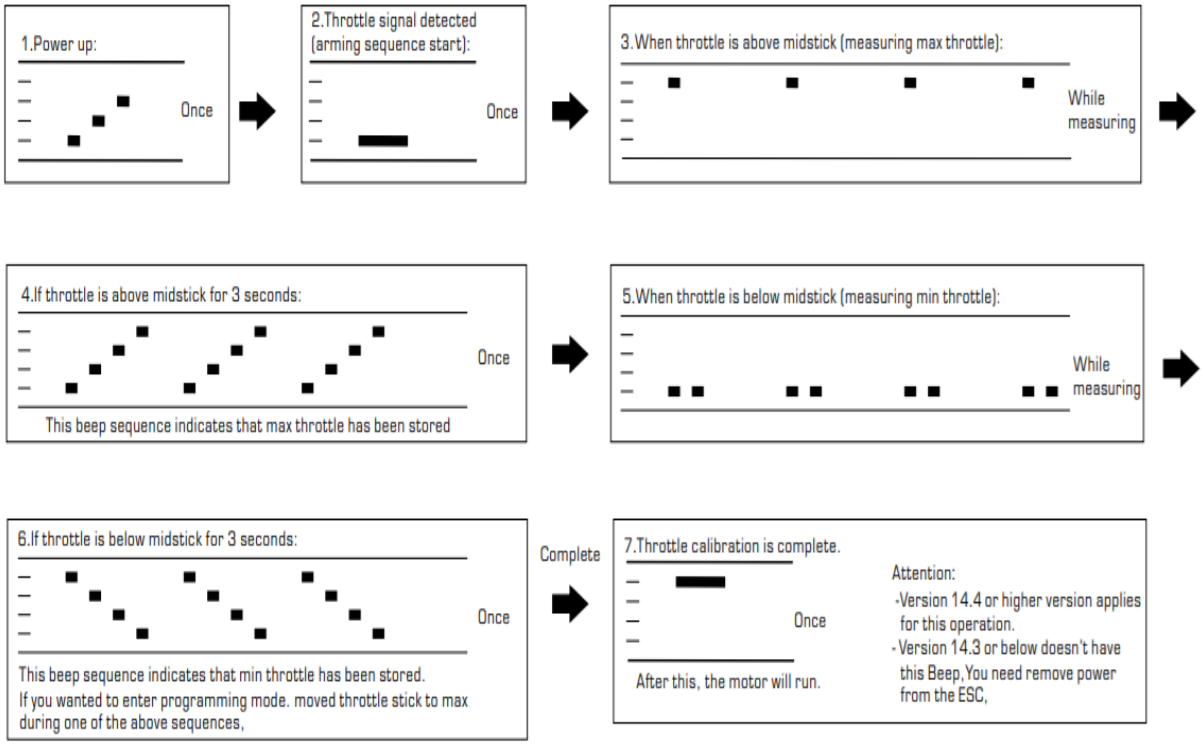


Figure 6: Calibration Routine

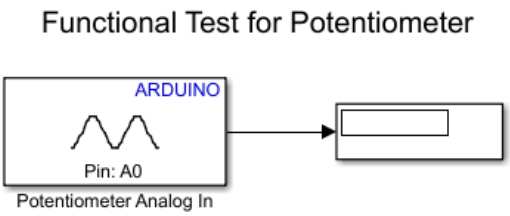


Figure 7

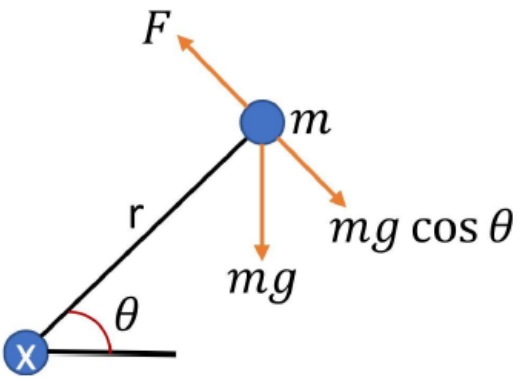


Figure 8: Lumped Model of T-RECS

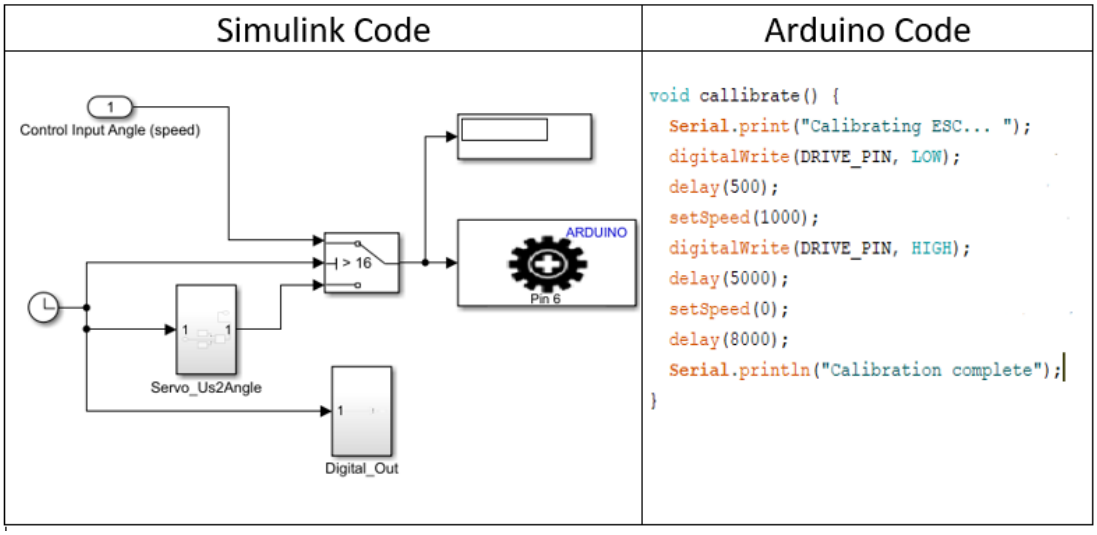


Figure 9: Calibration Comparison Arduino Vs Simulink

Modifying (3)

$$\begin{aligned} L\ddot{\theta} &= (F_p - b\dot{\theta} - mg\cos\theta) \\ \frac{L}{r}\ddot{\theta} + b\dot{\theta} + mg\cos\theta &= F_p \end{aligned} \tag{5}$$

According to First order expansion of Taylor series

$$\cos(\bar{\theta} + \hat{\theta}) = \cos(\bar{\theta}) - \sin(\bar{\theta}).\hat{\theta} \tag{6}$$

Plugging in $cos\theta$ in (5)

$$\frac{L}{r}\ddot{\theta} + b\dot{\theta} + mg(\cos(\bar{\theta}) - \sin(\bar{\theta}).\hat{\theta}) = F_p \tag{7}$$

As L = mr^2

$$\dddot{\hat{\theta}} + \ddot{\hat{\theta}}\frac{b}{mr} - \frac{g}{r}\sin(\bar{\theta}).\hat{\theta}) + \frac{g}{r}\cos\bar{\theta} = \frac{F_p}{mr} \tag{8}$$

Laplace transformation of equation (8)

$$s^2\hat{\theta} + s\frac{b}{mr}\hat{\theta} - \frac{g}{r}\sin\bar{\theta}.\hat{\theta} = \frac{\hat{F_p}}{mr} \tag{9}$$

$$\hat{\theta}(s^2 + s\frac{b}{mr} - \frac{g}{r}sin\bar{\theta}) = \frac{\hat{F_p}}{mr} \tag{10}$$

$$\implies \frac{\hat{\theta}}{F_p} = \frac{\frac{1}{mr}}{\hat{\theta}(s^2 + s\frac{b}{mr} - \frac{g}{r}sin\bar{\theta})} \tag{11}$$

if $\bar{\theta} = 0$, gives us,

$$\begin{aligned} \implies \frac{\hat{\theta}}{F_p} &= \frac{\frac{-1}{mg\sin\bar{\theta}}}{1 - s\frac{b}{mg\sin\bar{\theta}} - s^2\frac{r}{g\sin\theta}} \\ Also, \implies \frac{\hat{\theta}}{F_p} &= \frac{K}{1 + 2\zeta s + \frac{s^2}{\omega^2{}_0}} \end{aligned} \tag{12}$$

$$
\begin{aligned}
K &= -\frac{1}{mg\sin\bar{\theta}} \\
\omega_0 &= \sqrt{\frac{-g\sin\bar{\theta}}{r}} \\
\zeta &= \frac{-b}{2mg\sin\hat{\theta}}
\end{aligned}
\tag{13}
$$

If, $\hat{\theta} = 0$, shows us that we have a pole at the origin and the other at higher frequency:

$$
\begin{aligned}
G(s) = \frac{\hat{\theta}}{U} = \frac{\frac{b}{m^2r^2}}{s(1+s\frac{mr}{b})} &= \frac{K}{s(1+\frac{s}{\omega_{pole1}})} \\
where, K &= \frac{b}{m^2r^2} \\
\omega_{pole1} &= \frac{b}{mr}
\end{aligned}
\tag{14}
$$

Considering the simplified model for experimentation analysis
And, since ω is linearly proportional to the drive signal, u

$$
\begin{aligned}
L\ddot{\theta} &= Tr - rb\dot{\theta} - mg\cos\theta \\
\ddot{\theta} &= (Fr - Lb\dot{\theta} - mgr\cos\theta)\frac{1}{L}
\end{aligned}
\tag{15}
$$

As F = Aω^2, where
F = thrust produced,
A = Proportional Coefficient,
ω = angular velocity

$$
\implies \ddot{\theta} + C_1\dot{\theta} + C_2\cos\theta = C_3u^2 \tag{16}
$$

From the analysis undertaken in Project 0 and re-implementing the small error due to inaccurate motor shaft into the potentiometer,

$$
\begin{aligned}
\ddot{\theta} + C_1\dot{\theta} + C_2 &= C_3u^2 \\
\implies \ddot{\theta} + 6\dot{\theta} + 220 &= 0.0081312u^2
\end{aligned}
\tag{17}
$$

$$
\ddot{\theta} + b\dot{\theta} + \frac{g}{l}\sin\theta = C_3u^2 \implies \ddot{\theta} + 6\dot{\theta} + 220\sin\theta = 0.0081312u^2 \tag{18}
$$

To model the TRECS system, the sensor calibration routine was undertaken on simulink. From there, both frequency response was undetaken under varying sampling times. I ran the T-RECS at several frequencies ranging from a period of T = 0.24 s to T = 2 s in Arduino IDE, but the found the testing for sampling time of 0.01 sec. In each iteration, the collected data on the behavior of the system's response including the pitch θ (measured position), u (controller offset) and (desired position) in relation to time. To achieve the best fit to experimental data as discussed during Project 0, a linearity assumption has to be made. Motor and propeller have a significant impact on the dynamics of the model and thus can be expected to have the strongest role on the model of the system.

Thus from analysis and a pure assumption of the effect of motor and propeller high frequencies. I propose the model of the T-RECS to be approximately.

$$
G(s) = \frac{\theta}{U} = \frac{\frac{K_{prop}}{m.r}}{(s^2 + s.\frac{b}{m.r} - \frac{g}{r}sin(\theta)) * (1 + \frac{s}{Pole_3}) * (1 + \frac{s}{Pole_4})} \tag{19}
$$

That is, with the values calculated from Project 0, Proposed Model of the T-RECS should be :

$$G(s) = \frac{SystemGain}{(1 + 1.47 * s + \frac{s^2}{20.5}) * (1 + \frac{s}{Pole3}) * (1 + \frac{s}{Pole4})} \tag{20}$$

with the parameters of Overall Gain of 29.1 and a complex pair of poles at 4.5 rad and damping coefficient of 0.74.

2.4 Note on Potentiometer

The potentiometer forms the important system for feedback loop closure. The potentiometer matstays linear between $-160°$ to 160∘. The rotor shaft of potentiometer should be rotated in a clockwise direction. The Potentiometer is read on the Simulink using analog read block. The value is mapped to angle using a custom function generator in simulink which is developed as follows

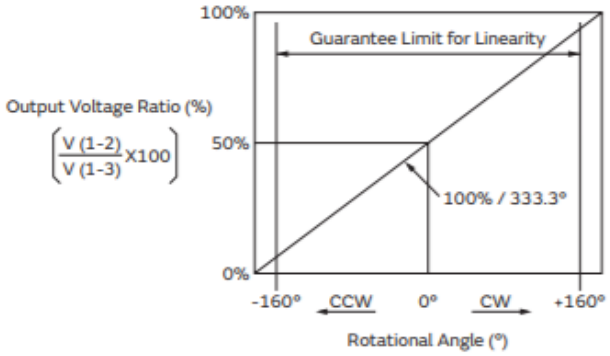


Figure 10: Output Voltage Vs Rotational Angle [2]

SIMULINK FUNCTION
function y = fcn(u)
y = interp1([3940, 2300], [-39, 15], double(u));

(21)

3 Controller Design

Initial goal for the controller was to compare and design the physical transfer function model with the simulated mode. However, given the amount of lag in the plant, small bandwidth and fast dynamics, the goal would be out of reach. Instead, the focus is solely on to adjust the gain of the controller with the intention to get a high crossover frequency as possible. The first attempt at a controller was in the form of a proportional controller. The process for determining stable condition is consisted of some hands-on trial and error and a logical assumption that it should be before the roll-off in the plant in order to increase phase margin at the cross-over frequency. The real system is not linear, and it was therefore not possible to achieve an exact match, but reasonable results were achieved.

3.1 Proportional Control

When the system is placed under feedback control. The initial analysis under a proportional gain, showcases the instability of the system near the linearization point of commanded angle of -20

degrees. When an external force is put on the TRECS the system immediately leads to unstable condition.

Under feedback the system tries to stabilize itself, but consistently keeps oscillating to reach a steady state of error, which I noticed to be around 3-5 degrees on multiple runs. The rise time under proportional is inconsistent over multiple trials, to make any conclusive analysis about settling time or rising time.

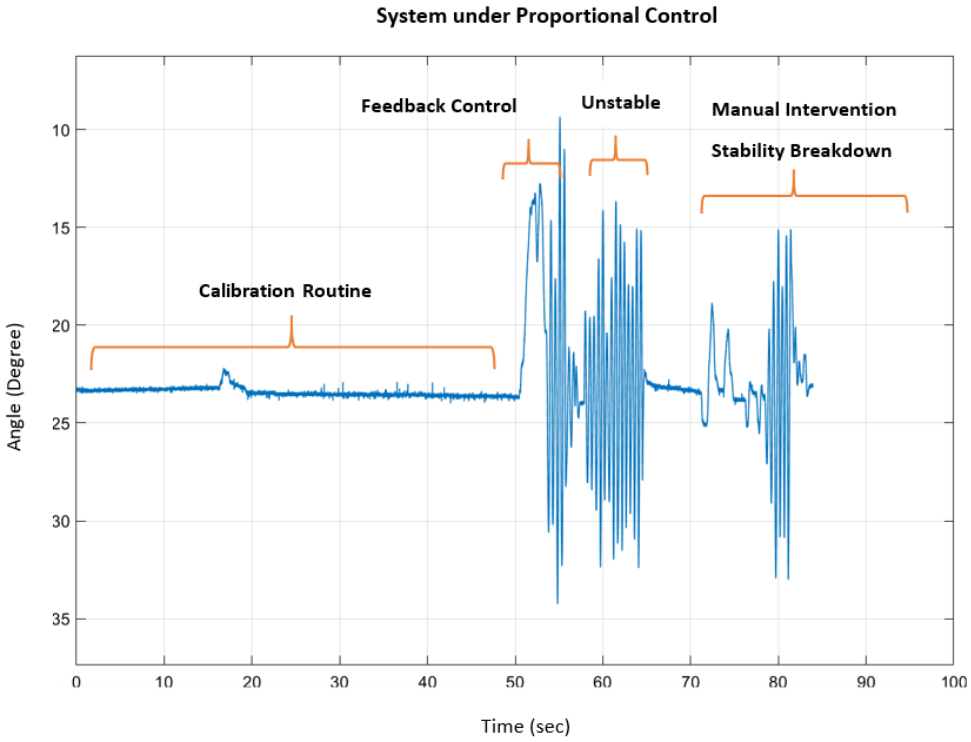


Figure 11: Scope - Proportional Control

3.2 PD Control

When the system is placed under Proportional derivative feedback control. Under the consideration that PD control is tuned for transfer function based to control for interpolated data from the potentiometer. For optimum control, lower derivative gain showed lower steady state error. And thus was tuned accordingly with filter coefficient of 300.

T-RECS

$$C(s) = K_p + K_d.\frac{N}{1 + \frac{N}{s}} \tag{22}$$

For PD Control, the following parameters gave the optimum result, but took around 7-8 seconds to settle.

$$C(s) = 1.2 + 0.8.\frac{100}{1 + \frac{100}{s}} \tag{23}$$

3.3 PID Control

For PID Control, the tuning took considerably longer, as more parameters involved inter-dependencies. The next logical step was to include an integrator in the plant. This would improve DC signal track-

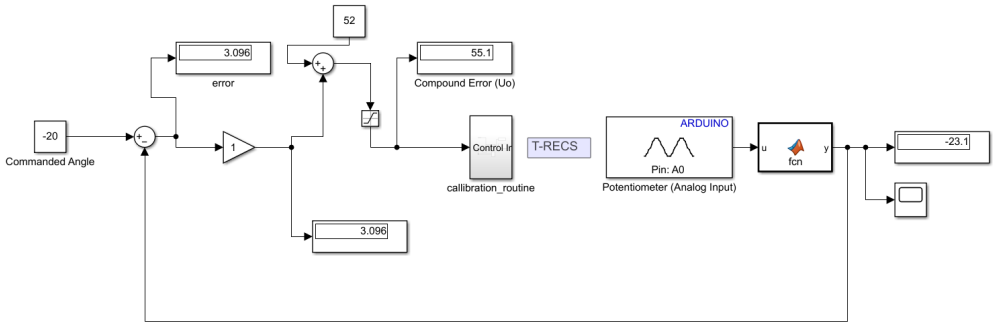


Figure 12: TRECS system for Proportional Control

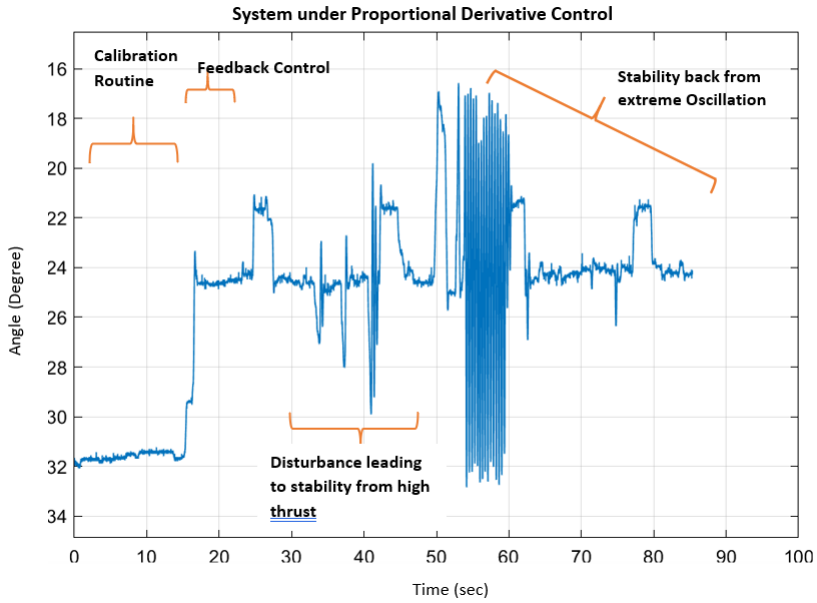


Figure 13: Scope - Proportional Derivative Control

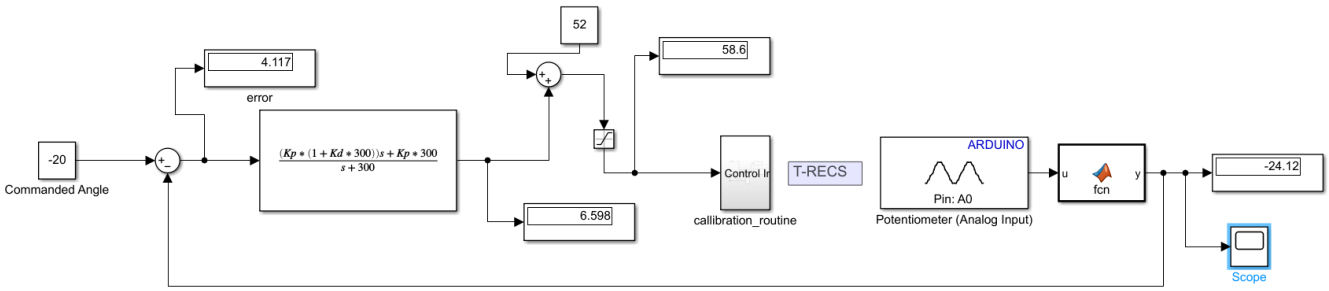


Figure 14: TRECS system for Proportional Derivative Control

ing, but at the cost of losing phase margin. In order to mitigate the loss in phase margin, the second zero that results from the PID controller would have to be placed close to the origin. This would result in very slow integral action. Before introducing the integral term to the controller, the team decided to attempt to minimize the effects of the nonlinear dynamics of the system.

$$C(s) = K_p + \frac{K_i}{s} + K_d.\frac{N}{1 + \frac{N}{s}} \tag{24}$$

$$C(s) = 1.5 + \frac{1.2}{s} + 0.2.\frac{300}{1 + \frac{300}{s}} \tag{25}$$

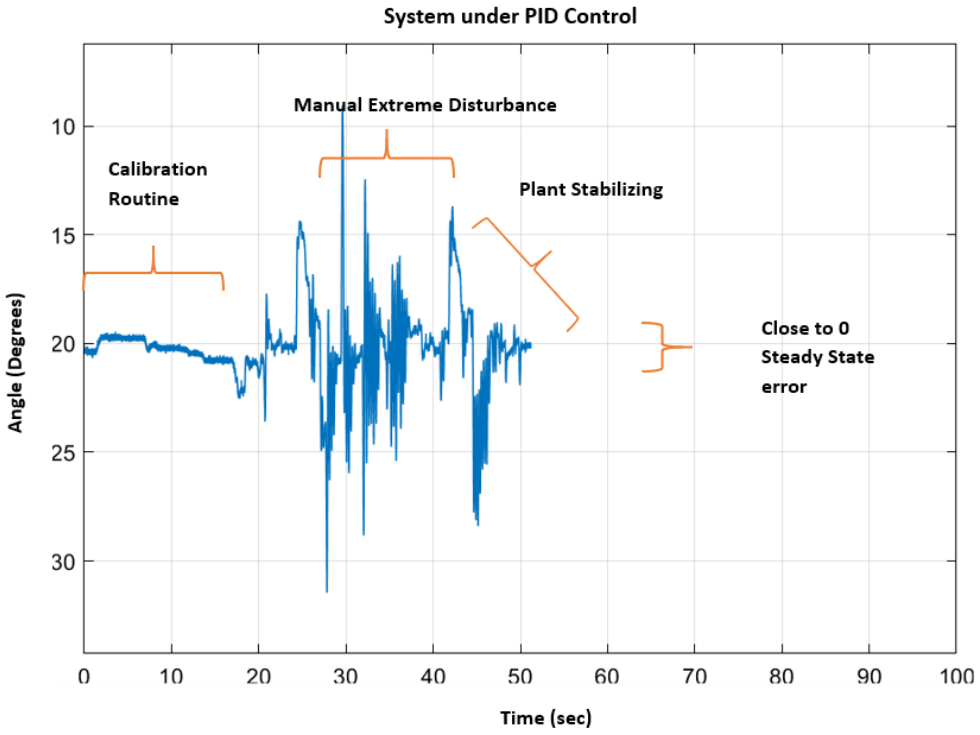


Figure 15: Scope - PID Control

The figure 14 shows the results of what appeared to be the most suitable controller for one of the TRECS systems. This controller transfer function is equation (25). Being a digital system, I believe the controller also contained an unavoidable pole3 and pole4 (due to sampling effects), but this could be ignored for the most part. This controller provided quick and smooth transitions to step inputs with acceptable overshoot. Unfortunately, reducing the gain of this controller also resulted in poorer DC signal tracking.

The effectiveness of this method was tested by using only the gravitational correction term as the drive signal. If approximated well enough, the arm would hold steady at any angle to which it is placed. This was indeed the case -the arm stayed at the desired position fairly closely between -39 and +10 degrees. But within ± 8 degrees. The Simulink sends the motor controller (ESC) a “drive” command in the form of a duty cycle. This drive sets the rotational velocity of the propeller. The force generated by the propeller as a function of rotational velocity is unknown, and expected to be non-linear (air drag tends to follow a square law).

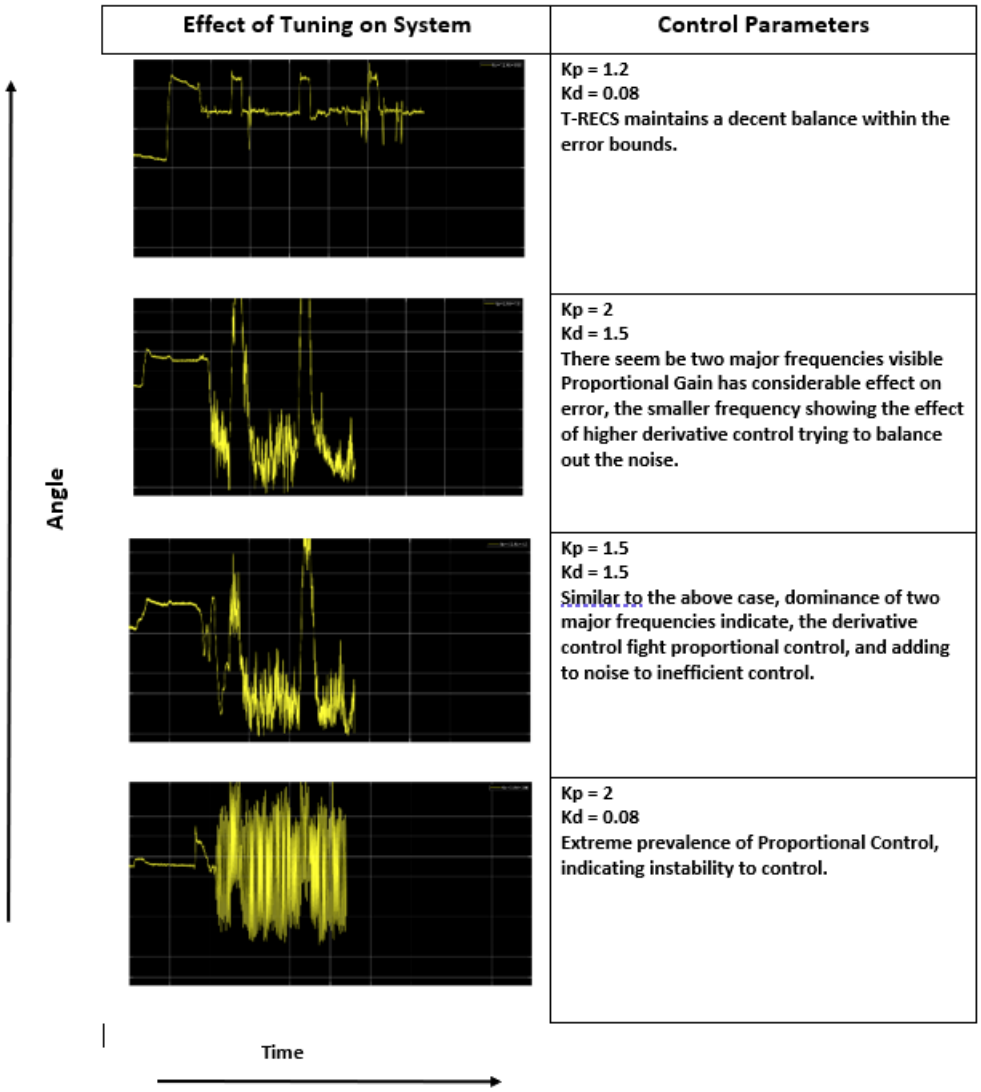


Figure 16: Tuning TRECS for PD Control

4 Conclusion

This experiment set out to observe the response behavior of the controller to provide insight into how adjustments could be used to further stabilize the system. By looking at the system response, we were able to create transfer functions based on the measurements and control the T-RECS system under feedback control to stabilize at a commanded angle.

References

[1] @miscArduinoD68:online, author = , title = Arduino Due — Arduino Official Store, howpublished = https://store.arduino.cc/usa/due, month = , year = , note = (Accessed on 04/15/2021)

[2] https://www.digikey.com/en/products/detail/murata-electronics/SV03A103AEA01R00/4359898