First Semester Progress Report

Programmable Flight Controller

Brendon Camm	Lucas Doucette	Dylan Humber
B00649176	B00685962	B00695554

Submitted: December 6, 2016

This page left intentionally blank.

Abstract

The following report summarizes the progress made throughout the first term of the Quad Copter Programmable Flight Controller project and the plan moving forward toward the final design. The goal of the project is to design a programmable quad copter flight controller that will have the capability of responding to live control inputs and react appropriately to disturbances present. The objective of this report is to summarize the deliverables set forth for first term and for the final design of the controller, the background of the proposed project, simulations performed and testing results from the first term. The report also outlines the work plan and distribution of tasks followed for the duration of the first term and discusses the suggested scheduling of tasks for the second term of the project.

Authorship

- Abstract - Lucas Doucette
- Objectives and Deliverables - Brendon Camm
- Background and Significance - Brendon Camm
- Proposed Approach and Validation - Brendon Camm
- Preliminary Results - Lucas Doucette and Dylan Humber
- Work Plan and Milestones - Dylan Humber
- Distribution of Tasks - Dylan Humber
- Budget - Dylan Humber
- Appendices - Lucas Doucette and Dylan Humber

Contents

1 List of Acronyms

1. BLDC - Brushless Direct Current
2. FOV - Field of View
3. GUI - Graphical User Interface
4. PD - Proportional Derivative
5. PI - Proportional Integral
6. PID - Proportional Integral Derivative
7. POC - Proof of Concept
8. PWM - Pulse Width Modulation
9. RF - Radio Frequency
10. USB - Universal Serial Bus

2 Objectives and Deliverables

2.1 Overall Objective

The main objective of the project is to develop a programmable flight controller that responds appropriately to control inputs and disturbances. The flight controller will recieve control inputs over Wi-Fi from a base station. The base station can be any device that is Wi-Fi enabled and has the appropriate software installed. The software on the base station will be a graphical

user interface (GUI) that allows the user to send control inputs to the drone and view statistics of the drone during operation.

The objectives of the project have been broken down into what will be accomplished in the first semester (Short term objectives) and what will be accomplished in the second semester (Long term objectives). These objectives along with specific details of each can be viewed in sections 2.2 and 2.3 respectively.

2.2 Short Term Objectives (October 2016 - December 2016)

2.2.1 Simulation

The simulations will allow us to gain an understanding of how the controller will respond to specific inputs. The simulation can then be tuned until the output is within the constraints set by Dr. Rhinelander. We will be simulating both the flight dynamics and controller using MATLAB and Simulink exclusively.

2.2.2 Construction of the Drone

The drone parts will arrive separately and assembly will be required. The extent of the assembly will be to attach the 4 Brushless Direct Current (BLDC) motors and batteries to the base of the drone. On top of the assembly the preliminary layout of the required hardware will be decided on. The layout is subject to change as we begin the final assembly in the second semester.

2.2.3 Initial Design of the Controller

A preliminary design of the flight controller will be constructed in software.

2.2.4 Initial Testing

The initial design of the controller will be tested using the brushless direct current (BLDC) motors supplied by Dr. Rhinelander. The initial tests will allow us to gain insight on what changes to the flight controller must be main in order to meet the constraints. Along with the testing of the controller tests will be conducted to characterize the BLDC motor and electronic speed controller (ESC)

2.3 Long Term Objectives (Dec 2016 - April 2017)

2.3.1 Graphical User Interface Design

The graphical user interface (GUI) will be installed on any base station intended to be able to operate the drone. The key features of the GUI include: A means to access the controller, displays drones position (Coordinates and altitude) and the ability the load a new build onto the drone. Some minor features will include: displays the current software build on the drone, current flight time and total flight time.

2.3.2 Base Station Configuration

This will entail installing the GUI onto the base station and configuring the base station network adapter to be able to communicate with the raspberry-pi on the drone.

2.3.3 Network Tests

The intent of the network test is to gain an understanding of the network strength at various distances. A preliminary idea of how these tests will be conducted is to ping the raspberry-pi from the base station to see the time of response at these distances.

2.3.4 Final Controller Tests

The final controller tests will be identical to the tests run during the first semester of the project. The purpose of these tests will be to verify that the appropriate changes were made in order to meet the defined constraints so that flight tests of the drone may be completed.

2.3.5 Flight Tests

The flight tests will be run to validate that each constraint has been met to the best of our ability. The tests include, but are not limited to: looking at the drones response to disturbances, response to control inputs, response to loss of communication and verify that the GUI is reporting the expected data.

2.4 Deliverables

The final deliverables will be as follows:

- Raspberry Pi based communication interface software
- Arduino based flight controller program

- Base station software that relays control signal to quadcopter as well as displaying status information in a GUI

- Detailed documentation for architecture of overall and of the programming of every subsystem.

2.4.1 Raspberry Pi based communication interface software

The flight controller will consist of both a Arduino microcontroller and Raspberry Pi 3 microprocessor that is Linux based. In order to establish communication between the two a script must be run. The script to achieve to communication will be supplied to Dr. Rhinelander.

2.4.2 Arduino based flight controller program

The flight controller software will be well commented to ensure that it is made clear how the software works.

2.4.3 Detailed documentation for software

The documentation will contain specific details regarding the software architecture as well as any specific details about how to run or load the software onto a platform. The documentation will also include instructions on how to use the provided GUI as well as any troubleshooting techniques if any are required.

2.4.4 GUI

The GUI will be loaded on to a Universal Serial Bus (USB) (USB will be supplied by Brendon Camm) so that it may be installed on any device that Dr. Rhinelander desires to act as a base station. The GUI will be able to be installed on Linux, Windows and MAC devices.

3 Background and Significance

Machine learning is defined as the science of getting computers to act without being explicity asked. This is achieved by the development of computer programs that can teach themselves to grow and change when exposed to different sets of data. There are two traditional types of machine learning algorithms: Batch learning algorithms and on-line machine learning algorithms. Batch learning algorithms require a set of predefined training data that is shaped over the period of time to train the model that the algorithm is running on. On-line learning uses an initial guess model that forms covariates from that initial guess then passes them through the algorithm to form an evolved model a new set of covariates are formed from the evolved model and then fed back to make a new prediction. The loop runs continuously so that the evolved model is constantly growing and learning to adapt to certain situations. Dr. Rhinelander's research is concerned with on-line machine learning algorithms therefore the drone we are developing will configured to adapt with these algorithms.

Quadrotor drones have been on the rise in popularity in the last several years due to their simplistic mechanical design and many practical uses.

The application of these drones vary from a hobbyist flying around their neighbourhood to military personnel carrying out high risk missions. A video recording device of some sort is generally attached to the drone and the video feed is relayed to a basesation for the operator to gain a field of view (FOV) of an area of interest. Having the capability to have a continuous video feed allows the drone to be used for many practical applications including but not limited to: Traffic condition monitoring and surveillance missions. While these drones are very sophisticated and advanced devices they are are missing one aspect that is very important to further Dr. Rhinelanders research: They are not totally configurable.

As mentioned, Dr.Rhinelanders research is concerned with on-line machine learning algorithms and without a platform that is completely configurable his research would be limited. Before the machine learning algorithms are implemented onto the drone it must first be able to be controlled. This is where we come in, we have been tasked by Dr. Rhinelander to develop a flight controller that recieves control inputs over Wi-Fi. Having a completely open source flight controller will allow for the addition of the machine learning algorithms to the flight controller software so that the drone can learn to partially, and eventually fully fly on it's own and make intelligent decisions.

4 Proposed Approach and Validation

We have decided to approach the design of the controller by breaking it down into two phases, each phase will consist of four months. The first phase will deal purely with simulation and proof of concept testing of the hardware

to gain a better understanding of what we were working with. The second phase will be to apply what we've learned during the first phase to develop the flight controller software.

4.1 First Phase

Before we can begin developing the flight controller we must first decide on which type of contol will be optimal for our application. The types of control we will consider are Proportional Integral and Proportional Integral Derivative. Each type of control will be applied to a simple simulation that outputs an the reaction torque of a 1kg body to a step input, which ever control supplies the most steady value of non-zero average torque will be the control type we will move forward with. Upon deciding which control type we will be using more advanced simulations will be run.

The more advanced simulation will consist of a 3D representation of the drone that is connected to various subsystems. The response of the system to a step input will be studied and the system will be tuned to obtain a reasonable response. The response we're going to achieve will simulate that the drone is hovering in a location, this response should be resemblant to that of a sinusoidal function. Upon completion of the simulations Proof of Concept (POC) testing will be performed.

The POC testing will consist of the following items: determining methods to control the motor speed, characteristics and limitations of the hardware and software and which communication protocol is optimal.

The main method we will be implementing to test control of the motor speed is the apply different pulse lengths to the motor. This will be achieved

by using Pulse Width Modulation (PWM) applied at different duty cycles. A script will be written that applies a voltage across a potentiometer that can then be varried to apply different duty cycles. The duty cycle will be displayed on a serial terminal to easily determine the current draw vs duty cycle characteristic of the motor.

Upon validating that the motor speed can be controlled using PWM characterization of the Electronic Speed Controllers (ESC) and BLDC motors will be performed. The ESC tests will attempt to prove that the ESC's are capable of controlling the motor speed using a variable input supplied by a potentiometer using the PWM script. To test the lift capability of a single BLDC motor a weight will be attached to a support system with the motor and blade attached to it. The weight appartatus will be placed on a zeroized scale and then power will be applied to the motor, as the propeller speed increases the reduction in weight read from the scale will be considered the lift capability. The test will be performed at various pulse width's to form a current draw versus pulse width characterization of the motor with a specified weight attached to it.

To determine which communication protocol (Wi-Fi or Bluetooth) will be optimal for our application basic range tests will be performed. To test Wi-Fi range, a Wi-Fi communicating device tethered to a Wi-Fi output from a cell phone will given to a user, the user will then walk a certain distance until communication is lost. To test the Bluetooth range a similar test will be conducted, communication will be established between a Raspberry Pi 3 and Playstation 4 controller a user will then walk a certain distance with the controller until communication is lost. Which ever tests supplies the

longest distance before communication is lost will be deemed optimal for our application.

To validate that communcation can be established between the Raspberry Pi 3 and Arduino the I2C bus on the Raspberry Pi will be enabled and an open source script will be run. This simple tests will prove that communication between the microprocessor and microcontroller can be achieved.

4.2 Second Phase

The second phase of the project will consist of applying what we've learned throughout the first phase to design the flight controller and the GUI. The two main goals of the second phase will be that the flight controller is able to receive control inputs from the GUI and responses appropriately and that the GUI functions as outlined in the Objectives and Deliverables section.

The development of the flight controller and GUI will be done in parallel. The leads for each of these tasks can be found in the Distribution of Tasks section. The reason we will be approaching it this way is the ensure that no one is reliant on another person to get their work done but this will also require close contact between the team leads to ensure that the flight controller and GUI are compatible. Weekly meetings and daily contact will ensure the compatibility of the two.

To validate that the GUI is functioning as intended a few basic tests will be performed: The controller is able to receive control inputs from the GUI over Wi-Fi, appropriate Global Position Sensor (GPS) coordinates are displayed as the drone travels, new software builds are able to be loaded onto the hardware from the GUI and once a new software build is loaded the name

of the build is located on the GUI. These tests will validate the the GUI is functioning as intended.

The flight controller tests will consist of applying various types of control inputs and disturbances and observing how the controller responds. This will validate that the controller operates within the given constraints.

5 Preliminary Results

5.1 SimuLink System Models

Using SimuLink, a model of the quadcopter system and control system was built. This was used to perform preliminary hovering tests. The SimuLink model was broken down into several blocks. Each of these blocks was built to handle key aspects of the system. The system can be seen in it's current state in the following figure. Each of the blocks shown are discussed in detail below.

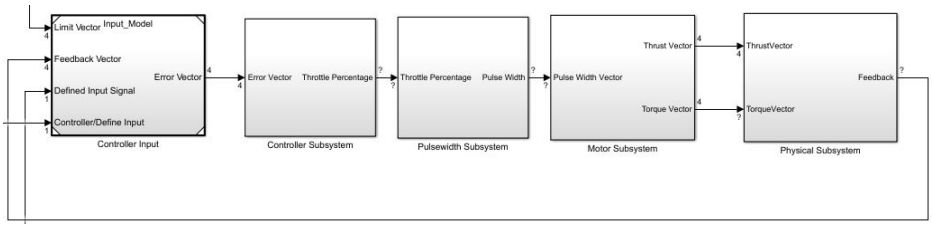


Figure 1: SimuLink Control System Architecture

5.1.1 Input Model

This block is the front end of the simulation. The four inputs to this block specify the origin of the control signal as well as the limits or sensitivity on the signal. As well as a port for sensor feedback. The final output is the error vector for the control system. The inputs are discussed in further detail below:

1. **Limit Vector**: This input takes a four dimensional vector of limits. These limits respectively apply to roll, pitch, yaw, and altitude. Roll, pitch, and yaw should be specified in degrees.

2. **Feedback Vector**: This input takes a four dimensional vector of the current values of roll, pitch, yaw, and altitude. This is then subtracted from the control signal to create the error signal.

3. **Defined Input Signal**: This input takes a four dimensional vector of control signal. Values for this should be normalized between -1 and 1.

4. **Controller/Define Input**: This input specifies whether to use a control signal from **Input 3** or from a controller (e.g. Joystick, DualShock gamepad) attached to the computer.

5.1.2 Controller Model

This subsystem contains the control system for the quadcopter. It currently consists of a PI controller for all four of our control signals. The gain values for these controllers have not yet been finely tuned. The outputs of the

controllers are limited to be between 0 and 100. This is to represent a percentage of our throttle.

5.1.3 Pulse Width Subsystem

This subsystem takes the percentage of throttle and translates it into a pulse width within the applicable range for our speed controller. Our lowest throttle is at 1127 μs while our highest throttle is at 1860 μs. Our output is given by the equation:

$$PulseWidth = Throttle\% \cdot (1860\mu s - 1127\mu s) + 1127\mu s$$

5.1.4 Motor Characterization

This subsystem determines the thrust and torque from each motor depending on their respective pulse width. Thrust is determined by using a lookup table; extrapolated from our experimental data. As we were unable to characterize the torque or the speed of the rotor, the output torque is an estimate based on the value of the thrust.

5.1.5 Physical Subsystem

This subsystem houses the physical simulation that has been created using SimuLink's Simscape Multibody package. It currently consists of a rudimentary visualization of a quadcopter. No more complex than a cross representing the body between the motors. In the simulation a force and torque is applied at the end of these arms to simulate the action created by the motors.

The inputs for this subsystem are a vector of the magnitude of the motor thrusts and a vector of magnitude of the motor torques. The simulation accounts for the direction of rotation of the motors by applying the torques along either the X or Y axis of the model, depending on the rotation. The subsystem also outputs the current roll, pitch, yaw, and altitude to be fed back.

5.2 Hover Test

Using the models and systems discussed in the previous subsection, our preliminary flight simulation was constructed. The SimuLink model Dynamic_Simulation.slx was created. The altitude controller was tested by providing a step input. After some tuning of the controller, we were able to obtain the following step response.

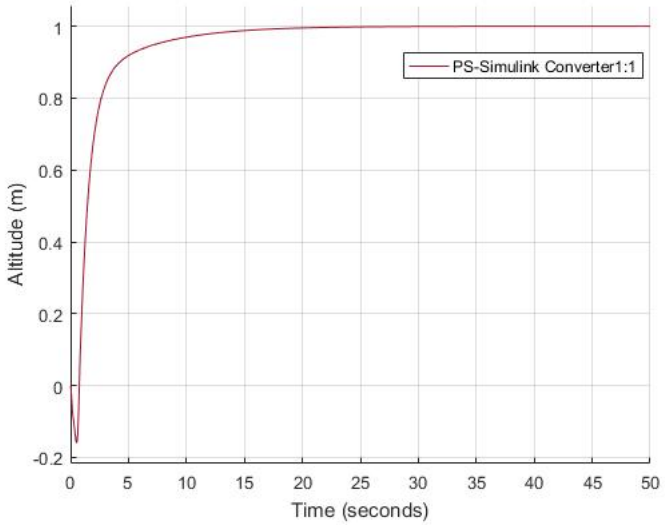


Figure 2: Hovering Step Response, for 1 metre altitude.

Currently the simulation does not account for a hard stop at 0 altitude. At the beginning of the response above, the quadcopter can be seen to fall briefly. Once the rotors have generated sufficient thrust, the quadcopter continues towards the requested 1 meter altitude.

5.3 Proof of Concept and Testing Performed

To date, several tests and proof of concepts have been performed to determine methods of controlling hardware, limitations of the hardware or software and to determine feasibility of communication protocols. The test subjects include:

- PWM
- Electronic Speed Controller
- Motor Lift Characteristics
- Wi-Fi Range
- I2C Communication Channel
- Playstation Controller Integration

5.3.1 PWM

Using an Arduino micro controller, a potentiometer and a simple DC motor, a circuit was devised to test a PWM output of the Arduino to drive a motor circuit. The motor circuit was isolated using a mosfet as the switching

operator and a diode to ensure there would be no damaging back emf in the system. A 9V battery powered the isolated circuit.

Through programming, the voltage across the potentiometer was taken into the Arduino as an analog input and mapped to a digital output as a PWM duty cycle. The script was successful in providing an input controlled PWM value to the motor. The code was modified to suit the needs of the electronic speed controllers by applying an input controlled pulse width as a function of time in replacement of a duty cycle as the ESC's require a pulse width range of 1060 μs and 1860 μs.

5.3.2 Electronic Speed Controller

The Afro ESC 12Amp BEC UltraLite Multirotor ESC V3 was tested at St. Mary's University with the assistance of Dr. Rhinelander. An Arduino running a script to map a potentiometer to a PWM duty cycle was used as an attempt to control the motor through the ESC. A 12V power supply fed the ESC while the Arduino controlled the duty cycle of the speed controller producing a voltage output to control the motor speed.

The ESC testing was successful, the testing proved the Arduino's capability to control the motor with a variable input. The testing had flaws as a PWM duty cycle was used instead of a timed pulse width input. The duty cycle had potential of operating correctly as the range of times could have been calibrated to a range in the duty cycle although this method proved to be difficult due to low values causing the ESC to enter calibration mode. The script used to operate the ESC was re-written as a timed pulse width to ensure complete compatibility and ease of future integration.The pulse

width script was tested using the ESC and was successful. The provided motor was successfully driven under no load conditions for the full range of pulse width values.The Arduino script and motor driver schematic can be seen in Appendix Listing A.1 and figure A.1 respectively.

5.3.3 Motor Lift Characteristics

Using a 12V power supply, the provided ESC and the Multi-Star Elite motor the characteristics of the motor's lifting capacity were tested. A weight was attached to a support system with the motor and blade seated on top. The apparatus was placed on a scale and the scale's reading was zeroed. As the rotor speed increased, the reduction in weight read by the scale was considered the lift capacity.

The test was performed beginning at a pulse width of 1127μs which was found through experimentation to be the cut in pulse width for motor operation. The test was performed at increments of 25μs. The resultant current draw values and lift values were documented. The lift values were used to create a simulink lookup table to characterize the motor's available force for simulation purposes.

During the load testing, it was noticed that the current draw from the individual motor was high. The power supply being used had a current limit of 3 A therefore the maximum current draw allowed during testing was 2.95 A to ensure no brown out due to lack of supply. The current limiting factor resulted in the test ending at a pulse width of 1525μs and a lift value of 137.5g.The resulting plot of the load testing is found in Appendix figure A.2

5.3.4 Wi-Fi and Bluetooth Range

A simple proof of concept regarding the range of Wi-Fi communications was performed. The test incorporated a a Wi-Fi communicating camera tethered to a Wi-Fi output from a cell phone. A user walked down Spring Garden holding the cell phone and found the approximate distance at which the phone and the camera lost communication. It was found that the range was approximately 100 ft with line of sight available with no Wi-Fi boosting technology. Bluetooth communications were also tested using the Raspberry Pi 3 connected to a Playstation 4 controller although the communication channel held a strong connection for only approximately 10 ft, this distance was considered insufficient for the scope of the project.

5.3.5 I2C Communication Channel

The flight controller will consist of both an Arduino micro controller and a Raspberry Pi 3 Linux based computer system. The flight controller will require a communication channel between each component to transmit data between each system. To enable these communications, an I2C bus is planned to be utilized.

The I2C bus was enabled on the Raspberry Pi and a sample open source script was run on the Arduino to begin I2C communications. The test resulted in the Raspberry Pi recognizing the Arduino on I2C bus.

5.3.6 Playstation Controller Integration

Using an open source python script, a Playstation 4 controller was integrated with the built in Bluetooth communication channel of the Raspberry Pi 3.

All inputs available from the Playstation controller were taken in as values to the Raspberry Pi, confirming the compatibility of systems. Based on the Playstation controller integration testing, the feasibility of the Bluetooth communication was put into question. Although successful, it was decided that based on range constraints, Bluetooth would not be used as the communication channel, a hardwired serial connection to a base station will be utilized.

6 Work Plan and Milestones

6.1 First Semester (Sep - Dec 2016)

The objectives for this semester, as outlined in our design memo (October 28th), are as follows:

- Flight Simulation
- Assembly of Quadcopter
- Initial Controller Design
- Initial Testing

The flight simulation and initial controller design have been completed and exist within our SimuLink simulation. Additionally, the schematics of the physical implementation have been finished. With the simulation complete and initial hover test has been carried out in simulation. This is discussed in further detail in Section 5.2.

The frame and motors of the quadcopter have been assembled by Dr Rhinelander. However, one of the motors is currently on loan to us in order to perform the characterization discussed in Section 5.3.3.

The Gantt chart submitted with the design memo may be found in Appendix WHO KNOWS WHICH ONE?!

6.2 Second Semester (Jan - Apr 2017)

In the second semester of the project, our milestones build off of the ground work finished this past semester. The principal milestones are as follows:

- Refinement of simulation.
- Refined controller tuning.
- Base Station GUI.
- Raspberry Pi programming.
- Arduino programming.
- Documentation.

As communication with the quadcopter and the basestation goes through the Raspberry Pi, the development of the two will coincide. This will allow for a flexible platform to be built. The base station will be designed to display data that the Raspberry Pi sends it, dynamically allocating screen space based on the volume of data or display settings. This means that if a sensor is added to the platform at a later date, the base station GUI will not have to be updated to display the data.

Implementation of the control loop on the Arduino can begin during the refinement of the simulation and controller tuning. However, these will need to be finished before the Arduino implementation can be finished. These will begin in parallel with the base station GUI and Raspberry Pi programming.

Documentation will be an ongoing effort throughout the semester. Due to it's flexibility, this will be written in LaTeX. Packages such as the listings package will be used to make the documentation of our code straightforward. This process can be automated in such away that as modules are added or changed, their documentation can be flagged for updating. This will ensure that we are able to provide clear and thorough documentation throughout the project.

7 Distribution of Tasks

7.1 First Semester (Sep - Dec 2016)

For this semester, general tasks where distributed as follows. Exact assignments are labelled in the first semester Gantt chart in Appendix WHAT LETTER MAN?.

- Brendon
  - General Research
  - Preliminary SimuLink Tests
- Dylan
  - MIMO Controller Research

  - SimuLink flight simulation

- Lucas
  - Proof of concept hardware tests
  - Hardware research

7.2 Second Semester (Jan - Apr 2017)

For the coming semester, many of the tasks will be undertaken by at least two members of the group. However, each of these division will have a group lead within our group. More specific division of roles may be found in the our Gantt chart for the second semester. This chart is found in Appendix SAME AS ABOVE, please keep in mind that this Gantt chart will be revised in January. Once deliverable dates for ECED 4901 (project course two) have been released, and a timeline discussion with Dr Rhinelander have occured the Gantt chart will be revised. The leads of the general development portions are as follows:

- Brendan
  - Basestation GUI

- Dylan
  - Raspberry Pi Implementation
  - Documentation

- Lucas
  - Arduino Implementation

8 Budget

For the following reasons, this project has no anticipated costs. Firstly, Dr Rhinelander and Reiland Systems have absorbed the cost of the quadcopter. Secondly, the control hardware (Raspberry Pi and Arduino) are provided by the group for the duration of the project. At the end of the project ownership of the control hardware will be retained by their respective owners, as well as ownership of the quadcopter being retained by Dr Rhinelander. Lastly, all of our deliverables are provided either as open software or within software platforms that our client already has access too (e.g. SimuLink).

A Proof of Concept and Testing

Listing A.1: ESC Driver Script

```
#include<Servo.h>

//Define Constants

Servo esc;
const int MotorPin = 9;
const int Var_Resistor = A0;
int Resistor_Value = 0;
int time_value = 0;

void setup() {

//begin serial monitor & baud rate
Serial.begin(9600)

//set up motor connection
esc.attach(MotorPin);

//Enable Variable Resistor Input
pinMode(Var_Resistor,INPUT);
```

```
}

void loop() {

//Read the resistor value
Resistor_Value=analogRead(Var_Resistor);

//Var_Resistor will be between 1-1024
// need to convert to 1060-1860
//1060-1860 is a range of 800, 800/1023=0.78201

//Conversion into req'd pulse width range
time_value=(Resistor_Value*0.78201)+1060;

//write the time signal to the servo motor
esc.writeMicroseconds(time_value);
Serial.println(time_value);


}
```

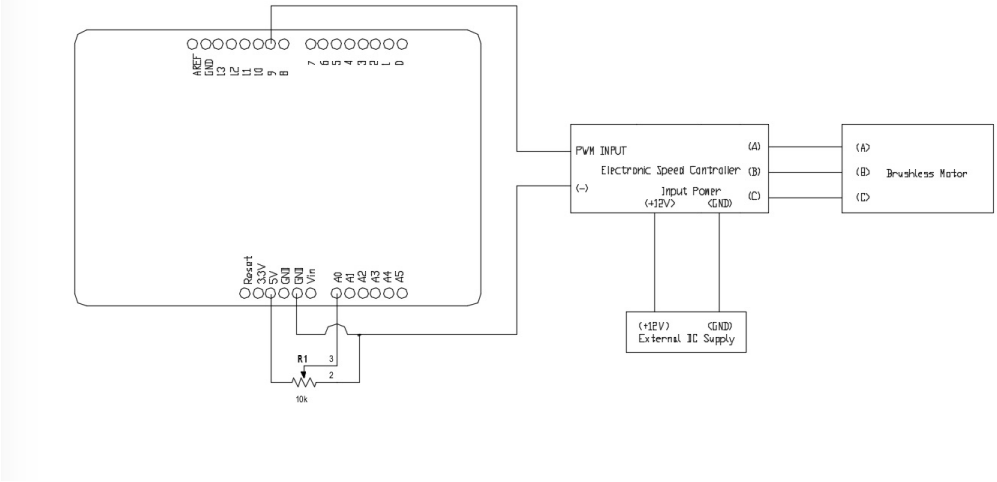


Figure A.1: Motor Driver Circuit POC Schematic

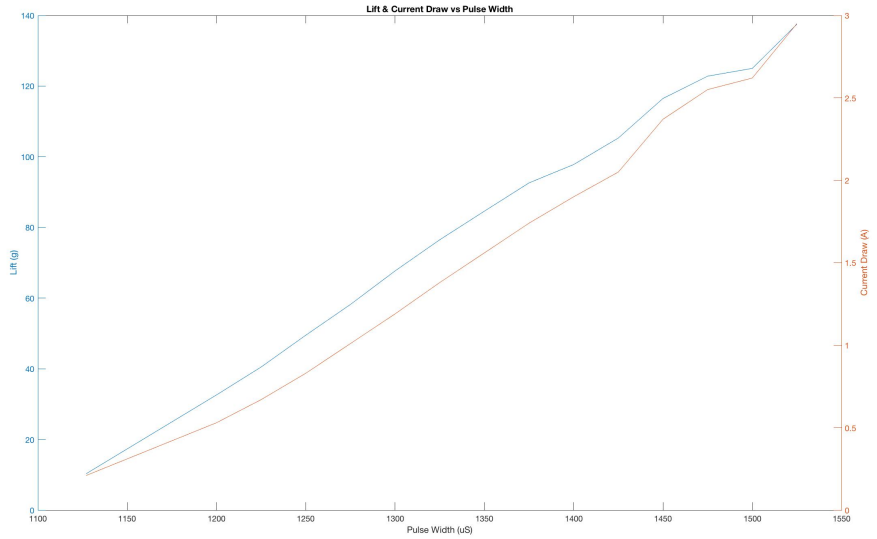
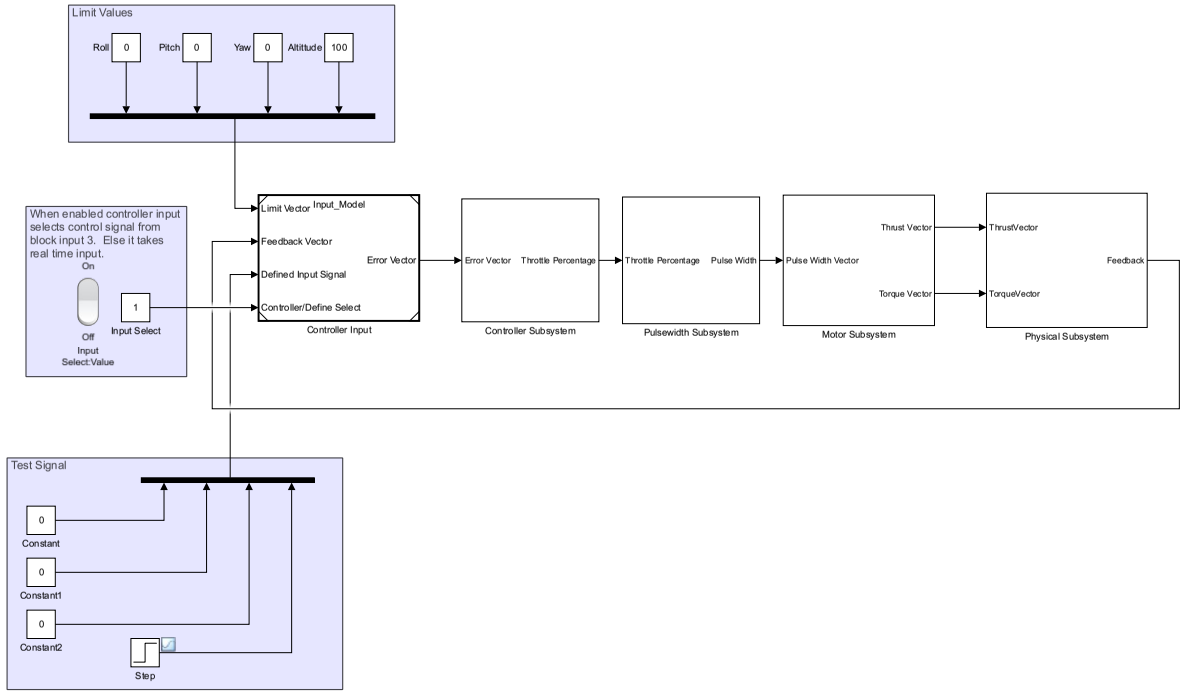


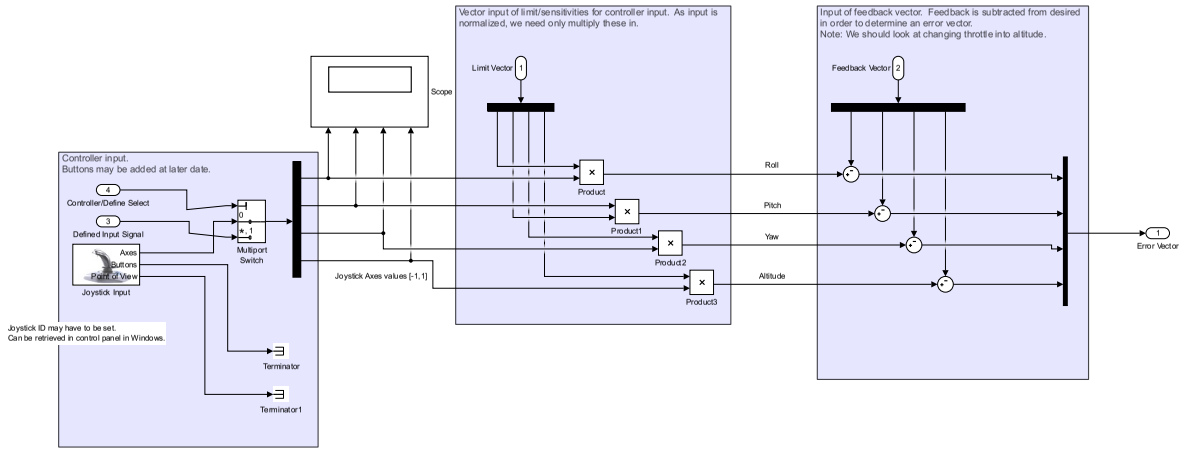
Figure A.2: Motor Characterization Testing

B SimuLink Models

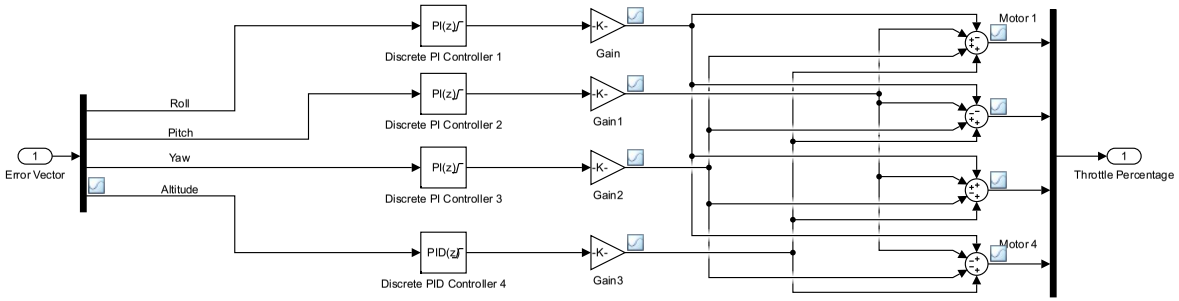
B.1 Dynamic Simulation



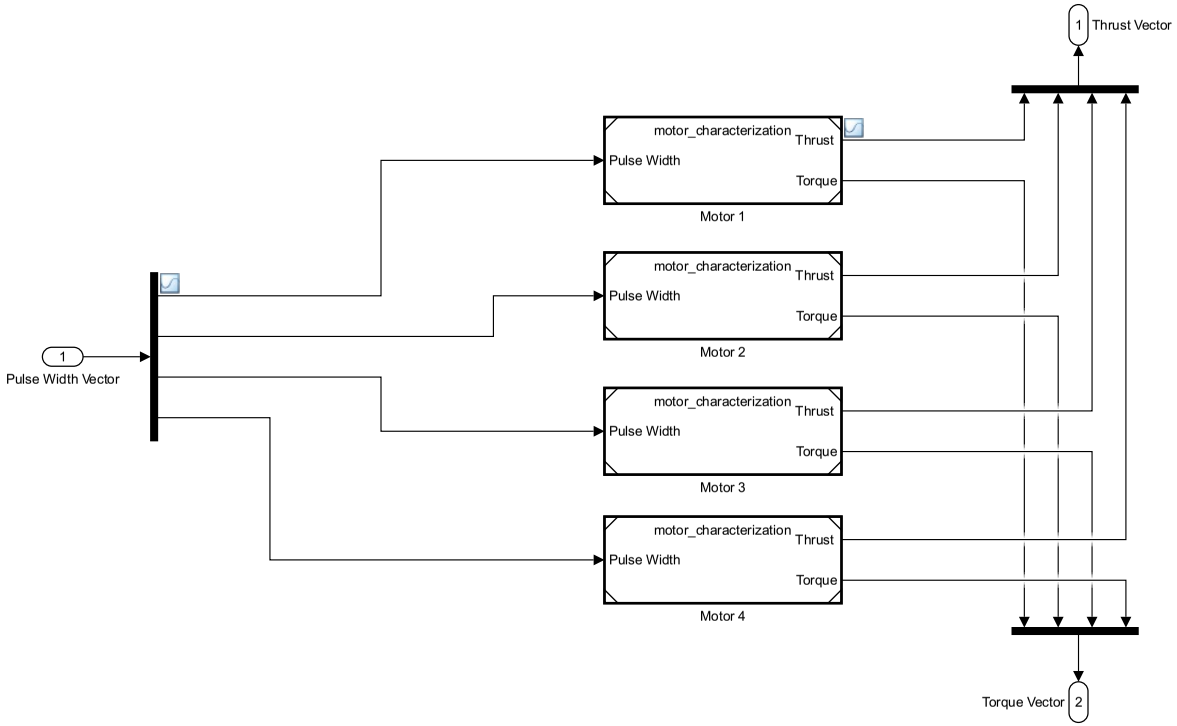
B.2 Input Model Subsystem



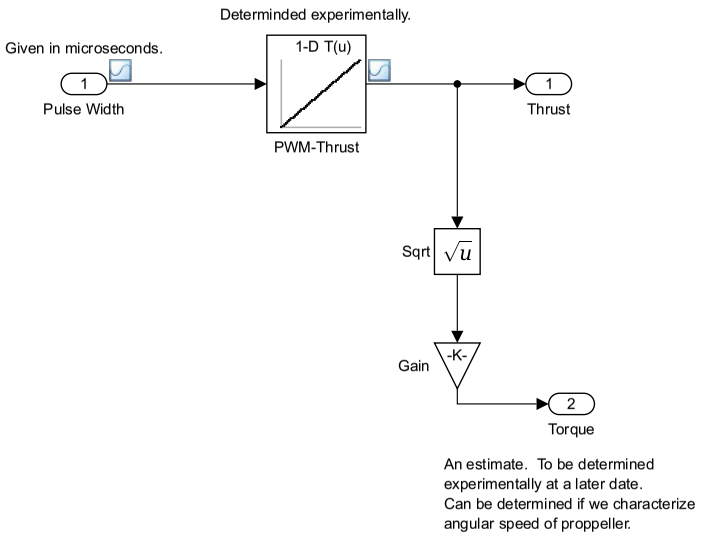
B.3 Controller Subsystem



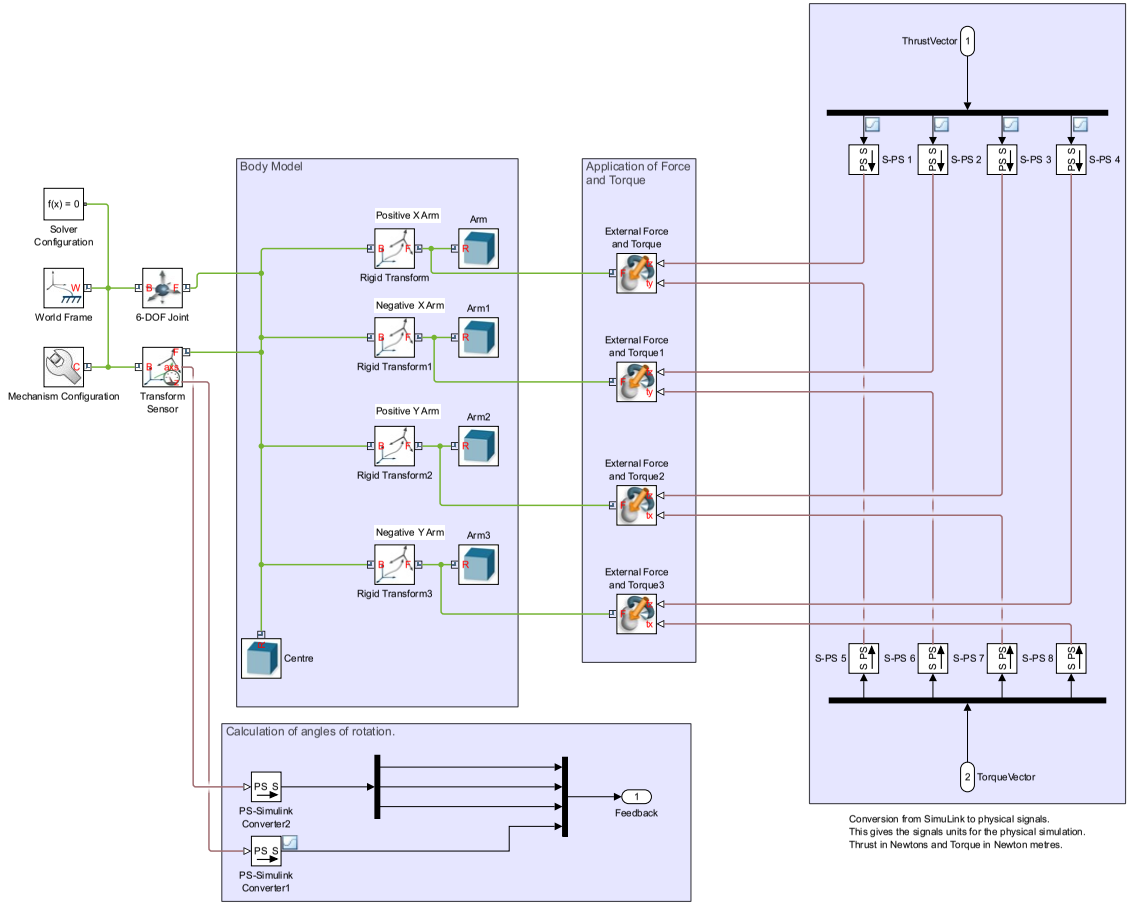
B.4 Motor Subsystem



B.5 Motor Characterization Model



B.6 Physical Subsystem



B.7 Pulsewidth Subsystem