Marquette University
e-Publications@Marquette

Master's Theses (2009 -) | Dissertations, Theses, and Professional Projects

Flight Control and Hardware Design of Multi-Rotor Systems

Nathan M. Zimmerman
Marquette University

Recommended Citation
Zimmerman, Nathan M., "Flight Control and Hardware Design of Multi-Rotor Systems" (2016). *Master's Theses (2009 -)*. Paper 370.
http://epublications.marquette.edu/theses_open/370

FLIGHT CONTROL AND HARDWARE DESIGN OF MULTI-ROTOR SYSTEMS

by

Nathan M. Zimmerman, B.S.

A Thesis Submitted to the Faculty of the Graduate School
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

August 2016

ABSTRACT

FLIGHT CONTROL AND HARDWARE DESIGN OF
MULTI-ROTOR SYSTEMS

Nathan M. Zimmerman, B.S.
Marquette University

This thesis overviews crucial concepts involved in achieving quadcopter flight such as orientation estimation and control system implementation. This thesis also presents researchers with comprehensive hardware and software specifications for a quadcopter system. The primary application for this system would be for research with regards to the implementation of advance control techniques as well as data acquisition. Key constructs of this system include hardware software specifications for a flight controller, the radio system, and the sensorless brushless motor controllers.

Firstly, the thesis starts by developing a reference frame and a mathematical model for the quadcotper system. Next, flight orientation estimation is determined through an assortment of MEMS sensors such as an accelerometer, gyroscope, and magnetometer. Each sensor will be individually addressed as to its strengths and weaknesses with regards to orientation estimation. An algorithm will then be proposed for the data fusion of these various sensors. This fused data will then be fed into a control system that will efficiently stabilize the quadcopter.

Finally, this thesis will overview methods of integrating lidar data directly into the quadcopter's control system. Real-world lidar data is used and a computational geometry algorithm, ICL, is employed to translate the point cloud data into relevant control parameters.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my parents who cultivated my passion for engineering and provided endless encouragement. Secondly, I would like to thank my advisor, Dr. Cris Ababei, who provided a TA-scholarship and academic support which made this thesis possible. I also would like to thank my other committee members, Dr. Susan Schneider and Dr. Ronald Brown, for taking the time to review this thesis and provide feedback.

I would also like to thank all of the engineers who significantly invested in me during my development as an engineer. Specifically, I would like to thank Brian Booth from John Deere who invested in teaching me fundamentals of hardware engineering. Likewise, I would like to thank Bradly Schleusner and Nicholas Butts from Appareo Systems who taught me vital fundamentals of writing embedded software. In addition, I would like to thank Dr. James Richie for an incredibly useful antenna theory class which made the RF portions of this thesis possible. I would also like to thank Kellen Carrey for his feedback as well as assistance in developing the RF controller. Finally, I would like to thank my fellow lab mates, Milad Ghorbani and Wenkai Guan, who managed to put up with the noise this thesis generated as well provide useful feedback and support.

TABLE OF CONTENTS

ABSTRACT . . . . . **i**

ACKNOWLEDGEMENTS . . . . . **ii**

TABLE OF CONTENTS . . . . . **iii**

LIST OF TABLES . . . . . **vii**

LIST OF FIGURES . . . . . **viii**

CHAPTER 1 Problem Statement, Objective and Contributions . **1**

1.1 Problem statement 1

1.2 Objectives 2

1.3 Previous Work 2

1.4 Contributions 3

1.5 Thesis Organization 4

CHAPTER 2 Quadcopter Background Information **6**

2.1 Common Quadcopter Motor Types 7

2.2 Quadcopter Configurations and Frame Design 9

2.3 Quadcopter System Architecture 10

CHAPTER 3 Control Goals and Orientation Establishment . . . **12**

3.1 Quadcopter's Reference Frame 12

3.2 Control Objective 13

3.3 Purpose of Mathematical Derivation 14

3.4 Mathematical Model 14

3.4.1 Forces 16

3.4.2 Torques 19

3.5 Inertia and Angular Acceleration 20

CHAPTER 4 Flight Orientation Estimation **23**

4.1 Introduction 23

4.2 3 Axis Accelerometer 23

4.2.1 Introduction 23

4.2.2 3D MEM Accelerometer 24

4.2.3 Accelerometer Orientation Equations . . . 26
4.2.4 Accelerometer Drawbacks . . . 26
4.3 3 Axis Gryoscope . . . 28
4.3.1 Introduction . . . 28
4.3.2 MEMS Gyro . . . 29
4.3.3 MEMS Gyro Drift . . . 30
4.4 3 Axis MEMS Magnetometer . . . 30
4.4.1 Introduction . . . 30
4.4.2 Hard-Iron and Soft-Iron Interference . . . 31
4.4.3 Magnetometer Tilt Compensation . . . 34
4.4.4 Magnetometer Drawbacks . . . 34
4.5 Sensor Fusion . . . 37
4.5.1 Introduction . . . 37
4.6 Orientation Estimation Conclusion . . . 40
CHAPTER 5 Control System Implementation . . . 42
5.1 Introduction . . . 42
5.2 PID Basics . . . 42
5.3 PID Software Implementation . . . 44
5.4 GUI and PID Tuning . . . 46
5.5 Software Architecture . . . 48
CHAPTER 6 Radio Frequency System Implementation . . . 50
6.1 Introduction . . . 50
6.2 Selecting a Frequency Band . . . 51
6.3 Creating a link budget with Friis transmission . . . 53
6.4 2.4GHz Microstrip and Coplanar Waveguide Transmission Line . . . 54
6.5 RF Baluns . . . 57
6.6 RF Packet Structure and Collision Avoidance . . . 57
6.7 Conclusion . . . 60
CHAPTER 7 Hardware Implementations . . . 62
7.1 Introduction . . . 62

7.2 Flight Controller Hardware 63
7.3 Remote Controller 64
7.4 Electronic Speed Controller Hardware 68
7.4.1 Brushless Motor Operation 70
7.4.2 Introduction to BEMF Zero Cross Driving 71
7.4.3 Brush-less Motor Driving Hardware 71
7.4.4 ESC Conclusion 75
CHAPTER 8 ICL and Lidar Integration 76
8.1 Introduction 76
8.2 Lidar use for positioning 77
8.3 Method of parsing Lidar Data 79
8.4 Iterative Closest Line 81
8.5 Lidar Integration with Quadcopter Control System 84
8.6 Control System Testing, Simulation and Conclusion 85
CHAPTER 9 Conclusion and Future Work 87
9.1 Flight Performance 87
9.2 Conclusions 89
9.3 Future Work 90
REFERENCES 91
APPENDIX A Hardware schematics and PCBs 95
A.1 Hardware Schematics 95
A.2 Flight Controller Schematic 96
A.3 Flight Controller Schematic 97
A.4 Flight Controller Schematic 98
A.5 Flight Controller Schematic 99
A.6 ESC Schematic 100
A.7 ESC Schematic 101
A.8 ESC Schematic 102
A.9 RF Controller Schematic 103
A.10 RF Controller Schematic 104

A.11 RF Controller Schematic 105
A.12 ESC Dev Board 106
A.13 ESC Dev Board 107
APPENDIX B Transmission Line Calculations 108
B.1 Microstrip Line Z_o Calculation 108
B.2 Coplanar Waveguide Z_o Calculation 109

LIST OF TABLES

6.1 Title 47 part 15.245 ISM Bands. 52

8.1 Specifications of the two lidars from Fig.8.2. 78

LIST OF FIGURES

2.1 Examples of quadcopter implementations [1],[2],[3],[4]. 6

2.2 Examples of sensored(a) and sensor-less(b) brush-less permanent magnet motors [5],[6]. 7

2.3 Example of ESC [7]. 8

2.4 (a): **X**, (b): **H**, (c): + configuration quadcopters. In this figure, arrows represent propeller direction, wires represent rigid frame, and the center represents the quadframe body. 9

2.5 Marq Drone electrical architecture. 10

3.1 Quadcopter reference frame. 12

3.2 Mechanical dimensions of quadcopter. l represents the length of the quadcopter and h represents the height. 15

3.3 Quadcopter free body diagram. 16

3.4 Grams thrust (T_i) per duty cycle input. Duty cycle input represents power percentage of a 3 cell lipo battery. 17

4.1 2D macro scale spring accelerometer. 24

4.2 Accelerometer MEMs image [8]. 25

4.3 Accelerometer noise before and during motor operation. 27

4.4 Classical gyroscope or gyrostat. 29

4.5 MEMS 3 Axis Gyroscope [9]. 29

4.6 MEMS 3 Axis Magnetometer [10]. 31

4.7 Milli-Gauss (mG) output of an uncalibrated magnetometer when rotated in 3 dimensions. 32

4.8 Output of a calibrated and normalized magnetometer when rotated in 3 dimensions. 33

4.9 Plot demonstrating successful tilt compensation of magnetometer readings. Here θ and ϕ are actuated without significant change in ψ_C. . . . 35

4.10 Demonstration of magnetic intensity (M), magnetic inclination (δ), and magnetic declination (σ). 36

4.11 Global variation in: (a) magnetic intensity, (b) magnetic declination, (c) magnetic inclination [11]. 36

4.12 Complementary filter acting on raw accelerometer and gyro inputs. . . 37

4.13 Complementary filter with magnetometer integration. 38

4.14 Code snippet of 9DOF fusion filter. 39

4.15 Complementary filter validation test measuring and computing roll. . . 40

4.16 Complementary filter validation test measuring and computing pitch. . 40

4.17 Complementary filter validation test measuring and computing yaw. . . 41

5.1 Diagram of quadcopter plant and control system. 43

5.2 Expanded view of quadcopter's control system. 44

5.3 Code snippet of PID controller. 45

5.4 Code snippet for calculating PID results. 45

5.5 Code snippet of pitch, roll, and yaw thrust calculations. 46

5.6 Code snippet of open loop throttle control. 47

5.7 Code snippet of control loop. 47

5.8 Quadcopter javascript GUI. 48

5.9 High level quadcopter software flow chart. 49

6.1 RF Marq Drone hardware: (a) Flight controller, (b) Radio controller . 50

6.2 Block diagram of Marquette Drone RF system 51

6.3 PCB transmission lines: (a) Microstrip line, (b) Coplanar waveguide. . 55

6.4 Plot of how ϵ_r changes with respect to frequency [12]. 55

6.5 PI tuning network example located on the RF controller. 56

6.6 RF Baluns: (a) Integrated Balun [13], (b) Discrete Balun [14]. 58

6.7 RF packet implementation. 58

6.8 RF packet implementation observed on an oscilloscope by probing the balun voltage. 60

7.1 Block diagram of flight controller 64

7.2 Image of Marq Drone flight controller. 65

7.3 Block Diagram of flight controller. 66

7.4 Image of Marq Drone remote. 66

7.5 LiPo under voltage, over current, over charge, reverse bias, and backfeed protection circuit. 67

7.6 Block Diagram of Marq ESC. 69

7.7 Picture of ESCs: (a) Marq ESC, (b) Marq ESC Dev board. 69

7.8 Diagram of brushless motor and 6 driving switches. 72

7.9 6 step trapezoidal commutation sequence. 72

7.10 Software block diagram of zero cross brushless motor driving. 73

7.11 Brushless motor phase voltage (yellow) and zero cross detection comparator (green). 74

8.1 Lidar plot of room 77

8.2 (a) Hokuyo UTM-30LX lidar used in [15] . (b) Neato XV-11 lidar device used in this project. 78

8.3 Lidar moved to the left by $\approx 600mm$ 79

8.4 ICP Iteration 80

8.5 ICP iteration. 82

8.6 Pseudocode of the ICL algorithm 83

8.7 ICP and ICL integration results acting on dataset in Fig. **8.3**. 84

8.8 Quadcopter control diagram with lidar and sonar inputs added. 85

8.9 (a) Quadcopter simulation with external disturbance. (b) Plot of positional drift elimination. 86

9.1 Roll performance in hover test of quadcopter 87

9.2 Pitch, roll, and yaw performance during hover test of quadcopter. . . . 88

Acronym	Definition
BEMF:	Back Electro-Motive Force.
CPWG:	CoPlanar WaveGuide.
EMF:	Electro-motive Force.
FPU:	Floating Point Unit.
ESC:	Electronic Speed Controller.
FET:	Field Effect Transistor.
FSK:	Frequency Shift Keying.
ICP:	Iterative Closest Point.
ICL:	Iterative Closest Line.
IMU:	Inertial Measurement Unit.
ISM:	Industrial Scientific Medical.
I/O:	Input / Output.
LiDAR:	Light Detection And Range.
LiPo:	Lithium-ion Polymer.
MCU:	Micro-Controller Unit.
MEMS:	MicroElectroMechanical System.
PID:	Proportional Integral Derivative.
PPM:	Pulse Position Modulation.
QFN:	Quad Flat No-leads.
RF:	Radio Frequency.

SPI: Serial Peripheral Interface.

UART: Universal Asynchronous Receive Transmit.

UHF: Ultra High Frequency.

USB: Universal Serial Bus.

Math/symbol	Definition
θ	Pitch
ϕ	Roll
ψ	Yaw
$\hat{x}$	Estimate of x
$\bar{x}$	Unit vector of x direction
$\cdot$	Scalar product
$\times$	Cross product
$UPPERCASE$	Matrix
A^{-1}	Inverse of matrix A
$\|x\|$	Absolute value of x
$atan2()$	Four quadrant arc tangent function
$atan()$	Traditional two quadrant arc tangent function

CHAPTER 1

Problem Statement, Objective and Contributions

1.1 Problem statement

This thesis proposes quadcopter control theory, inertial measurement unit (IMU) orientation fusion, and provides a comprehensive software/hardware platform. To establish a reference frame, this thesis specifies a mathematical coordinate system and then uses this system to determine control goals. The remainder of the thesis focuses on how to meet the proposed control goals. An overview of existing IMU sensors is presented as well as their trade-offs with respect to quadcopter flight. An algorithm is then proposed to overcome existing sensor limitations by fusing IMU data to obtain orientation. Once known orientation is established, this thesis then focuses on specifying multi-rotor kinematics, software, and hardware involved in flight control. Finally, this thesis explores means of autonomous control with a lidar sensor.

The primary problem this thesis attempts to solve is in reducing barriers to entry for advance control techniques. When building a multi-rotor vehicle (drone), designers are faced with the choice of paying for a custom designed aerial vehicle, building their own vehicle from scratch, or sacrificing controllability for a inexpensive off-the-shelf system. While numerous inexpensive off-the-shelf multi-rotor platforms are available, they often consist of proprietary modules even when advertised as open-source. Common examples of these

black-box modules are sensor-less brushless motor controllers, flight controllers, and radios. These modules are often proprietary and have limited hardware specifications [16],[17]. Consequently for a researcher, the control and the modifiability of these modules is limited.

1.2 Objectives

A main objective of this work is to provide researchers with a functional, fully specified, and stabilized quadcopter. This system will be specified from scratch hardware and software with the intent of eliminating as many black box components as possible. In addition, this flight system will have an emphasis on theoretical control as well as IMU data collection making it a prime candidate for future research.

1.3 Previous Work

Admittedly, much work has been done in the area of advance control of multi-rotor systems and these works are too numerous for a comprehensive listing. Consequently, key examples will be provided that were used as a reference for the development of this thesis. For example, Robert Mahony presents a comprehensive method for modeling, orientation, and control of a quadcopter with state space methods [18]. Another example of advance control is where researchers at University of Zurich implemented a quadcopter with a model predictive control were able to perform extreme acrobatic maneuvers [19],[20]. Numerous other control techniques have been applied to quadcopters as

well such as PID, LQR, LQR-PID, and H_∞ [21], [22].

With regards to open source multi-rotor systems, Open Pilot and Clean Flight are perhaps two of the most popular open software flight controller systems [16],[17]. These frameworks support a broad range of multi-rotor vehicles from tri-copters to octo-copters. With regards to open source software and hardware systems, the Pixhawk and Sparky systems feature an open source flight controllers [23],[24]. While these systems feature some open hardware and software, they integrate with systems that are proprietary.

In regards to indoor autonomous control, this is an active area of research for all types of remote vehicles. Mapping of unknown environments has been conducted with mutli-rotor vehicles utilizing lidar and employing the iterative closest point (ICP) algorithm [15],[25],[26]. However, in these cases the multi-rotor vehicle was operated by a human. A team at MIT achieved autonomous indoor control of an aerial vehicle by combining lidar data and IMU data with an extended Kalman filter as well as a Gaussian particle filter [27]. However, in this case the environment had been pre-mapped and pre-determined trajectories were used. Comprehensive simultaneous mapping and control is still an ongoing area of research.

1.4 Contributions

This thesis developd a functional quadcopter system with a higher level of integration than most other open source options. In addition, this thesis develops a computationally efficient lidar parsing algorithm. The key contributions of this

flight system are as follows:

- Provides open source software / hardware files for a functional stabilized auto-leveling flight controller
- Provides open source software / hardware files for a remote control system
- Offers practical design insights for other researchers attempting to construct their own aerial vehicles
- Proposes a computationally efficient implementation of ICL for lidar data translation

1.5 Thesis Organization

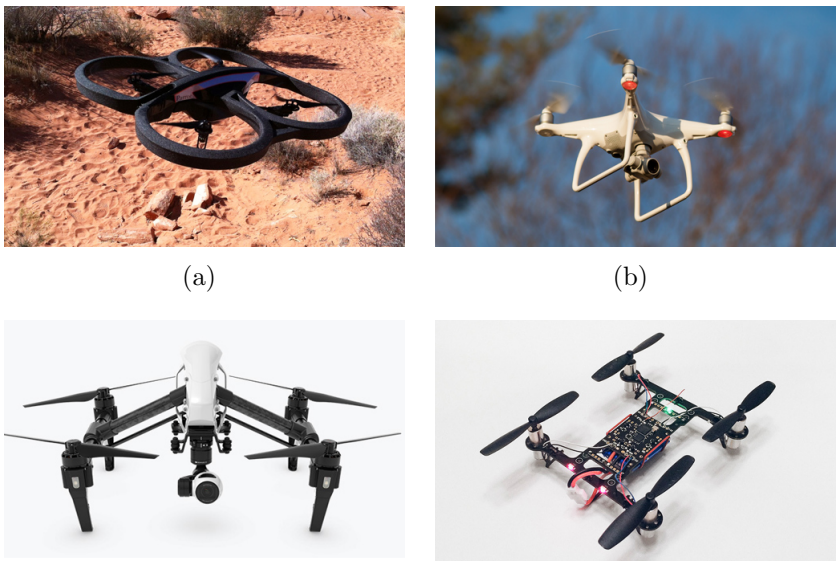
Chapter 2 presents fundamental background information and terminology associated with multi-rotor systems that will be used throughout this thesis. Chapter 3 overviews the the mathematical reference frame for the multi-rotor system and specifies control objectives. Chapter 4 describes sensors used for quadcopter orientation estimation as well as a fusion algorithm to combine various sensors. Chapter 5 proposes a control system to achieve auto-leveling flight. This chapter also overviews the software architecture used in order to execute the desired control system. Chapter 6 overviews the quadcopter's radio system and introduces applicable theory required for its design and construction. Chapter 7 describes other hardware components of the quadcopter such as the flight controller and electronic speed controllers. Chapter 8 overviews lidar integration with a quadcopters control system. This chapter also specifies an

algorithm for parsing lidar data under significant processing and memory constraints. Finally, Chapter 9 overviews the findings of this thesis, presents flight data, and discusses future work.

CHAPTER 2

Quadcopter Background Information

In order to provide context for future chapters, this chapter introduces the basic inputs and outputs common to a quadcopter system. As the name quadcopter implies, a quadcopter is a multi-rotor aircraft with four propellers. Beyond the similarity of four propellers, there is significant design diversity. This design diversity includes but is not limited to motor type and frame design. Examples of various quad-copter implementations are shown in Fig. **2.1**.



(a) (b)

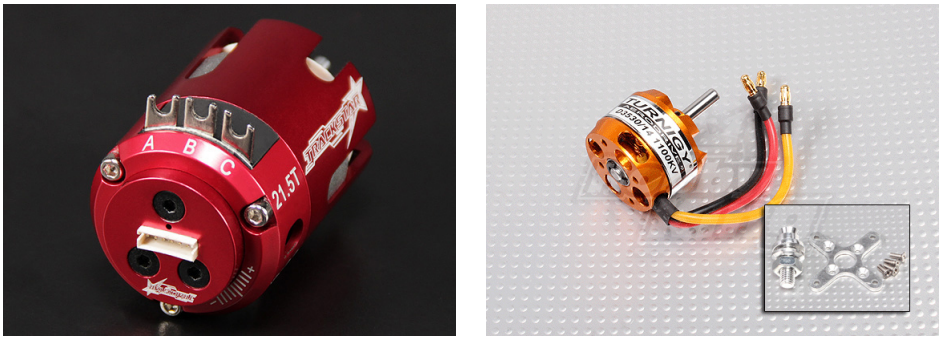
(c) (d)

Figure 2.1: Examples of quadcopter implementations [1],[2],[3],[4].

The quadcopters in Fig. **2.1** are referenced throughout this chapter as demonstrations of different types of design. A defining aspect of these quadcopters is the motor type that they utilize. Consequently, the next section will overview the common types of quadcopter motors.

2.1 Common Quadcopter Motor Types

Low-cost commercial quadcopters generally use electric DC motors such as brushed and brushless permanent magnet motors. In contrast to gas motors, there exist small electric motors that are light-weight, low-cost, and of simple construction. These features make small electric motors ideal for low-cost commercial quadcopters. Among these electric motors, two common types of DC electric motors exist which are brushed and brushless motors. As the name implies, brushed DC motors are mechanically commutated with a brush, are powered by DC, and are explained in detail in [28]. The control simplification and cost of brushed DC motors makes them a popular choice for micro-size quadcopters such as the Turnigy Micro-X shown in Fig. **2.1**(d). However, the brush which mechanically commutates the motor results in friction losses as well as limited motor life span. Consequently, larger quadcopters often use brushless permanent magnet DC motors which are electronically commutated. Brushless motors are also split into two common types which are sensored and sensor-less motors. Examples of these two motor types are illustrated in Fig. **2.2**.



(a) (b)

Figure 2.2: Examples of sensored(a) and sensor-less(b) brush-less permanent magnet motors [5],[6].

In order to electronically commutate a permanent magnet motor, the rotor's position must be known. This can be achieved by using Hall effect sensors or by using sensor-less driving techniques. Sensored motor operation simplifies driving complexity but results in heavier and more exspensive motor as seen in Fig. **2.2**. Instead of using sensors, sensor-less techniques such as back electro-motive force (BEMF) zero-cross detection and field oriented control can be implemented [29],[30]. Sensor-less driving will be discussed in more detail in Chapter 7. Sensor-less operation is desired since sensor-less motors have reduced weight, cost, and complexity. As a result of this, sensor-less brushless motors are common in quadcopters. However, in order to drive these motors, a DC to AC 3 phase sensor-less motor driver is needed. In terms of popular multi-rotor vernacular, these are commonly referred to as electronic speed controllers (ESC).

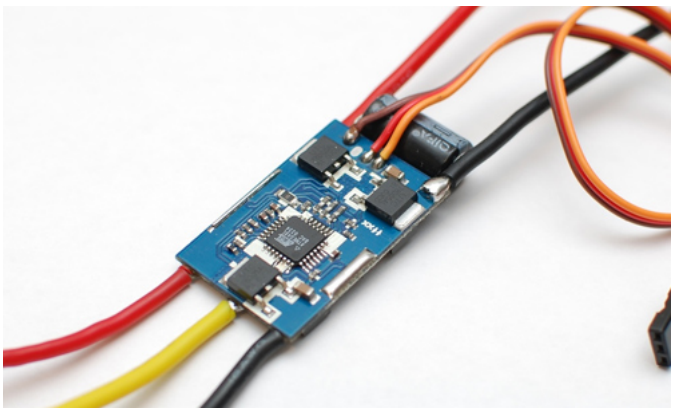
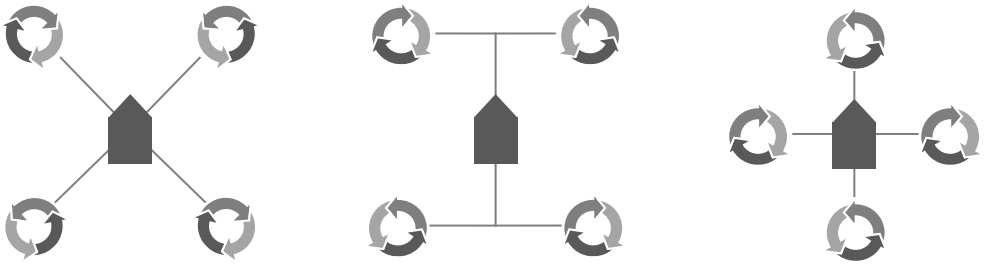


Figure 2.3: Example of ESC [7].

These ESCs represent an integral part of the quadcopter system architecture since their output controls the orientation of the quadcopter by varying the speed of the propellers. These devices will be discussed in greater detail in Chapter 7.

2.2 Quadcopter Configurations and Frame Design

Quadcopters have various configurations though the most common types are the **X** configuration, the **H** configuration, and the + configuration. These various configurations are illustrated in Fig. **2.4**.



(a) (b) (c)

Figure 2.4: (a): **X**, (b): **H**, (c): + configuration quadcopters. In this figure, arrows represent propeller direction, wires represent rigid frame, and the center represents the quadframe body.

Each type of configuration offers advantages and disadvantages. The **X** configuration is the most commonly utilized motor configuration since it is simple to construct, ideal for a forward facing camera, and is symmetrical. Quadcopters in Fig. **2.1**(a,b,c) are using this type of configuration. A disadvantage of this configuration is an increase in control complexity. In contrast to the **X** configuration, the + configuration is the simplest to mathematically model and control. However, this configuration is least ideal for a forward facing camera. Consequently, very few commercial drones are sold in this configuration and they generally only appear in research or DIY projects. Finally, the **H** configuration is sometimes built for mechanical convenience as seen in Fig. **2.1**(d). In contrast, the DJI Inspire in Fig. **2.1**(c) was designed as a **H** configuration quadcopter to

achieve improved camera perspective [31]. While this configuration can be ideal for forward facing cameras, it is also not symmetrical about its center. This lack of symmetry should be taken into account and will be discussed later in this thesis. These three configurations including minor deviations represent the majority of quadcopter configurations in common use today.

2.3 Quadcopter System Architecture

A final step in presenting necessary background information for this quadcopter thesis is to briefly overview the electrical architecture. A block diagram of the architecture unique to the quadcopter designed in this thesis is shown in Fig. **2.5**.

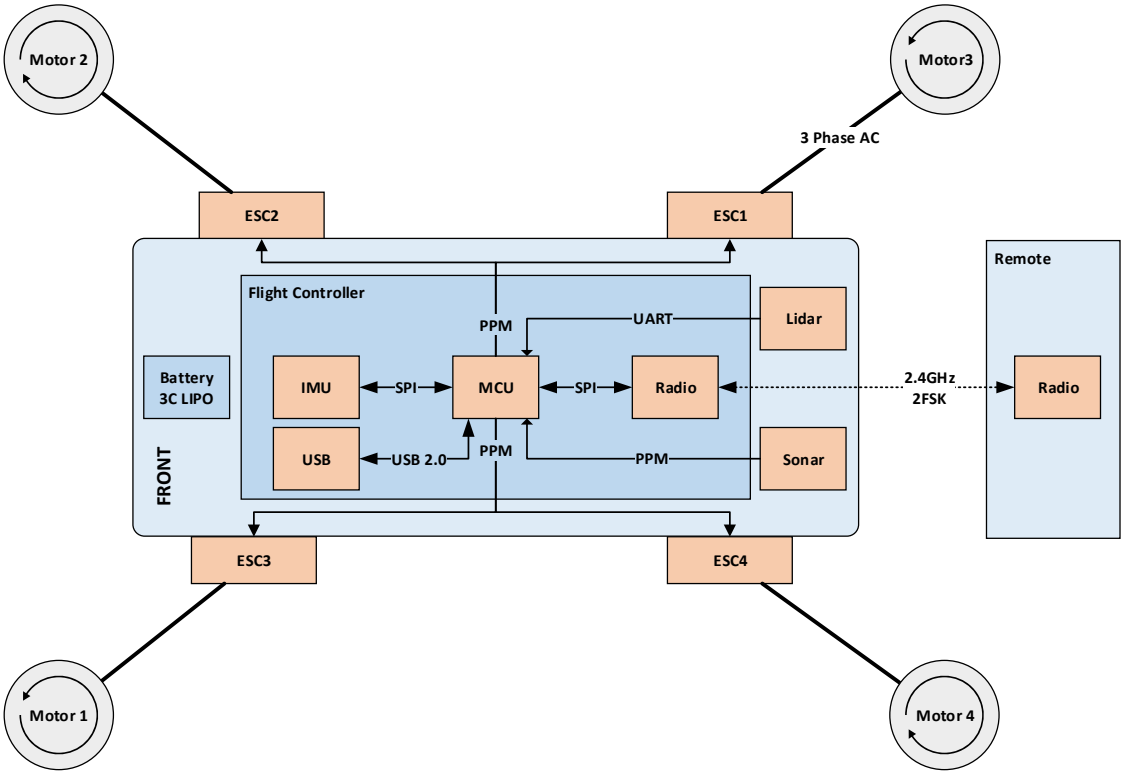


Figure 2.5: Marq Drone electrical architecture.

The underlying system for the quadcopter in this thesis is named the "Marq Drone" system. This system consists of a flight controller, a radio, and ESCs. Core components of the flight controller include a micro-controller unit(MCU), a radio, and an inertial measurement unit (IMU). The flight controller communicates with the ESCs through pulse position modulation (PPM). More information about PPM can be found here [32]. In addition to communicating with ESCs, the flight controller communicates with a lidar sensor through a universal asynchronous receive and transmit (UART) interface. To communicate with devices external to the quadcopter, a USB interface can be used for data acquisition or programming. Another communication method is through a wireless 2.4GHz frequency shift keyed(FSK) interface for receiving flight commands and transceiving flight data.

Now that all the components involved have been introduced, the I/O of the system can be identified. For the flight controller itself, feedback inputs are from the IMU, sonar, and lidar sensors. The outputs of the system are the four individual PPM signals that are sent to the ESCs. Now that the I/O of the system has been briefly introduced, a control scheme will be derived that uses these I/O to achieve stable flight.

CHAPTER 3

Control Goals and Orientation Establishment

In this chapter, a reference coordinate system for the quadcopter will be presented in addition to control goals. Like other multi-rotor vehicles, the primary control objective is to keep the quadcopter's orientation controlled and its altitude above ground non zero. With this in mind, a commonly used reference coordinate system will be presented to explicitly define this control objective.

3.1 Quadcopter's Reference Frame

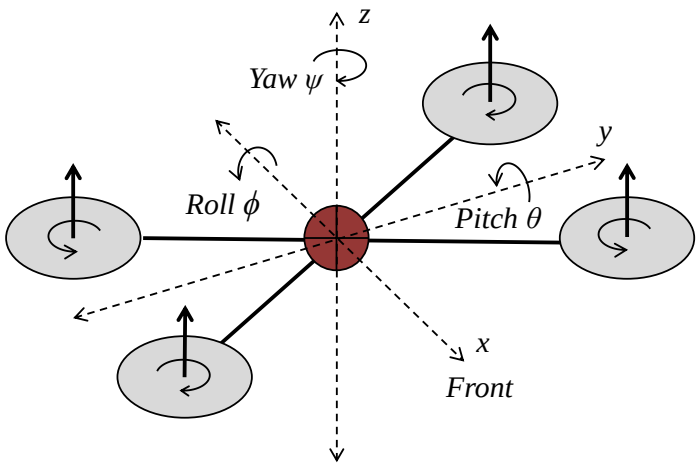


Figure 3.1: Quadcopter reference frame.

Fig. 3.1 demonstrates a reference frame for quadcopters. The reference frame uses Cartesian x,y,z system of coordinates for position and uses Euler angles pitch (θ), roll (ϕ), and yaw (ψ) to denote orientation. The origin the axes represents the center of the quadcopter as well as the center of mass.

3.2 Control Objective

A primary control objective for the quadcopter is to be able to control orientation. With this in mind, the first set of control goals can be summarized in equation **3.1** where θ_c, ϕ_c, ϕ_c represent a user commanded rotation, where $\dot{\theta}, \dot{\phi}, \dot{\psi}$ represent the rotational velocities of the quadcopter, and where T_c represents the user commanded throttle.

$$\begin{aligned} &\dot{\theta} = 0 \qquad \dot{\phi} = 0 \qquad \dot{\psi} = 0 \\ &\theta \to \theta_c \qquad \phi \to \phi_c \qquad \psi \to \psi_c, \qquad T \to T_c \end{aligned} \tag{3.1}$$

These control parameters, $\theta_c, \phi_c, \psi_c, T_c$, allow a user to maneuver the quadcopter to anywhere in three dimensional space. Consequently, many commercial systems give users these four degrees of freedom to operate a quadcopter. However, the user must also act as a control system in order to regulate the quadcopter's height to keep it above the ground. To achieve this, the user must observe the quadcopter's height and constantly adjust the throttle, T_c, such that the height with respect to ground, z, is roughly constant. Also, if a user desires a quadcopter to stay at a fixed position in space, they must also observe the x, y position of the quadcotper and adjust θ, ϕ accordingly.

While these parameters allow a user to control a quadcopter, these four degrees of freedom are not enough for stable autonomous flight. Without a user to observe and control the quadcopters x, y, z position with respect to the room, stable flight is not possible. Therefore as a secondary control objective, equation **3.2** demonstrates the control needed for autonomous flight where x_c, y_c, z_c

represents a user desired position.

$$\begin{aligned} \dot{x} = 0, \qquad \dot{y} = 0, \qquad \dot{z} = 0 \\ x \to x_c \qquad y \to y_c \qquad z \to z_c \end{aligned} \tag{3.2}$$

To meet the autonomous control objectives in equation **3.2** , additional sensors are needed such as lidar, GPS, and sonar. These methods will be introduced later in this thesis.

3.3 Purpose of Mathematical Derivation

A mathematical derivation of a multi-rotor frame provides a basis for update laws and helps a designer gain intuition regarding the system. In simple cases, these update laws can be derived from visual inspection of the airframe. However, this method becomes less effective as motor count is increased, as asymmetries are introduced to the air frame, and as motor angles are changed. Consequently, the incentive of the following sections is to translate input of motor speed to angular velocities $(\dot{\theta}, \dot{\phi}, \dot{\psi})$ such that the control system can drive pitch (θ), roll (ϕ), and yaw (ψ) to their desired values.

3.4 Mathematical Model

Now that control goals have been defined for user-assisted quadcopter flight, a model of the quadcopter must be derived to meet the proposed goals. For this thesis, a modified **X** configuration quadcopter design was selected. This was chosen for mechanical convenience and mounting simplicity. A diagram shown in

fig. **3.2** dimensions this modified **X** configuration quadcopter.

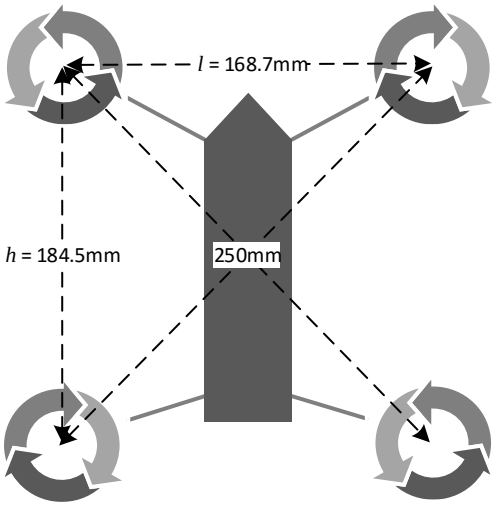


Figure 3.2: Mechanical dimensions of quadcopter. l represents the length of the quadcopter and h represents the height.

These dimensions are relevant to the physics based model derivation. Note that this quadcopter is not symmetrical about its center. Fig. **3.3** demonstrates a more detailed free body diagram of forces and torques acting upon the quadcopter.

In absence of system input, the dominant force acting on the quadcopter is mass·gravity ($-m \cdot g$). To counteract this force, thrust vectors ($[T_0, T_1, T_2, T_3]$) are generated by the propellers. Note, the aerodynamic effects that are not shown in Fig. **3.3** since these forces are assumed to be negligible. Wind and drag are examples of aerodynamic effects that have been set to zero in order to simplify modeling of the system.

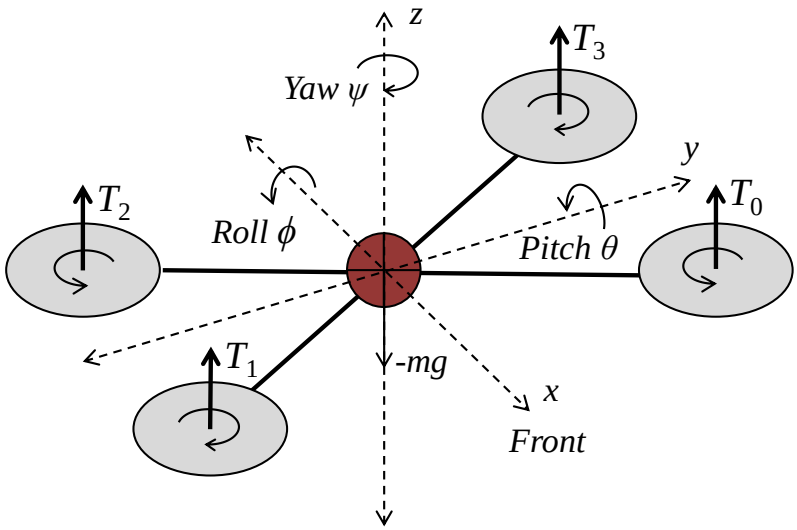


Figure 3.3: Quadcopter free body diagram.

3.4.1 Forces

Propellers produce a force perpendicular to their plane of rotation. It follows that this force is related proportionally to the propeller's angular velocity. It can be shown that a propeller's thrust can be approximated by equation **3.3** where T_i is a force vector for motor index i, where k is a scaling constant, and where ω is the angular velocity of the propeller [33].

$$T_i = k \cdot \omega^2 \tag{3.3}$$

The sum of thrust force vectors, T, is shown in equation **3.4** where $\bar{x}, \bar{y}, \bar{z}$ are Cartesian unit vectors.

$$T = \sum_{i=0}^{3} T_i = k \cdot \begin{bmatrix} 0 & 0 & \omega^2 \end{bmatrix} \cdot \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} \tag{3.4}$$

Note that all these forces are in the z direction. The implication of this is that the quadcopter cannot transverse a room's x, y plane without some rotation. This has aerial footage implications which is why some multi-rotor platforms use angled motors as seen here [34]. Another important note is that while a ESC is aware of motor speed, ESCs do not commonly send this information back to a controller. ESCs are simply circuits to which a PPM signal is sent to. Without taking external measurements, it is not possible to accurately correlate a PPM signal to propeller's velocity or thrust. To approximate the PPM signal to thrust relationship, the Marq Drone was put upside down on a weight scale and then varying PPM signals were sent to the motors. The results of this testing are shown in Fig. **3.4**.

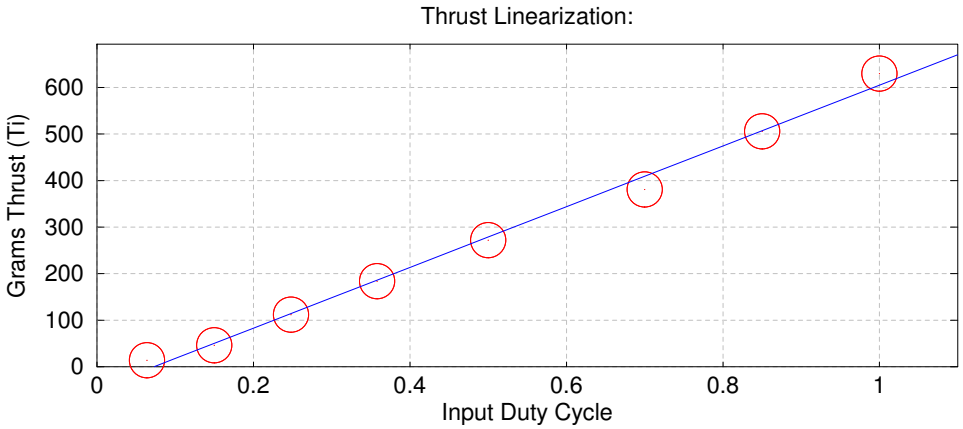


Figure 3.4: Grams thrust (T_i) per duty cycle input. Duty cycle input represents power percentage of a 3 cell lipo battery.

As observed in Fig. **3.4**, there was a roughly linear relationship between input duty cycle and thrust. Consequently, linear regression was used to approximate the thrust for a given motor in terms of the PPM duty cycle as shown in equation **3.5**.

$$\hat{T}_i = 652.82 \cdot PPM_i - 47.68. \tag{3.5}$$

Naturally, this equation is only applicable to this specific combination of propellers and motors used in the Marq Drone. This experiment would need to be performed again should the motors or propellers change.

Now that an estimate T_i has been formulated, it is possible to calculate the required thrust to keep the quadcopter in the air.

$$\begin{aligned} \sum_{i=0}^{3} \hat{T}_i \cdot \bar{z} &= m \cdot g \\ \bar{z} &= cos(\theta) \cdot cos(\phi) \\ \hat{T}_i &= \frac{m \cdot g}{4 \cdot cos(\theta) \cdot cos(\phi)} \end{aligned} \tag{3.6}$$

Assuming $\hat{T}_i$ is set equal to the result calculated in equation **3.6**, the quadcopter should roughly maintain its height regardless of minor rotations in θ, ϕ. However, it is important to note that thrust vectors T_i occur at some r_i distance away from the center of mass which results in torques acting on the frame.

3.4.2 Torques

From physics, we know that torque, τ, is equal to the cross product of distance and force. For control purposes, it is useful to calculate the torque for each of the quadcopters rotation axes as seen in equation **3.7**. In equation **3.7**, height (h) and length (l) are defined from Fig. **3.2**.

$$\begin{aligned} \tau &= r \times F \\ \tau_\phi &= r_{\bar{y}} \times T = (l/2) \cdot (T_0 + T_3 - T_1 - T_2) \\ \tau_\theta &= r_{\bar{x}} \times T = (h/2) \cdot (T_0 + T_1 - T_2 - T_3) \\ \tau_\psi &= a \cdot (T_0 + T_2 - T_1 - T_3). \end{aligned} \tag{3.7}$$

These equations are similar to what shown in [33] and adapted for an asymmetrical **X** configuration quadcopter. As observed in equation **3.7**, the pitch and roll torques (τ_θ, τ_ϕ) are defined with respect to the quadcopter's frame geometry. Had the quadcopter frame been symmetrical, l, h would have been equal which would have resulted in a simpler equation. Without taking this asymmetry into account at the control system level, the quadcopter's pitch and roll would react differently. In contrast to τ_θ, τ_ϕ, it is harder to define the torque about the yaw axis (τ_ψ) geometrically. This torque is slightly less intuitive because it arises from having spinning masses attached to the frame. Without taking this into account, the quadcopter would continuously spin about the yaw axis. However, this problem is not unique to just quadcopters. A helicopter, for example, uses a tail rotor in order to counteract the yaw torque introduced by the primary blade. Similarly, in order to counteract this torque on the quadcopter frame, two propellers are selected to spin clock wise and two

propellers are selected to spin counterclockwise. These alternating motor directions are illustrated in Fig. **3.3**. While geometry and force vectors have allowed for derivations of torque, these cannot be converted to angular rates without observing inertia.

3.5 Inertia and Angular Acceleration

Classical physics defines a relationship between torque and angular acceleration as shown in equation **3.8** where ω is angular velocity, where $\alpha, \dot{\omega}$ represent angular acceleration, and where I_{xyz} represents the inertia matrix.

$$\tau = I_{xyz} \cdot \alpha = I_{xyz} \cdot \dot{\omega} = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{YX} & I_{YY} & I_{YZ} \\ I_{ZX} & I_{ZY} & I_{ZZ} \end{bmatrix} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} \tag{3.8}$$

The angular accelerations can be solved for by equating equation **3.8** to equation **3.7**. Approximations can be used to simplify the inverting and calculation of the inertia tensor matrix. Note, if rotations are defined about an object's center of mass and the mass is symmetrically distributed, the inertia matrix is diagonal [35]. Furthermore, values of I_{xx}, I_{yy}, I_{zz} can be approximated by simplifying a 3D body into basic shapes such as rectangles. For a 3D rectangle, the inertia about an axis can be calculated as $I_{xx} = \frac{1}{12} \cdot m \cdot ((y_{length})^2 + (z_{length})^2)$. Similar equations would apply for I_{yy}, I_{zz}. Using this approximation, equation **3.9** was calculated for the quadcopter frame and is expressed in grams meter squared.

$$I_{xyz} = \begin{bmatrix} 0.0907 & 0 & 0 \\ 0 & 0.5005 & 0 \\ 0 & 0 & 0.5422 \end{bmatrix} \cdot g \cdot m^2 \tag{3.9}$$

This is an approximation since in reality, the mass will not be perfectly symmetrically distributed. Consequently, using this approximation and taking the inverse of the tensor matrix, the results are shown in equation **3.10** where k represents a given discrete time sample.

$$\begin{bmatrix} \ddot{\phi}(k) \\ \ddot{\theta}(k) \\ \ddot{\psi}(k) \end{bmatrix} = I_{xyz}^{-1} \cdot \begin{bmatrix} (l/2) \cdot (T_0(k) + T_1(k) - T_2(k) - T_3(k)) \\ (h/2) \cdot (T_0(k) + T_3(k) - T_1(k) - T_2(k)) \\ a \cdot (T_0(k) + T_2(k) - T_1(k) - T_3(k)) \end{bmatrix} \tag{3.10}$$

This equation can be used in the formation of a state space system. The state space system is approximated using a discrete derivative. Also, equation **3.6** can be divided by mass to produce an approximation of z. Consequently, a discrete state space representation of the four desired degrees of freedom is shown in equation **3.11** where ΔT represents the time of a fixed sampling period.

$$\begin{bmatrix} \dot{\phi}(k+1) \\ \dot{\theta}(k+1) \\ \dot{\psi}(k+1) \\ \dot{z}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\phi}(k) \\ \dot{\theta}(k) \\ \dot{\psi}(k) \\ \dot{z}(k) \end{bmatrix} + \begin{bmatrix} \frac{l}{2 \cdot I_{XX}} & \frac{l}{2 \cdot I_{XX}} & -\frac{l}{2 \cdot I_{XX}} & -\frac{l}{2 \cdot I_{XX}} \\ \frac{h}{2 \cdot I_{YY}} & \frac{-h}{2 \cdot I_{YY}} & \frac{-h}{2 \cdot I_{YY}} & \frac{h}{2 \cdot I_{YY}} \\ \frac{a}{I_{ZZ}} & \frac{-a}{I_{ZZ}} & \frac{a}{I_{ZZ}} & \frac{-a}{I_{ZZ}} \\ \frac{g}{4 \cdot cos(\theta) \cdot cos(\phi)} & \frac{g}{4 \cdot cos(\theta) \cdot cos(\phi)} & \frac{g}{4 \cdot cos(\theta) \cdot cos(\phi)} & \frac{g}{4 \cdot cos(\theta) \cdot cos(\phi)} \end{bmatrix} \cdot \begin{bmatrix} T_0(k) \\ T_1(k) \\ T_2(k) \\ T_3(k) \end{bmatrix} \cdot \Delta T \tag{3.11}$$

These equations will form the basis of the control system update laws. However, in order to apply them, the orientation of the quadcopter must be known.

CHAPTER 4

Flight Orientation Estimation

4.1 Introduction

In order to meet the control goals defined by equation **3.1**, estimation of the quadcopter's orientation is required. With modern advancements in electronics, determining orientation can be done inexpensively, efficiently, and quickly with micro electrical mechanical system (MEMS) based sensors. Unfortunately, currently, there is no single affordable MEMS sensor that directly measures θ, ϕ, ψ. Consequently, the combination of multiple MEMS sensors is required in order to accurately estimate orientation. This chapter overviews the operation of relevant MEMS sensors as well as their strengths and weaknesses with respect to multi-rotor aircraft. This chapter then establishes the minimum number of MEMS sensors needed to achieve the desired goals.

4.2 3 Axis Accelerometer

4.2.1 Introduction

3 axis accelerometers are devices that are designed to measure Cartesian $\bar{x}, \bar{y}, \bar{z}$ accelerations. It follows that the output of the device is an acceleration vector denoted as $A = [A_x, A_y, A_z]$. Fig. **4.1** is a macro scale illustration of how accelerometers measure acceleration.

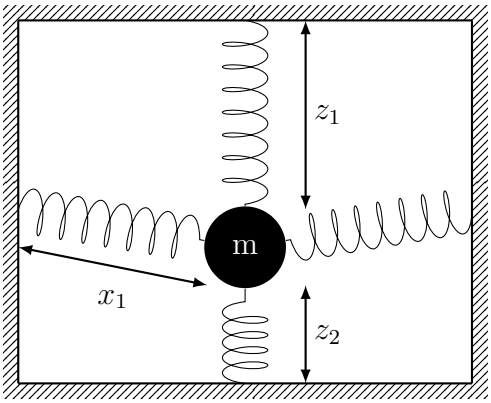


Figure 4.1: 2D macro scale spring accelerometer.

In Fig. **4.1** a mass is suspended by springs. Acceleration acting on the mass is determined by spring deflection. Due to gravity, an accelerometer will state that an object at rest is experiencing an acceleration approximately $9.8m/s^2$. This acceleration is represented in Fig. **4.1** with the to equation **4.1** can be derived for A_x and A_y. Also, note that in Fig. **4.1**, the x dimension springs are also experiencing deflection due to gravity. However, the magnitude of deflection for the x dimension springs are of equal magnitude. When using equation **4.1** for the x direction, the net acceleration would be 0.

4.2.2 3D MEM Accelerometer

deflection of the z springs. This gravity reference is crucial for steady state orientation estimation. Using these principles, an equation is formulated to calculate the acceleration per axis as shown in equation **4.1** specifically for the z axis.

$$\begin{aligned} F = m \cdot A, \quad F = k \cdot z \\ A_z = \frac{k \cdot (z_1 - z_2)}{2 \cdot m} \end{aligned} \tag{4.1}$$

Given one is using a 3D accelerometer, similar equations

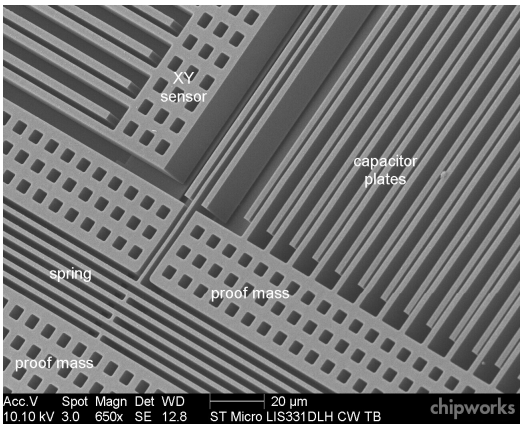


Figure 4.2: Accelerometer MEMs image [8].

In order to achieve low form factor, low power consumption, and low cost, modern accelerometers are designed using MEMS based technology. While they operate under the same principles illustrated in Fig. **4.1**, there is little visual similarity since the springs are often of micron size etched in a silicon based process. Fig. **4.2** is a picture of the internals of a modern MEMs accelerometer. These structures take a variety of shapes although they generally consist of spring loaded masses in combination with parallel plates. When the device is accelerating, this applies force to the internal springs of the MEMS structure. This force results in the distance between capacitive plates changing. The MEMS device measures the capacitance of the plates which is converted to a deflection distance. Using the deflection distance of the plates, the device can then calculate acceleration by taking into account the spring system's mass and a spring coefficient [36]. The accelerometer can then transmit this acceleration value through a serial bus or an analog signal.

4.2.3 Accelerometer Orientation Equations

To make use of accelerometer readings with respect to orientation, a conversion from Cartesian acceleration to Euler angles is required. It can be shown that equation **4.2** and equation **4.3** convert from Cartesian acceleration to Euler angles θ and ϕ [37].

$$\theta = atan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right) \cdot \frac{180}{\pi} \tag{4.2}$$

$$\phi = atan2\left(\frac{A_y}{A_z}\right) \cdot \frac{180}{\pi} \tag{4.3}$$

Note that in the case of ϕ in equation **4.3**, the $atan2$ function is used which is a four quadrant version of the traditional arc tangent function. Also note that there is no equation for the yaw (ψ) orientation angle. This is because a ψ orientation will not register as any measurable change in the Cartesian acceleration vector. This is a fundamental physical limitation of the 3 axis accelerometer.

4.2.4 Accelerometer Drawbacks

1. Incomplete Orientation Assessment

While vectors contain magnitude and directional information, they do not intrinsically have a rotational component. It follows that the yaw(ψ) of a given object cannot be determined with a 3 axis accelerometer. Consequently, the sole use of an accelerometer is not enough for the control goals **3.1** to be achieved.

Without yaw feedback, any unbalanced ψ torques about the quad-copter would result in the aircraft spinning in air. Naturally, this would make navigation difficult and it follows that the accelerometer is not generally used as a stand-alone orientation sensor for multi-rotor aircraft.

2. Sensitive to High Frequency Vibrations

Another noteworthy issue with accelerometers is that they are sensitive to high frequency vibrations. While this is not an issue for some applications, quadcopters often experience a high level of vibration due to their motors spinning at high speeds. The effect that this can have is illustrated in Fig. **4.3**.

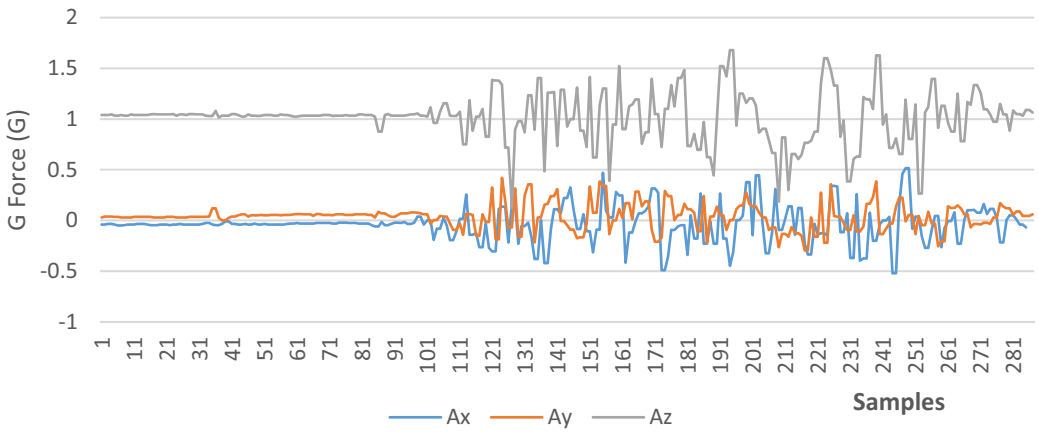


Figure 4.3: Accelerometer noise before and during motor operation.

As seen in Fig. **4.3**, when at rest, there is negligible noise on the accelerometer readings. However, when the quadcopter's motors are turned on, a significant amount of high frequency noise is registered on the sensor. While low pass filtering would remove this noise, the delay of such a filter can lead to flight instability.

3. External Acceleration Produces Unknown Orientation

Accelerometers measure a gravity reference when an object at rest. However, the accuracy of this reference can be lost if the quadcopter is accelerating. For example, if the quadcopter in Fig. **3.3** is moving upwards at a rate twice that of gravity, the accelerometer would yield $A_z = -1g, A_x = 0g, A_y = 0g$. Using this data, equation **4.3** would state that the quadcopter is upside-down. Consequently, equation **4.3** is only valid and useful for orientation when the quadcopter is not accelerating.

4.3 3 Axis Gryoscope

4.3.1 Introduction

In contrast to the accelerometers, gyroscopes measure rotation. For decades, gyroscopes have been used as an integral part in naval and aerospace navigational systems. The first gyroscopes were generally spinning masses held in a gimbal frame. These frames allowed the mass to spin freely and maintain its axis of rotation regardless of the orientation of the external frame. Due to conservation of angular momentum, the spinning mass would resist any changes to its rotation. This structure is visualized in Fig. **4.4** .

To electronically measure changes in rotation using a classical gyroscope, electrical potentiometers are connected to the gimbal frames. While these devices are capable of a large degree of accuracy, they are large, heavy, and expensive.

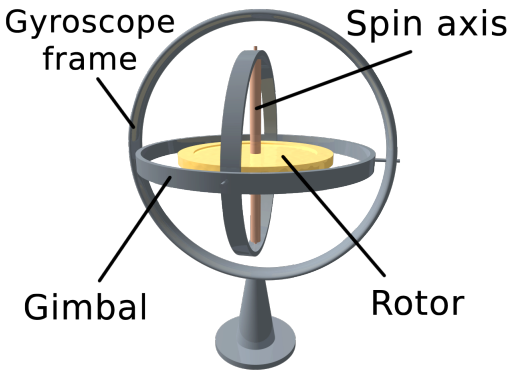


Figure 4.4: Classical gyroscope or gyrostat.

4.3.2 MEMS Gyro

A MEMS version of the gyroscope is illustrated in Fig. **4.5**. Due to the silicon design process, they are small, low power, and cost effective.

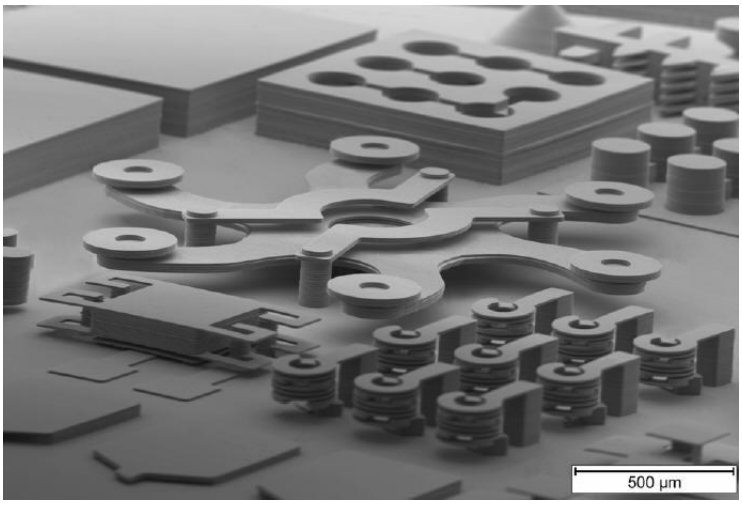


Figure 4.5: MEMS 3 Axis Gyroscope [9].

Unlike the macro version shown in Fig. **4.4**, MEMS gyroscopes do not contain a spinning disk. Owing to the fact that it is difficult to achieve a large angular momentum in a microscopic space, MEMS gyroscopes operate on a different physical principle. This physics principle is called the "coriolis effect" [38]. Consequently, these types of gyros are called "Coriolis vibratory gyroscopes". As

the name suggests, these contain vibrating structures. By measuring forces exerted on these vibrating structures, velocity of rotation can be determined [39]. It is important to note that MEMS gyroscopes measure angular velocity in contrast to macro-scale gyroscopes that measure orientation angles directly. To estimate orientation angles, it follows that the MEMS gyroscope readings must be integrated as shown in equation 4.4.

$$\theta = \int_0^t \dot{\theta} \cdot dt \qquad \phi = \int_0^t \dot{\phi} \cdot dt \qquad \psi = \int_0^t \dot{\psi} \cdot dt \tag{4.4}$$

4.3.3 MEMS Gyro Drift

As a result of integrating the readings MEMS gyroscope over time, steady state orientation error accumulates. This accumulated error is commonly called gyro drift. As a result, a MEMS gyroscope must continuously be re-calibrated or must be high pass filtered. Consequently, MEMS gyroscopes are often used in combination with other sensors.

4.4 3 Axis MEMS Magnetometer

4.4.1 Introduction

Magnetometers are devices that measure magnetic field. The most common and oldest form of a magnetometer is a compass. Compasses, which point in the direction of earth's magnetic north, have been used in navigation for hundreds of

years. For modern low power electronics, a MEMS solution also exists. An example of a MEMS magnetometer is shown in Fig. **4.6**.

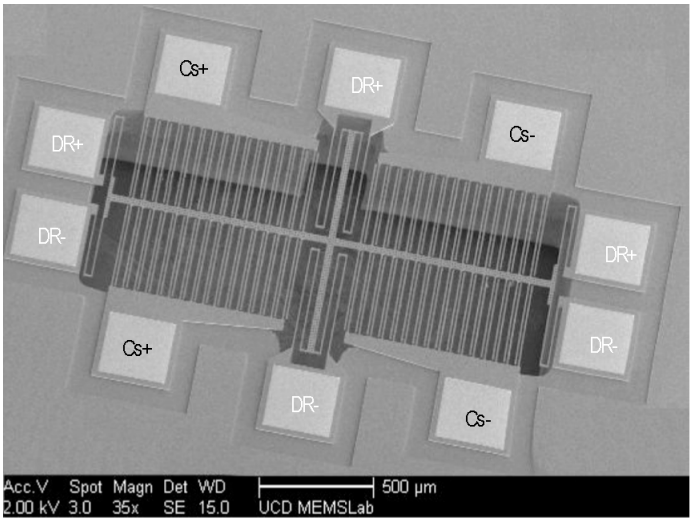


Figure 4.6: MEMS 3 Axis Magnetometer [10].

Common MEMS magnetometers function by making use of the Lorentz force in order to measure magnetic field [10]. This force is then converted to a electronic signal similarly to how an accelerometer measures acceleration. This force creates deflection on a spring system which is sensed by parallel capacitive plates. Primarily, these sensors are used to measure the earth's magnetic field which gives orientation insight. When these devices are used in conjunction with an accelerometer, steady state estimation accuracy of θ, ϕ, ψ can be achieved. While these devices can provide useful insight into orientation, they also have their own drawbacks.

4.4.2 Hard-Iron and Soft-Iron Interference

A magnetometer's proximity to both ferrous and non ferrous metal is a large source of error. These errors are called hard-iron and soft-iron interference for

ferrous and non ferrous materials respectively [40]. Electronic assemblies intrinsically contain both types of material and consequently are susceptible to both as soon as a magnetometer is bonded to a PCB. Unfortunately, since the earth's magnetic field is weak, these errors can dominate the output readings of a magnetometer.

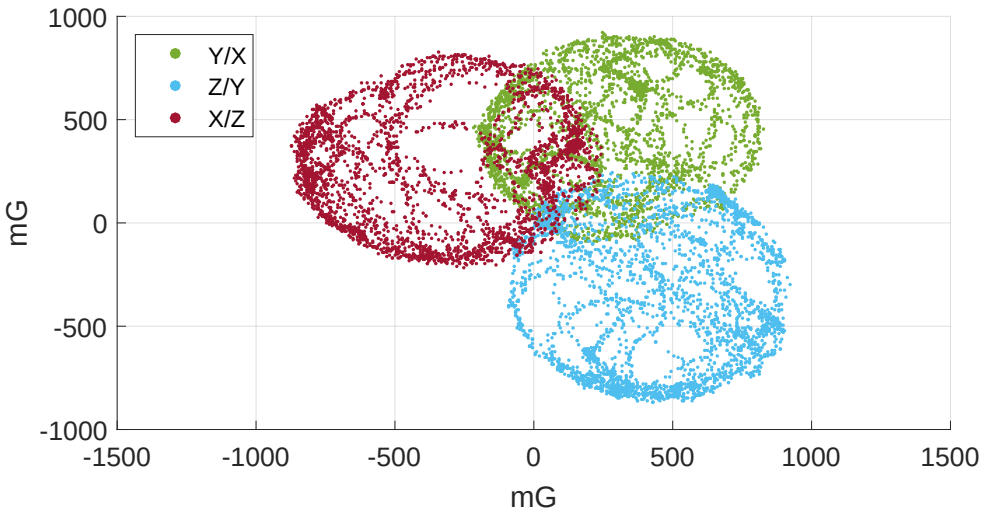


Figure 4.7: Milli-Gauss (mG) output of an uncalibrated magnetometer when rotated in 3 dimensions.

Consequently, for magnetometer readings to be valid, a calibration must be done after a magnetometer has been bonded to the PCB. The uncalibrated magnetometer readings observed on the quadcopter is shown in Fig. **4.7**. For practical purposes, a simple offset and scaling calibration can be performed. A method of performing this scaling is shown in equation **4.8** where M_x, M_y, M_z represents the milli-Gauss values on their respective axises.

$$\hat{M}_x = 2 \cdot \frac{M_x - min(M_x)}{max(M_x) - min(M_x)} - 1$$
$$\hat{M}_y = 2 \cdot \frac{M_y - min(M_y)}{max(M_y) - min(M_y)} - 1 \tag{4.5}$$
$$\hat{M}_z = 2 \cdot \frac{M_z - min(M_z)}{max(M_z) - min(M_z)} - 1$$

Note that this calibration procedure should only be performed during a window of time in which the quadcopter is being rotated about all its axes. If a windowed calibration is not performed, eventually the min and max values will be corrupted by external fields and will permanently skew the values of M_x, M_y, M_z. This calibration procedure was performed on the quadcopter's IMU and the results can be seen in Fig. **4.8**.

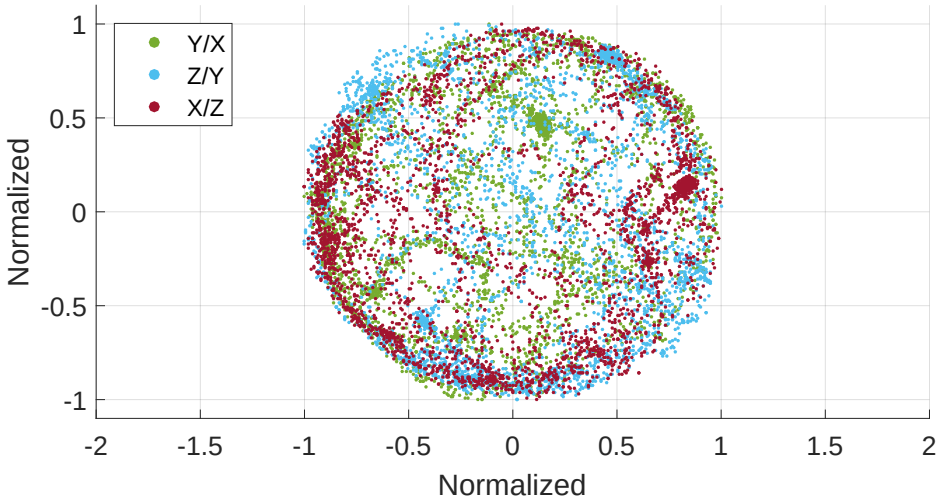


Figure 4.8: Output of a calibrated and normalized magnetometer when rotated in 3 dimensions.

As seen in Fig. **4.8**, the magnetometer data is normalized and no longer has significant scaling or offset variation between the axes. While the result of this normalization is unit-less, this is of no consequence since future calculations are done using ratios. For example, assuming the quadcopter is level ($\theta = 0, \phi = 0$),

yaw (ψ) can be calculated with the basic equation shown in equation **4.6**.

$$\psi = atan2(\frac{\hat{M}_y}{\hat{M}_x}) \tag{4.6}$$

Note this equation is no longer valid if the magnetometer is not level since it ignores the M_z. Consequently, if a magnetometer is titled, additional calculations are required.

4.4.3 Magnetometer Tilt Compensation

Magnetometer accuracy can be improved by performing tilt compensation based on observed θ, ϕ. The end goal of tilt compensation is to make equation **4.6** valid by de-rotating the axes by θ, ϕ. Using rotation matrices along with equation **4.6**, equation **4.7** is derived [41].

$$\hat{\psi}_C = tan^-1\left(\frac{\hat{M}_z \cdot sin(\phi) - \hat{M}_y \cdot cos(\phi)}{\hat{M}_x \cdot cos(\theta) + \hat{M}_y \cdot sin(\theta)sin(\phi) + \hat{M}_z \cdot sin(\theta)cos(\phi)}\right) \tag{4.7}$$

For this equation to work properly, it is imperative that all IMU outputs match the axis as shown in Fig. **3.1**. This equation was validated with IMU data and a comparison between compensated and uncompensated yaw is shown in Fig. **4.9**.

4.4.4 Magnetometer Drawbacks

Susceptible to external fields

As mentioned previously, the earth's magnetic field is extremely weak. Consequently, external ferrous material can significantly skew orientation readings if in close proximity to the sensor. A method that was implemented to

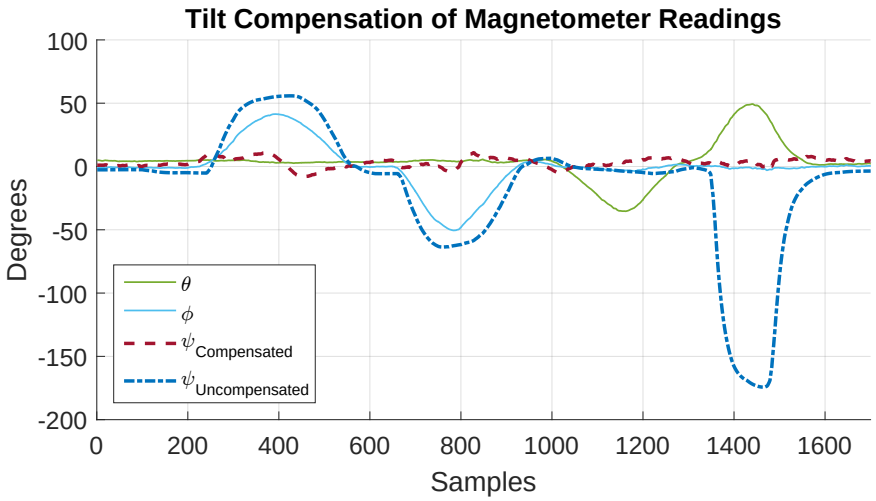


Figure 4.9: Plot demonstrating successful tilt compensation of magnetometer readings. Here θ and ϕ are actuated without significant change in ψ_C.

combat this is to ignore the magnetometer data if the data observed is significantly beyond the normalization in Fig. **4.8**.

Global Location Dependent

Earth's magnetic field direction and strength vary across the surface of the earth. The earth's magnetic field vector points towards magnetic north as opposed to the true geographical north. The angular difference between these two locations is defined as magnetic declination. In addition, the earth's magnetic field has a varying vertical component. The angular difference between the surface of the earth and the earth's magnetic field is defined as magnetic inclination. This is illustrated in Fig. **4.10** where σ is magnetic declination and where δ is magnetic inclination.

Geographical variations in magnetic intensity, inclination, and declination are shown in Fig. **4.11**.

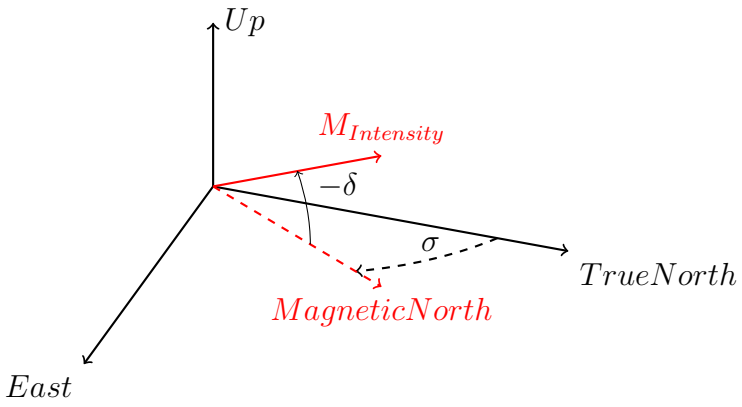
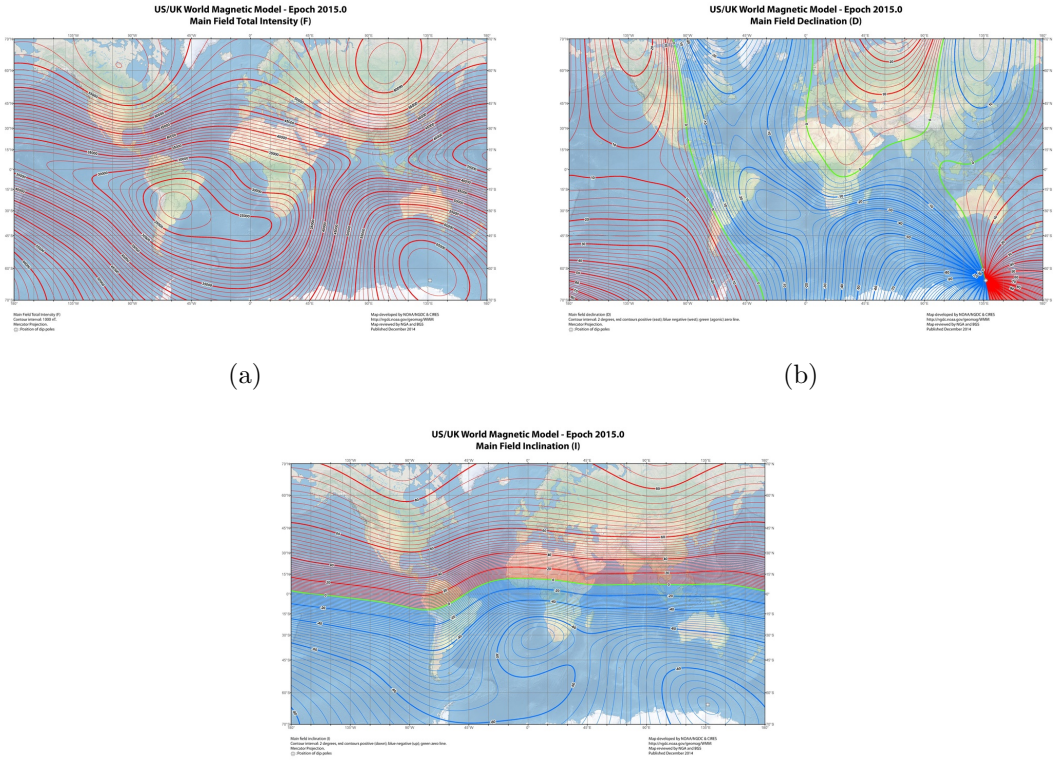


Figure 4.10: Demonstration of magnetic intensity (M), magnetic inclination (δ), and magnetic declination (σ).



(a) (b)

(c)

Figure 4.11: Global variation in: (a) magnetic intensity, (b) magnetic declination, (c) magnetic inclination [11].

These variations have implications for quadcopter flight. Variations in field strength would affect the results of the calibration procedure in equation **4.8**. Consequently, the quadcopter should be re-calibrated per continent of operation. Variations in magnetic declination do not actively affect user operated flight.

However, absolute heading error can have implications in autonomous flight. However, if the quadcopter's location is known, the map in Fig. **4.11**(b) can be used for correction. Finally, magnetic inclination can be accounted for likewise using the map in Fig. **4.11**(c). Another method of compensating for magnetic inclination is to use the equation in equation **4.7** which effectively ignores the z component of the magnetic field.

4.5 Sensor Fusion

4.5.1 Introduction

The process of combining data from multiple sensors and coming up with a collective estimate is commonly called sensor fusion. For orientation, there are multiple different sensor fusion algorithms such as gradient descent and Kalman filtering [42]. However, due to the high frequency vibrations, large system accelerations, low computational power available, and fast control loop update requirements, not all algorithms are ideal for quadcopter flight use. With this in mind, a simple complementary filter was selected. This work was inspired by [43] and is shown in Fig. **4.12** where subscript k represents the sample number.

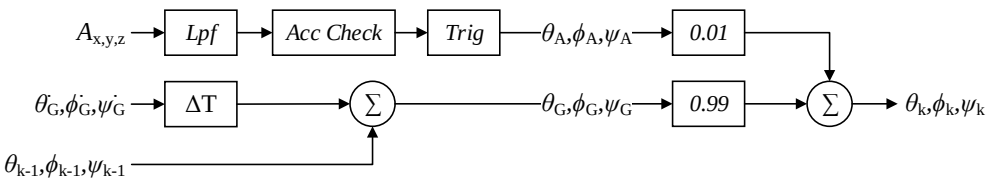


Figure 4.12: Complementary filter acting on raw accelerometer and gyro inputs.

A block diagram representation of a basic complementary filter is shown in Fig.

4.12 where $A_{x,y,z}$ is the raw accelerometer data and $G_{\dot{\theta},\dot{\phi},\dot{\psi}}$ is the raw gyroscope data. This filter addresses weaknesses such as what was discussed in 4.2.4 and 4.3.3. While gyroscopes have good dynamic response and noise immunity, they have long term drift. In contrast, accelerometers have poor dynamic response but are not susceptible to drift in the same manner. Consequently, a high pass filter is used on the gyroscope data and a low pass filter is used on the accelerometer data. However, before the data is combined, it is important to verify validity of the accelerometer data as mentioned in 4.2.4. Also, before combining the gyroscope and accelerometer data, the accelerometer's data must be converted to orientation angles using equation **4.2**.

However, a weakness of this filter is that it does not incorporate a magnetometer and would consequently have steady state ψ error. To counteract this, a magnetometer can be added using equation **4.8**. This data can be fused with the gyroscope reading of ψ in the same complementary fashion as to how the accelerometer and gyro data were fused. A block diagram form of this filter is shown in Fig. **4.13** where $M_{x,y,z}$ represents the magnetometer readings scaled by equation **4.8**.

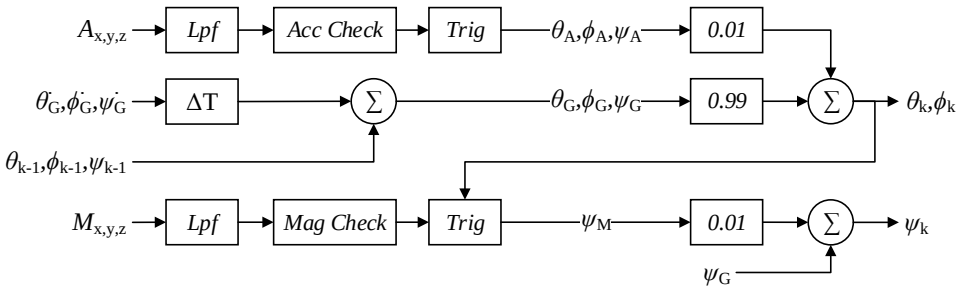


Figure 4.13: Complementary filter with magnetometer integration.

This diagram can be easily implemented in C code. Due to the filtering and trigonometry operations, it is highly recommended that this filter be used on a micro-controller with a hardware floating point unit. Pseudo C code of the proposed filter is shown in Fig. **4.14**.

```
#include <math.h>
#define DeltaT 0.002 // Filter is run every 2ms.
void IMU_Comp_Filter(){
  float Accel_Mag, Pitch_Accel, Roll_Accel,Num,Den,Yaw_Mag; //temp vars

  Ax = Ax + 0.1*(Ax_Raw - Ax); // Acceleration Low Pass Filter
  Ay = Ay + 0.1*(Ay_Raw - Ay);
  Az = Az + 0.1*(Az_Raw - Az);

  Pitch+= Gx * DeltaT; // Apply Gyro Integral
  Roll+= Gy * DeltaT;
  Yaw+= Gz * DeltaT;

  Accel_Mag = fabsf(Ax) + fabsf(Ay) + fabsf(Az); //Should be near 1g
  if (fabsf(Accel_Mag-1)<0.2) //If not experiencing external acceleration
  {
    Pitch_Accel = (180/PI) * atan(-Ay,Az); // Apply Trig
    Roll_Accel = (180/PI) * atan2(Ax,sqrt(Ay*Ay + Az*Az));

    Pitch = 0.99*Pitch + 0.01*Pitch_Accel; //Combine Gyro & Accel Data
    Roll = 0.99*Pitch + 0.01*Roll_Accel; // Fuse gyro and accelerometer
  } // End of typical complementary filter

  Mx = Mx + 0.1*(Mx_Raw - Mx); // Magnetometer Low Pass Filter
  My = My + 0.1*(My_Raw - My);
  Mz = Mz + 0.1*(Mz_Raw - Mz);

  Mag_Mag = fabsf(Mx) + fabsf(My) + fabsf(Mz); //Should be near 1
  if(fabsf((Mag_Mag-1)<0.2) //If not experiencing external fields
  {
    Num = Mz*sin(Roll) - My*cos(Roll);
    Den = Mx*cos(Theta)+My*sin(Pitch)*sin(Roll)+Bz*sin(Pitch)*cos(Roll);
    Yaw_Mag = (180/PI) * atan2(num,den); // Apply Trig
    Yaw = 0.99*Yaw + 0.1*Yaw_Mag; // Fuse gyro and magnetometer
  }
}
```

Figure 4.14: Code snippet of 9DOF fusion filter.

4.6 Orientation Estimation Conclusion

The proposed IMU fusion filter is a computationally efficient and simple filter that provides complete orientation assessment of a quadcopter. The filter was validated using real world test data by placing the drone on a measured rotary bed. Data was then simultaneously collected from the IMU as well as from potentiometers attached to the bed itself. The results of the testing are shown in Fig. **4.15**, Fig. **4.16** and Fig. **4.17**.

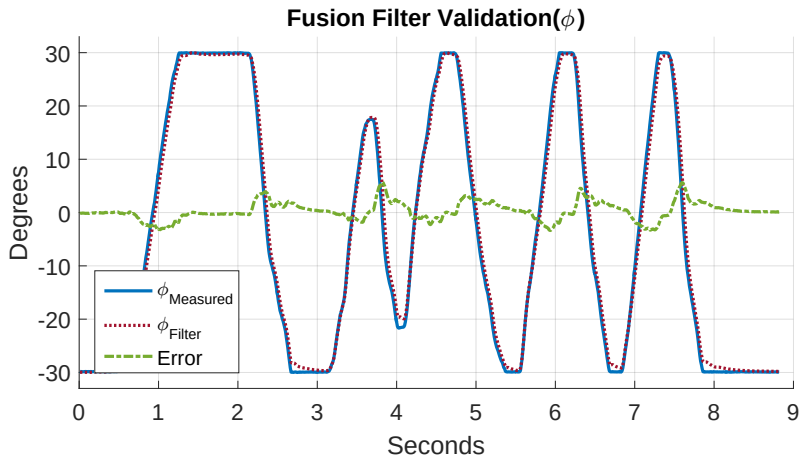


Figure 4.15: Complementary filter validation test measuring and computing roll.

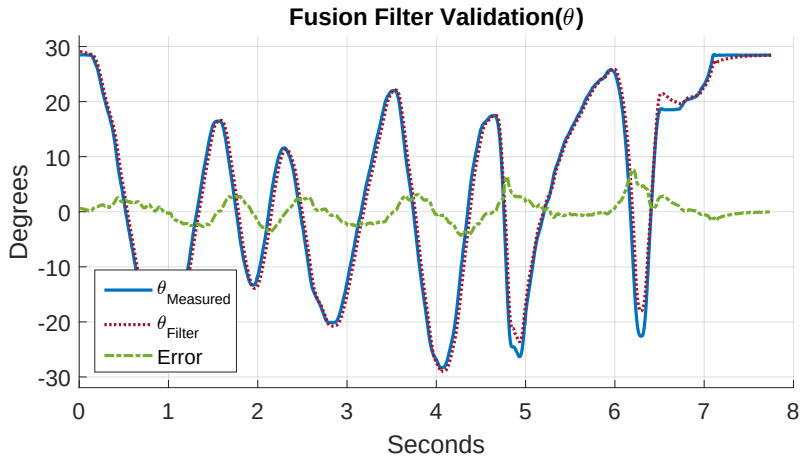


Figure 4.16: Complementary filter validation test measuring and computing pitch.

As observed in validation plots, the steady state error is negligible and the

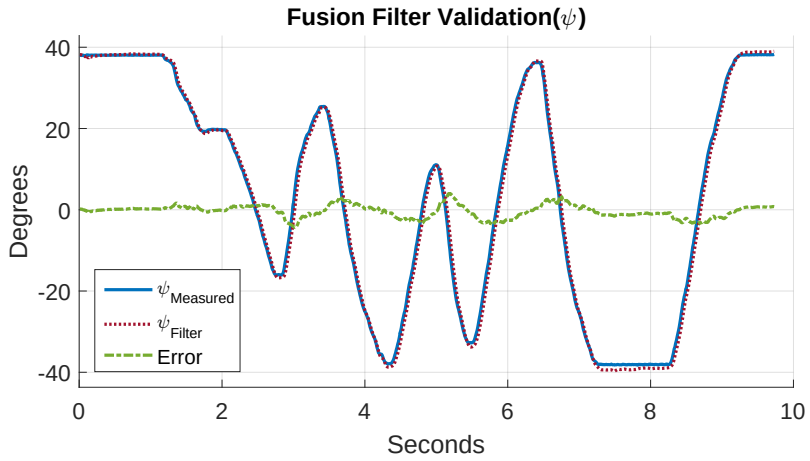


Figure 4.17: Complementary filter validation test measuring and computing yaw.

transient error is minimal. A weakness of this filter is that it relies on filter coefficients that must be tunned on a case by case basis. Another weakness of this filter is that it is using Euler angles (θ, ϕ, ψ). Euler rotation angles become undefined in edge cases such as $\theta = 90$. Consequently, this filter implementation is not designed for acrobatic maneuvers. A strength of this filter is that is it is easy to understand and implement. In addition, with added data checks for valid accelerometer and magnetometer information, this filter is ideal for use on multi-rotor aerial vehicles.

CHAPTER 5

Control System Implementation

5.1 Introduction

This chapter will overview the control system used to achieve stable flight and meet the control objectives listed in **3.1**. This control system was needed in order to handle real world disturbances and to account for unknown offsets. In order to drive the quadcopter's orientation (θ, ϕ, ψ) to desired values, a series of proportional integral derivative (PID) controllers were implemented.

5.2 PID Basics

A PID controller is a control loop that updates based upon the error observed between a desired output and the measured output. This control loop's response is characterized by coefficients K_p, K_i, K_d and is shown in equation **5.1** where e represents error and $u(t)$ represents the PID output [44].

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{5.1}$$

The magnitude of coefficients K_p, K_i, K_d determines how responsive the controller is to proportional, integral, and derivative errors respectively. Control loops such as PID act on systems to stabilize them if needed. In addition, a PID

loop can improve the transient response of a system. Systems themselves in controls context are often referred to as a "plant". In the case of this thesis, the plant is the quadcopter system. This plant along with control is visualized in Fig. **5.1**, where $u(t)$ is the input to the plant, where $r(t)$ is the desired set point of the plant, and where $e(t)$ is the error between the output of the plant and the desired set point.

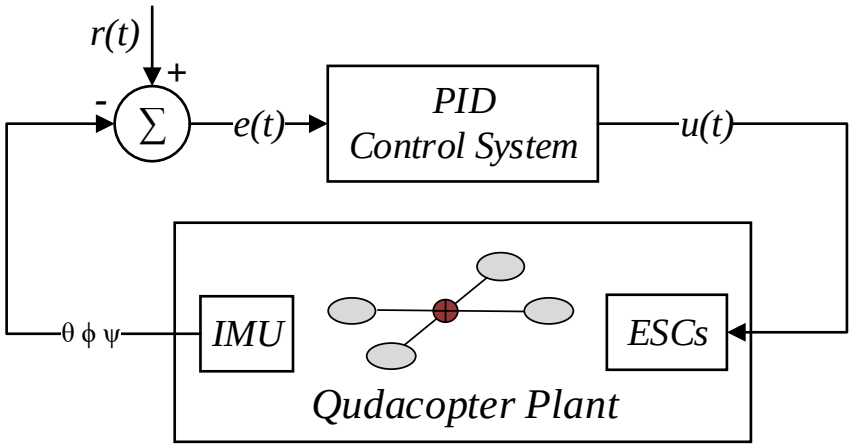


Figure 5.1: Diagram of quadcopter plant and control system.

The input to the plant, $u(t)$, represents the speed commanded to each of the quadcopter's motors. For simplicity, $u(t)$ will be redefined to vector T where $T = [T_0, T_1, T_2, T_3]$. T_0, T_1, T_2, T_3 represent the four individual motor thrust vectors. $r(t)$, the desired quadcopter outputs, is the user commanded orientation angles θ_c, ϕ_c, ψ_c as well as throttle T_c. Recall that these outputs were previously defined in equation **3.1**. Note that this system is linearized at a current point in time and consequently assumes that superposition is upheld. It follows that the input to the system, u(t), is calculated as a summation of the individual control components. This control concept is shown in Fig. **5.2** and the calculations for

each block are demonstrated in section 5.3.

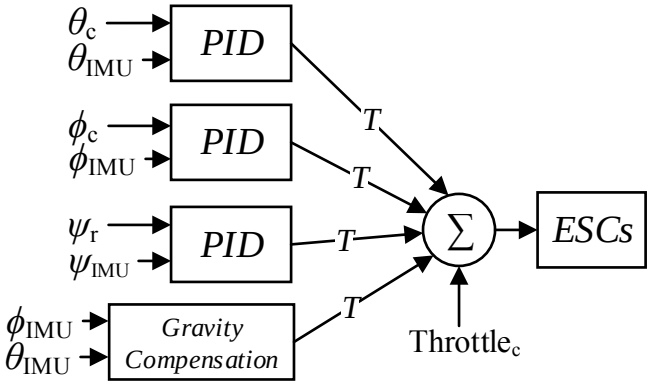


Figure 5.2: Expanded view of quadcopter's control system.

With the input and output relationship established, the following section overviews the PID code developed for the system as well as how T is updated such that the control goals are achieved.

5.3 PID Software Implementation

The first step in implementing the control shown Fig. **5.2** was to write a simple PID controller in software as shown in Fig. **5.4**.

This code provides a PID output per a given input, target, and PID parameter structure. This PID block can then be used to act on pitch, roll, and yaw.

With these control outputs, we can now finally use the model developed in equation **3.11** which relates how θ, ψ, ϕ relate to motor speeds T. This model determines how to update motor speeds T respectively. To illustrate this, Fig. **5.5** shows how equation **3.11** was applied in code to update T based on a given control error. Note, apart from simulation, a motivation for this derivation is

```
#define I_THR 200 // Max threshold value for integral
#define DeltaT 0.01 // Filter is run every 10ms.
float PID_Output(float Input, float Target, PID_STRUCT * PID){
  float Error = Target-Input; // Calcualte Setpoint Error
  float P = (PID->KP) * Error; // Proportional
  float I = (PID->KI) * Error * DeltaT + PID->Integral_Error; //
    Integral
  float D = (PID->KD) * (Error- PID->Prev_Error) * DeltaT;//
    Derivative
  PID->Prev_Error = Error; //For next derivative term
  PID->Integral_Error += (PID->KI) * Error * DeltaT; //For next I
    term
  if(fabsf( I ) > I_THR){ // Integral windup prevention
    PID->Integral_Error = (PID->Integral_Error>0) ? I_THR : -I_THR
    ;
    I = (PID->Prev_Error>0) ? I_THR : -I_THR; // Limit I to +-
    I_THR
  }
  return (P+I+D); // Return PID result
}
```

Figure 5.3: Code snippet of PID controller.

```
void  Get_Control_Errors(){
  cntrlRoll = PID_Output(IMU_Roll, Remote_Roll, &Roll_PID);
  cntrlPitch = PID_Output(IMU_Pitch, Remote_Pitch, &Pitch_PID);
  cntrlYaw = PID_Output(IMU_Yaw, Remote_Yaw, &Yaw_PID);
}
```

Figure 5.4: Code snippet for calculating PID results.

such that pitch and roll perform similarly. Consequently, constants A, B are formed for pitch and roll respectively. For normalization, pitch and roll were divided by A such that their response would be roughly equal. Yaw rate is due to different inertia physics and thus was not scaled by A.

The final control variable is height(z) which is controlled by the user. Consequently, as far as the controller is concerned, this is open loop control. However, to simplify user control, aspects of equation **3.11** can still be taken into account. Specifically when the quadcopter has a non-zero roll or pitch, more collective thrust is needed in order to ensure the quadcopter maintains its height. The code designed to achieve this throttle control is demonstrated in Fig. **5.6**.

```
#define A (H * DeltaT / (2*IXX)) // Constant acting on pitch
#define B (L * DeltaT / (2*IYY)) // Constant acting on roll
void Apply_Pitch(){
  T[0] += (1/A) * cntrlPitch;
  T[1] += (1/A) * cntrlPitch;
  T[2] -= (1/A) * cntrlPitch;
  T[3] -= (1/A) * cntrlPitch;
}

void Apply_Roll(){
  T[0] += (1/B) * cntrlRoll;
  T[1] -= (1/B) * cntrlRoll;
  T[2] -= (1/B) * cntrlRoll;
  T[3] += (1/B) * cntrlRoll;
}

void Apply_Yaw(){
  T[0] +=cntrlYaw;
  T[1] -=cntrlYaw;
  T[2] +=cntrlYaw;
  T[3] -=cntrlYaw;
}
```

Figure 5.5: Code snippet of pitch, roll, and yaw thrust calculations.

With the implementation of the PID system and update laws, these segments of code can be combined to formulate the overall control system. Note that to achieve a stable quadcopter, it is recommended to update motor speeds at least at a rate of 100 Hz. This control loop code is illustrated in Fig. **5.7**.

5.4 GUI and PID Tuning

Optimal selection of PID coefficients is a common problem when designing a PID control system. Given one has an accurate physical model, the optimal PID coefficients can be selected based upon desired overshoot and speed for example. However, without a physical model, values can be tunned on the fly. Initially, quadcopter parameters to achieve flight were tunned on the fly. A javascript GUI was written to help for debug purposes. This GUI had the capacity to tune the PID parameters of the quadcopter on the fly. A snapshot of this GUI is shown in

```
#include <math.h>

void Apply_Gravity_Compensation(){// additional thrust if titling
  float MG_Factor = (MASS * GRAVITY / 4) * cos(IMU_Pitch) * cos(
    IMU_Roll);
  T[0] +=MG_Factor;
  T[1] +=MG_Factor;
  T[2] +=MG_Factor;
  T[3] +=MG_Factor;
}

void Apply_Throttle(){ // Take joystick value of throttle and
    apply it to all motors
  T[0] +=Remote_Throttle;
  T[1] +=Remote_Throttle;
  T[2] +=Remote_Throttle;
  T[3] +=Remote_Throttle;
}

#define MOTOR_SPEED_MAX 1000.0 // Max speed normalized to 1000.
#define MOTOR_SPEED_MIN 100.0

void Apply_Throttle_Limits(){ // Ensure motor values are not
    beyond their min/max values
  uint16_t i = 0;
  for(i=0; i<4; i++)
  {
    if(T[i]>(MOTOR_SPEED_MAX))
      T[i] = MOTOR_SPEED_MAX;
    else if(T[i]<(MOTOR_SPEED_MIN))
      T[i] = MOTOR_SPEED_MIN;
  }
}
```

Figure 5.6: Code snippet of open loop throttle control.

```

void Run_Control_Loop()
{
  Get_Control_Errors();
  Apply_Throttle();
  Apply_Gravity_Compensation();
  Apply_Pitch();
  Apply_Roll();
  Apply_Yaw();
  Apply_Throttle_Limits();
}
```

Figure 5.7: Code snippet of control loop.

Fig. **5.8**.

This GUI was useful for collecting and saving live flight data, testing motor

Figure 5.8: Quadcopter javascript GUI.

orientation, and adjusting PID values. Being able to update and visualize data on the fly greatly simplified PID tuning and general development.

5.5 Software Architecture

The control loop in Fig. **5.7** along with the IMU code in Fig. **4.14** account for the majority of flight critical code running on the quadcopter. An overall software architecture is shown in Fig. **5.9**.

To keep control timing and IMU collection constant, the software was implemented in three separate threads. These threads primarily managed IMU filter updates, control system updates, and radio link updates. For this quadcopter software implementation, IMU data is received on a consistent

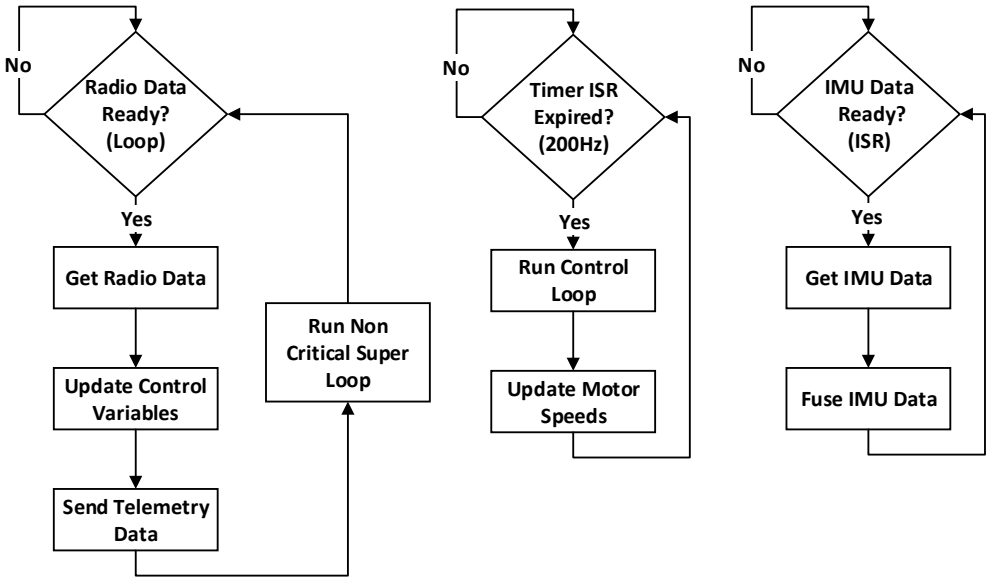


Figure 5.9: High level quadcopter software flow chart.

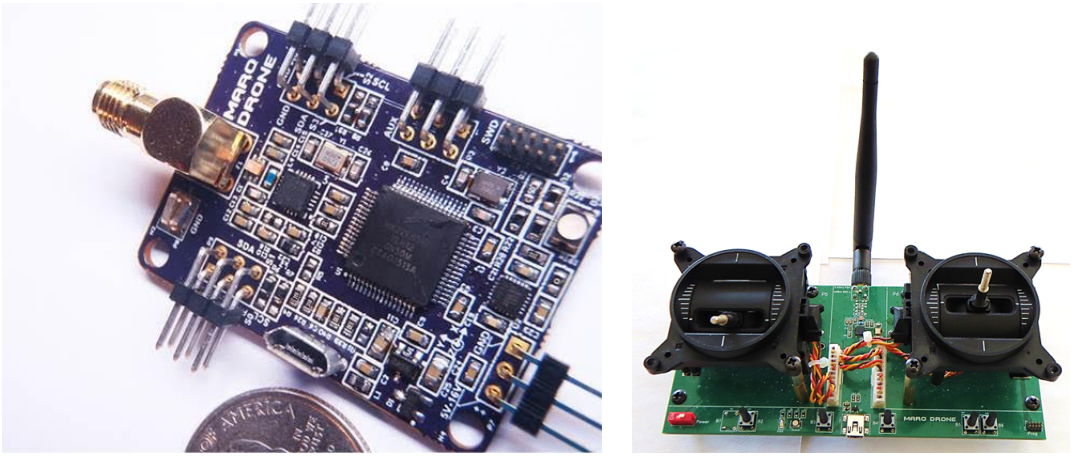
interrupt basis. Gyro and accelerometer readings are updated and filtered at 500 Hz. The magnetometer is updated at 100 Hz which is the maximum for the selected device. These updates are performed based on an interrupt to give the IMU filter constant timing characteristics. Likewise, the control system is also triggered through interrupts. The control system executes every 200 Hz and uses the most up to date data from both the controller and the IMU filter to perform its calculations. Less timing critical tasks are then executed in a super loop such as checking for new radio data, transmitting telemetry data, handling USB commands, and other non flight related features. Full source code is provided at [45].

CHAPTER 6

Radio Frequency System Implementation

6.1 Introduction

This chapter overviews the radio frequency (RF) system used with the Marq Drone and explains concepts regarding its design. Radio frequency design at ultra high frequency (UHF) is challenging which is why a chapter is devoted to it. This chapter will overview a minimal amount of information required to design a legal and efficient RF link. The RF hardware designed for the Marq Drone is shown in Fig. **6.1**.



(a) (b)

Figure 6.1: RF Marq Drone hardware: (a) Flight controller, (b) Radio controller

The Marq Drone utilizes a flight controller with an integrated radio as well as a remote controller with an integrated radio. The primary motivation behind this was to eliminate use of the PPM protocol as well as have the capacity to collect data in real time. This hardware was designed to help understand the nature of

the underlying system as well to be able to implement a more flexible architecture as shown previously in Fig. **2.5**. The schematics of these pieces are included in Appendix A. The CC2500 transceiver was selected for this RF system and its selection is explained in latter sections. A RF block diagram of this system is presented in Fig. **6.2**.

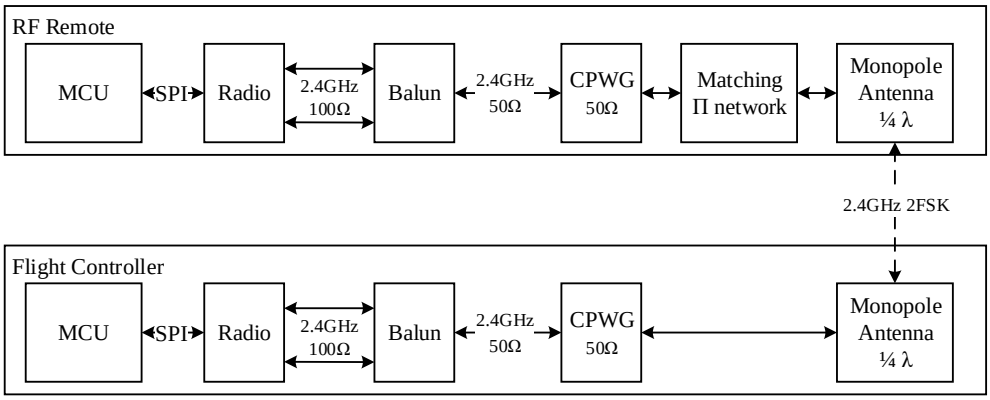


Figure 6.2: Block diagram of Marquette Drone RF system

These RF blocks will be overviewed in the following sections. Before overviewing these blocks, basic design considerations are addressed such as frequency of operation and range calculations.

6.2 Selecting a Frequency Band

Frequency of operation is the principal attribute of any RF system. The frequency determines the size of the radio system, the required power of the RF system, and the legality of a system. Firstly, one must select a radio frequency that is legal to operate in their region. For many countries, there exists Industrial, Scientific, and Medical (ISM) bands of operation where RF

transmission does not require a license. For the United States, ISM operation is covered in FCC Title 47 Part 15 [46] and is shown in Table 6.1.

Fundamental frequency (MHz):	**Field strength of fundamental (mV/m)**	**Field strength of harmonics (mV/m)**
902-928	500	1.6
2435-2465	500	1.6
5785-5815	500	1.6
10500-10550	2500	25.0
24075-24175	2500	25.0

Table 6.1: Title 47 part 15.245 ISM Bands.

The 2.4 GHz band is widely accepted in most countries and has become popular for consumer electronics. Consequently, many affordable RF devices now exist for this band of operation. An advantage of this band is the size of its quarter wavelength. The calculations for the 2.4 GHz quarter wavelength are shown in equation **B.2** where λ is wavelength, where c is the speed of light, and where f_o is the frequency of operation.

$$\begin{aligned} \frac{\lambda}{4} &= \frac{c}{4 \cdot f_o} \\ \frac{2.99 \cdot 10^8 m/s}{4 \cdot 2.4 GHz} &\approx 3.12 cm \approx 1.23 inches \end{aligned} \tag{6.1}$$

Having a quarter wavelength of roughly 1.23 inches allows for the antenna to be easily mounted to both the drone and the remote controller. While smaller antennas could be used for lower frequency bands, these antennas would be fundamentally limited in their efficiency.

6.3 Creating a link budget with Friis transmission

Another crucial aspect of an RF system is designing for a desired transmission range. Traditionally this is done by creating a link budget. To create a link budget, the power output of the transmitter must be known, the antenna gain of both the transmitter and receiver must be known, and the receiver sensitivity must be known. With these inputs and assuming Friis transmission, the Friis transmission equation can be used to calculate range [47]. For the Marq Drone system, quarter wavelength monopoles were used for the transmitter and receiver that have an ideal antenna gain of 5.19 dBi. In regards to the transiever, the CC2500 was selected which has a max transmission power of 1dBm and a receiver sensitivity of -83 dBm at 500kBaud. The standard representation of the Friis transmission equation as shown in equation **6.2** where P_{rx} is the receiver sensitivity in watts, where P_{tx} is the transmitter power in watts, where r is the distance in meters, and where G_{tx}, G_{rx} are the isotropic gains of the transmitter and receiver antennas.

$$P_{rx} = P_{tx} \cdot G_{tx} \cdot G_{rx} \left(\frac{c}{4\pi \cdot r \cdot f_o} \right)^2 \tag{6.2}$$

Next, this equation is solved for range in terms of units that are commonly specified such as decibel milli-watts (dBm) and decibel isotropic (dBi).

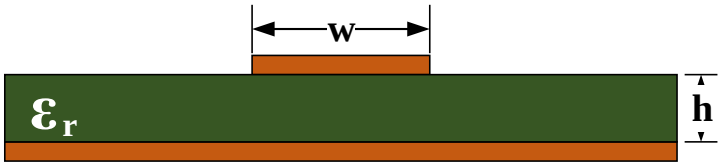
$$r = 10E\left(\frac{P_{tx,dbm} - P_{rx,dbm} + G_{tx,dbi} + G_{rx,dbi} + 20log_{10}(\frac{c}{4 \cdot \pi \cdot f_o}) - 10}{20} \right) \approx 82m \tag{6.3}$$

Note this is a highly optimistic estimate of the expected range since it assumes perfect antenna gains and no additional losses. However, this number is a ball park range and one cannot expect more than this value. Often it is assumed that the observed value will be 10 times less than an ideal measurement which would leave the range to be slightly under 100 feet which is acceptable for indoor flight.

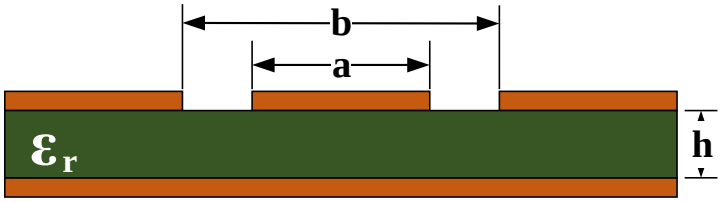
6.4 2.4GHz Microstrip and Coplanar Waveguide Transmission Line

Transmission lines are required for high frequency signals where parasitic inductance and capacitance begin to affect the characteristics of the signal transmitted. 2.4 GHz is a high enough frequency where these parasitic effects will have adverse influence over signals transmitted. For optimum power transmission, the source impedance must match the load impedance and the transmission line impedance. In general, both sources and antennas are matched to 50 Ω impedance. This leaves the transmission line impedance which must be designed to 50 Ω. Consequently, a transmission line is needed on the PCB between the source and the antenna. There are two common transmission lines that can be easily created on a PCB and they are shown in Fig. **6.3**.

Calculations for these transmission lines are lengthy and are included in Appendix B. Note that it is imperative that the correct ϵ_r be used for both equations and that this parameter changes with frequency. Fig. **6.4** is a demonstration of how frequency affects the dielectric constant of a PCB substrate. For most materials, the dielectric constant trends downwards as the frequency increases.



(a)



(b)

Figure 6.3: PCB transmission lines: (a) Microstrip line, (b) Coplanar waveguide.

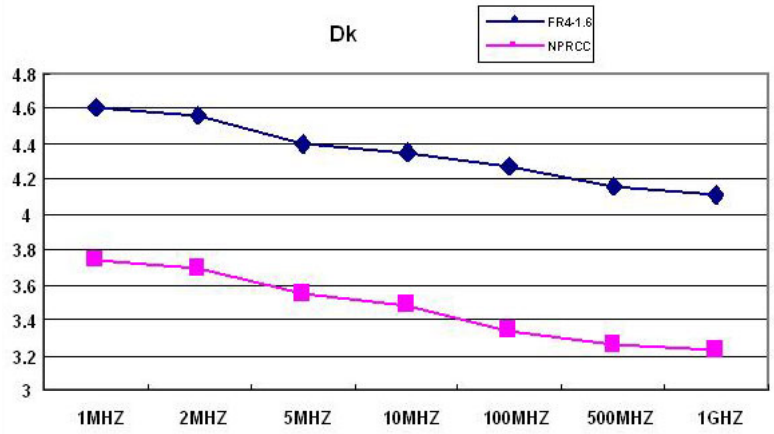


Figure 6.4: Plot of how ϵ_r changes with respect to frequency [12].

Failing to take the material's frequency characteristics into account can result in lossy transmission lines. While the transmission line equations in Appendix B are frequency independent, the material characteristics cause a real world frequency dependency.

With regards to transmission line selection, coplanar wage guides are often preferred since they result in smaller transmission lines. In addition, coplanar

waveguides are more ideal for adding shunt tunning elements. Parallel and series element tuning networks can be used to match a load impedance (antenna) to the transmission and the source impedance. Tunning strategies are beyond the scope of this thesis but are overviewed at the following reference [48]. An example matching network is shown in Fig. **6.5**.

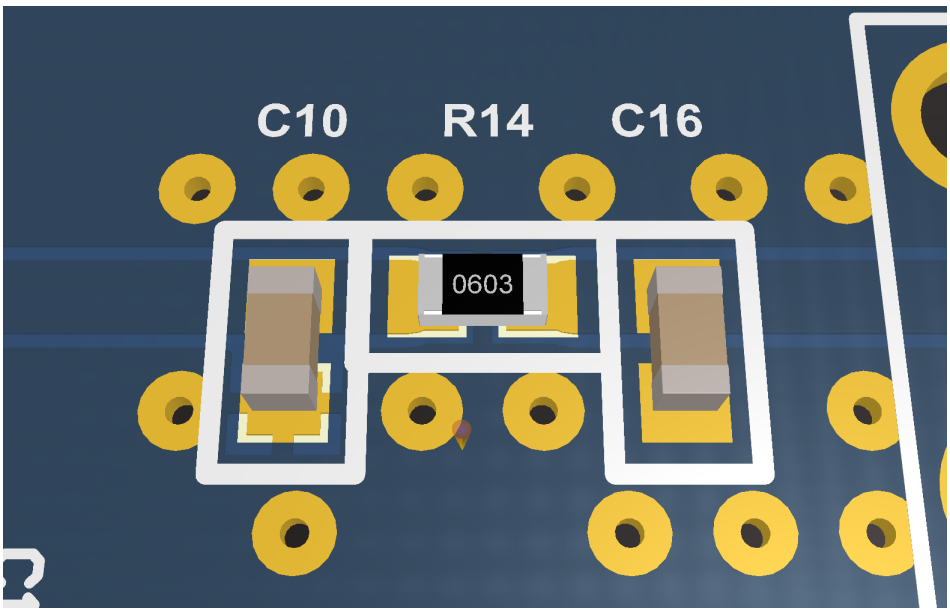


Figure 6.5: PI tuning network example located on the RF controller.

In this network, shunt and series elements can be changed to capacitors or inductors as needed. This network should be located close to the load that is intended to be matched. Vias were added around the network to reduce ground loop impedance. Without vias, small value capacitors can act as inductors. These vias are left un-tented to allow for connection of RF pigtails. RF pigtails are used to validate the impedance of the network with a vector network analyzer. Note, if RF performance is satisfactory, the network can be left unpopulated except for a zero ohm resistor on the series element.

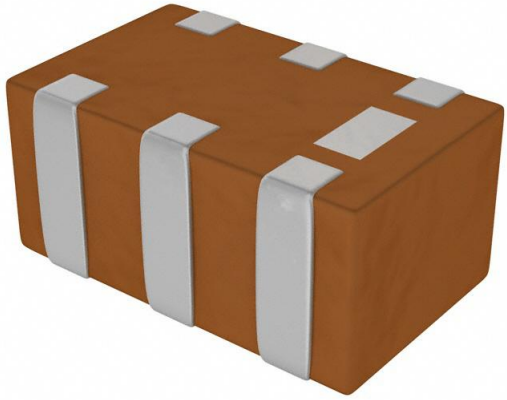
6.5 RF Baluns

RF sources and loads often exist in differential and single ended forms. The CC2500 transceiver has a differential RF output which was intended for differential loads such as loop antennas or dipole antennas. However, differential antennas are a mechanical inconvenience so it is often desired to use single ended loads such as a monopole antenna or a helical antenna. Consequently, baluns are used to convert from differential signals to a single ended signals and vice versa. PCB mounted baluns can be made of passive discrete electrical components such as inductors and capacitors [14]. However, the validation and the tunning of these baluns is difficult without access to a high frequency oscilloscope. An alternative is to use an integrated balun that requires minimal additional discrete components. Examples of both an integrated balun and discrete balun are illustrated in Fig. **6.6**.

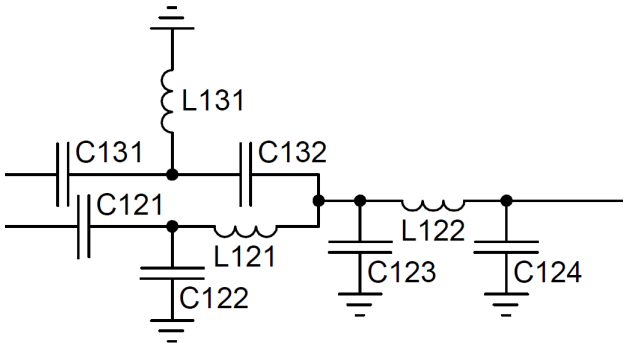
For this thesis, an integrated balun was used to save space and reduce design complexity [13].

6.6 RF Packet Structure and Collision Avoidance

Once the hardware design of the system was completed, a method of sending RF packets was designed. This method is only an example and is by no means optimal. RF packet structures vary for application as to whether throughput or range is prioritized. The RF packet structure used in this thesis is shown in Fig. **6.7**.



(a)



(b)

Figure 6.6: RF Baluns: (a) Integrated Balun [13], (b) Discrete Balun [14].

RF Preamble				Sync		Data							Check sum	
						Sync		Index	Data					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 6.7: RF packet implementation.

This packet contains four major components which are a preamble, sync, data, and checksum. The preamble size can vary and is used to give the receiver an opportunity to lock onto the signal being transmitted. Longer preambles increase the probability of the packet being received. Sync bytes inform the receiver when the preamble has ended. The data segment of the packet in this case consists of

two sync bytes, an index byte, and four bytes of data. Finally a checksum is appended to the packet to help verify the validity of the data received. Note the preambles are not checksummed since it takes time for the receiver to lock onto the signal and initial corruption is expected. RF packets are discarded should the data checksum fail. Packeting of data as opposed to a continuous stream helps save power as well as gives the opportunity for bidirectional communication between transceivers. However, a disadvantage of packeting data is overhead. In this case, 15 bytes of data are used to convey 5 bytes of information. This limits the effective baud rate at which information is transfered. Another bandwidth limitation is the speed that bytes are encoded on top of the 2.4 GHz carrier frequency. For this implementation, two frequency shift keying (2-FSK) modulation was used. Bytes were encoded at 250K baud rate which meant that at best, information would be conveyed at 250K speed in addition to packet overhead. A lower baud would increase probability of reception meanwhile a higher baud would allow for faster transmission. 250K was selected as a compromise between these two goals. Then, to achieve bi-directional communication, a ping system was used in which the remote would periodically send out control packets and the flight controller would send back telemetry information when it received a packet. Naturally, the period of this exchange must be longer than the time required to transmit 2 packets. A packet exchange is shown in Fig. **6.8** by probing the voltage at the balun.

The balun has an integrated bandpass filter which simplifies observation of transmitted and received packets. While the presence of packets can be observed by a RMS magnitude, the individual byte information is not possible to obtain

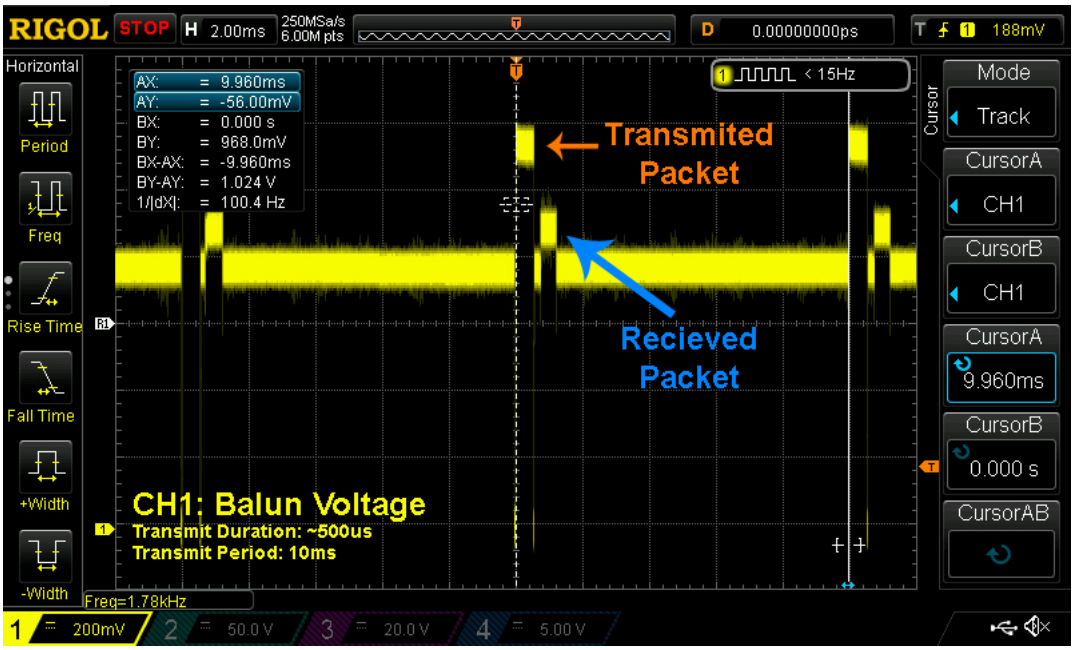


Figure 6.8: RF packet implementation observed on an oscilloscope by probing the balun voltage.

without a GHz class oscilloscope. Having the remote transmit periodically and the flight controller echoing back helps prevent any chance of packet collision. An initial design where both the controller and flight controller periodically transmitted independently resulted in phases of signal loss. During the phases of signal loss, both the remote and controller were periodically transmitting at the same time and periodic collisions would occur.

6.7 Conclusion

Designing a custom RF system allows for great flexibility and efficient data acquisition but comes at the cost of design complexity. For a custom RF system, first a frequency band is selected based on the antenna size desired and legal bands of operation. After a frequency has been selected, antenna types and a transceiver are selected given the mechanical and power constraints of the

system. RF transmission lines and baluns are required in order to get a signal from point A to B without significant loss. Finally, a scheme must be formulated for the transmission of bidirectional packets. In the end, an efficient, customizable, and bidirectional link was established between the quadcopter and the remote. This is advantageous over the majority of off-the-shelf systems since data is not encoded into PPM signals, the data is bidirectional, and the source files are available for future expansion.

CHAPTER 7

Hardware Implementations

7.1 Introduction

This chapter overviews the electronic hardware design of the quadcopter. Electronic hardware components of this system include the radio system, flight controller, and motor controllers. A picture of the quadcopter is shown in Fig. 7.1.

These pieces of hardware were designed to help understand the nature of the

underlying system as well to be able to implement a more flexible architecture as shown previous in Fig. **2.5**. Hardware involved in this project was designed in Altium and schematics/PCBs can be viewed fully in Appendix A. The following sections overview the hardware design for the quadcopter.

7.2 Flight Controller Hardware

The flight controller consists of a MCU, a IMU, a radiocore, and external IO. The flight controller is responsible for stabilizing the quadcopter and executing user control. To simplify the software, a microprocessor was needed with sufficient computational power for numerous floating point instructions. The Cortex M4 MK22FN512 by Freescale was chosen for its FPU, large flash space, large RAM space, USB interface, low per unit cost, and hardware simplicity. With regards to the IMU, the MPU-9250 by Invensense was selected since it is a compact integrated solution with a 3 axis accelerometer, gyro, and magnetometer. Another advantage of the MPU-9250 is that it had already been flight tested in another multi-rotor platform [24]. For the radiocore, the CC2500 by Texas instruments was selected due to its low cost and high reconfigurability. For motor control, four hardware PWM pins were exposed on external headers for PPM communication with ESCs. Finally, serial IO and analog IO were exposed on an additional external header for additional expansion. This architecture is illustrated in block diagram form in Fig. **7.1**.

The board itself was assembled using solder paste and a toaster reflow oven. The primary hardware challenges involved in flight controller operation was the

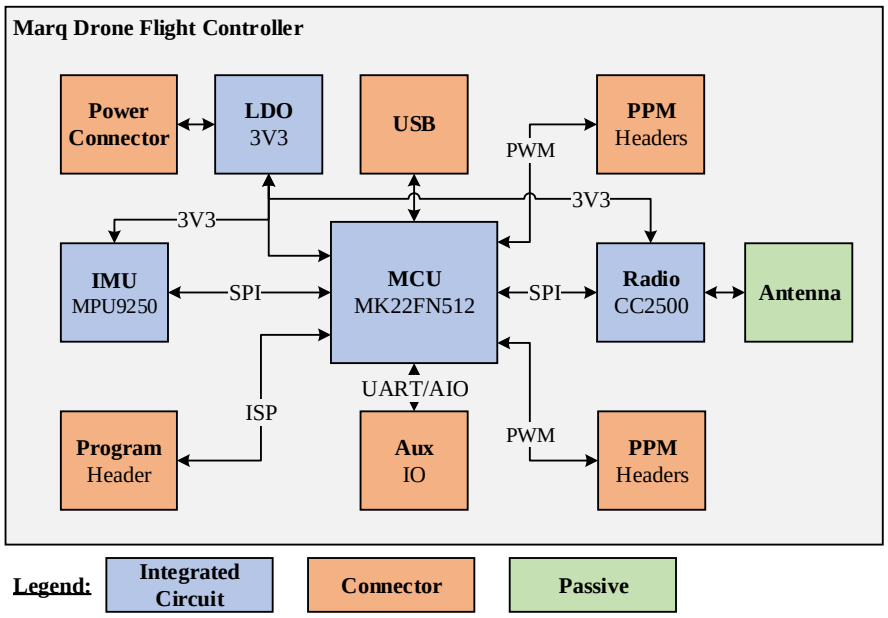


Figure 7.1: Block diagram of flight controller

development of the radio core. The micro-controller as well as the IMU are much lower frequency devices which accordingly required less attention to detail. Clean power sourcing, decoupling, electrostatic discharge (ESD) protection, transient protection and other basic hardware principles were implemented in its design. The assembled design is shown in Fig. **7.2**.

7.3 Remote Controller

The remote controller hardware consists of two joysticks, a micro-controller with a USB interface, a radiocore, push buttons, and a rechargeable battery. A system level block diagram for the remote is shown in Fig. **7.3**. This diagram illustrates all the major hardware blocks utilized by the controller. Further information can derived from the source code as well as the schematic.

For the joysticks, two Frsky Taranis Gimbals were selected. These joysticks were

selected since they were primarily designed for remote flight control. To simplify the mechanical design, these joysticks mount directly to the PCB which avoids requiring an injection molded or 3D printed enclosure. With regards to the microcontroller, the Freescale Kinetis Cortex M0+ KL25Z128 microcontroller was selected due to its low cost, availability, and USB interface. The radiocore is the same as the flight controller which is the CC2500 and discussed in Chapter 6. Push buttons were added for flight commands such as emergency cutoff. Finally, for remote portability, a 1 cell 4.2V LiPo cell battery was used such that the remote could be operated independently of a computer. This cell would be recharing any time the USB interface was plugged in. In addition, the battery system had hardware protection against reverse battery, current back-feeding, over current, and under-voltage. A picture of the assembled controller is illustrated in Fig. **7.4**.

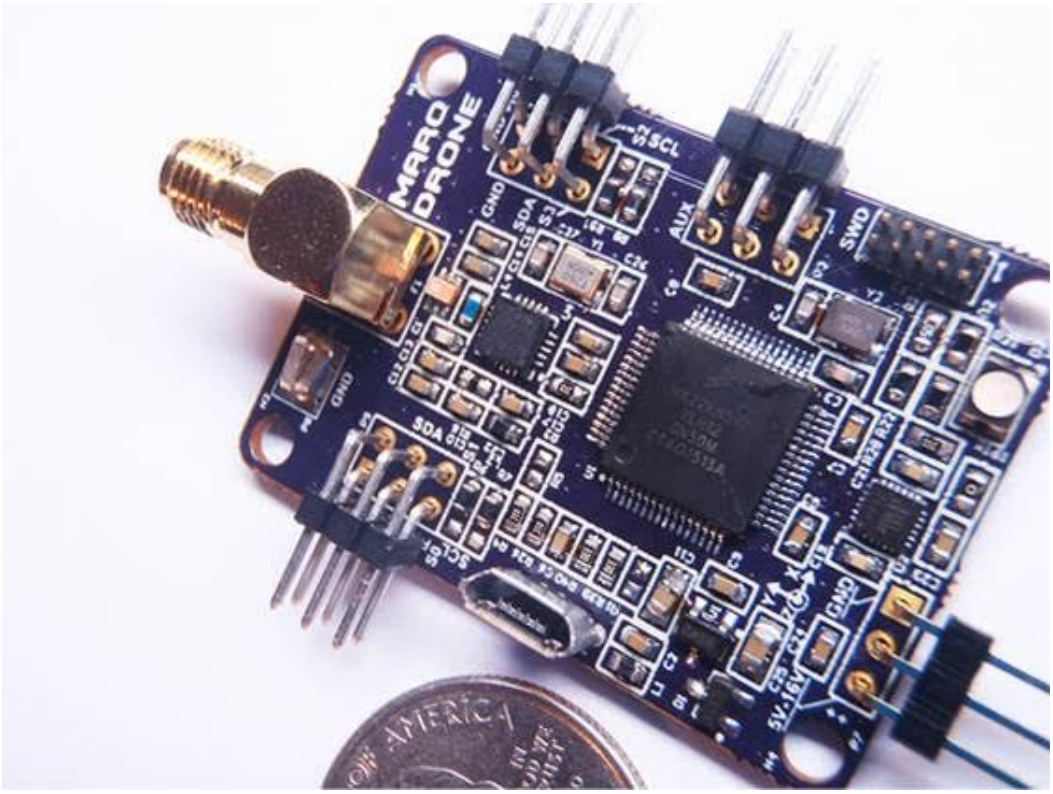


Figure 7.2: Image of Marq Drone flight controller.

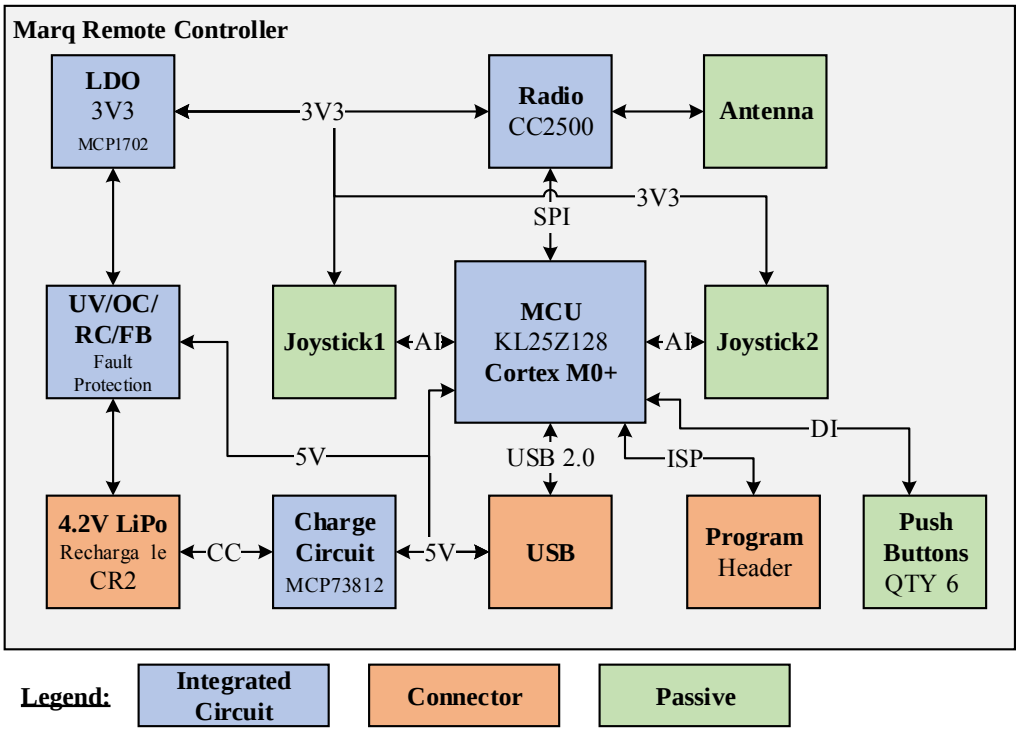


Figure 7.3: Block Diagram of flight controller.

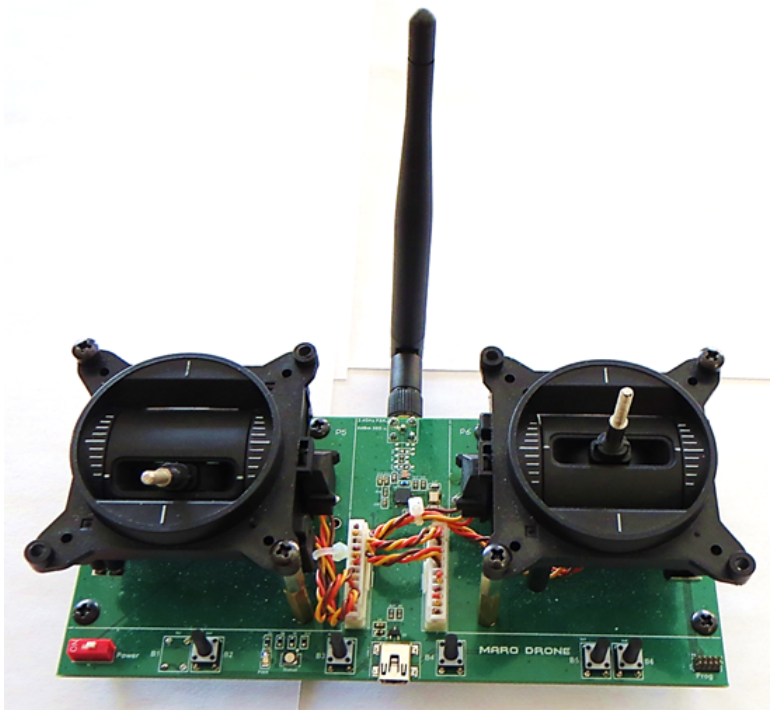


Figure 7.4: Image of Marq Drone remote.

Apart from the RF design, the only other challenging aspect of the remote controller design was the LiPo battery protection system. Care must be taken such that Lipo batteries are not overcharged. Overcharging LiPos will break down their internal chemistry and potentially cause fires. Consequently, the MCP73812 charge IC was used to regulate the current being used to charge the LiPo and to cutoff charging at the batteries nominal voltage. Another danger with LiPo batteries is draining them past their minimum rated voltage. Overdraining can result in a break down of chemistry which leads to inability to recharge and sustain charge. Finally, drawing more than the battery's rated current can cause the battery to overheat and potentially catch fire and or break down in chemistry. Note, the energy capacity contained in a small single cell LiPo battery is very minimal which significantly reduces the potential hazards the battery can generate. The circuit derived to prevent under voltage, over current, and reverse battery is shown in Fig. **7.5**.

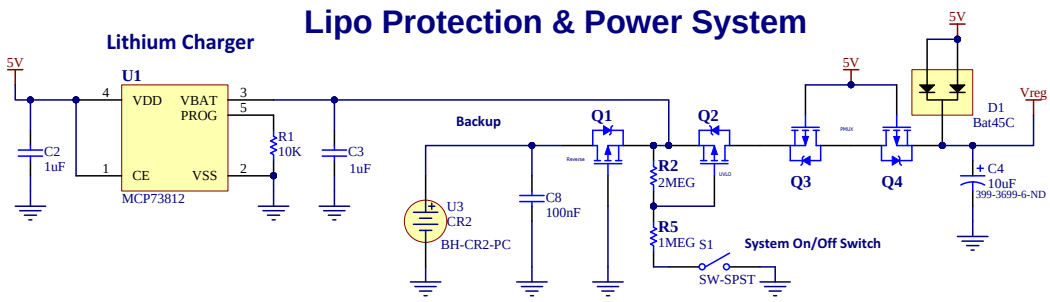


Figure 7.5: LiPo under voltage, over current, over charge, reverse bias, and back-feed protection circuit.

U1 prevents overcharging of the LiPo. PFET Q1 prevents damage to U1 and other ICs in the event of reverse bias. Q2 in combination with R2 and R5 creates an under voltage lockout as well as an over current lockout. When the LiPos battery voltage is low, the bridge divider on the gate of Q2 forces Q2 into the

cutoff region of operation. Similarly, when significant current is drawn, the internal resistance of the battery combined with the R2 and R5 bridge forces Q2 into cutoff. Back to back PFETs Q3 and Q4 prevent USB power from back-feeding the battery when plugged in. FETs were used as opposed to diodes on the batteries current path to eliminate voltage drop out on the 3.3V rail.

7.4 Electronic Speed Controller Hardware

The brushless sensorless motor controllers, or ESCs, are made up primarily of a microcontroller, gate drivers, and high power FETs. The MCU selected was the Cortex M0+ KL25Z128 Kinetis by Freescale. This was the same processor used on the remote and is pin by pin compatible with the micro-controller on the flight controller. The gate driver selected was the ADP3110 by ON Semiconductor since it supported dual high side low side NFET driving, supported 3.3V logic signals, is low cost, and has a easy to solder package. The primary switching FET selected was the PSMN011 by NXP since it supports a max drain current of 61A, has an RDSON of 10 $m\Omega$, and since it utilizes a small QFN package. A block diagram of this system is Fig. **7.6**.

This ESC was designed to be compact and have capacity to drive atleast 10A of current. To achieve this, double sided population was required. Another useful aspect of this design was the USB interface for data acquisition as well as programming. This ESC was created first as a larger dev board with more debugging features such as headers for probing. Once the theory of operation was confirmed, this board was shrunk to a 0.9" by 1.6" board. Pictures of both of

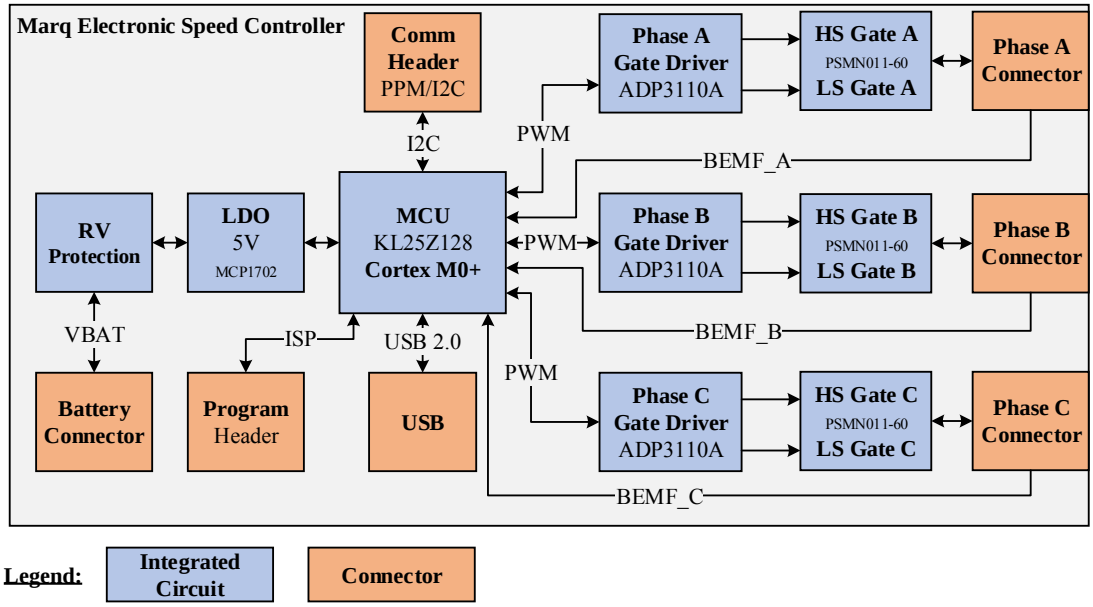
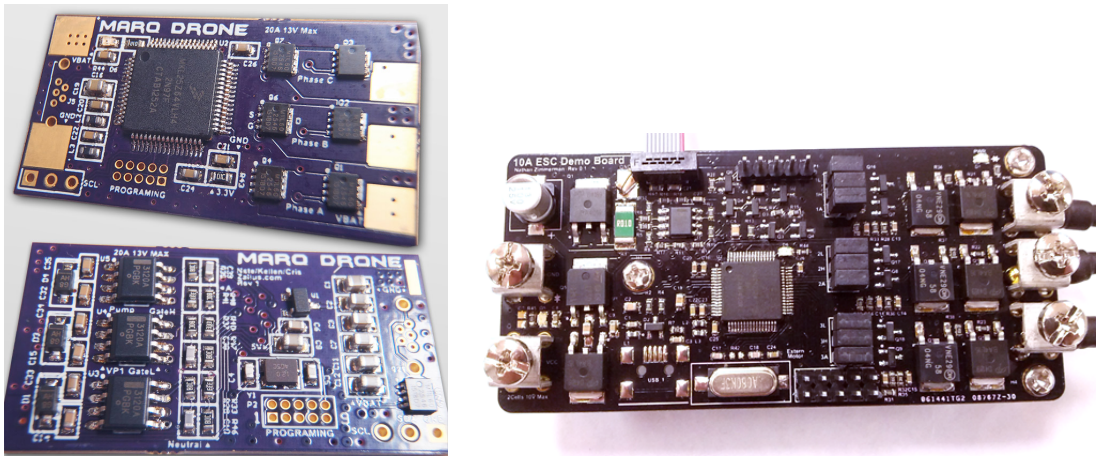


Figure 7.6: Block Diagram of Marq ESC.

these boards are shown in Fig. 7.7



(a) (b)

Figure 7.7: Picture of ESCs: (a) Marq ESC, (b) Marq ESC Dev board.

With regards to method of operation, recall these ESC are effectively sensorless brushless motor controllers as stated in Section 2.1. This controllers uses is back EMF zero crossing detection [29]. The following section will briefly describe brushless motor basics as well as the BEMF zero cross detection driving method

used in this controller.

7.4.1 Brushless Motor Operation

Brushless motors are synchronous 3 phase permanent magnet motors. Like a DC brushed motor, brushless motor consists of a stator which contains coils and a rotor which contains permanent magnets. If the coils are energized correctly, the coils create a rotational torque that acts on the magnets on the rotor. Since the magnets are adhered to rotor, this rotational torque causes the motor shaft to spin. However, if the coils on the stator are not energized correctly with respect to the polarity and position of the magnets, rotational torque is not generated. Consequently, knowledge of the rotors position is required in order to properly drive a DC motor. For brushed motors, a mechanical device called a brush is used such that stator coils are always properly energized with respect to the rotor. However, with a brushless motor, this mechanical linkage from stator to the rotor is removed in order to reduce friction and extend motor lifetime. Without the mechanical linkage between the rotor and the stator, external sensing methods of the rotor are required. Hall effect sensor can be used in order to detect the rotors position. However, hall effect sensors are expensive and require additional wires as well as mechanical protection. Another method of driving brushless motors is with sensor-less techniques such as back electromotive force (BEMF) zero cross detection.

7.4.2 Introduction to BEMF Zero Cross Driving

BEMF zero cross detection driving is a brushless 3 phase motor driving method that relies on estimating the rotors position by using voltage generated by the motor's rotation. Naturally, brushless motors generate voltage when rotating similar to how a generator produces power. This voltage generated by the brushless motor is called BEMF. During operation, it is possible to view the BEMF signal of a DC 3 phase brushless DC motor on a disconnected phase. Within the BEMF waveform, there is an event called the "zero crossing". This event is used to determine the rotor's position in order to drive the motor correctly. The following section will introduce basic driving concepts as well as how to observe and utilize zero crossing events.

7.4.3 Brush-less Motor Driving Hardware

Brushless motors are 3 phase loads. To drive current in any direction through these three phases, three H-bridge switches are used. Consequently, there is a total of six switches and only two switches are turned on at a given time. This leaves one phase disconnected and this phase's voltage is representational of the BEMF of the motor. This concept is illustrated in Fig. **7.8** where the phase A high side switch is on, where the phase C low side switch is on, and where phase B voltage is the BEMF signature.

These 6 switches must be actuated in a specific sequence in order for the motor to spin. This sequence is often called the "6 step trapezoidal commutation sequence". This sequence is illustrated in Fig. **7.9**. If actuated correctly, the this creates a rotational torque on the rotor of the motor. Note that, in

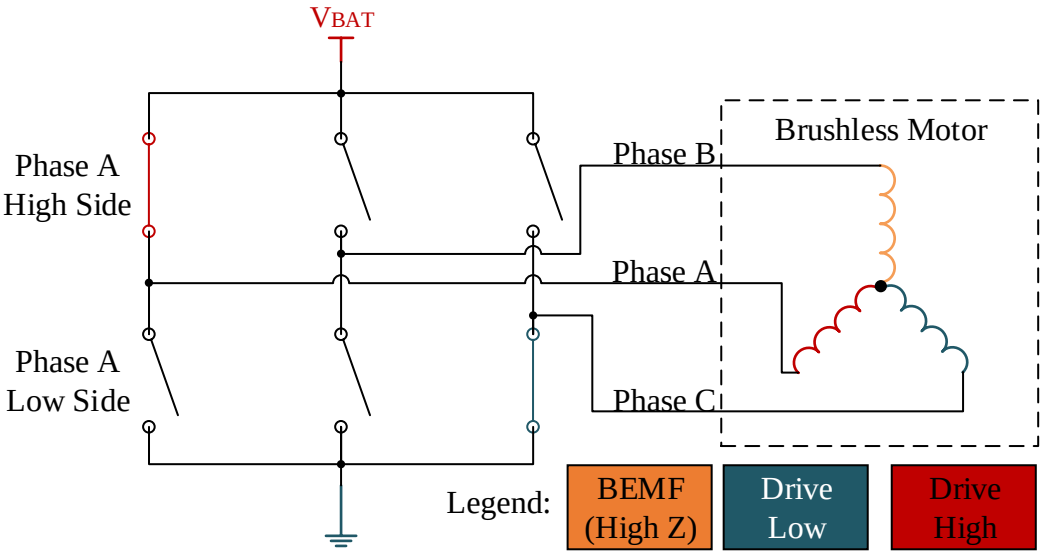


Figure 7.8: Diagram of brushless motor and 6 driving switches.

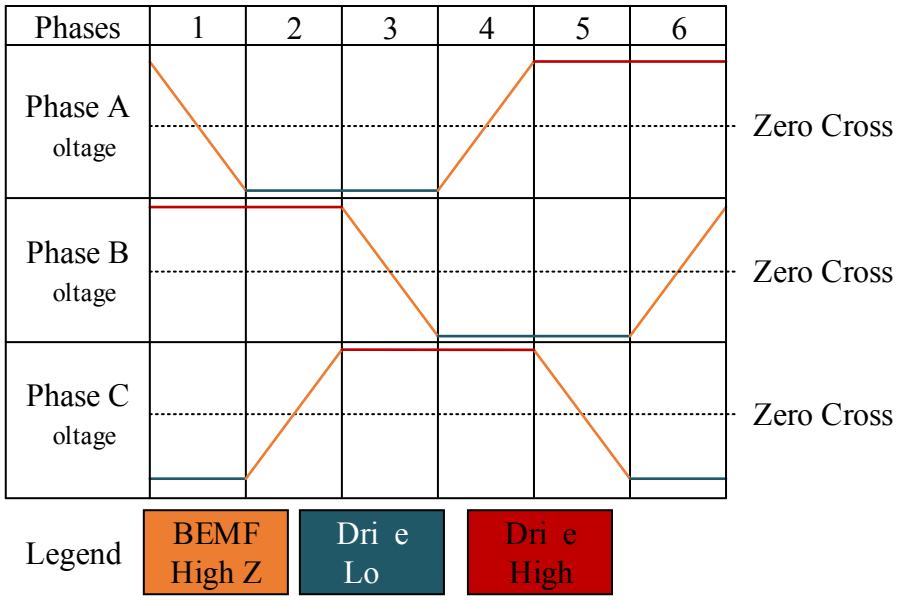


Figure 7.9: 6 step trapezoidal commutation sequence.

sequence, one phase is always producing BEMF. The BEMF phase is used in order to determine the position of the rotor such that power can be applied productively. The position of the rotor can be determined by detecting zero crossing events. A zero crossing is where the BEMF crosses over the neutral

voltage of the motor. This zero crossing event represents a rotor mid point between a commutation step. To utilize this zero crossing event, an assumption can be made that the rotor is not rapidly accelerating within a commutation step. This assumption is fair for brushless motors on a quadcopter since the torque load is nearly constant. With this assumption, one can assume that the time it takes for the mid point to be observed will also be the approximate time it takes for the rotor to reach the next commutation phase. A basic software flowchart demonstrating this concept is shown in Fig. **7.10**.

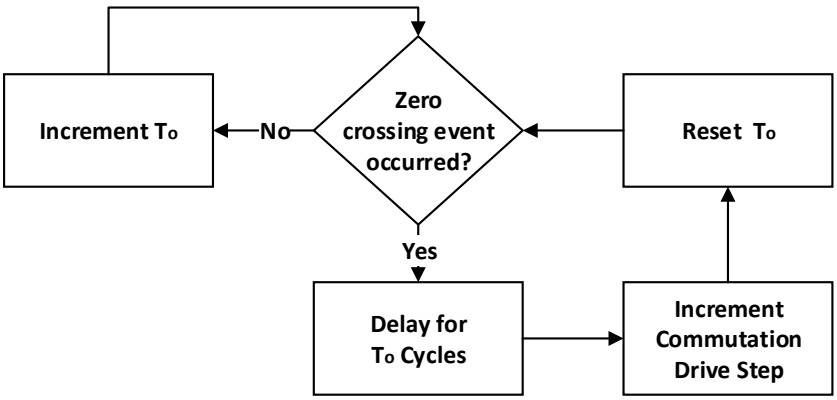


Figure 7.10: Software block diagram of zero cross brushless motor driving.

While this driving scheme is simple, complexity arises in detecting the zero cross event. The zero crossing event can be determined by using a hardware comparator that compares the motor neutral voltage with the BEMF waveform. However, most brushless motors for multi-rotors do not expose the neutral signal line of the motor. Consequently, this neutral voltage must be approximated by summing the three phase voltages. This is further complicated by significant BEMF noise generated by PWM switching. The diagram shown in Fig. **7.9** illustrates 100% duty cycle which is not a common mode of operation. It follows

that to vary the speed of the motor, the duration of on time is reduced with PWM. These switching signals create voltage transients that couple onto the BEMF phase. A brushless motor phase voltage is shown in Fig. **7.11** along with the hardware comparator readings.

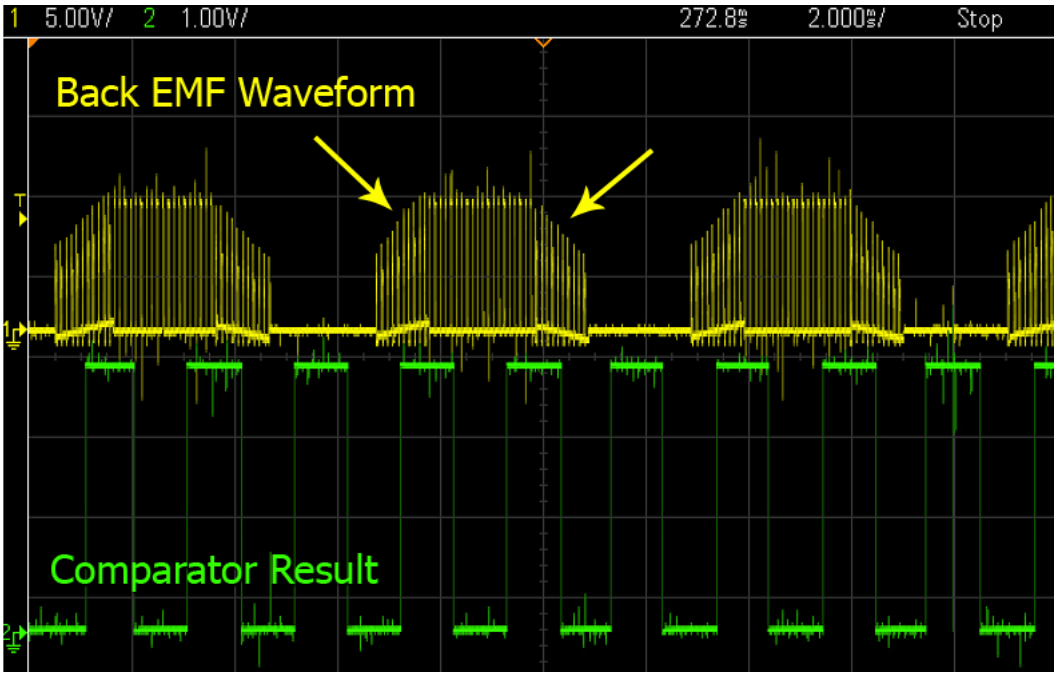


Figure 7.11: Brushless motor phase voltage (yellow) and zero cross detection comparator (green).

To achieve these results, hardware filtering was implemented on the MCU where comparator samples were windowed and transitions were generated only after a certain amount of consecutive agreeing samples were observed. This filtering along with the software method proposed in Fig. **7.10** allowed for effective brushless motor operation.

7.4.4 ESC Conclusion

Brushless motors are efficient motors that can be driven through various sensorless techniques such as zero cross detection. Section 7.4.3 overviews both the hardware and software required to implement BEMF zero-cross driving of brushless motors. While sensorless driving was achieved, there was not enough time to incorporate the drivers shown in Fig. **7.7** into the final quadcopter.

CHAPTER 8

ICL and Lidar Integration

8.1 Introduction

This chapter overviews a strategy to meet the secondary control objective proposed in equation **3.2** which is the elimination of positional drift. Positional drift is one obstacle of many that must be overcome in order to achieve autonomous indoor flight of quadcopters. Unlike land based robots, there is no significant static frictional force to keep a quadcopter in place when hovering. Consequently, there is a variety of factors that can cause positional drift in a quadcopter. One source of drift results from equation **3.4** being an approximation. Frame imperfections and air turbulence results in a summed force vector that isn't purely vertical. Another source of drift can be from external forces such as air flow in a room. A final source of drift can occur from imperfections in the control system or imperfections in the orientation estimation. As a result of these sources, positional drift is a common problem for quadcopters. The sensors overviewed in Chapter 4 are incapable of measuring positional drift directly. Consequently, to directly sense positional drift, more sensors are needed. For outdoor flight, GPS in combination with IMU and barometer sensors have been used to reduce positional drift with an extended Kalman filter [49]. However, GPS is not reliable indoors and barometers do not provide sub meter accuracy. Consequently, other sensors are used such as optical

flow cameras, external cameras, radio beacons, and lidars. This thesis will explore the use of a lidar sensor.

8.2 Lidar use for positioning

Lidar is a rotating laser range finder that provides 2-dimensional position information. A laser range finder calculates distance by measuring the time required for a laser to strike a target and bounce back. This time of flight measurement can be converted to distance using the speed of light. The lidar's distance output is then encoded into polar coordinates ($r\angle\theta$). This data can then be converted to Cartesian coordinates using the equations: $x = r \cdot cos(\theta)$, $y = r \cdot sin(\theta)$. An example 2D lidar dataset recorded in a room is shown in Fig. 8.1 where points represent distance from the lidar.

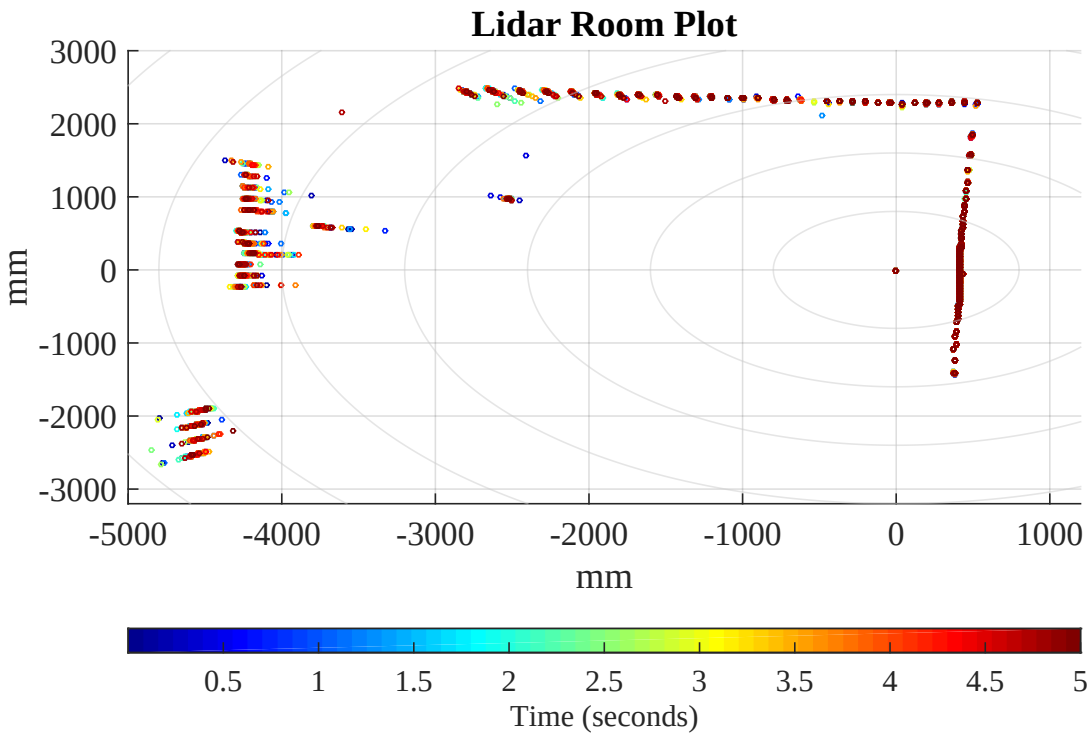
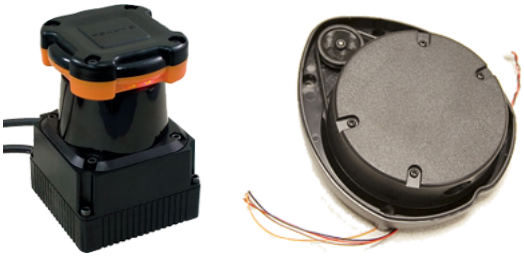


Figure 8.1: Lidar plot of room

In this plot, the change in color represents a change in time. For this dataset, the lidar was in a stationary position. With regards to the physical hardware, this dataset was taken with the XV-11 lidar [50]. This lidar was used in order to help eliminate positional drift cheaply and in a computational efficient manner. Currently, this lidar is the most affordable lidar on the market which provides motivation for its use. Commonly, in other research, more expensive lidars such as the Hokuyo UTM-30XL lidar was used [15],[26]. Pictures of the XV-11 lidar and the Hokuyo UTM-30XL are shown in Fig. **8.2**.



(a) (b)

Figure 8.2: (a) Hokuyo UTM-30LX lidar used in [15] . (b) Neato XV-11 lidar device used in this project.

While the XV-11 lidar is more affordable, this price reduction comes at the cost of performance. Table 8.2 overviews the technical performance between the XV-11 lidar and the UTM-30LX Lidar.

Lidar device:	**Hokuyo UTM-30LX**	**Neato XV-11**
Points per revolution	1,440	360
Revolution speed (ms)	25	250
Max range (m)	30	4
Cost (USD)	4,825	150

Table 8.1: Specifications of the two lidars from Fig.8.2.

While lidar can provide crucial distance information, this data comes at the cost of significant computational requirements. For example, the UTM-30LX

generates 57600 points per second. Once points are obtained, a significant amount of post processing must occur in order to transform this data into useful positional information. As a result, systems developed with lidars often consist of GHz processors with gigabytes of RAM. This chapter will focus on developing an algorithm to achieve this translational information with significantly less computation.

8.3 Method of parsing Lidar Data

In order to eliminate the quadcopter's positional drift $(\Delta x, \Delta y)$, an algorithm is required in order to estimate $(\hat{\Delta x}, \hat{\Delta y})$ from the lidar datasets. An example dataset of positional drift is shown in Fig. **8.3** where the lidar was moved left by roughly 600mm.

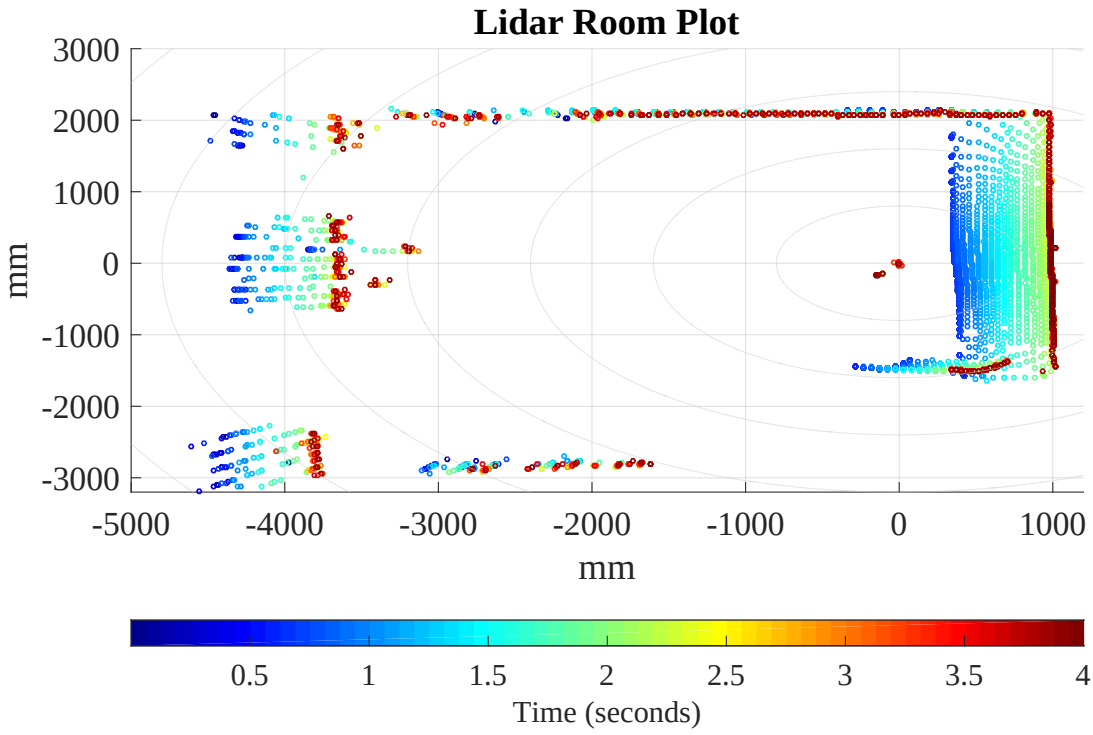


Figure 8.3: Lidar moved to the left by $\approx 600mm$

To estimate information from this lidar data, various computational geometry methods are employed such as iterative closest point (ICP), and iterative closest point line (ICPL) [25][51]. These methods are commonly used for solving point cloud translation problems. In addition to determining positional translation, these algorithms can also be employed to detect rotational differences and scaling differences between data sets. However, in order to limit scope, complexity, and computation, the lidar will be employed only for translational determination. The magnetometer for example, can be used to determine rotation. While ICP can estimate translation, ICP becomes less accurate with less data points and when datasets that are taken at a further distance away. For example, if ICP is employed on the dataset in Fig. **8.3**, ICP would register positional changes in both X and Y as shown in Fig. **8.4**.

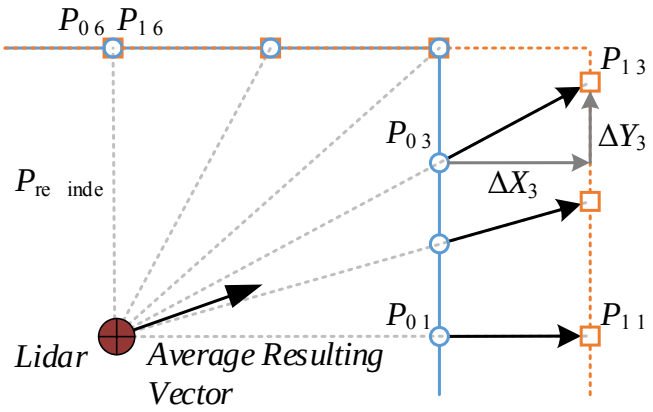


Figure 8.4: ICP Iteration

ICP operates with gradient descent and an example iteration is shown in equation **8.1** where x and y are arrays of N lidar points, where σ represents the gradient descent learning rate, and where $\Delta x, \Delta y$ represent the resulting lidar translation.

$$\delta x = E\left[\sum_{i=0}^{i=N}(x_i - x_{cp})\right], \qquad \delta y = E\left[\sum_{i=0}^{i=N}(y_i - y_{cp})\right]$$
$$\Delta x = \Delta x + \delta x, \qquad \Delta y = \Delta y + \delta y \tag{8.1}$$
$$x = \sum_{i=0}^{i=N}(x_i + \sigma \cdot \delta x), \qquad y = \sum_{i=0}^{i=N}(y_i + \sigma \cdot \delta y)$$

Note that additional computation is required in order to find (x_{cp}, y_{cp}) which represent the coordinates of the closest point from a previous lidar data set to a present given point (x_i, y_i). Convergence requirements and the value of the learning rate would be determined by a designer on a application basis. However, due to the geometry nature of ICP acting on lidar data, this algorithm may not converge to a correct value. In order to achieve accurate indoor drift measurement over time, a more accurate algorithm is needed.

8.4 Iterative Closest Line

This section overviews a specific implementation of the iterative closest line (ICL) algorithm proposed for parsing lidar data efficiently and accurately. ICL is based on a closest point to line metric as opposed to ICP which is based on a closest point to point metric. In ICL, the point P_{cp} with coordinates (x_{cp}, y_{cp}) is replaced with a point P_{CL} with coordinates (x_{CL}, y_{CL}). The point P_{CL} is located on line between the two previous closest points. In addition, P_{CL} is also the closest point on that line to the point being compared. The concept of P_{CL} points is shown in Fig. **8.5** where lidar points are denoted in the form of $P_{lidarRevolution,index}$.

The point (x_{CL}, y_{CL}) is calculated using equation **8.2** where points

$(x_1, y_1), (x_2, y_2)$ represent the two closest points from a prior lidar revolution with respect to a new lidar point (x_3, y_3). Note that in equation **8.2**, a, b, c represent parameters of the general line equation which is expressed as $a \cdot x + b \cdot y + c = 0$. Lines were formulated in this manner as opposed to slope intercept form in order to mitigate potential devisions by 0.

$$
\begin{aligned}
a &= y_2 - y_1, \quad b = x_1 - x_2, \quad c = -by_1 - ax_1 \\
x_{CL} &= \frac{b(bx_3 - ay_3) - ac}{a^2 + b^2} \\
y_{CL} &= \frac{a(-bx_3 + ax_3) - bc}{a^2 + b^2}
\end{aligned}
\tag{8.2}
$$

With the calculation procedure of (x_{CL}, y_{CL}) complete, the next step in ICL algorithm is to formulate and minimize a cost function J. The ICL cost function J shown in equation **8.3** where i represents the index of N lidar points.

$$
J = \sum_{i=1}^{N} (x_i - x_{CL})^2 + (y_i - y_{CL})^2 \tag{8.3}
$$

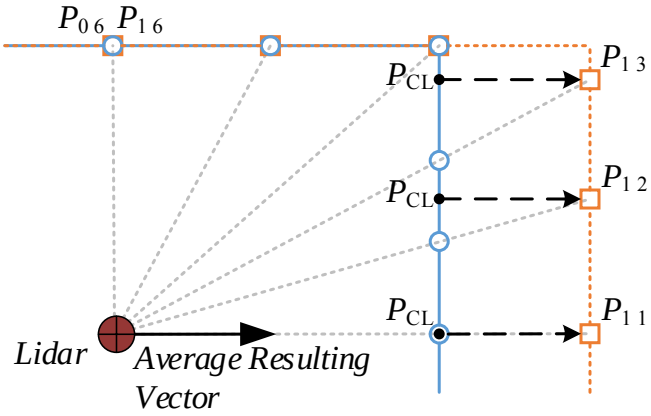


Figure 8.5: ICP iteration.

This cost function can be minimized iteratively through gradient descent. To use gradient descent, the partial derivatives must be calculated as shown in equation 8.4.

$$\begin{aligned} \delta x_k &= \sum_{i=1}^{N} \frac{\partial J_{k,i}}{\partial x} = \sum_{i=1}^{N} (x_{k,i} - x_{CL}) \\ \delta y_k &= \sum_{i=1}^{N} \frac{\partial J_{k,i}}{\partial y} = \sum_{i=1}^{N} (y_{k,i} - y_{CL}) \end{aligned} \tag{8.4}$$

Now that the necessary mathematical concepts have been overviewed, a basic pseudocode example for ICL is shown in Fig. 8.6.

```
int N = 360; // 360 points per revolution
float α = 0.5 // Learning Rate
int iterationCount = 10 // Fixed number of iterations
float δx,δy,Δx,Δy,x_CL,y_CL =0; //intermediate variables
float x[] = {x1, ... xn}; //Array of recent X lidar points
float y[] = {y1, .. yn}; // Array of recent Y lidar points
float xPrevious[] = {x1, ... xn}; //Array of previous X lidar
    points
float yPrevious[] = {y1, .. yn}; // Array of previous Y lidar
    points

void ICL(){

for( j = 0; j<iterationCount;j++)
{
  for( i = 0; i < N; i++){ //accumulate delta terms.
    x_CL = findCL(x[i],xPrevious); //Find closest point line
    y_CL = findCL(y[i],yPrevious);
    δx = δx + (x[i] - x_CL);
    δy = δy + (y[i] - y_CL);
  }
  for( i = 0; i < N; i++){ // Simultaneous update all points
    x[i] = x[i] - α * δx/N;
    y[i] = y[i] - α * δy/N;
  }
  Δx = Δx + δx/N; // accumulate displacements
  Δy = Δy + δy/N; // accumulate displacement
  δx = 0; δy = 0; // Reset delta values for next iteration
}
printf(Δx , Δy) // Output results

}
```

Figure 8.6: Pseudocode of the ICL algorithm

While this code example only covers a single lidar revolution, a more complete example can be found at [45]. The performance of this algorithm in comparison to vanilla ICP is shown in Fig. **8.7** acting upon the dataset shown in Fig. **8.3**.

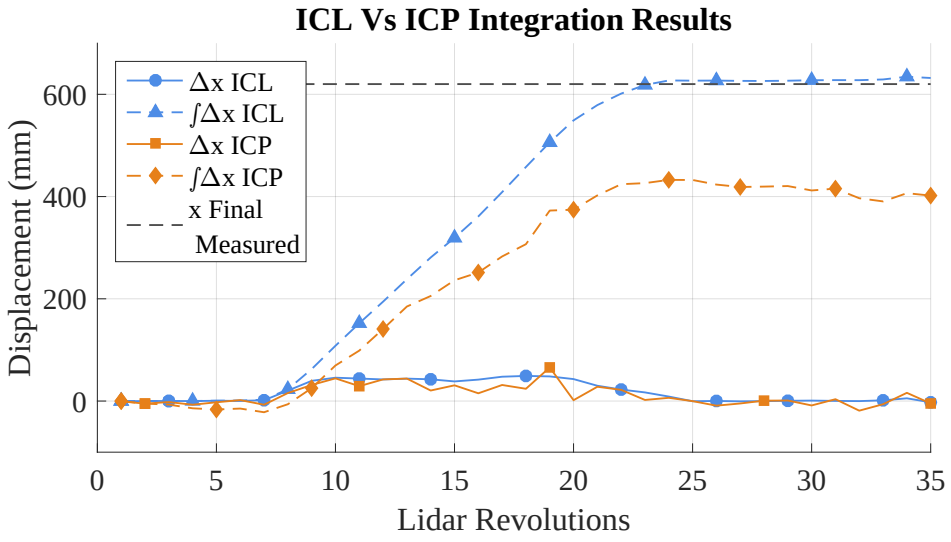


Figure 8.7: ICP and ICL integration results acting on dataset in Fig. **8.3**.

.

ICL and ICP were performed on every consecutive set of lidar data. Δx represents the rate of drift while $\int \Delta x$ represents the total accumulated drift. As seen in Fig. **8.7**, ICL error is smaller over time which results in the correct integration distance over time. In contrast, ICP has non negligible error resulting in significant accumulated error over time. Consequently, this implementation of ICL is a preferred candidate for parsing lidar data in real time.

8.5 Lidar Integration with Quadcopter Control System

This section will overview a simple proposed method of integrating lidar information into the quadcopter's control system such that positional drift is reduced. As shown in Sections 8.3 and 8.4, ICP and ICL can act on lidar data to

obtain positional drift estimations $\Delta x, \Delta y$. An additional PID control loop can

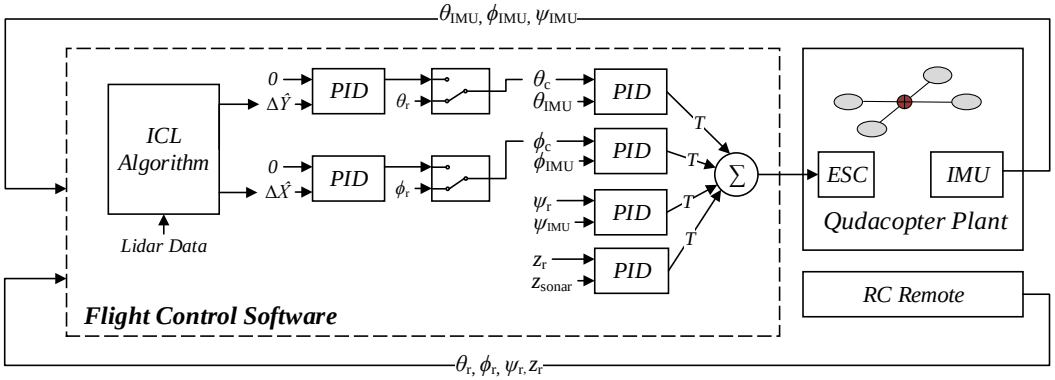
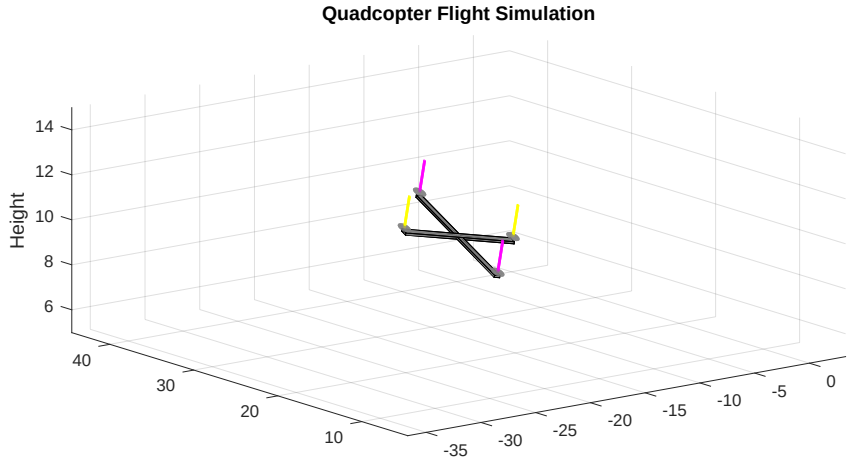


Figure 8.8: Quadcopter control diagram with lidar and sonar inputs added.

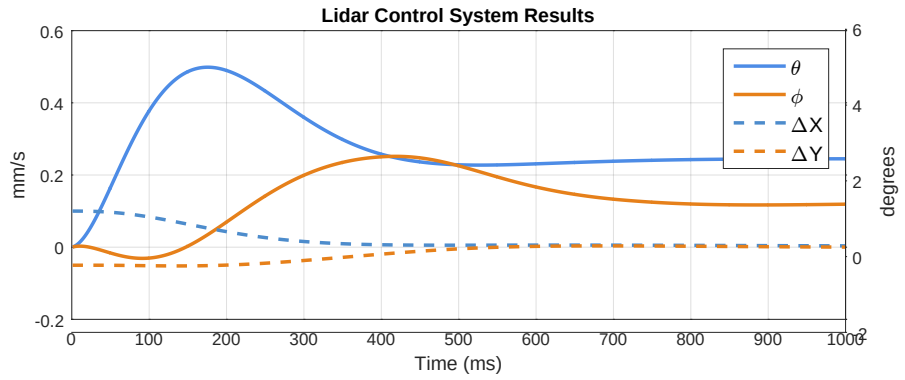
then be added acting on the quadcopter's ϕ, θ orientation angles to drive $\Delta x, \Delta y$ to zero respectively. This control scheme is shown in Fig. **8.8** and is an extension of the original control system shown in Fig. **5.2**.

8.6 Control System Testing, Simulation and Conclusion

The control system proposed in Fig. **8.8** was tested only in simulation. Flying a quadcopter with the XV-11 lidar would be challenging due to its heavy weight in addition to significant torque oscillations that would be coupled to the quad-frame. Consequently, this control scheme was tested in Matlab using a similar mathematical model [33]. In this model, an external disturbance was added causing drift in both the x and y planes. The control system in Fig. **8.8** then acted upon ϕ, θ to counteract this drift.



(a)



(b)

Figure 8.9: (a) Quadcopter simulation with external disturbance. (b) Plot of positional drift elimination.

CHAPTER 9

Conclusion and Future Work

9.1 Flight Performance

This section overviews the performance of meeting the control objectives described in equation **3.1**. Using the control system proposed in Chapter 5 and the hardware shown in Chapter 7, the quadcopter was able to achieve flight with minimal steady state error. A segment of data taken during a flight hover test is shown in Fig. **9.1**.

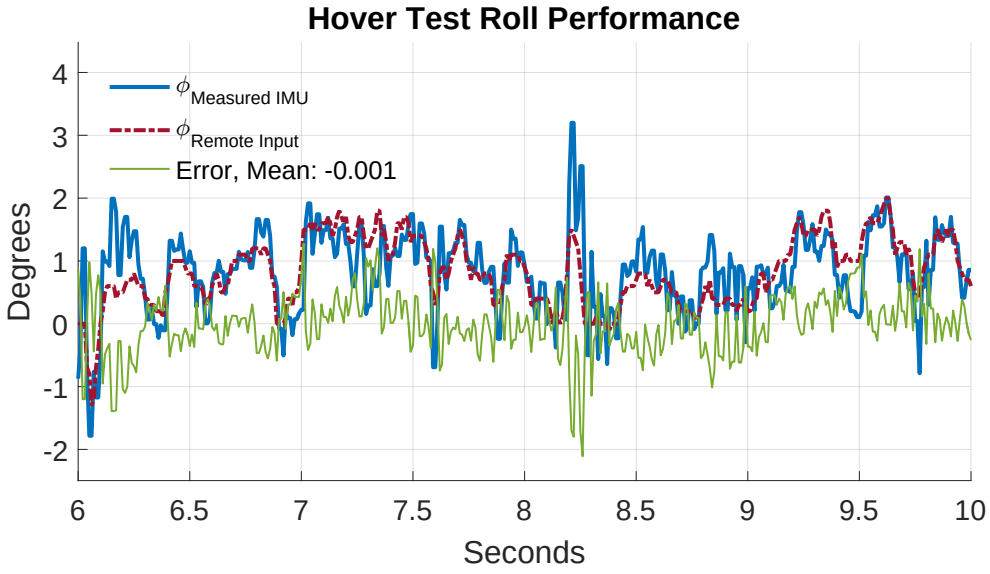


Figure 9.1: Roll performance in hover test of quadcopter

Note that in this flight test, a human was controlling the quadcopter attempting to make the qudacopter hover in place. As a result of this, the user commanded roll, ϕ_{Remote} is not 0. Consequently, the error $\phi_{Remote} - \phi_{Measured}$, can be used as a performance metric. This data was collected through the RF link and sent to a

computer through the USB interface on the remote. For the entire duration of this test, the error averaged to be 0.001^o. However, the standard deviation of the error is 0.4816 degrees. In other-words, roughly 70% of the time the error is within 0.5^o of the target. This noise error is in part due to the noise of the user generated control signal and in part due to jitter/noise in the control system. While pitch and roll have similar performance, yaw had the most control error over time. This is in part due to the control system gains and it is also in part due to the quadcopter design not having a control input that acts directly on yaw. Recall from the physical model that yaw control is achieved by a differential torque from clockwise and counter clockwise spinning motors. In contrast, pitch and roll can be controlled directly by the forces exerted by the propellers on the frame. The control system errors for pitch, roll, and yaw are shown in Fig. **9.2**.

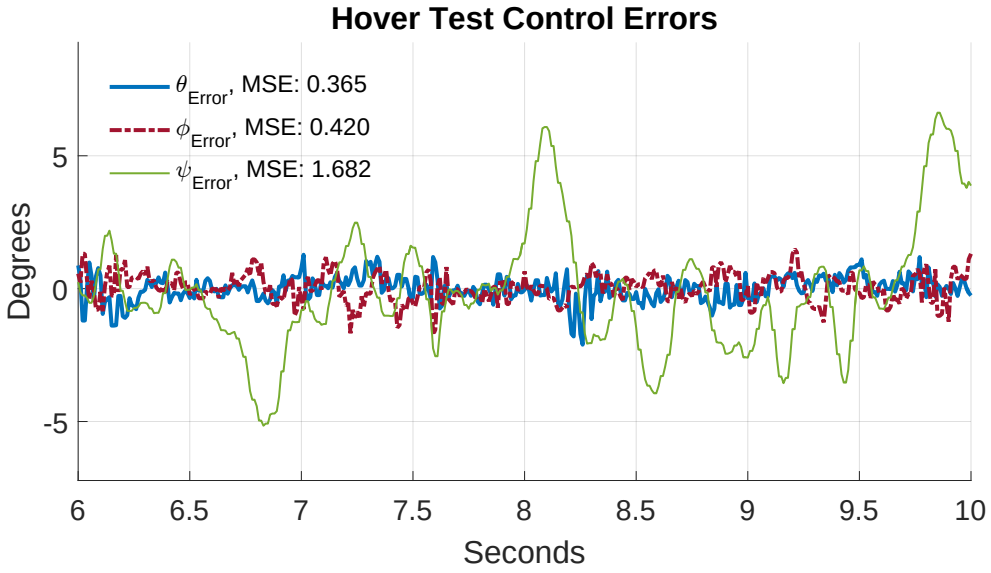


Figure 9.2: Pitch, roll, and yaw performance during hover test of quadcopter.

Using the dataset in Fig. **9.2**, the mean squared error (MSE) was generated for each control axis. The MSE tracks the controllability of each axis. For reference, a flight video of the drone can be seen at [52].

9.2 Conclusions

The main purpose of this thesis was to develop both the hardware and software for a quadcopter system. Chapter 3 developed a control update laws as well as control objectives. Chapter 4 developed a 9 degrees of freedom fusion filter to provide accurate estimations of orientation. Chapter 5 demonstrated a control system to achieve stable auto leveling flight using a series of PID controllers. Chapter 6 overviews the RF hardware involved in creating the transcieving link between the remote and the quadcopter. Chapter 7 specifies the other hardware involved in the system such as the flight controller and electronic speed controllers. Chapter 8 demonstrates theory involved with autonomous flight and using a lidar to maintain position while indoors. Finally, the conclusion demonstrated flight results and performance of the overall system.

With regards to contributions, this thesis provides reference source files, multi-rotor design insights, and a computationally efficient lidar parsing algorithm. Specifically, this thesis provides open source software / hardware files for a functional stabilized auto-leveling flight controller. It also provides open source software / hardware files for a remote controller. It offers design insights into achieving multi-rotor flight with a 9 degree of freedom orientation fusion algorithm as well as a PID control system. Finally, this thesis presents a gradient descent based ICL algorithm that can parse lidar data in real time with limited computational resources.

9.3 Future Work

This section describes potential follow-up work to the current Marq Drone Design. Currently, the IMU filter operates on Euler angles when ideally it should be based upon quaternions to avoid singularities. In addition, quaternions also allow for true linear spherical interpolation where as low pass filtering Euler angles does not produce a linear response. With regards to the control system, the first thing that would be improved is the PID control system performance. The current balancing action is somewhat noisy and the yaw control is lacking. In addition to PID control, other control techniques could be implemented such as linear quadratic regulator (LQR) as well as model predictive control (MPC). Another improvement would be to integrate the designed electronic speed controllers into quadcopter system. A final improvement would be to mount the lidar to the quadcopter and implement the ICL algorithm in real time.

References

[1] Halftermeyer, "Parrot AR.Drone 2.0 Image", 2012, [Online]. Available: https://commons.wikimedia.org/wiki/File:Parrot_AR.Drone_2.0_take-off,_Nevada.jpg.

[2] DJI, "Phantom Image", 2016, [Online]. Available: https://commons.wikimedia.org/wiki/File:DJI_Phantom_4_in_Flight_March_2016.jpg.

[3] HobbyKing, "DJI Inspire 1", 2016, [Online]. Available: http://www.dji.com/product/inspire-1.

[4] C Deets, "Turnigy Micro-X Quadcopter", 2016, [Online]. Available: http://charliedeets.com/turnigy-micro-x-quadcopter-from-hobby-king/.

[5] Hobby King, "Turnigy TrackStar 21.5T Sensored Brushless Motor", 2016, [Online]. Available: http://www.hobbyking.com/hobbyking/store/__71364__Turnigy_TrackStar_21_5T_Sensored_Brushless_Motor_1855KV_ROAR_approved_AR_Warehouse_.html.

[6] Hobby King, "Turnigy D3530/14 1100KV Brushless Outrunner Motor", 2016, [Online]. Available: http://www.hobbyking.com/hobbyking/store/__18225__turnigy_d3530_14_1100kv_brushless_outrunner_motor.html.

[7] A. Aras, "ESC 35A", 2012, [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/d/d1/ESC_35A.jpg.

[8] D. Warren, "Motion Sensing in the Iphone 4", 2010, [Online]. Available: http://www.memsjournal.com/2010/12/motion-sensing-in-the-iphone-4-mems-accelerometer.html.

[9] Senodia, "What is MEMS?", 2013, [Online]. Available: http://senodia.com/technology.aspx?id=275.

[10] R. Dahai, W. Lingqi, Y. Meizhi, C. Mingyang, Y. Zheng, and H. Muzhi, "Design and analyses of a mems based resonant magnetometer", *Sensors*, vol. 9, no. 9, pp. 6951, 2009.

[11] NOAA, "Maps of Magnetic Elements", 2016, [Online]. Available: http://www.ngdc.noaa.gov/geomag/WMM/image.shtml.

[12] N. Ya, "Resin Coated Copper Foils For Building Up PCB", 2008, [Online]. Available: http://www.npc.com.tw/electronic_materials_div/images-1/pdf/08---c4-01-2.htm.

[13] Johanson Technology, "2.45 GHz Impedance matched Balun-Filter for Texas Instruments' CC2500, CC2510, and CC2511 Chipsets", 2012, [Online]. Available: http://www.johansontechnology.com/datasheets/baluns/Balun_Filter_Matched_2450BM15B0003_v4.pdf.

[14] Texas Instruments, "Low-Cost Low-Power 2.4 GHz RF Transceiver", 2016, [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2500.pdf.

[15] S. Winkvist, "Low computational SLAM for an autonomous indoor aerial inspection vehicle", Master's thesis, University of Warwick, the Netherlands, 2013.

[16] Clean Flight, "Clean Flight", 2016, [Online]. Available: http://cleanflight.com.

[17] Open Pilot, "Open Pilot", 2016, [Online]. Available: https://www.openpilot.org.

[18] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor", *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sept 2012.

[19] M. Mueller and D. Raffaello, "A model predictive controller for quadrocopter state interception", in *Control Conference (ECC), 2013 European*. IEEE, 2013, pp. 1383–1389.

[20] S. Lupashin, A. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 1642–1648.

[21] L. Argentim, W. Contrimas, P. Santos, and R. Aguiar, "Lqr and lqr-pid on a quadcopter platform", in *IEEE Int.Conference on Electronics, Informatics and Vision*, 2013.

[22] G. Raffo, M. Ortega, and F. Rubio, "An integral predictive/nonlinear h infinity control structure for a quadrotor helicopter", in *Automatica*, 2013, pp. 1–7.

[23] Pixhawk, "PIXHAWK is the all-in-one unit, combining FMU and IO into a single package", 2016, [Online]. Available: https://pixhawk.org/.

[24] Tau Labs, "Sparky 2", 2016, [Online]. Available: https://github.com/TauLabs/TauLabs/wiki/Sparky2.

[25] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces", in *Int. Journal of Computer Vision*. 1994, pp. 19–25, IEEE.

[26] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time", in *Robotics: Science and Systems Conference*, July 2014.

[27] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing", in *IEEE International Conference on Robotics and Automation*, 2012, pp. 19–25.

[28] C. Hubert, *Electric Machines*, 2nd edition, 2002.

[29] NXP, "3-Phase BLDC Motor Control with Sensorless Back EMF Zero Crossing Detection Using 56F80x", 2016, [Online]. Available: cache.freescale.com/files/product/doc/AN1914.pdf.

[30] NXP, "Sensorless PMSM Field-Oriented Control", 2016, [Online]. Available: http://www.nxp.com/doc/DRM148.

[31] DJI, "Inspire 1", 2015, [Online]. Available:

http://www.dji.com/product/inspire-1.

[32] G Lazaridis, "Pulse Position Modulation and Differential PPM", 2011, [Online]. Available: http://www.pcbheaven.com/wikipages/Pulse_Position_Modulation/.

[33] A. Gibiansky, "Quadcopter Dynamics and Simulation", 2012, [Online]. Available: http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics.

[34] CyPhy, "CyPhy LVL 1", 2015, [Online]. Available: http://www.cyphyworks.com/robots/lvl1/.

[35] R. Hibbeler, *Statics and Dynamics*, 12th edition, 2010.

[36] ST Microchip, "Accelerometer and Gyroscopes Sensors: Operation, Sensing, and Applications", 2015, [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/5830.

[37] ST Microchip, "Tilt measurement using a low-g 3-axis accelerometer", 2013, [Online]. Available: http://www.st.com/web/en/resource/technical/document/application_note/CD00268887.pdf.

[38] V. Bahatia, *Classical Mechanics: With introduction to Nonlinear Oscillations and Chaos*, Narosa Pub House, 1st edition, 1997.

[39] A. Trusov, "Overview of MEMS Gyroscopes: History, Principles of Operations, Types of Measurements", 2011, [Online]. Available: http://www.alexandertrusov.com/uploads/pdf/2011-UCI-trusov-whitepaper-gyros.pdf.

[40] Freescale, "Layout Recommendations for PCBs Using a Magnetometer Sensor", 2013, [Online]. Available: cache.freescale.com/files/sensors/doc/app_note/AN4247.pdf.

[41] Freescale Semiconductor, "Tilt Sensing Using a Three-Axis", 2013, [Online]. Available: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf.

[42] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays", Tech. Rep., University of Bristol, Department of Mechanical Engineering, Apr. 2010.

[43] P. Jan, "Reading a IMU Without Kalman: The Complementary Filter", 2013, [Online]. Available: http://www.pieter-jan.com/node/11.

[44] N. Nise, *Control Systems Engineering*, 6th edition, 2011.

[45] N Zimmerman, "Open source hardware/software quadctoper system", 2016, [Online]. Available: https://github.com/NateZimmer/MarqDrone.

[46] Federal Communications Commission, "Rules and Regulation for Title 47", 2016, [Online]. Available: https://www.fcc.gov/general/rules-regulations-title-47.

[47] C. Balanis, *Antenna Theory: Analysis and Design*, 4th edition, 2015.

[48] Maxim Integrated, "Impedance Matching and the Smith Chart: The Fundamentals", 2002, [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/742.

[49] ArduPilot Team, "Extended Kalman Filter Navigation Overview and Tuning", 2016, [Online]. Available: http://ardupilot.org/dev/docs/extended-kalman-filter.html#extended-kalman-filter.

[50] Neato Robotics, "LaserSmart Mapping & Navigation", 2016, [Online]. Available: https://neatorobotics.com/lasersmart-mapping-navigation.

[51] A. Censi, "An icp variant using a point-to-line metric", in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 19–25.

[52] N Zimmerman, "Marq Drone Flight Video", 2016, [Online]. Available: https://www.youtube.com/watch?v=cArrBF8TGZQ.

[53] S. Madgwick, "Estimation of imu and marg orientation using a gradient descent algorithm", in *IEEE International Conference on Rehabilitation Robotics*, 2011.

[54] B. Wadell, *Transmission Line Design Handbook*, 1st edition, 1991.

[55] K. Lauszus, "Flight Controller for Quad Rotor Helicopter in X-configuration", Master's thesis, Technical University of Denmark, the Netherlands, 2015.

[56] M. Thompson, "Lorentz Force MEMS Magnetometer", 2010, [Online]. Available: http://www-bsac.eecs.berkeley.edu/publications/search/send_publication_pdf2client.php?pubID=1271176542.

[57] I. Ltkebohle, "BWorld Robot Control Software", 2008, [Online]. Available: http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/.

[58] Freescale Semiconductor, "Tilt Sensing Using a Three-Axis", 2013, [Online]. Available: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf.

APPENDIX A

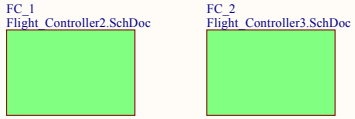
Hardware schematics and PCBs

A.1 Hardware Schematics

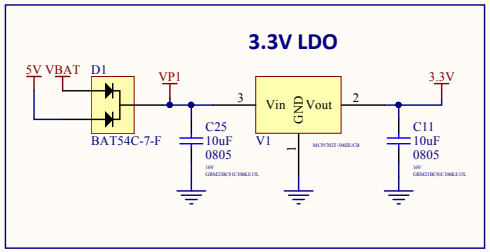
The following are electrical schematics used in the creation of the quadcopter hardware. This hardware was developed in an educational licensed version of Altium at Marquette University. The PCB for these electronic assemblies were manufactured through www.oshpark.com. The boards were then populated and reflowed at the Marquette Embedded Systems Laboratory.

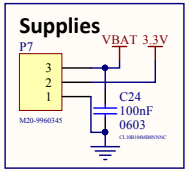
Marq Drone Flight Controller

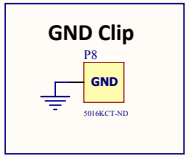
2. Power
3. Micro

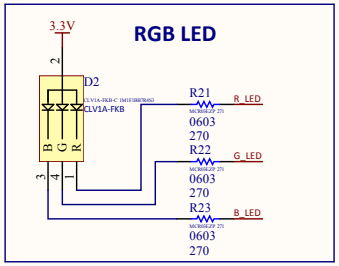


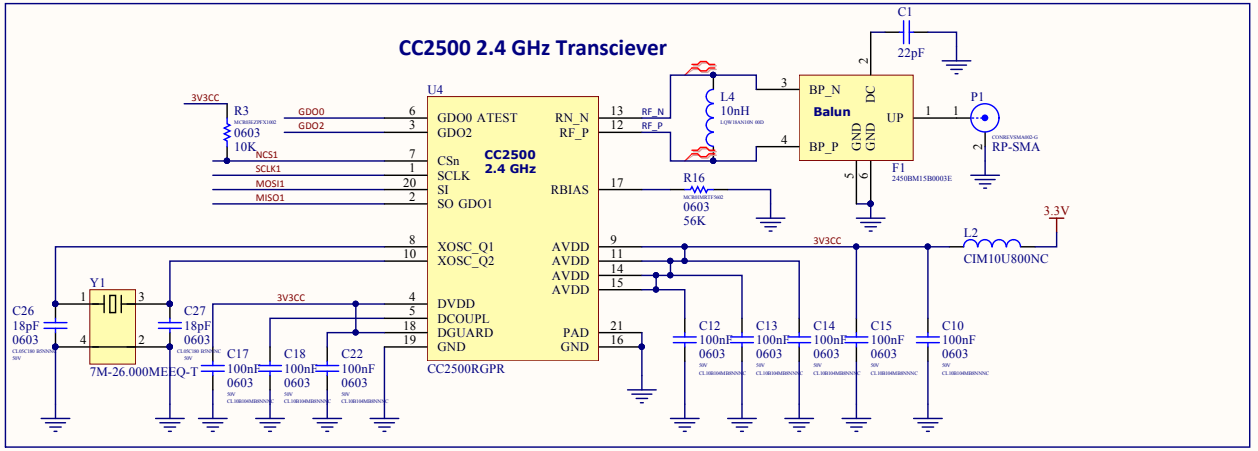
Power & Connectors & Sensors

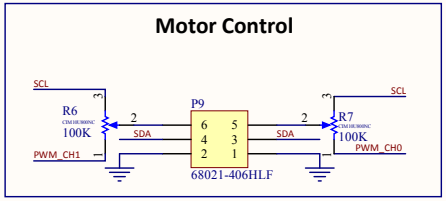


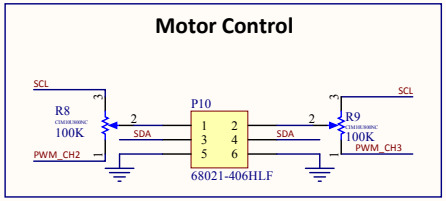


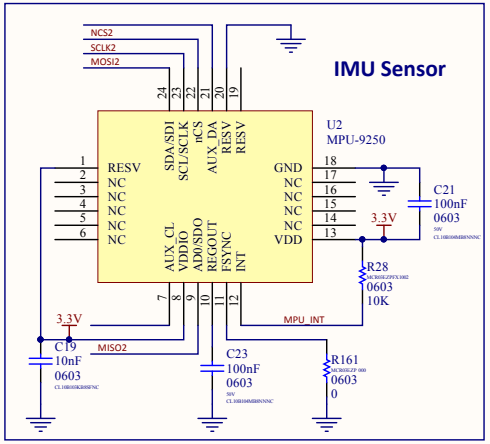


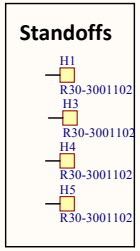




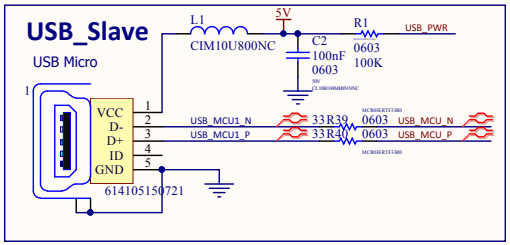


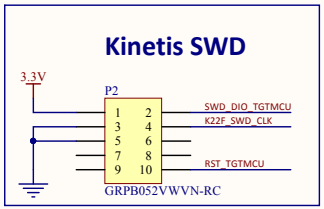


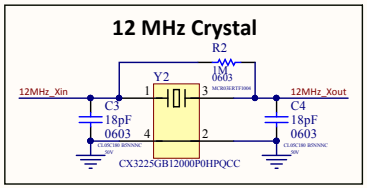


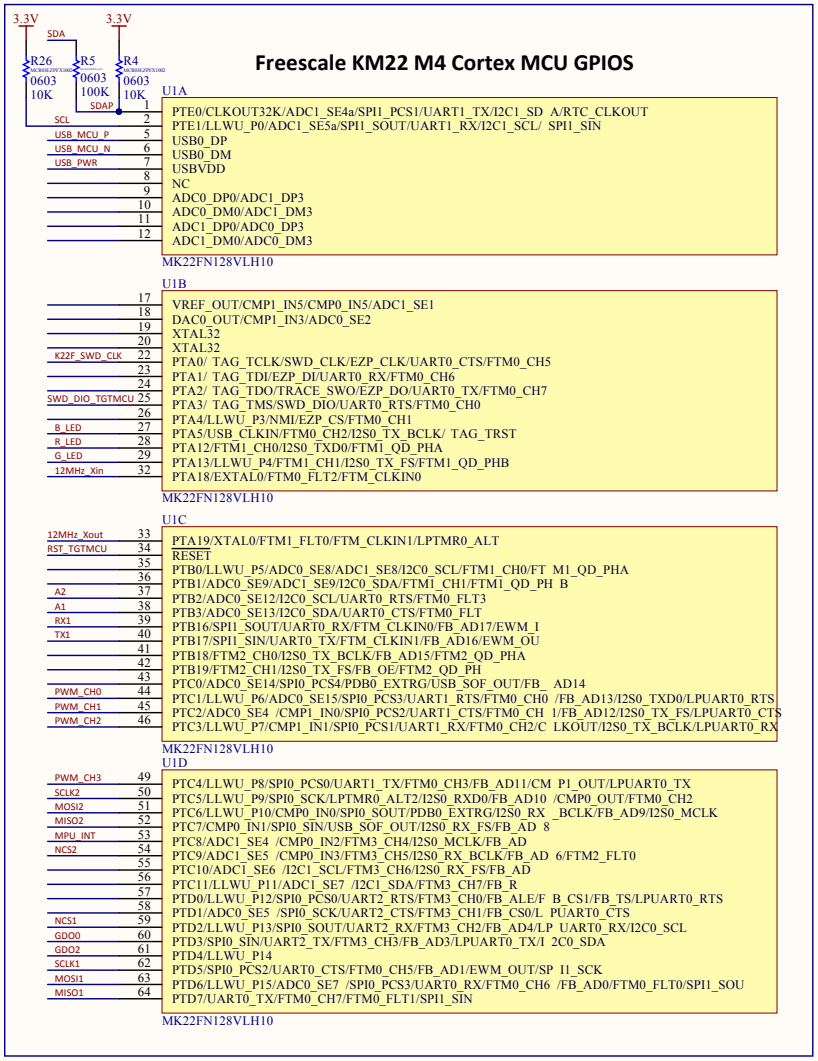


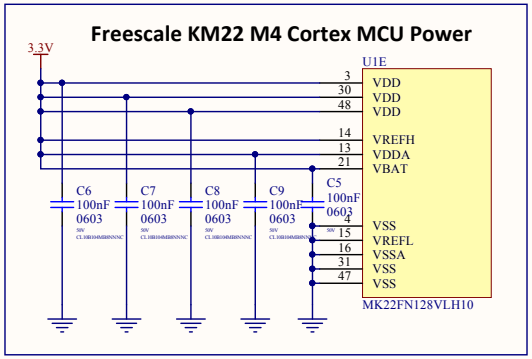
Micro

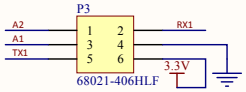

USB_Slave
USB Micro
L1
CIM10U800NC
5V
C2
100nF
0603
R1
0603
100K
USB_PWR
VCC
D-
D+
ID
GND
USB_MCU1_N
USB_MCU1_P
33R39
33R40
0603
USB_MCU_N
USB_MCU_P
614105150721


Kinetis SWD
3.3V
P2
SWD_DIO_TGTMCU
K22F_SWD_CLK
RST_TGTMCU
GRPB052VWVN-RC

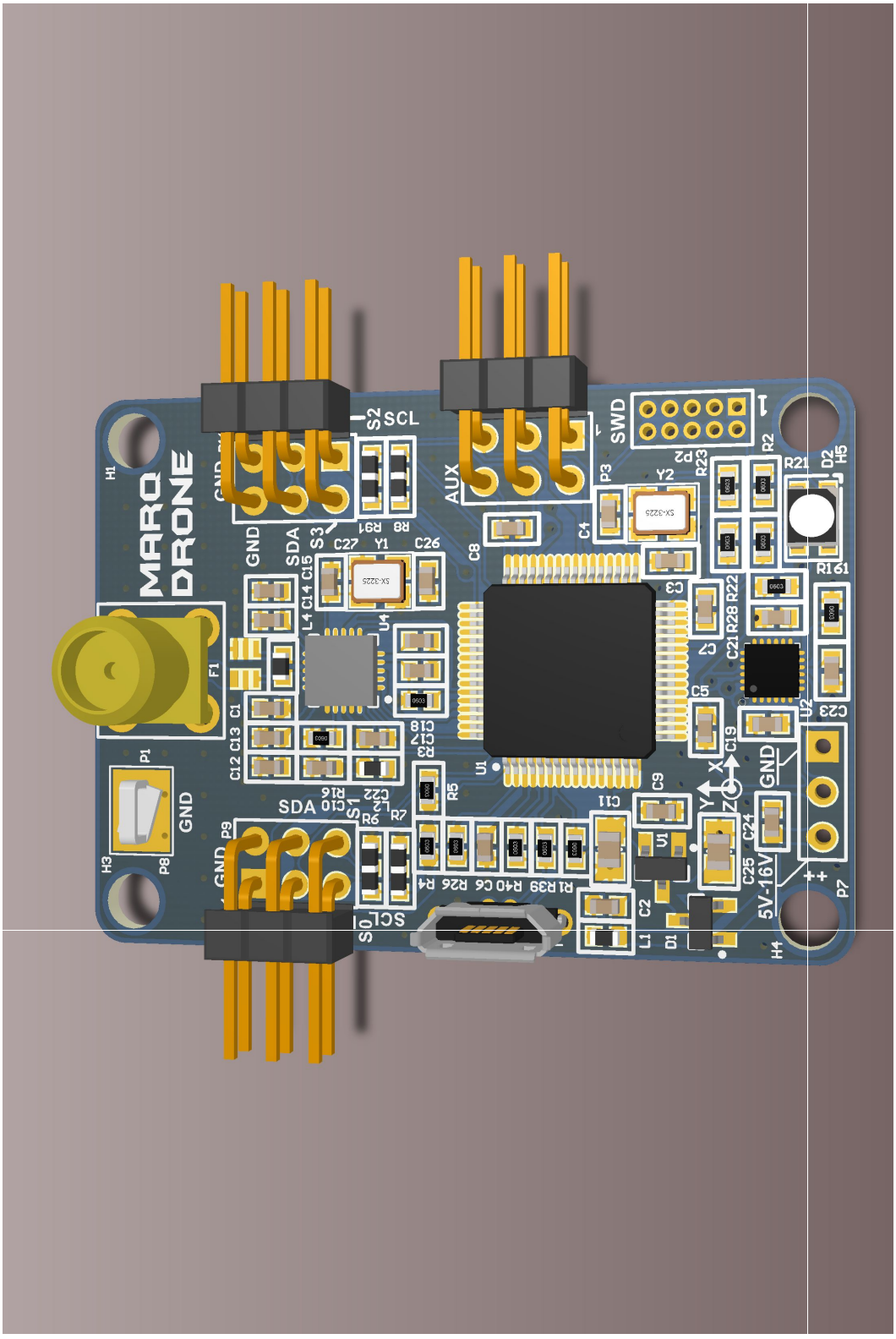

12 MHz Crystal
R2
1M
0603
Y2
12MHz_Xin
12MHz_Xout
C3
18pF
0603
C4
18pF
0603
CX3225GB12000P0HPQCC


Freescale KM22 M4 Cortex MCU GPIOS
3.3V
SDA
R26
R5
R4
0603
10K
100K
SDAP
SCL
U1A
PTE0/CLKOUT32K/ADC1_SE4a/SPI1_PCS1/UART1_TX/I2C1_SD A/RTC_CLKOUT
PTE1/LLWU_P0/ADC1_SE5a/SPI1_SOUT/UART1_RX/I2C1_SCL/ SPI1_SIN
USB0_DP
USB0_DM
USBVDD
NC
ADC0_DP0/ADC1_DP3
ADC0_DM0/ADC1_DM3
ADC1_DP0/ADC0_DP3
ADC1_DM0/ADC0_DM3
MK22FN128VLH10
U1B
VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE1
DAC0_OUT/CMP1_IN3/ADC0_SE2
XTAL32
XTAL32
PTA0/ TAG_TCLK/SWD_CLK/EZP_CLK/UART0_CTS/FTM0_CH5
PTA1/ TAG_TDI/EZP_DI/UART0_RX/FTM0_CH6
PTA2/ TAG_TDO/TRACE_SWO/EZP_DO/UART0_TX/FTM0_CH7
PTA3/ TAG_TMS/SWD_DIO/UART0_RTS/FTM0_CH0
PTA4/LLWU_P3/NMI/EZP_CS/FTM0_CH1
PTA5/USB_CLKIN/FTM0_CH2/I2S0_TX_BCLK/ TAG_TRST
PTA12/FTM1_CH0/I2S0_TXD0/FTM1_QD_PHA
PTA13/LLWU_P4/FTM1_CH1/I2S0_TX_FS/FTM1_QD_PHB
PTA18/EXTAL0/FTM0_FLT2/FTM_CLKIN0
MK22FN128VLH10
U1C
PTA19/XTAL0/FTM1_FLT0/FTM_CLKIN1/LPTMR0_ALT
RESET
PTB0/LLWU_P5/ADC0_SE8/ADC1_SE8/I2C0_SCL/FTM1_CH0/FT M1_QD_PHA
PTB1/ADC0_SE9/ADC1_SE9/I2C0_SDA/FTM1_CH1/FTM1_QD_PH B
PTB2/ADC0_SE12/I2C0_SCL/UART0_RTS/FTM0_FLT3
PTB3/ADC0_SE13/I2C0_SDA/UART0_CTS/FTM0_FLT
PTB16/SPI1_SOUT/UART0_RX/FTM_CLKIN0/FB_AD17/EWM_I
PTB17/SPI1_SIN/UART0_TX/FTM_CLKIN1/FB_AD16/EWM_OU
PTB18/FTM2_CH0/I2S0_TX_BCLK/FB_AD15/FTM2_QD_PHA
PTB19/FTM2_CH1/I2S0_TX_FS/FB_OE/FTM2_QD_PH
PTC0/ADC0_SE14/SPI0_PCS4/PDB0_EXTRG/USB_SOF_OUT/FB_ AD14
PTC1/LLWU_P6/ADC0_SE15/SPI0_PCS3/UART1_RTS/FTM0_CH0 /FB_AD13/I2S0_TXD0/LPUART0_RTS
PTC2/ADC0_SE4 /CMP1_IN0/SPI0_PCS2/UART1_CTS/FTM0_CH 1/FB_AD12/I2S0_TX_FS/LPUART0_CTS
PTC3/LLWU_P7/CMP1_IN1/SPI0_PCS1/UART1_RX/FTM0_CH2/C LKOUT/I2S0_TX_BCLK/LPUART0_RX
MK22FN128VLH10
U1D
PTC4/LLWU_P8/SPI0_PCS0/UART1_TX/FTM0_CH3/FB_AD11/CM P1_OUT/LPUART0_TX
PTC5/LLWU_P9/SPI0_SCK/LPTMR0_ALT2/I2S0_RXD0/FB_AD10 /CMP0_OUT/FTM0_CH2
PTC6/LLWU_P10/CMP0_IN0/SPI0_SOUT/PDB0_EXTRG/I2S0_RX _BCLK/FB_AD9/I2S0_MCLK
PTC7/CMP0_IN1/SPI0_SIN/USB_SOF_OUT/I2S0_RX_FS/FB_AD 8
PTC8/ADC1_SE4 /CMP0_IN2/FTM3_CH4/I2S0_MCLK/FB_AD
PTC9/ADC1_SE5 /CMP0_IN3/FTM3_CH5/I2S0_RX_BCLK/FB_AD 6/FTM2_FLT0
PTC10/ADC1_SE6 /I2C1_SCL/FTM3_CH6/I2S0_RX_FS/FB_AD
PTC11/LLWU_P11/ADC1_SE7 /I2C1_SDA/FTM3_CH7/FB_R
PTD0/LLWU_P12/SPI0_PCS0/UART2_RTS/FTM3_CH0/FB_ALE/F B_CS1/FB_TS/LPUART0_RTS
PTD1/ADC0_SE5 /SPI0_SCK/UART2_CTS/FTM3_CH1/FB_CS0/L PUART0_CTS
PTD2/LLWU_P13/SPI0_SOUT/UART2_RX/FTM3_CH2/FB_AD4/LP UART0_RX/I2C0_SCL
PTD3/SPI0_SIN/UART2_TX/FTM3_CH3/FB_AD3/LPUART0_TX/I 2C0_SDA
PTD4/LLWU_P14
PTD5/SPI0_PCS2/UART0_CTS/FTM0_CH5/FB_AD1/EWM_OUT/SP I1_SCK
PTD6/LLWU_P15/ADC0_SE7 /SPI0_PCS3/UART0_RX/FTM0_CH6 /FB_AD0/FTM0_FLT0/SPI1_SOU
PTD7/UART0_TX/FTM0_CH7/FTM0_FLT1/SPI1_SIN
MK22FN128VLH10

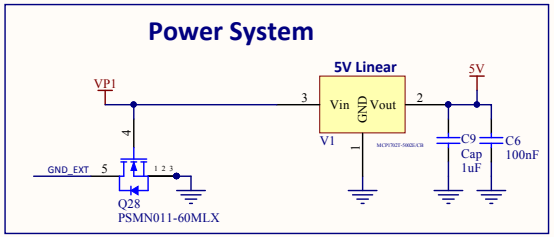

Freescale KM22 M4 Cortex MCU Power
3.3V
U1E
VDD
VDD
VDD
VREFH
VDDA
VBAT
VSS
VREFL
VSSA
VSS
VSS
C6
C7
C8
C9
C5
100nF
0603
MK22FN128VLH10

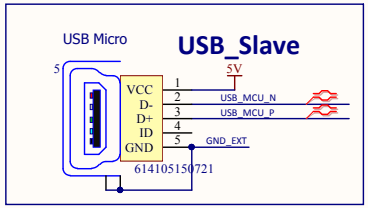

P3
A2
A1
TX1
RX1
3.3V
68021-406HLF

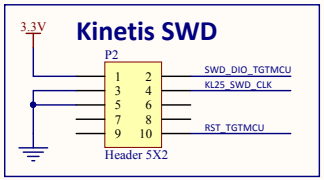
A.5 Flight Controller Schematic

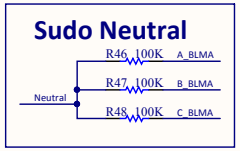


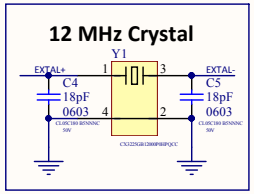
Sensorless Brushless Motor Controller

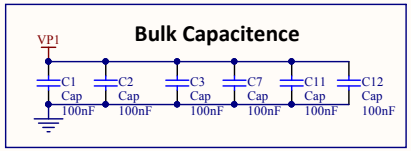


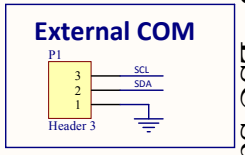


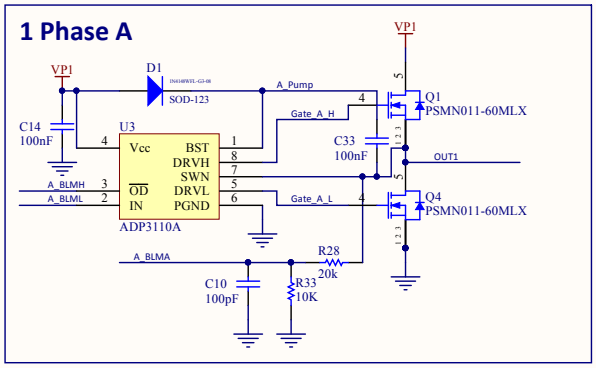


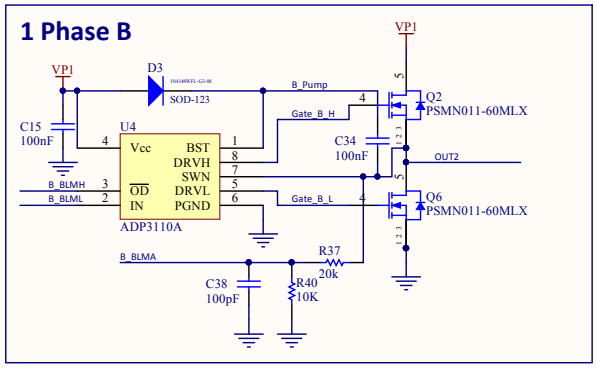


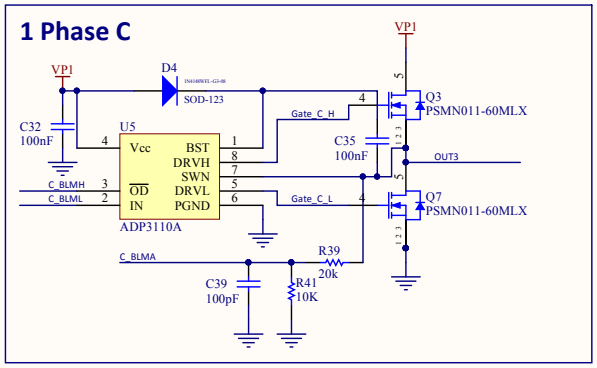


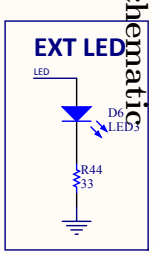


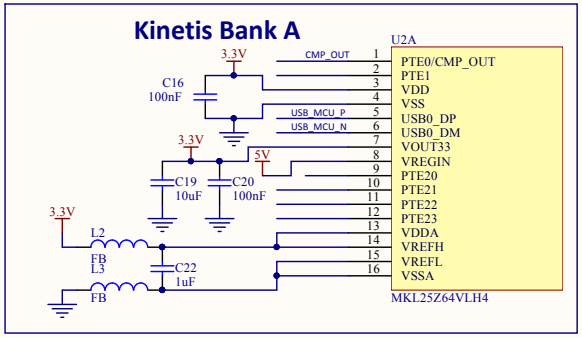


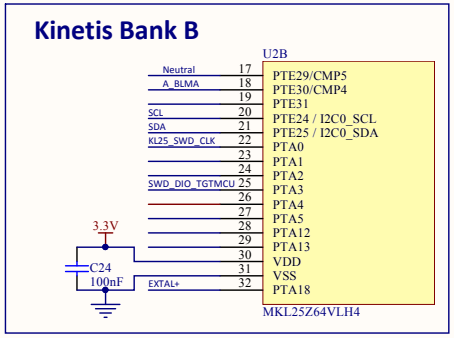


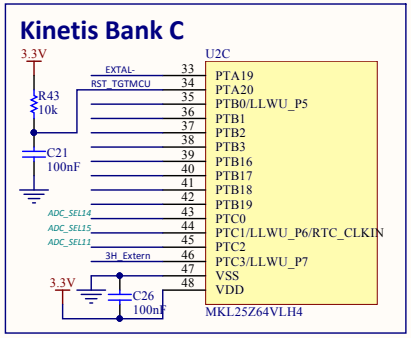


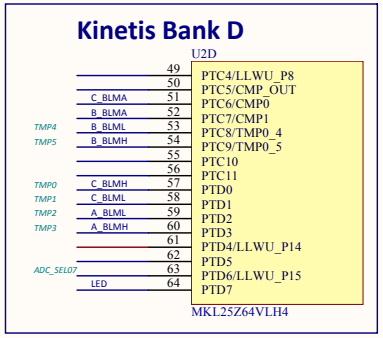










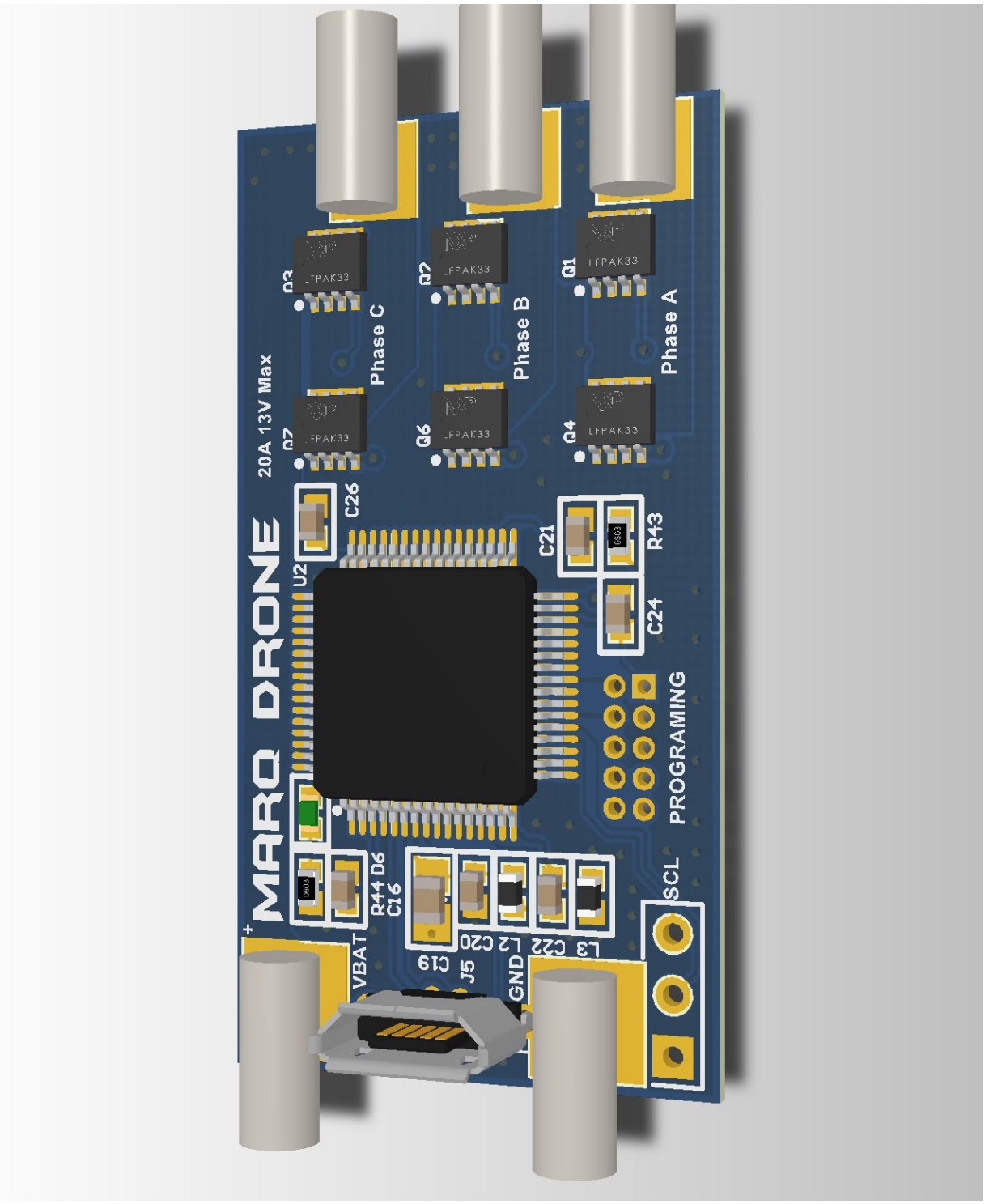


A.6 ESC Schematic

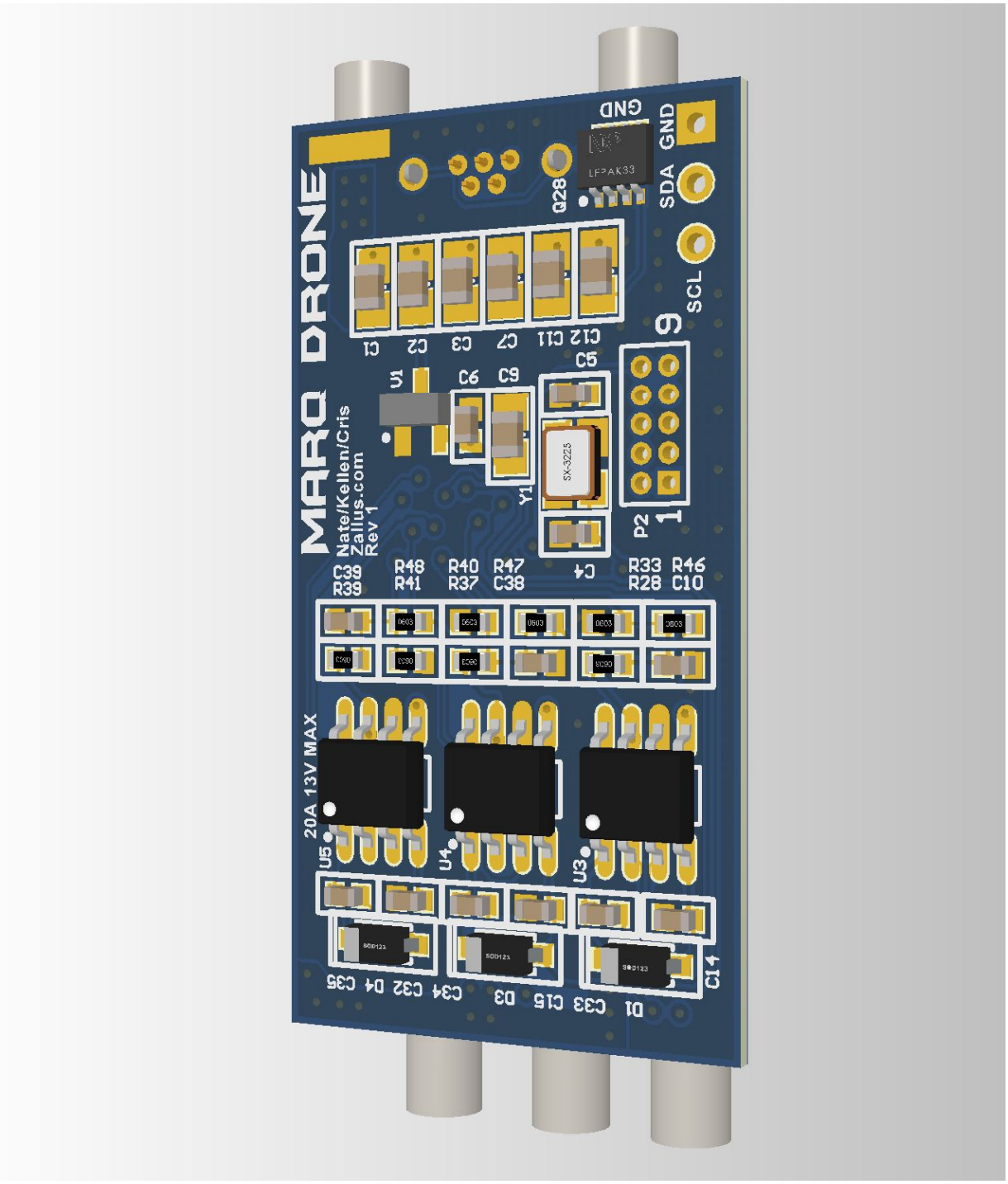
100

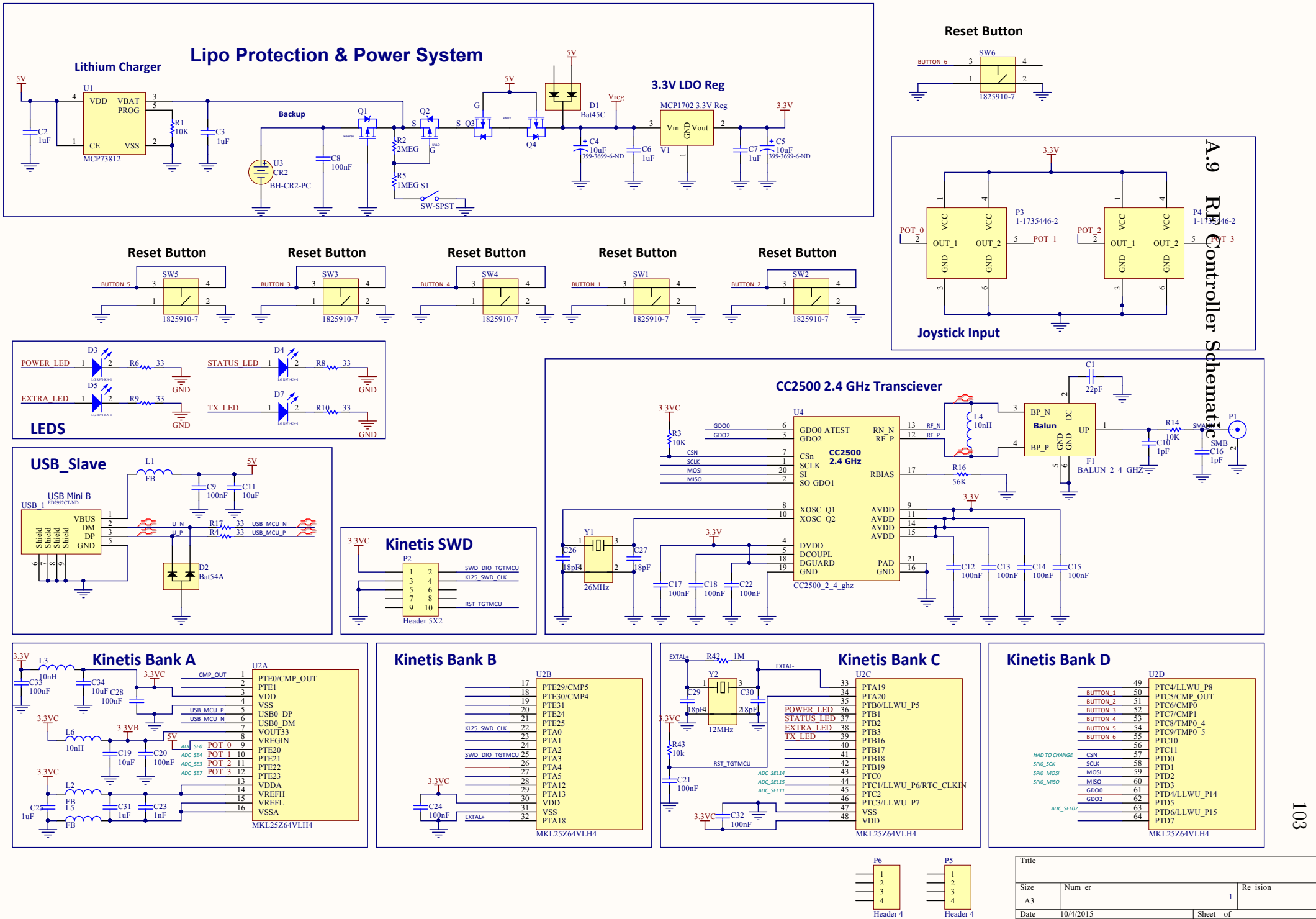
Title	Sensorless Brushless Motor Controller	
Size A3	Num er	Re ision Rev 1
Date 9/7/2015		Sheet of
File C Users .. ESC3.SchDoc		Dra n B Nathan Zimmerman

A.7 ESC Schematic

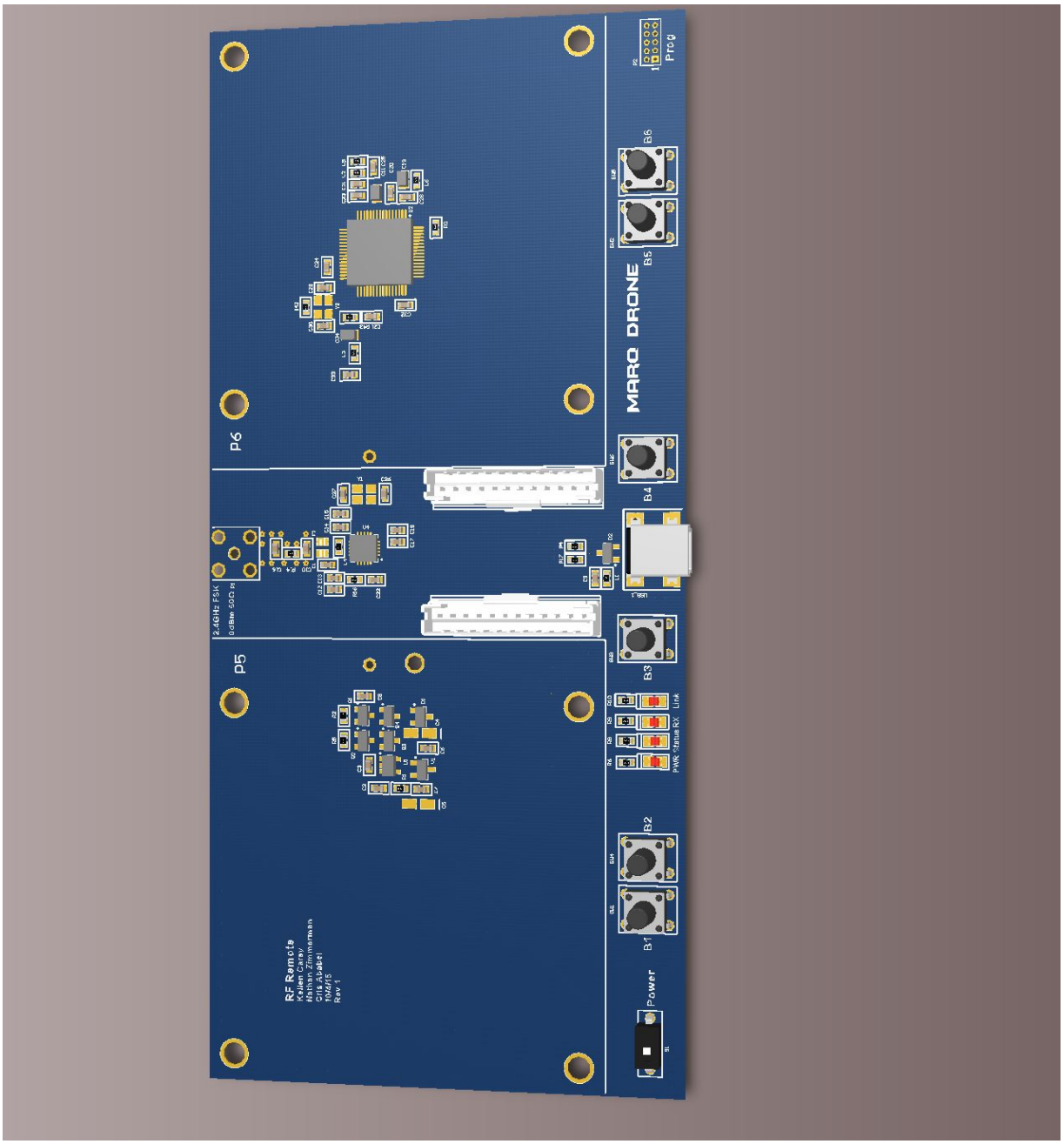


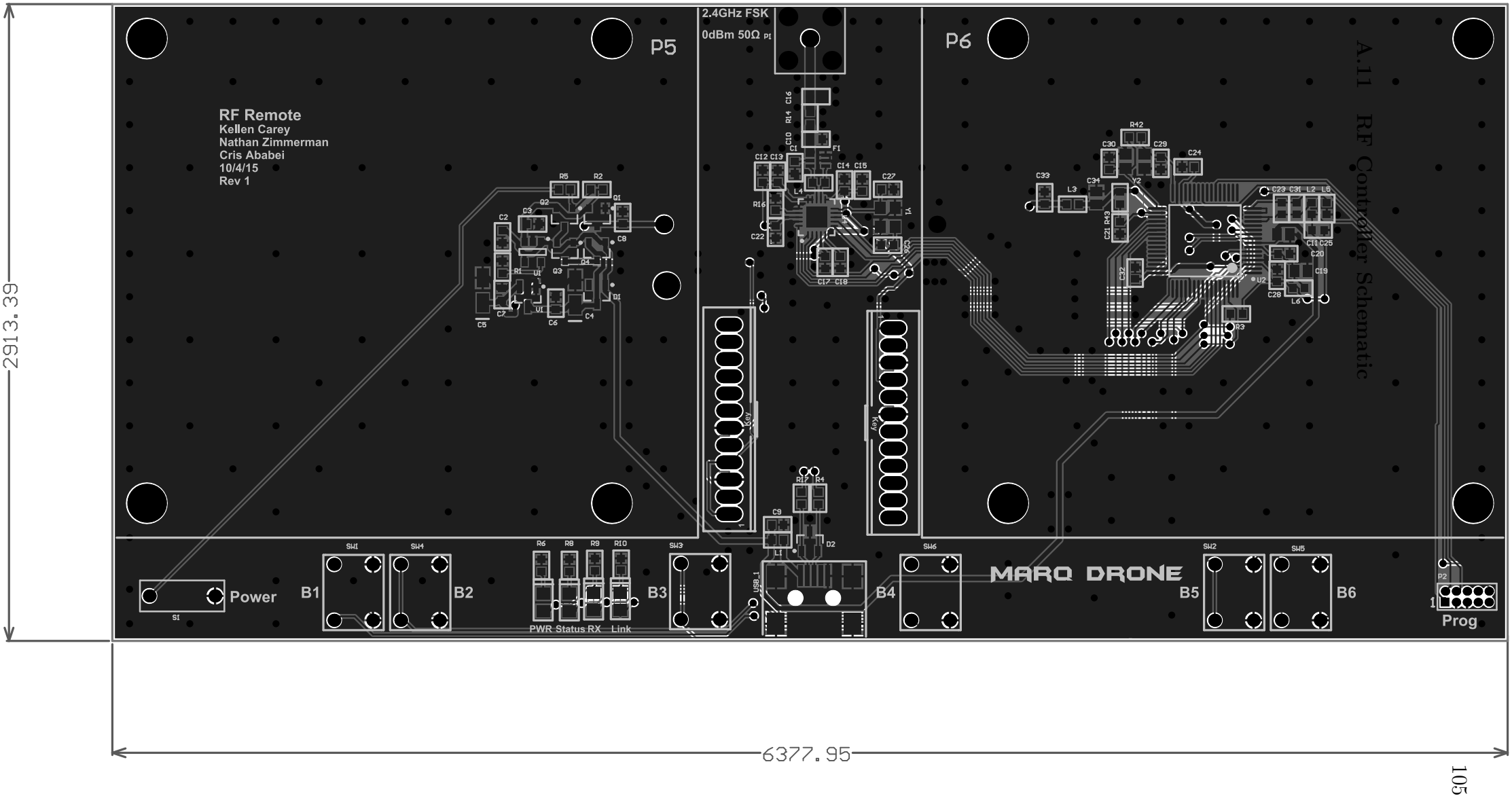
A.8 ESC Schematic




A.9 RF Controller Schematic
Lipo Protection & Power System
Lithium Charger
U1
MCP73812
Backup
U3
BH-CR2-PC
3.3V LDO Reg
MCP1702 3.3V Reg
Reset Button
SW6
SW5
SW3
SW4
SW1
SW2
1825910-7
Joystick Input
P3
P4
1-1735446-2
LEDS
POWER_LED
STATUS_LED
EXTRA_LED
TX_LED
USB_Slave
USB Mini B
CC2500 2.4 GHz Transciever
CC2500_2_4_ghz
Kinetis SWD
Header 5X2
Kinetis Bank A
Kinetis Bank B
Kinetis Bank C
Kinetis Bank D
MKL25Z64VLH4
P6
P5
Header 4
Title
Size
A3
Num er
1
Re ision
Date
10/4/2015
Sheet of
File
C Users .. RF Controller.SchDoc
Dra n B
Kellen Care
103

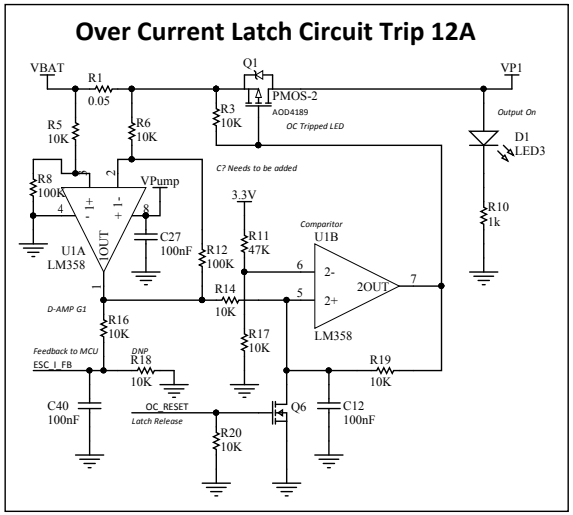
A.10 RF Controller Schematic

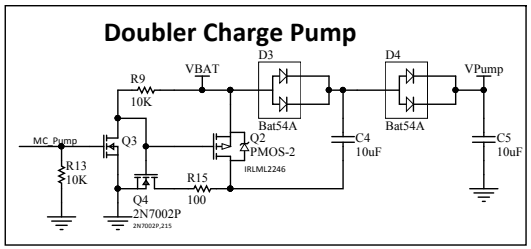


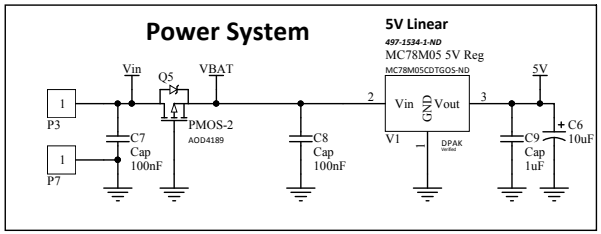

RF Remote
Kellen Carey
Nathan Zimmerman
Cris Ababei
10/4/15
Rev 1
P5
P6
2.4GHz FSK
0dBm 50Ω
A.11 RF Controller Schematic
Power
B1
B2
B3
B4
B5
B6
PWR Status RX Link
Prog
MARQ DRONE
2913.39
6377.95
105

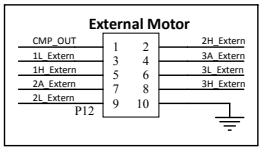
Sensorless 10A ESC REV2.0

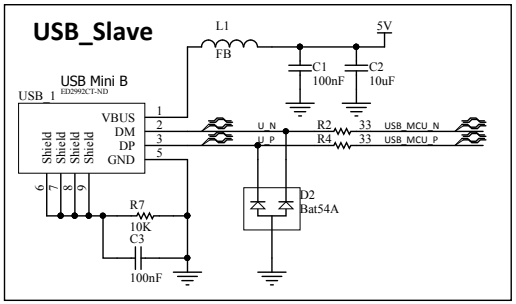
Nathan Zimmerman

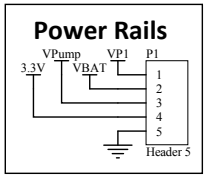


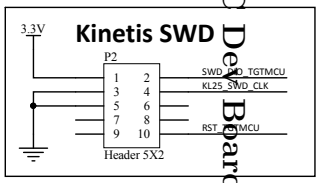


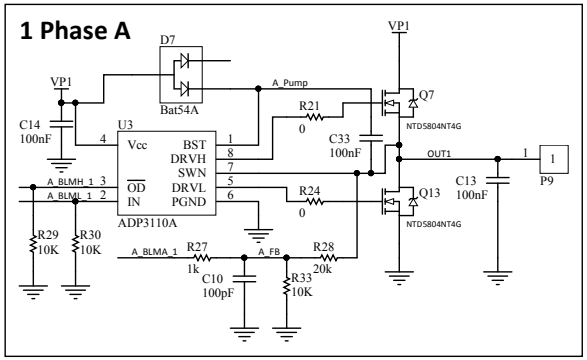


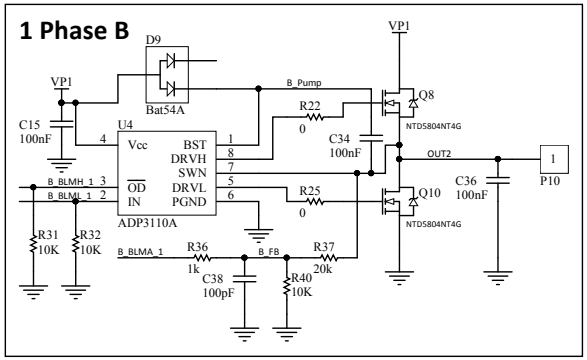


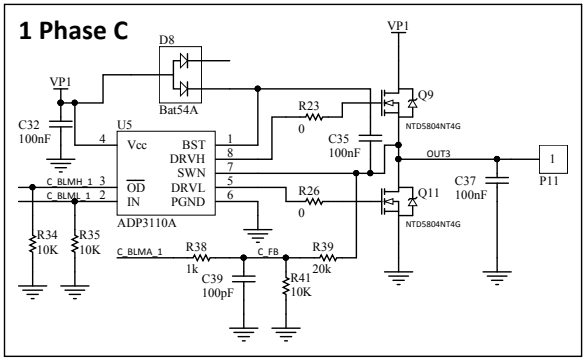


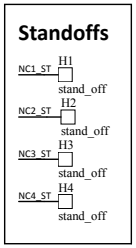


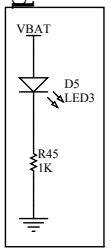


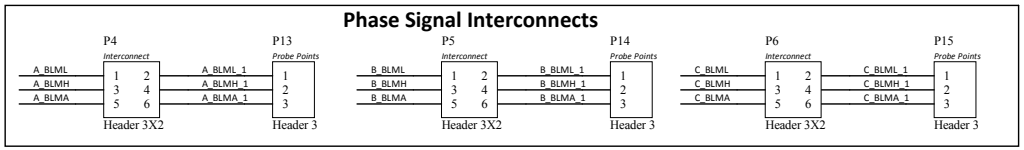


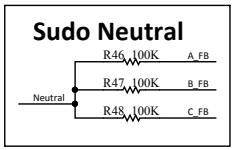


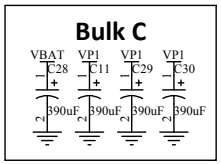


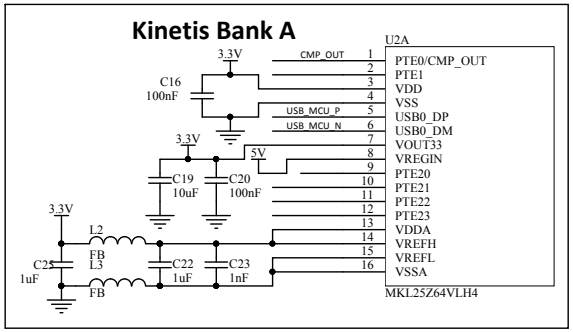


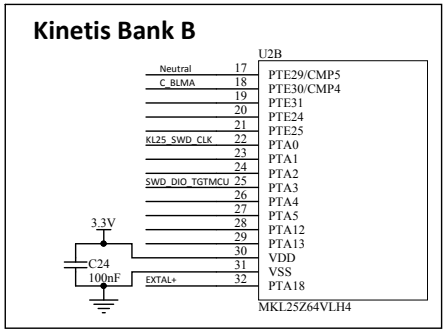


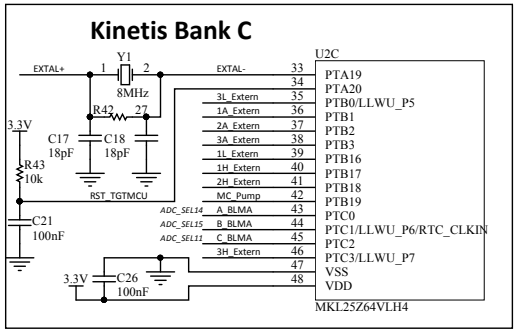


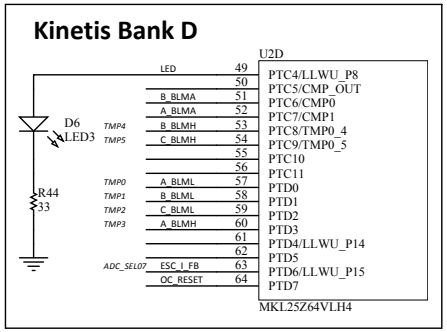








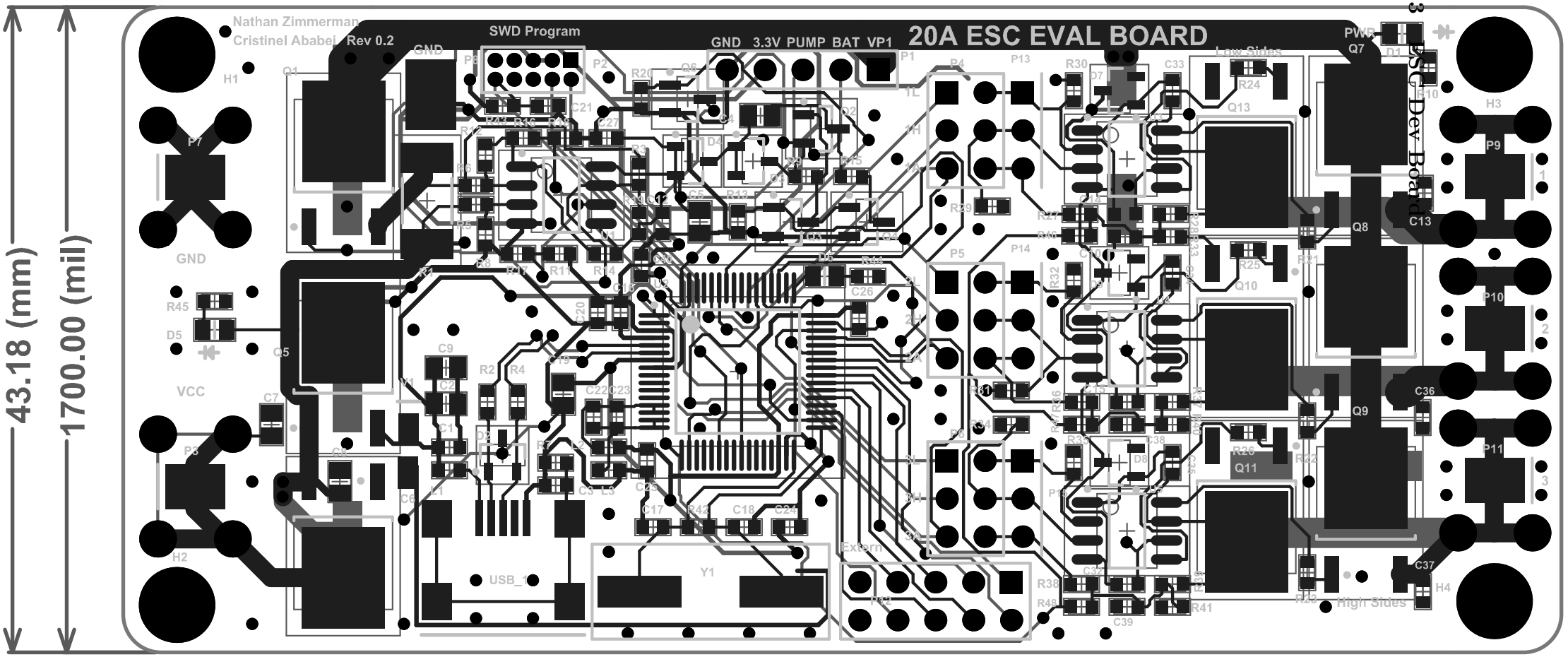




A.12 ESC Dev Board

106

ESC Rev 2



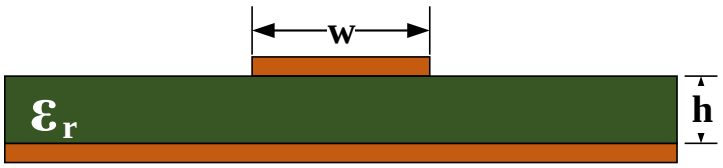
Nathan Zimmerman
Rev 2

APPENDIX B

Transmission Line Calculations

B.1 Microstrip Line Z_o Calculation

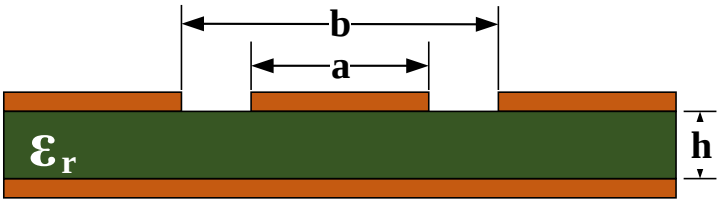
The following are empirical formulas used to calculate the characteristic impedance of a microstrip line where w is the width of the track, ϵ_r is the relative permittivity, t (not shown) is the copper weight, and h is the height of the dielectric[54]. Care should be taken that the correct relative permittivity is selected with respect to the frequency of operation.



$$
\begin{aligned}
\epsilon_{eff} &= \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2} \cdot \left[\frac{1}{\sqrt{1 + \frac{12h}{W}}} + 0.04\left(1 - \frac{W}{h}\right)^2 \right] \\
\Delta W &= \frac{t}{\pi} \cdot ln\left[\frac{4e}{(t/h)^2 + \left(\frac{1/\pi}{w/t+1.1}\right)^2} \right] \\
\Delta W' &= \Delta W \left(\frac{1 + 1/\epsilon_r}{2} \right) \\
W' &= W + \Delta W' \\
Z_o &= \frac{120\pi}{2 \cdot \sqrt{2} \cdot \pi \cdot \sqrt{\epsilon_r + 1}} \cdot ln\Bigg(1 + \\
&\frac{4h}{W'} \cdot \left[\frac{14 + 8/\epsilon_{eff}}{11} \cdot \frac{4h}{W'} + \sqrt{(\frac{14 + 8/\epsilon_{eff}}{11})^2 \cdot (\frac{4h}{W'})^2 + \frac{1 + 1/\epsilon_{eff}}{2} \cdot \pi^2} \right] \Bigg)
\end{aligned}
\tag{B.1}
$$

B.2 Coplanar Waveguide Z_o Calculation

The following are empirical formulas used to calculate the characteristic impedance of a coplanar waveguide where a is the width of the track, b is the width of the waveguide, and ϵ_r is the relative permittivity[54]. Other variables used in this equation are intermediate variables and K is an elliptical integral.



$$
\begin{aligned}
k &= \frac{a}{b} \\
kl &= \frac{tanh(\frac{\pi \cdot a}{4 \cdot h})}{tanh(\frac{\pi \cdot b}{4 \cdot h})} \\
kl' &= \sqrt{1 - kl^2} \\
k' &= \sqrt{1 - k^2} \\
K(k) &= \int_0^{\pi/2} \frac{1}{\sqrt{1 - k^2 \cdot (sin(\theta))^2}} d\theta \\
\epsilon_{eff} &= \frac{1 + \frac{\epsilon_r \cdot K(k') \cdot K(kl)}{K(k) \cdot K(kl')}}{1 + \frac{K(k') \cdot K(kl)}{K(k) \cdot K(kl')}} \\
Z_o &= \frac{60\pi}{\sqrt{\epsilon_{eff}}} \cdot \frac{1}{\frac{K(k)}{K(k')} + \frac{K(kl)}{K(kl')}}
\end{aligned}
\tag{B.2}
$$