Draft Report

Programmable Flight Controller

Brendon Camm	Lucas Doucette	Dylan Humber
B00649176	B00685962	B00695554

Submitted: March 27 , 2017

This document was written in LaTeX. It was compiled with pdflatex. The source of the document may be viewed at:
https://github.com/Brendoncamm/SYP/tree/master/Documents/Second%20Semester%20Report

Abstract

The following report summarizes the progress made throughout the Quad Rotor Drone Programmable Flight Controller project and the future recommendations based upon project findings. The goal of the project is to design a programmable quadcopter 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 the project and to present findings, results and testing procedures. The report also outlines the design process of the project. The Quad Rotor Flight Controller project has three major components; flight and stabilizing simulations, implementation of software, and a GUI interface. **Authorship**

- Abstract - Lucas Doucette
- Letter of Transmittal - Brendon Camm
- Simulations - Dylan Humber
- GUI - Brendon Camm
- Physical Implementation - Lucas Doucette
- Conclusions - Dylan Humber
- Future Recommendations - Group
- References - Group
- Appendices - Group

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 Introduction

2.1 Simulations

2.2 Physical Implementation

The physical implementation of the drone software and hardware requires a communication channel between a control input, a base station host, a wireless communication receiving unit, and a host for the controlling system. The objective of the physical implementation is to provide the software and hardware design required to successfully implement a quadrotor drone flight controller. It is desired

to have a controller that is capable of sensing disturbances in flight and stabilizing the system based upon the measured disturbances.

The objectives of the physical implementation for the project begin with successfully creating a WiFi communication channel between a controller input and a wireless device placed on the drone hardware. The drone is to have flight sensing capabilities, interfaced with both the on board wireless device and a grounded controlling station. The flight sensing is to include yaw, pitch, roll and altitude measurements with a polling rate sufficient for real time control of the drone. The flight sensing is to be combined with a set point from the wirelessly transmitted control input by a PI or PID loop to control the flight of the drone with stabilizing features. The code written for the control input, communication interface, and finally the controller is to be well documented to ensure ease of use and simplicity for updating and modifying after handing over the final product.

2.3 Graphical User Interface

The Graphical User Interface (GUI) requires a network connection between the base station host it lives on and the Raspberry Pi 3 to recieve the serialized dictionaries containing sensor data and controller information. The objective of the GUI is to provide the user with a medium to access information regarding the current flight, such as, the altitude at which the drone has flown or how the physical controller is configured. It is desired to have a real time plot of the altitude based on the sensor information and the ability to initialize the physical controller.

The objectives for the GUI for the project start with selecting an appropriate development framework that will function across Windows, Mac OS and Unix environments. The GUI is to have the ability to initialize the communications between the base station host and Raspberry Pi 3 to allow for the control inputs

to be sent using the base station host as well as receiving the data that is to be displayed on the GUI. The GUI is to be intuitive to avoid any unneccessary confusion with the end user. The code written and any other software used for the GUI will be provided and well documented to ensure that in event that the end user would like to modify anything after receiving the final product that this can be done effortlessly.

3 Design Process

3.1 Simulations

3.2 Physical Implementation

3.3 Graphical User Interface

3.3.1 Framework Selection

4 Design Solutions

4.1 Simulations

4.2 Physical Implementation

4.3 Graphical User Interface

5 Project Results

5.1 Simulations

5.2 Physical Implementation

5.3 Graphical User Interface

6 Discussion

6.1 Simulations

6.2 Physical Implementation

6.3 Graphical User Interface

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);


}
```

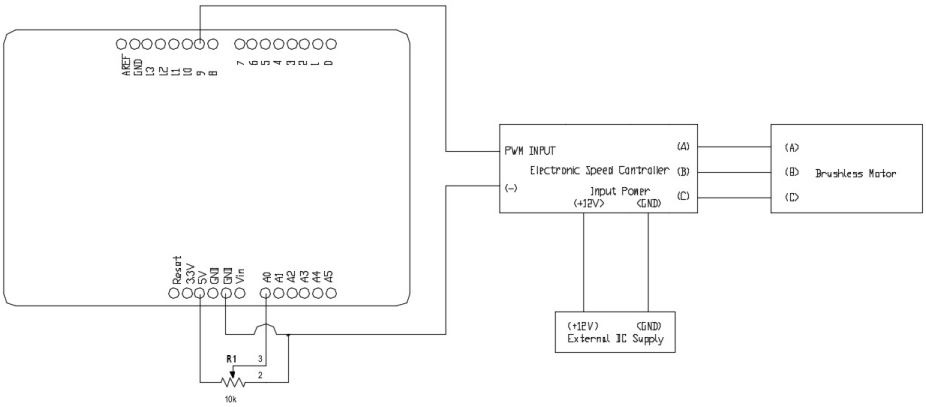


Figure A.1: Motor Driver Circuit POC Schematic

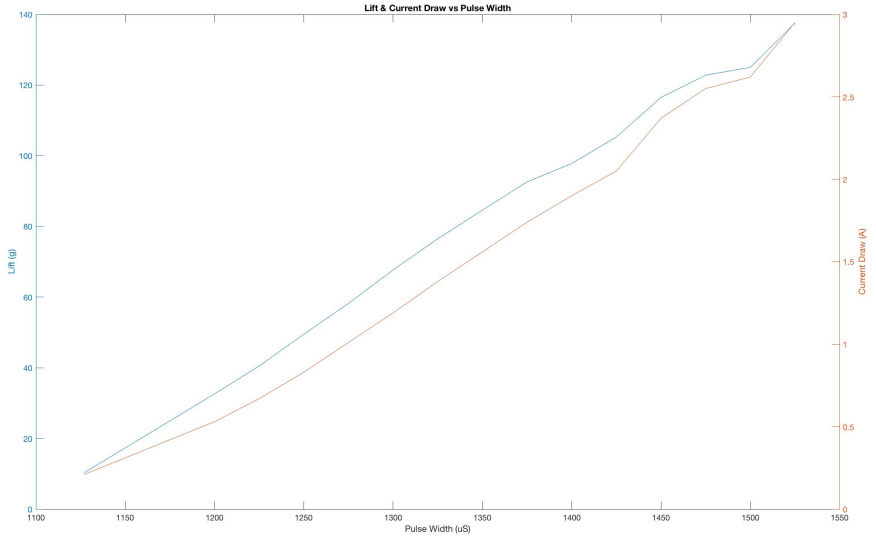
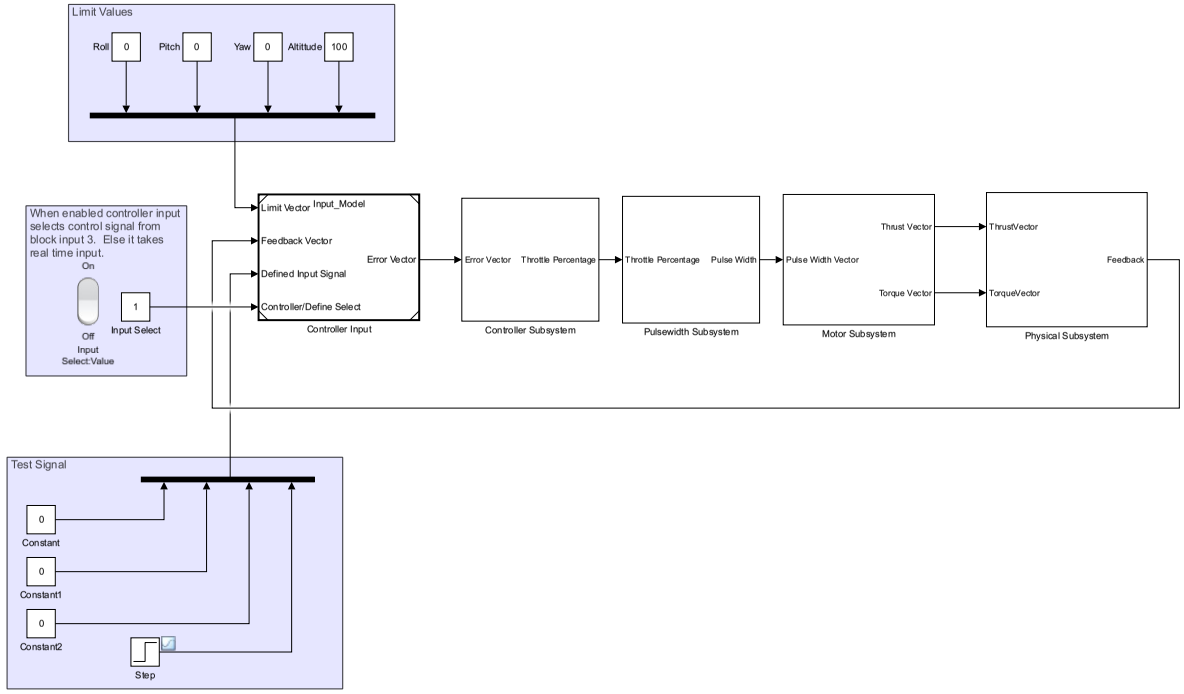


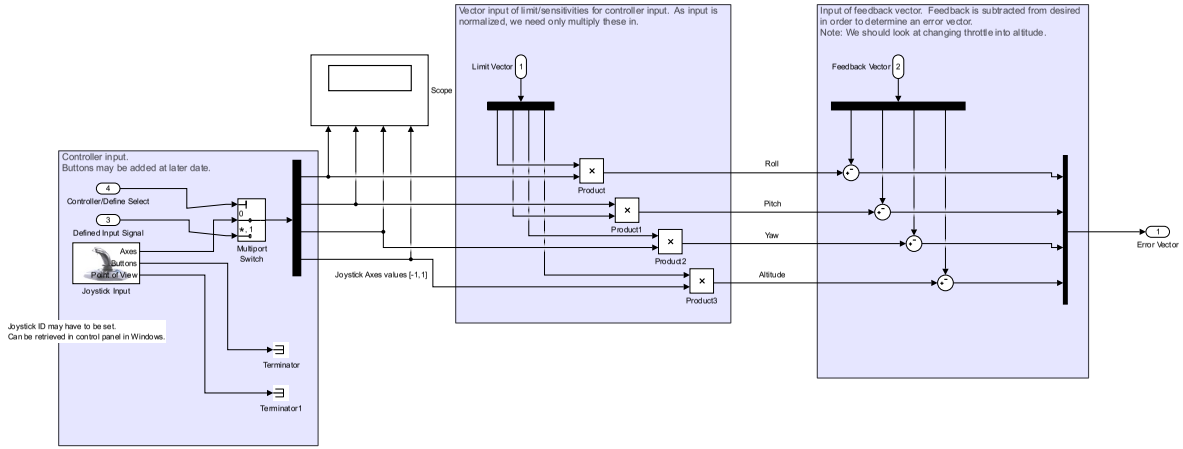
Figure A.2: Motor Characterization Testing

B SimuLink Models

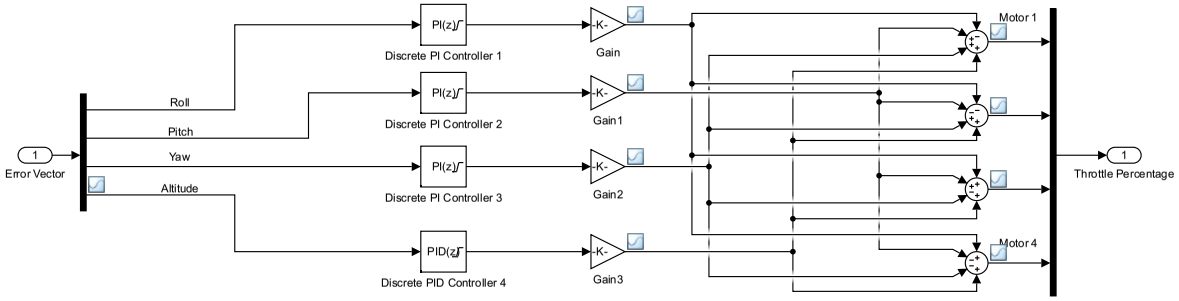
B.1 Dynamic Simulation



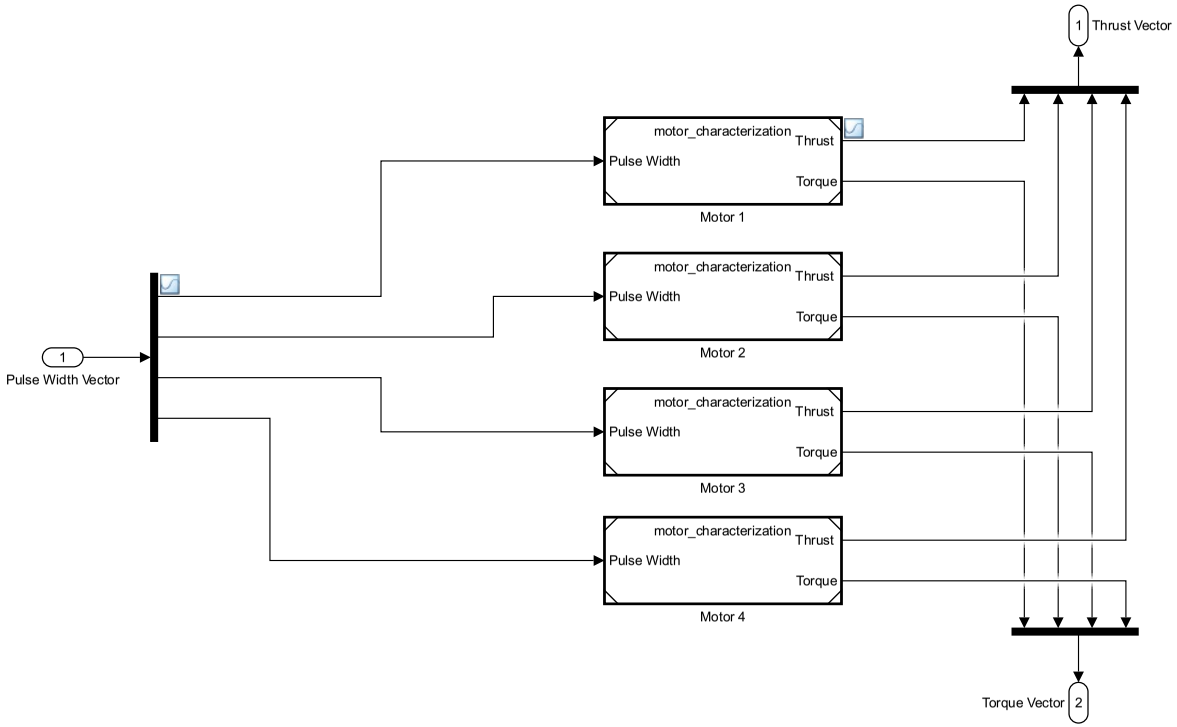
B.2 Input Model Subsystem



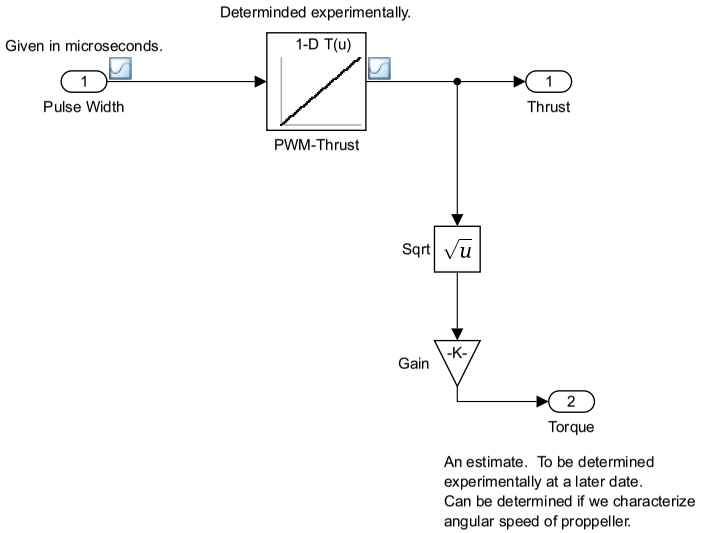
B.3 Controller Subsystem



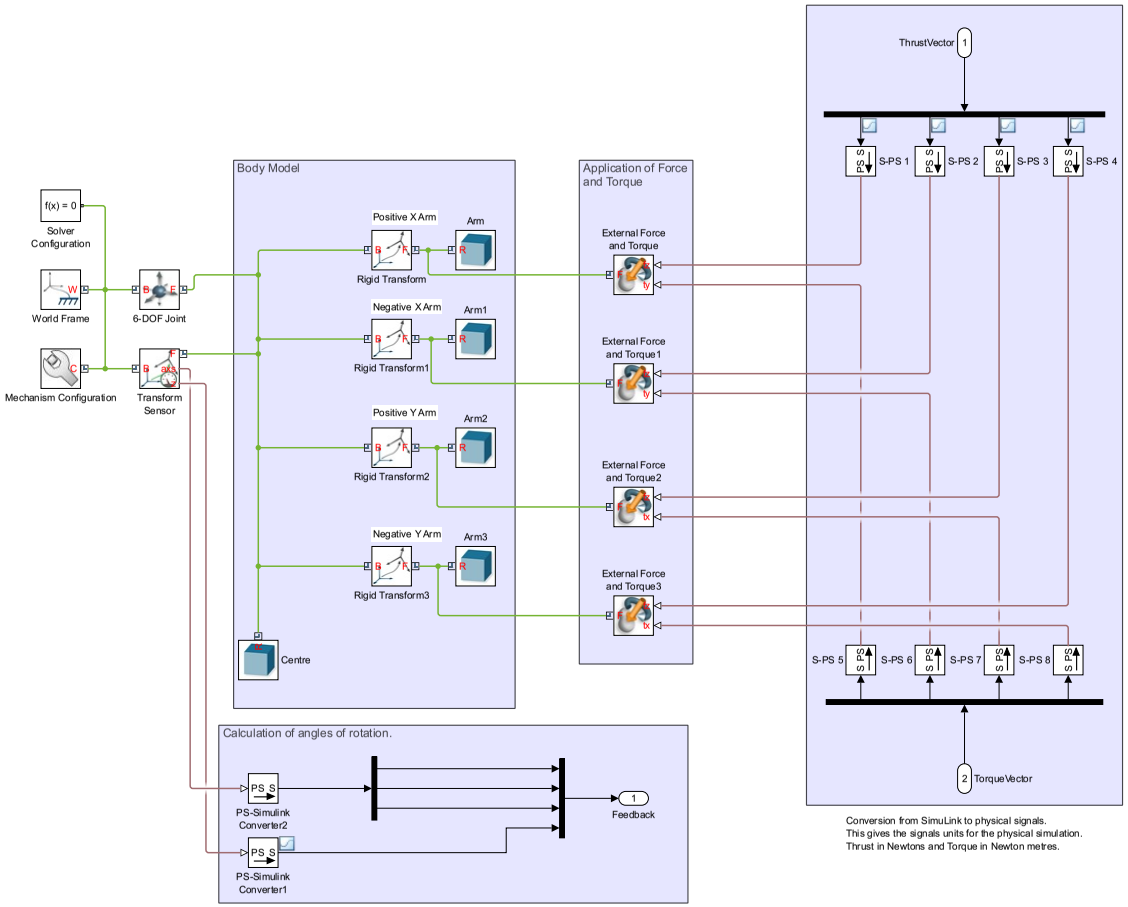
B.4 Motor Subsystem



B.5 Motor Characterization Model



B.6 Physical Subsystem



B.7 Pulsewidth Subsystem