Interim Report

DRAFT OF FINAL REPORT

Programmable Flight Controller

Brendon Camm
B00649176

Lucas Doucette
B00685962

Dylan Humber
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

Department of Electrical
and Computer Engineering
Dalhousie University
P.O. Box 1000
Halifax, Nova Scotia
B3J 1B6

March 26, 2017

Dr Jason Rhinelander
Reiland Systems Ltd.
PLACEHOLDER ADDRESS
Dartmouth, NS
POSTAL CODE
CC: Dr. Jason Gu
Dr José Gonzalez-Cueto
ECE Support Secretary

Dear Dr. Rhinelander:

Attached is Group 2s senior design interim report titled Programmable Flight Controller, written in fulfillment of the requirements of the Faculty of Engineering ECED 4901 Senior Design course. The report outlines the specifics of each portion of the project encounters throughout the 2016/2017 academic year.

The intention of this project is to develop a completely programmable flight controller for use on a quad-rotor drone. The controller and sensors will be implemented using an Arduino Uno Microcontroller and Raspberry Pi 3 computer. A graphical user interface was also developed to display information about the controller and sensors. The sections enclosed in the attached report discusses the simulation, implementation and testing of the controller as well as the design and testing of the graphical user interface. In addition, this report outlines the management side of the project including the Gantt Chart showing the time-lines, task division and each individual group members responsibilities. The report concludes with sections highlighting some of the challenges that were encountered over the course of the project including a new legislation regarding recreational drone usage introduced by the federal government in March of 2017 which severely disrupted our planned flight testing following this section will be recommended future improvements.

Please contact the groups nominated project manager, Dylan Humber (at dylan.humber@gmail.com) should you have any questions about the report.

Best Regards,

Dylan Humber

Brendon Camm

Lucas Doucette

Contents

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

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. RPi - Raspberry Pi
11. USB - Universal Serial Bus

2 Introduction

2.1 Background and Significance

Machine learning is defined as the science of getting computers to act without being explicitly 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 co-variates 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.

Quad rotor 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 base station 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.Rhinelander's 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 receives 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.

2.2 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 receive 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.

2.3 Simulations

The simulations will allow us to gain an understanding of how the controller will respond to specific inputs. The simulation can then be tuned to test various PID constants for performance. We will be simulating both the flight dynamics and controller using MATLAB and Simulink exclusively.

Additionally, the simulation will be able to take input from both predefined control sequences and HID on the operating system. As real time input from a HID is desired for the simulation, real-time rendering in 3D will be attempted.

2.4 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 quad rotor 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 Wi-Fi 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.

2.5 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 receive 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 unnecessary 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, this can be done effortlessly.

3 Methods

3.1 Simulations

3.2 Physical Implementation

The physical implementation of the system began with determining a method to interface a controller input with a base station host. To do this, research of available python libraries was performed. It was determined quickly that the most effective route would be to use Python's Pygame library which analyses a bluetooth or usb connected device and determines the types of control inputs applicable for the device. Initially, a PS4 bluetooth connected controller was successfully utilized, receiving all axis and button inputs through the Python interface.

The next challenge of the physical implementation was to determine the best method of Wi-Fi communication. It was determined that Python's socket library would be best suited, and a Raspberry Pi 3 with a Wi-Fi module available would act as the server in the server, client architecture. A preliminary communication server client script set was written and tested successfully between the base station laptop and the Raspberry Pi server host.

For the controller host, two possible solutions were considered. Utilizing an Arduino micro-controller as a permanent host of the control loop, communicating with the Raspberry Pi to receive control input data or using the Raspberry Pi for both communication and the controller. Using Arduino would provide a dedicated control loop and an intuitive interface to prototype with. The Raspberry Pi is capable of running the control loop, but issues with regards to a real time operating system were anticipated. Using an Arduino would require more mounting space on the drone as well as another communication channel between the RPi and Arduino to troubleshoot. Many other controllers could have been considered but both RPi and Arduino are owned by group members.

The controller required input from a 10 degree of freedom inertial measurement unit to calculate yaw, pitch, roll and altitude to poll and compare to the control set points. Eventually, alternative methods of calculating altitude were considered and discussed. The methods included infra-red distance sensing, ultrasonic sensing, multiple barometers coupled with Kalman filters, or a barometer accelerometer coupling. PI and PID loops were researched to be implemented on the controller. Several additional filtering techniques were considered for altitude measurement, including moving average filtering, averaging filtering and Kalman filtering with a feedback loop for comparison.

Communications between the RPi and the Arduino were researched, using the interface to interface bus or the serial communication channel were both consid-

ered. The data was required to be transmitted as single bytes, thus an algorithm to convert received bytes into floating point values needed research and development.

From a technical background perspective, research regarding the Python language and Python's available libraries was required. Server to client interfacing is also required technical background. Previous knowledge of the Python language assisted in directing the research and development flow throughout the project. PI and PID control loop knowledge was required and research time was used to learn filtering processes, Arduino library development, miscellaneous communication algorithms and utilization of the Adafruit 10 DOF IMU.

3.3 Graphical User Interface

3.3.1 Framework Selection

There is a multitude of frameworks available for graphical user interface development so in order to narrow down the choices the following restrictions were placed:

- Must function on Windows, Mac OS and UNIX environments
- Have the ability to display live plots
- Be compatibile with the physical controller initialization software
- Have well documented libraries for ease of development

With these restrictions placed and further research into various frameworks the selections narrowed down to Qt and PyQt5. Each of these use the Qt framework which is regarded as a reliable, cross-platform GUI development software with extremely well documented libraries which met the major requirements. Although both Qt and PyQt5 use the same framework there are a few key differences.

The most notable difference at first glance is that the two use different languages, Qt uses C++ where PyQt5 uses Python but the biggest difference is in the design suites. When developing using Qt you're able to use the QtCreator design suite which allows the user to design the look of the GUI using click and drag widgets, code the functionality of the widgets then simply compile all within the design suite. When using PyQt5 there isn't a complete design suite, instead the user must design the user interface using QtDesigner and then import the GUI file into the Python script, from here the widgets are coded is relatively the same other than the obvious differences between C++ and Python. To finalize the decision on which framework to use two basic GUI's with the same functionality were developed using Qt and then PyQt5.

The basic GUI was to have the ability to call the PS4 controller initialization script and manipulate the axis settings. Because the initialization script was written in Python it was very difficult and time consuming to do this using the C++ version of Qt compared to using PyQt5 where all that had to be done was import the script. Because each of the scripts that the GUI needs to call are written using Python it was decided to carry on using PyQt5. On top of the ease of importing the scripts using PyQt5 it also allows us to use the vast amount of Python libraries that are available. With the development framework decision made it was time use QtDesigner to design the layout of the GUI that will be intuitive for the end user.

3.3.2 GUI Layout

The layout of a GUI can make or break the end users experience, a clunky non-intuitive GUI will undoubtably cause the end user an unneeded headache. For this reason, a prototype of a graphical user interface was developed and sent to our client to get an idea of what he would be comfortable with and then this prototype was optimized based on his comments. The optimized layout was presented to each of the group members and a consensus was made to move forward using the new layout as it provides all of the required functionality in a clean manner.

A snapshot of the original prototype layout can be viewed in APPENDIX X, and the final layout can be viewed in the figures below. The purpose and functionality of the widgets will be discussed in the following section. With the layout of the GUI decided on it was time to begin developing the functionality of the widgets.

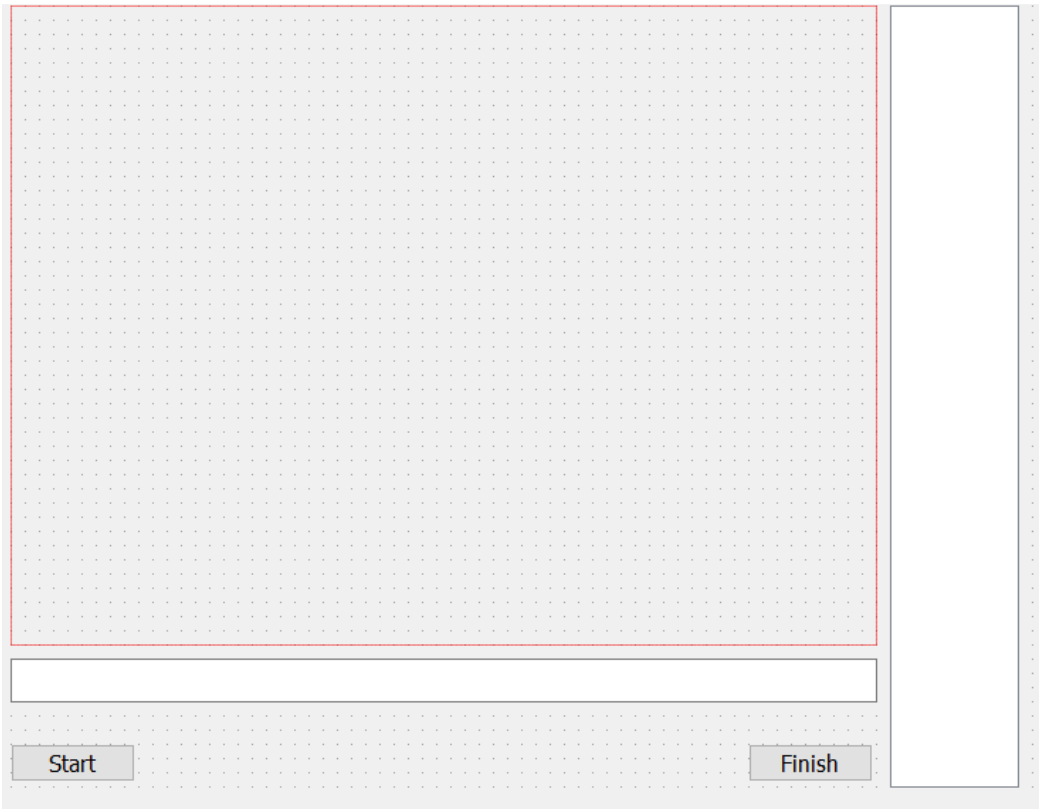

Figure 1: Final GUI Home Page

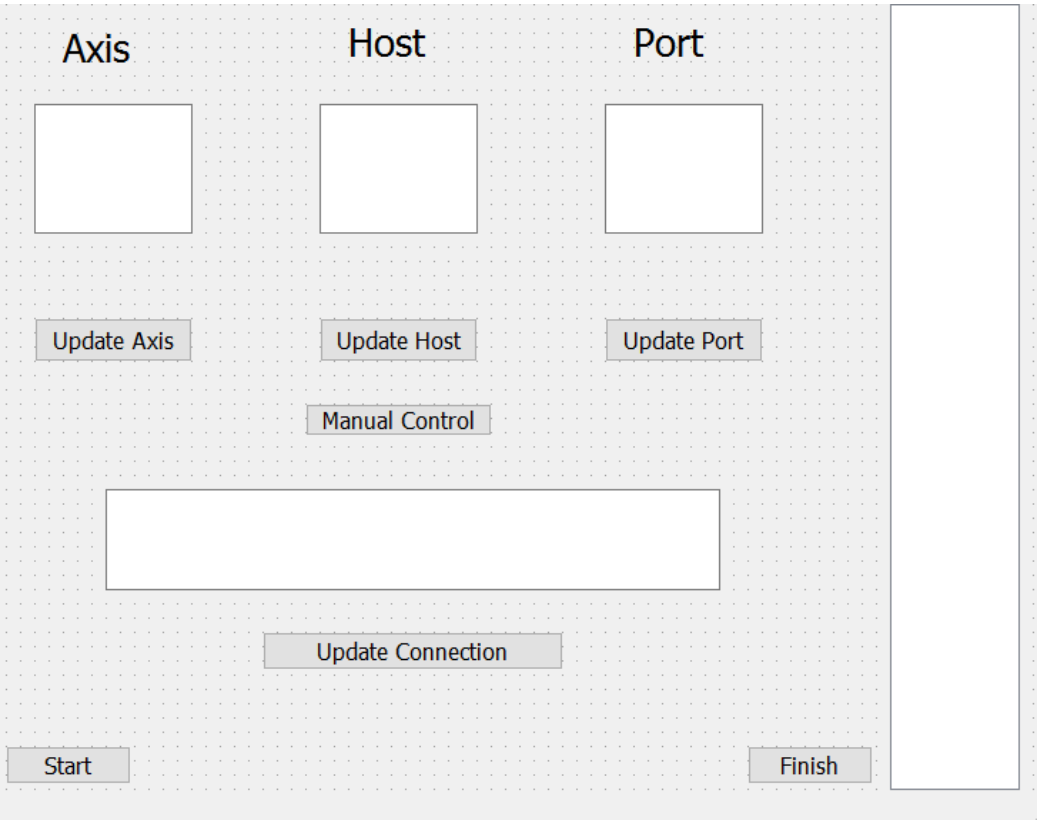

Figure 2: Final GUI Controller and Network Settings page

3.3.3 Widgets

4 Proposed 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