

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences



Development of an Open Architecture Quadrotor Micro Air Vehicle:
Manufacture, Hardware and Software Integration.

A thesis presented by

Pedro Antonio Guerrero Cardoso

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of:

Master of Science

With

Specialty in Manufacturing Systems

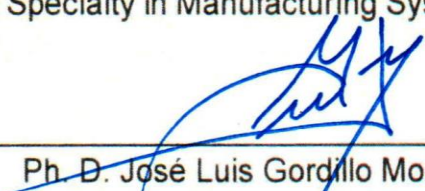
Monterrey, Nuevo León, June 2019

Instituto Tecnológico y de Estudios Superiores de Monterrey

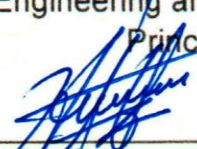
Campus Monterrey

School of Engineering and Sciences


The committee members hereby, certify that have read the thesis presented by Pedro Antonio Guerrero Cardoso and that is fully adequate in scope and quality as a partial requirement for the degree of Master of Science with Specialty in Manufacturing Systems



Ph. D. José Luis Gordillo Moscoso
Tecnológico de Monterrey
School of Engineering and Sciences
Principal Advisor




Ph. D. Herman Castañeda Cuevas
ITESM, Monterrey, Nuevo León
Co-Advisor



Ph. D. Alejandro Enrique Dzul López
ITL, Torreón, Coahuila
Committee Member





Ph. D. Rubén Morales Menéndez
Dean of Graduate Studies
School of Engineering and Sciences

Monterrey, Nuevo León, December 10th, 2019

Declaration of Authorship

I, Pedro Antonio Guerrero Cardoso, declare that this thesis titled “Development of a Quadrotor MAV and Codification of the Control Equations” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly in candidature for a research degree at this University
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- When I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what have contributed by myself.



Pedro Antonio Guerrero Cardoso
Monterrey, Nuevo León, June 2019

©2019 by Pedro Antonio Guerrero Cardoso
All rights reserved

Dedication

This work is dedicated to:

Pedro Marcelo Guerrero Montes and María Teresa Cardoso Alvarado, my parents.

Raúl Javier Guerrero Cardoso and María Teresa Guerrero Cardoso, my siblings.

Andrea Alejandra Delgado Tallavas, my girlfriend.

My Friends made during my graduate studies and walked with me along this torture
(just kidding).

Thanks for all your patience, help, support and words during all this path. With your help
I could finish this work and get prepared for the next goal.

Special dedication to María Guadalupe Alvarado Peza, my grandmother that recently
passed away.

Acknowledgements

I want to express my feelings of gratitude to my professors during this studies and project. They gave to me the knowledges to improve and always try to reach a goal better than the proposed.

A special thanks to PhD. Herman Castañeda Cuevas, my thesis co-advisor, that borrowed to me many information sources to understand and complete the project and taught me more about robotics and control systems.

I want to thank to my class partners, specially to Armando Miranda, Salvador Leal and Fermín Aragón that helped me when some of the analysis were too complex to me and gave me more information to reach a better result.

Thanks to Manuel Sinco, Jesús Silva, Jesús Sánchez, Jorge Arizaga, Alexis Sánchez, Ramiro Campos, Benjamin Benzig, Thomas Somik, Henry Orozco, Román Flores, Anelise Chapa, Denisse Hernández and Rico Zia, everyone involved in this project as part of their professional formation.

My most special thanks to my family that supported me emotionally along this path to reach another step on my professional life.

Thanks to Tecnológico de Monterrey and CONACyT for giving me the opportunity through the scholarship of accomplish this personal and professional goal.

Total thanks to everyone.

Development of an Open Architecture Quadrotor Micro Air Vehicle: Manufacture, Hardware and Software Integration.

by

Pedro Antonio Guerrero Cardoso

Abstract

The thesis addresses the development and manufacturing methodology of a Micro Aerial Vehicle of the Quadrotor (Quad-MAV). The software employed to design the electronic diagram, the layout of the Printed Circuit Board (PCB) and the embedded software of the drone, were from open source, allowing free access.

The proposal of this thesis is to demonstrate that it is feasible to create a Quad-MAV with low cost components previously selected and using free software for the development of the product.

To prototype, the PCB layout and the electronic diagram are developed on Autodesk EAGLE and then the Gerber files are processed by the LPKF CircuitPro to manufacture the copper wires over a FR4 plate and later the Surface Mounted Devices (SMD) are placed and soldered on their respective space over the board.

The program of the microcontroller used in the prototype allows a stepped process to read the instructions of the Wi-Fi communication module and the data of the current state provided by the Inertial Measurement Unit (IMU), both processes, data and solve the control equations, allows that the system works. The code is written in C using the IDE MPLAB® X, the XC32 compiler and the firmware development tool, MPLAB® Harmony. The external modules, namely the Wi-Fi communication module and the IMU circuit, are coded and programmed using as base the programs provided in Arduino.

The experiments are conducted using a computer as a ground station to send the data to the Quad-MAV to ensure the communication and the integration of all the components in the final prototype of the board. Such experiments suggest that the construction of a Quad-MAV using low cost and commercial components is feasible giving access to an open source drone that can be adapted by the end-user to fulfill specific purposes.

List of Figures

Figure 1.1 A3Rs drone from Exyn Technologies used for gold mining.....	2
Figure 1.2 Basic Elements of a drone	2
Figure 1.3 Methodology of the Quad-MAV development.....	7
Figure 2.1 Quadrotor configurations.....	11
Figure 2.2 Final Functional Prototype.....	12
Figure 2.3 Aircraft angles and referential frame	13
Figure 2.4 Motor labeling and rotation for cross configuration in this thesis	14
Figure 2.5 Roll viewed from the front of the drone	15
Figure 2.6 Pitch viewed from the right side of the drone	16
Figure 2.7 A negative yaw seen from above the drone	16
Figure 2.8 VTOL and Hover	17
Figure 2.9 PD Controller Closed- Loop Diagram.....	18
Figure 3.1 Elements and connections of the quadrotor	22
Figure 3.2 Early version of the MAV PCB	23
Figure 3.3 New LEDs and power phases placement	24
Figure 3.4 Comparison of the input and output paths between regulators	25
Figure 3.5 Verification of the voltage regulator.....	26
Figure 3.6 Top layout with the surrounding paths	26
Figure 3.7 Places to fulfill with ProConduct	28
Figure 3.8 Continuity verification process.....	29
Figure 3.9 Component assembly verification step.....	29
Figure 3.10 Quadrotor motor numbering and corresponding pins.....	32
Figure 3.11 Data chains a) The longest case b) The shortest case	35
Figure 3.12 Simplified flow chart	36
Figure 4.1 IMU-PIC32-GND station communication setup	44
Figure 4.2 Data received from the PIC32 into the computer	45
Figure 4.3 Second stage setup	45
Figure 4.4 Console command and response from the PIC32 using PuTTY	46
Figure 4.5 Adjustment of PWM with command transmission.	46
Figure 4.6 PWM regulated on the LEDs (motor pins)	46
Figure 4.7 Normal Distribution of the sample of times.....	47
Figure 4.8 Experiment setup for frequency of the PWM and timer 4 period	48
Figure 4.9 First PWM frequency measurement.....	49
Figure 4.10 First Timer 4 frequency measurement.....	50
Figure 4.11 PWM frequency at 20 kHz.....	50
Figure 4.12 Period of the timer 4 for the FSM tasks	51
Figure 4.13 Testing setup to measure the force delivered by the motors.....	52
Figure 4.14 a) Drone anchored b) Drone free for final test.....	53
Figure A.1 Setting the XC32 compiler on the X IDE	59
Figure A.2 Starting the project.....	59
Figure A.3 Selecting the adequate path to the Harmony installation folder	59

Figure A.4 Configuration of the functions and pins to be used on the project	60
Figure A.5 Options to be modified are highlighted in blue	61
Figure A.6 Defining the number of instances	61
Figure A.7 Defining the OC instances and their properties	62
Figure A.8 Defining the number of timer instances	62
Figure A.9 Adjustment of timer Parameters	62
Figure A.10 Definition of the USART instances and disabling the interrupts	63
Figure A.11 Adjustment of the USART instance options	63
Figure B.1 Fiducials Placement	66
Figure B.2 Selection of the Insulation and Contour Ruting	67
Figure B.3 Board Production Wizard starting page	67
Figure B.4 Quantity of boards to be produced	68
Figure B.5 Finished product	68
Figure B.6 Placement of the ProConduct	69
Figure B.7 Heat treatment of the ProConduct	69
Figure B.8 Placement of the soldering	70
Figure B.9 Plate with all the component pads with solder	70
Figure B.10 Placement of the components	71
Figure D.1 Graph showing the real and ideal behavior of the Motor 1	75
Figure D.2 Graph showing the real and ideal behavior of the Motor 2	76
Figure D.3 Graph showing the real and ideal behavior of the Motor 3	76
Figure D.4 Graph showing the real and ideal behavior of the Motor 4	77

List of Tables

Table 1.1 Comparison of small-scale vehicles ^{1, 2}	3
Table 2.1 Motor-Direction-Orientation relationship over the space	14
Table 3.1 Comparison of the different IMU boards.....	24
Table 3.2 Process comparison table	27
Table 3.3 Parameters of the MAV	30
Table 3.4 Assignment of the Instance, Module and Pin for each motor	32
Table 3.5 Control gains values.....	39
Table 3.6 Motor characteristics at 3.2 V	41
Table 3.7 Constants parameters employed to find the equation 42	41
Table 4.1 Data from the normal distribution	47
Table 4.2 Values obtained from the motor force measurement.....	52
Table A.1 Autogenerated functions employed.....	64
Table C.1 Cost of the components of the drone	73
Table D.1 Measurement obtained from the motors	75

Contents

Abstract	ix
List of Figures	xi
List of Tables	xiii
1. Introduction	1
1.1 Context and background	1
1.2 Quad-MAV definition	3
1.3 State of the art	3
1.4 Problem statement.....	4
1.5 Objective	4
1.6 Methodology	5
1.7 Achievement and contributions	8
1.8 Composition	8
2. Quad-MAV Behavior and Control	11
2.1 Vehicle's configuration	11
2.1.1 Quadrotor configuration	11
2.2 Quadrotor attitude, orientation and altitude behavior	12
2.2.1 Roll and movement along the "Y" axis	15
2.2.2 Pitch and movement along the "X" axis	15
2.2.3 Yaw motion	16
2.2.4 Altitude and movement along "Z" axis	17
2.3 Control equations.....	18
3. Design and Manufacture	21
3.1 Electronic design.....	21
3.1.1 Electronic diagram and interactions.....	21
3.1.2 Replaced Elements.....	22
3.1.3 Modification of the electronic layout.....	24
3.2 Manufacturing process.....	27
3.3 Physical parameters of the drone	30
3.4 Development of the embedded firmware	30
3.4.1 PWM Implementation	31
3.4.2 FSM definition.....	33
3.5 Adjustment of the real model embedded parameters	37
3.5.1 Adjustment of the periods of the timers	37
3.5.2 Adjustment of constants in control equations.....	38
3.6 Discrete control equations and motor velocity calculus.....	39

4. Experiments and analysis	43
4.1 Data collection test.....	44
4.1.1 Data reception and transmission from between the IMU, WiFi and microcontroller	44
4.2 Timers testing	48
4.2.1 PWM frequency and timer 4 period	48
4.3 Prototype testing	51
4.3.1 Slope for the control of motor velocity.....	51
4.3.2 Full prototype test	53
5. Conclusion	55
5.1 Contributions.....	55
5.2 Limitations.....	56
5.3 Future Work	56
Appendix A MPLAB® Configuration and Code Functions	58
A.1 MPLAB® configuration	58
A.2 Main autogenerated functions	63
Appendix B Changes on the Manufacturing Process	66
B.1 Copper Roughing Process.....	66
B.2 ProConduct Process.....	68
B.3 Placement of the components	69
Appendix C Cost of the Components	73
Appendix D Motor Force Measurement	75
Bibliography	79
Curriculum Vitae	82

Chapter 1

1 Introduction

This section contains information and context related to the Unmanned Aerial Vehicles (UAVs) and the state of the art related to the micro scaled vehicles. The problem is stated, and the objectives, methodology and contributions of the thesis are presented in this chapter.

1.1 Context and background

An Unmanned Aerial Vehicle, shortened as UAV, can be defined as a machine that can travel through the air without the necessity of a human crew aboard as the first part of the name indicates it [1].

The UAVs and its modern uses started since the World War I according to [2], where the aerial torpedo was developed. Nowadays the UAVs are used in many areas, from military to research and even as a hobby. The drones on the military service are employed for tasks as intelligence and recognition, collecting data of the ground as the one in Figure 1.1, while in research tasks are used for improving the functionality of different algorithms like vision or flight stabilization, and as hobbies are used for photography and recording on areas or angles that for the human are not safe or impossible to reach. The UAVs currently can be used on numerous tasks, all depending on the specifications on the drone used and the purpose of the task that is programmed.

According to the initial team on the development of this project, on [3] and using other models of the UAVs in the state of the art, the elements listed below, and represented in the Figure 1.2, are present in all the developments of drones:

- Propulsion system. The actuators that allow the vehicle to fly and perform maneuvers in the air, it can be turbines like in the airplanes and most fixed-wing aircrafts or can be motors with propellers like in the helicopters and multi-rotor aircraft.
- Battery. Supplies the power to all the modules allowing them to work properly.
- Sensors. Measures the current state of the aircraft and gives the feedback to the main processing unit.
- Main Controller Unit (MCU). This unit is where all the information provided by the sensors and the ground station is processed through algorithms to control the state of the vehicle.
- Ground station. Base for the flying crew, including the pilot, that ensures and supervises the functionality of the UAV.

- Communication module. Module that receives the data from the ground station, sends the coded information to the MCU and feedbacks the current state of the vehicle to the ground station.

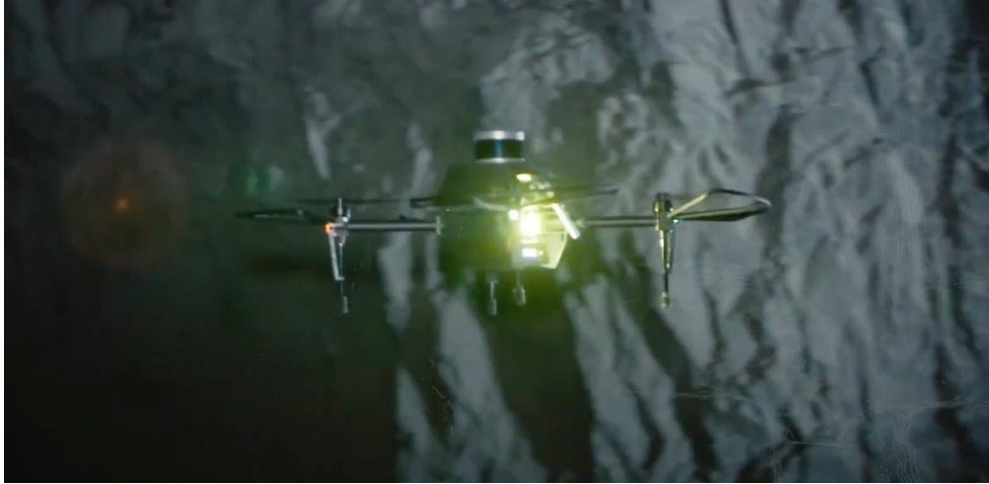


Figure 1.1 A3Rs drone from Exyn Technologies used for gold mining.

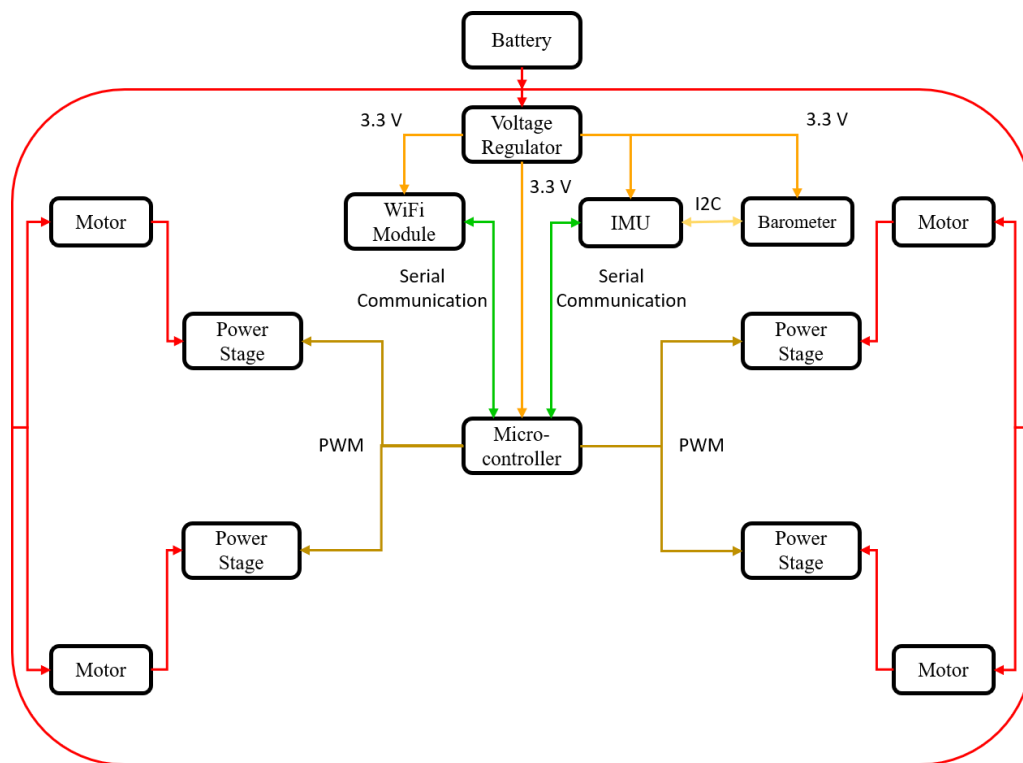


Figure 1.2 Basic Elements of a drone

Micro Aerial Vehicles (MAVs) are a sub-division of this kind of aerial vehicles that has some specifications being size, weight and flight range the most important. The Defense

Advanced Research Projects Agency (DARPA) started the investigation to develop this kind of aircraft to create an aerial vehicle capable of being used in combat or in urban areas, especially inside buildings, where the space is reduced [1].

The MAVs, due to their characteristics of weight and size can accomplish those tasks on places where a Small Aerial Vehicle or a regular sized UAV cannot be used as are indoors and mines, especially on places where the GPS signal is not available. The MAVs can also perform tasks of the ISR (Investigation, Surveillance and Recognition) type, since this kind of vehicles are harder to be detected than the regular and most of the small sized drones.

1.2 Quad-MAV definition

As previously mentioned, the small-scale UAVs has some restrictions and are classified according to the magnitude of this restrictions. Using the Table 1 and as exemplified in [1], a Micro Aerial Vehicles has a weight lower than 100 [g], a wingspan smaller and 15 [cm] and a flight range smaller than 10 [km].

Table 1.1 Comparison of small-scale vehicles^{1, 2}

Specs	Small Tactical	Miniature	Micro
Size	< 10 [m]	< 5 [m]	< 15 [cm]
GTOW	10 - 25 [kg]	< 10 [kg]	< 100 [g]
Speed	< 130 [m/s]	< 50 [m/s]	< 15 [m/s]
Altitude	< 3500 [m] AGL	< 1200 [m]AGL	< 100 [m] AGL
Range	< 50 [km]	< 25 [km]	< 10 [km]
Endurance	Up to 48 [h]	Up to 48 [h]	Up to 20 [min]

The characteristics of the MAVs makes them ideal for the tasks of the ISR kind, being able of maneuver inside buildings, mines and general closed spaces.

This thesis uses the concept Quad-MAV to refer to a Micro Aerial Vehicle of the quadrotor kind that was used to conduct the development of this project.

1.3 State of the art

With the purpose of making this project competitive, some other researches of the same kind where consulted. In [4], Elisa Capello, et. Al. conducted an experiment where the mathematical model of a Micro UAV of the quadrotor kind was used, this with the purpose of developing the Proportional-Derivative (PD) controller that later will be uploaded to the onboard controller. This document only presents the modeling and the results of the experimentation but never clarifies the hardware used nor the real size of the drone

1. GTOW (Gross Take-Off Weight): Maximum that the vehicle can Lift including its own weight.
2. AGL (Above Ground Level): Maximum height that the vehicle can fly, having as reference the platform from where it takes-off.

Other of the projects reviewed is from students from the National University of Singapore [5]. The micro aerial vehicle in that project used a frame made of ABS as a general structure of the MAV, while also as a protective case for the multiple PCBs of the electronic system. Each motor has a microcontroller and a power phase to control the speed, adding weight and cost. The most expensive component is the Inertial Measurement Unit (IMU) employed, being a VN-100 that costs around \$800.00 dollars, causing a very expensive product and avoiding the open source philosophy.

The MAV developed in [6] uses Gumstix Overo® to develop the whole system. While this is practical, since the electrical development is done and only requires integration, it also elevates the price since are bought modules and not completely designed. This document also allows a vision of how the PCB can work as the main frame of all the vehicle.

The current work in the development of this project is given on [3] at the Monterrey campus of Tecnológico de Monterrey. The drone at that stage of development counted with the basic pin configurations, the connection of the peripheral sensor modules, the placement of LED indicators, power stage and voltage regulation to feed the electronic components. The drone has as a characteristic that the main mechanical structure is at the same time the electronic board, reducing the complexity of the drone assembly process, and the components are from low cost, including the MCU and the sensors. This allows the development of a low-cost MAV capable of a steady flight.

1.4 Problem statement

Redesign a previous model of a Quadrotor Micro Aerial Vehicle with low cost components following a systematic procedure for design, manufacturing and testing, capable of lift its weight to later be programed to accomplish more complex tasks on indoor and outdoor environments. In consequence of above Developing MAV is a difficult and costly activity. As originally the MAV must be redesigned to reduce the complexity on the electronic design, built with low cost elements, follow the open-source philosophy and programed to receive the data from the sensors, the communication module and control the propulsion system ensuring the functional integration of all the components.

1.5 Objective

The objective of this thesis is, using a functional description and a not completely designed prototype, to systematically redesign the MAV, build a prototype with low cost components and following the open-source philosophy, allowing it to be modified for more specific purposes and develop the embedded software to ensure the functional integration of all the components.

The specific objectives of the project include the following points:

- Hardware design:
 - Review the design of the MAV
 - Modification of the electronic layout making it simpler to allow a faster identification of damages in the circuit and reduce the possibility of error occurrence in the prototype PCB.
 - Reduction of the number of electronic components to reduce the weight of the vehicle and update the discontinued components or modules.
 - Manufacture a functional prototype using the specialized equipment for the PCB manufacturing and assembling the surface mount devices (SMD) as well as the through-hole components, using the welding iron, heat gun, welding and flux.
- Embedded software design:
 - Develop an embedded software based on a finite state machine to reduce the potential bugs or error and ensure the good integration of the component's functions.
 - Establish the communication chain between components to ensure the correct data transmission to the MCU.
 - Program the controller to stabilize the hover function of the vehicle.

1.6 Methodology

This project takes as base a previous version of the vehicle developed by members of the School of Science and Engineering at ITESM [3]. It consists on the analysis of the status and performance of the previous prototype and reduce the number of components and complexity of the pads. This allows a small reduction of the cost and the weight that will act over the MAV when there is no other payload to be carried. All the elements must be integrated to the PCB that also works as the mainframe of the drone.

The main structure of the MAV is the one of a quadcopter with a wingspan smaller than 15 centimeters.

This Thesis Methodology can be seen in Figure 1.3, for the optimization of the functionality in the embedded system, as well as in the simplification of the electronic design for an easier assembly of the parts and reduce the error margin on the manufacturing of the prototype.

- 1 The electronic diagram and PCB template is generated using a CAM software specialized in this function, ensuring the layouts have all the proper connections and distribution of the components between layer and have an appropriate design.
- 2 After the design of the electronic template using the computer software, the circuit and its connections are tested in a protoboard to ensure the viability of the electronic design and that the paths are designed in the most adequate way according to the requirements of the project. For the elements that where replaced by the new version

or where added, each one is tested individually to understand the behavior of each one to avoid compromising the testing circuit integrity.

- 3 The manufacturing process is started after all the components are tested by separate at first and latter all together connected as in the final prototype (integration). The manufacturing of the PCB is done through a CNC machine that must be adjusted through a series of test-and-check steps to avoid the interruption of the process or reworkings that could damage a potential circuit. The soldering of the components is done after the top and bottom layer have continuity in the indicated places and there is no short in the circuit.
- 4 The program of each module (main controller, sensors and communication antenna) is tested at the end of the software development phase while the IMU module, the electronic configuration and the voltage regulator are tested at the end of the electronic development phase at the testing circuit to debug and clean all the potential errors. Once the program is ensured, it is loaded into the prototype to conduct to the experiments, gather the results and do the experimental analysis to improve the embedded code, electronic circuit or adjust the internal algorithms for stabilization.

The Figure 1.3 illustrates the development process followed. On the green cycle is the development of the embedded software that will be uploaded to the main processor unit. Every time a new step of the embedded code is added, it is tested individually to ensure a correct functionality and if any error is found then the software must be redesigned. The blue cycle shows the development of the electronic board. After the selection and new design of the board connection concludes, a testing process is followed to avoid communication errors or wrong connections among the components. If any test is failed, the board must be redesigned. The yellow cycle contains the manufacturing process, that includes the assembly and programing of the board. After the printing of the board finishes or every time a component is mounted, a continuity verification and capacitance measuring is done to verify the correct assembly of the prototype. After the drone is assembled the embedded software is programed inside the main processing unit and it is verified using the debugging tool. The Figure 1.3 has red arrows that indicates the path to follow in case of failure of the test. In case of failure of the prototype testing, the whole path must be done considering now the limitations that were found in the testing.

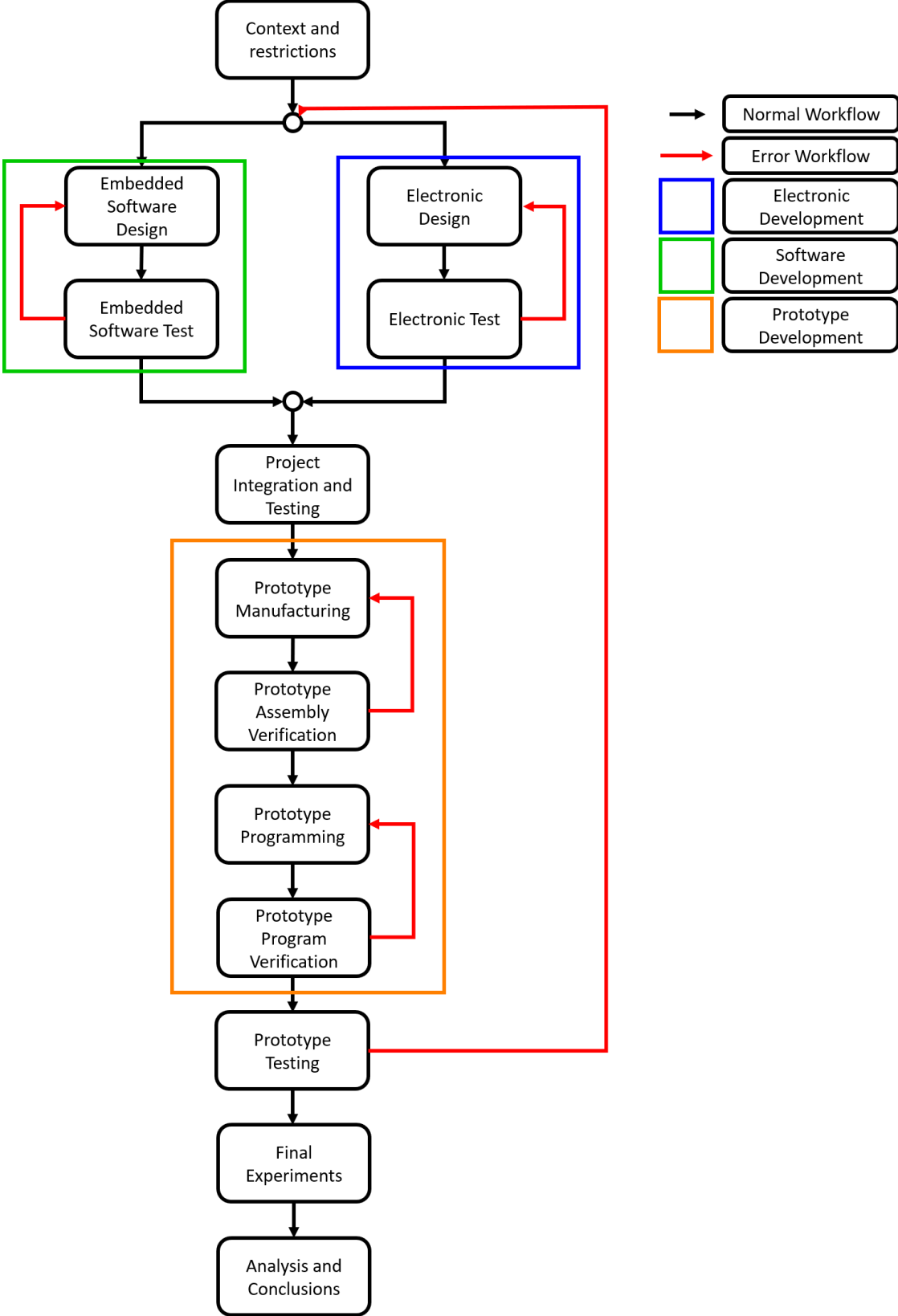


Figure 1.3 Methodology of the Quad-MAV development

1.7 Achievement and contributions

The contribution relies on a whole process for the manufacturing of an open source Quad-MAV, capable of take-off and perform a stable hover or flight.

The achievements of the project are a pair of prototypes capable of react to a command sent through a wireless communication, process the reference, read and process the current state given by the sensors, compare the reference and the current state and regulate the PWM that control the propulsion system speed.

The electronic design: composed by an open source project to be modified by anyone that want to develop a Quad-MAV. The electronics provide a stable energy source to all the components to avoid the interruption in the communication between the ground station and the drone, fluctuations in the process of the embedded system, or malfunction of the components leading to an early replacement. The devices used meet the specifications in size and weight to allow the vehicle to move freely through the air. The PCB where all the components and modules are connected works as the body of the drone.

The prototype: a functional prototype with SMD components to reduce the weight of the board and with pinholes to place the Wi-Fi communication module and the IMU sensor, allowing a fast removal in case that any of these components require to be reprogrammed or replaced for a new version. A set of pin headers must be placed with the purpose of connecting the programming device to upload new firmware developed for the main processor unit. The prototype can be divided into modules allowing an easier analysis of any possible failure at the circuit and making a faster reparation process. The Modules can be divided:

- Voltage regulation.
- Sensor module.
- Communication module
- Processing module.
- Power stage and propulsion module.

The embedded code: based in freeware that can be downloaded and read by anyone that have a computer and the adequate IDE to program microcontrollers of the PIC32 family. The configuration of the drivers and peripherals of the microcontroller must be clear to allow a translation of the code between different microprocessors platforms.

1.8 Composition

The thesis is organized as follows:

Chapter 2 gives information about what is a UAV, a MAV, classifications, the theoretical development of the project and the configuration of the system.

Chapter 3 holds the information about the selection of the components replaced, how where added or removed some elements, the coding of the main firmware, the manufacturing process and how the adjustment of parameters of the final prototype.

Chapter 4 makes emphasis on the process of experimentation of the prototype as well as the code debugging process, final testing of the prototype and results of experimentation.

Chapter 5 is about the conclusions of the project, limitations of the project and recommended future work.

Appendix A contains the configuration, steps and main functions employed on the embedded code.

Appendix B delivers the changes on the manufacturing process.

Appendix C gives information about the cost of every element on the drone board.

Appendix D shows the graphics obtained from the measurement of the force delivered by the motors during the experiments phase.

Chapter 2

2 Quad-MAV Behavior and Control

The chapter gives general information about the Quad-MAV: a detailed configuration is given. A general explanation of the behavior of the vehicle in attitude and altitude is addressed. Finally, the information about the controller employed and the equations used to stabilize the hover of the vehicle is delivered.

2.1 Vehicle's configuration

To develop the control equation that can stabilize the vehicle during the flight is necessary to understand the dynamic behavior of the Quad-MAV vehicle, detect the inputs, references and outputs.

Additionally, the general information about the possible configurations of the quadrotors and the difference between them is provided.

2.1.1 2.1.1 Quadrotor configuration

A quadrotor can have one of the 2 following configurations: the plus (“+”) configuration and the cross (“x”) configuration. The ‘+’ configuration has a rotor on the tip, one in the back and one on each side. This configuration advantage is found on the modeling and design of the control since only a pair of motors needs to be adjusted to move the vehicle over a single axis of the plane ‘XY’. The ‘x’ configuration in contrast requires the use of both pair of motors to accomplish the same movement, making more complex the modeling but reducing the stress and energy on each motor for movements on the axis ‘X’ or ‘Y’ due the share of payload among both pair actuators. Another advantage is that the cross configuration does not creates a *Yaw* caused by the *Roll* or *Pitch* control inputs [7].

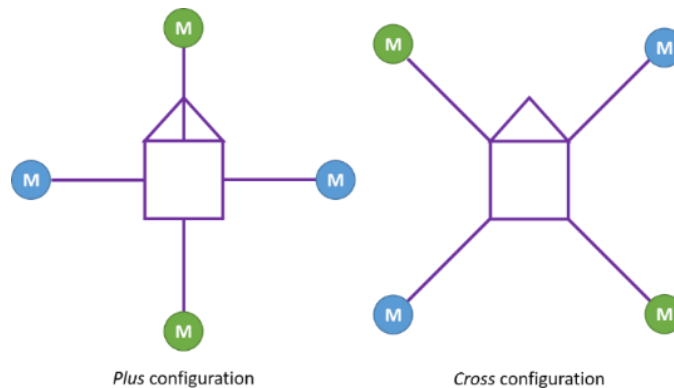


Figure 2.1 Quadrotor configurations

The drone developed in this thesis, shown in the Figure 2.2, is a UAV of the quadrotor type that uses the cross configuration for the maneuvering of the vehicle. Also, it enters on the classification of the Micro Aerial Vehicles (MAV) according to the characteristics of the vehicle. It has an average of 13.1 cm of wingspan and a GTOW of 71 grams after modifications made on the vehicle.

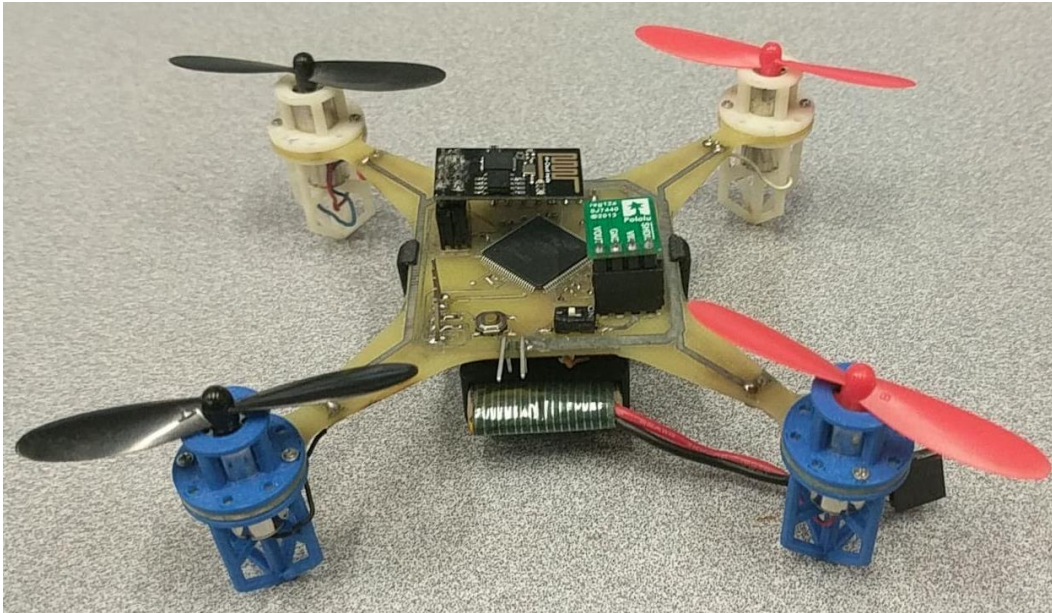


Figure 2.2 Final Functional Prototype

2.2 Quadrotor attitude, orientation and altitude behavior

The quadrotor is a underactuated robotic system due to the characteristics that has as vehicle. Every drone could be able to rotate over and through the three axes of the space, since there is no joint that impedes the movements, having in total 6-DOF (Degrees of freedom). In the case of the quadrotor, as the name indicates it, it has 4 rotors on the structure of the robotic system.

According to robotic theory as explained in [8], a robot is overactuated when the quantity of actuators is more than the degrees of freedom that the mechanism has, a fully actuated robot has one actuator for each DOF that it has while an underactuated robotic system has less actuators than degrees of freedom, therefore, not all can be controlled and have dependencies over others. In the case of a quadrotor, there are 6 DOF and only 4 actuators, meaning that 4 of the 6 positions can be controlled and the other two will have dependencies in the positions that can be controlled.

Every multirotor vehicle can control in an independent way the rotation over the 'Z' axis, called $Yaw(\psi)$, and the position or displacement through the same axis, being this the altitude position controlled by the $Thrust$ delivered by the spinning velocity of the motors.

The quadrotors can control all the angular positions at the same time. The rotation over the “X” axis is called *Roll*(ϕ) and the rotation over the “Y” axis is called *Pitch*(θ). The rotation over this last two axes has effects on the position through both axis. While controlling the *Roll* angle, the drone will start to move through the “Y” axis and controlling the *Pitch* angle will cause as an effect the movement through the “X” axis.

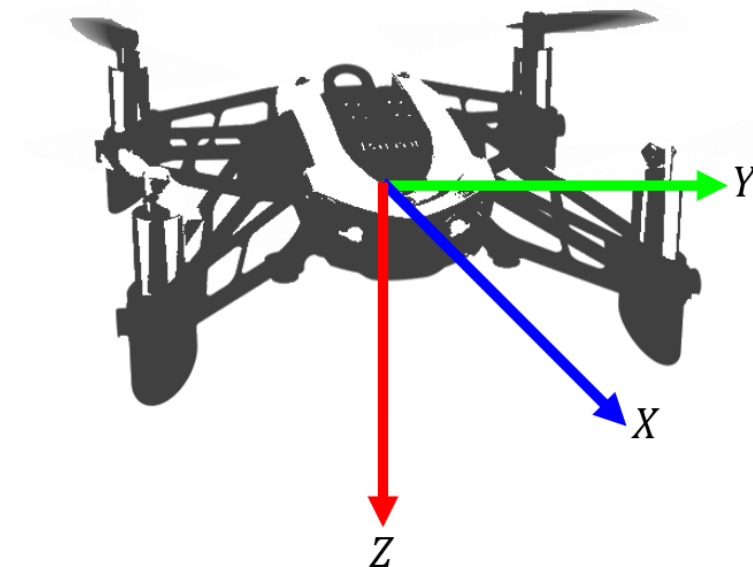


Figure 2.3 Aircraft angles and referential frame

In contrast to other robotic systems, the aerial vehicles have a different inertial frame. On the robotic systems and ground vehicles the “X” axis is looking toward the observer of the system, the “Z” is directed upward being the “Y” orthogonal to both. On the Aerial vehicles the “X” axis points toward the front of the vehicle and the “Z” axis points downward directly to the ground, meaning that the takeoff is a negative displacement and the landing a positive displacement.

For a positive movement along the “X” axis the quadrotor requires a positive *Pitch* angle, then the rotor labeled with the numbers 3 and 4 (Figure 2.4) needs to increase the spinning speed to deliver a higher force, in contrast to the motors 1 and 2 (Figure 2.4), that need to reduce the velocity to compensate the extra force delivered by the first pair of motors. The same occurs with the *Roll* angle. To produce a positive movement on the “Y” axis, the inclination angle must be negative, requiring the speed increment on the motors 2 and 4 to deliver a higher force to make the drone to spin over the “X” axis, and the reduction on the motors 1 and 3 to produce a countereffect for the extra force delivered by the motors pair located at the left side of the vehicle. The Table 2.1 compares the angular position with the displacement over the plane “XY” using the model of the drone found in this thesis.

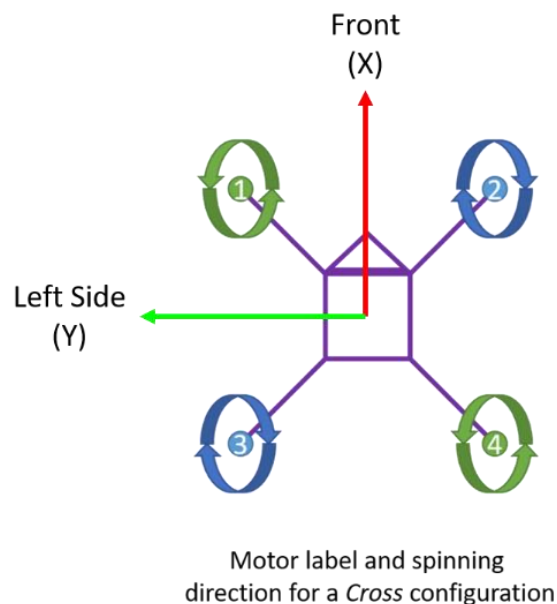


Figure 2.4 Motor labeling and rotation for cross configuration in this thesis

Table 2.1 Motor-Direction-Orientation relationship over the space

Displacement Axis	Displacement Direction	Rotation Angle	Rotation Direction	Motor				
				1	2	3	4	
X	+	Pitch	+	LS	LS	HS	HS	
X	-	Pitch	-	HS	HS	LS	LS	
Y	+	Roll	-	LS	HS	LS	HS	
Y	-	Roll	+	HS	LS	HS	LS	
Z	+	N/A		HS	HS	HS	HS	
Z	-			LS	LS	LS	LS	
N/A			Yaw	+	HS	LS	LS	HS
			Yaw	-	LS	HS	HS	LS

To make the vehicle to spin over the “Z” axis on certain direction it is necessary to increase the speed on the pair of that moves in the opposite direction since this spinning produces a torque on the opposite direction of the spinning and reduce the velocity of the pair that rotates in the same direction of the desired spinning.

The altitude position also depends on the *Thrust* of the four motors working at the same time, but in this case all four motors collaborate in the same acting direction. Each motor delivers a force that goes towards the ground having a reaction on the opposite direction that counters the effect of the gravity over the mass of the drone. To have a VTO (vertical takeoff) it is necessary that the force delivered by all the motors become greater than the weight of the drone, since this causes an upward acceleration. To hover the force must

1. HS: High Speed Motor.
2. LS: Low Speed Motor.

be equal to the weight since this causes a balance on the forces sum that allows a static position in the middle of the air when no other force is acting over the vehicle. For the VL (vertical landing) the force produced by the motors needs to be smaller, this having an opposite reaction respecting to the VTO. Since the motors collaborate in the same direction, for hovering each motor must deliver the same force to avoid a moment that causes a movement on the *Roll*, *Pitch* or *Yaw* angles.

2.2.1 Roll and movement along the “Y” axis

To produce a positive *Roll*, as explained previously, the motor 2 and the motor 4 must produce a higher force to rotate the vehicle until the desired angle is reached by the MAV's state. To counter the extra force delivered by the motors 2 and 4, the other pair (motors 1 and 3) must reduce the force to avoid a difference on the *Thrust* that can produce a movement along the “Z” axis.

This difference of forces delivered by the motors causes the drone to tilt toward the “Y” axis and simultaneously changes the direction of the force produced by the motors as shown in the Figure 2.5 The force produced now on the “Y” axis creates an acceleration that moves the vehicle through the positive direction of such axis.

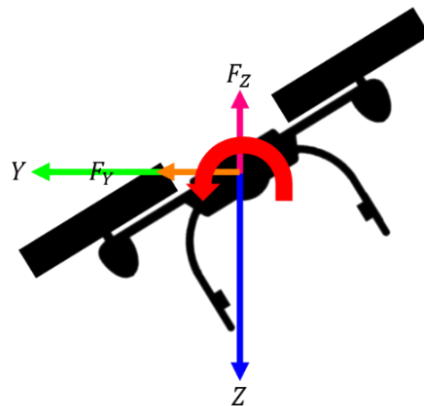


Figure 2.5 Roll viewed from the front of the drone

2.2.2 Pitch and movement along the “X” axis

The *Pitch* behaves like the *Roll*, since both rotations are executed over a horizontal axis that is orthogonal to the other horizontal axis and the vertical axis. The motors that produce a higher force are 1 and 2 while the motors 3 and 4 must reduce their spinning speed, until the desired angle is reached.

As in the previous rotation, the movement will start with a small angle. When the drone rotates in a negative direction over the ‘Y’ axis the displacement will be produced along the “X” axis in the negative direction as it can be observed on the Figure 2.6.

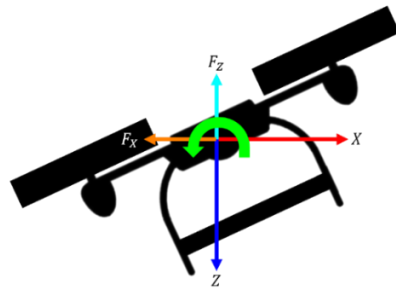


Figure 2.6 Pitch viewed from the right side of the drone

2.2.3 Yaw motion

The *Yaw* is characterized in the way that, at difference from the previous rotations, it does not produce a movement along any other axis. This is due the nature of the movement.

On the previous rotations the direction of the force produced by the motors changed according to the angular position. “Z” While in both cases the force before rotating over an axis the force has effect only over the axis. After the drone tilted the force started to act over the “Y” axis when it is a *Roll* rotation, and over the “X” axis when it is a *Pitch* rotation.

The *Yaw* does not produce any change on the direction of the force, only the orientation of the vehicle. To produce a positive twist over the “Z” axis it is necessary to increase the velocity of the motors that spins on the opposite direction of the desired rotation. The Figure 2.7 shows a negative rotation in *Yaw*, this caused by the increase on the rotation speed delivered by the motors 2 and 3 according to the Figure 2.4. To counter the extra force delivered by the previous motors mentioned, the other pair must reduce the spinning speed to keep the balance on the *Thrust* delivered by the four motors.

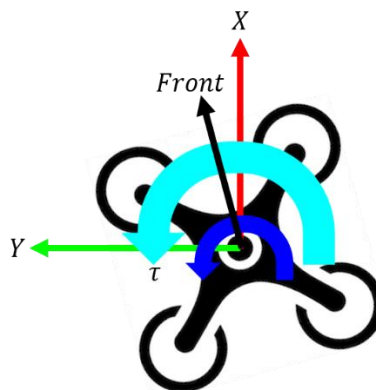


Figure 2.7 A negative yaw seen from above the drone

2.2.4 Altitude and movement along “Z” axis

The previous mentioned angular positions work based on a difference between the *Thrust Force* delivered by two pairs of motors. In the case of the altitude and the movement along the “Z” axis, the four motors must deliver the same force at the same time to avoid any angular movement.

A difference with the other displacements and movement, the movement along the “Z” axis deals with the force caused by the gravity as can be seen on the Figure 2.8. To perform a hover the sum *Thrust* delivered by the motors must counter the effect of gravity. To perform a Vertical Take-Off the sum of the forces delivered by the motors must be greater than the weight of the vehicle, this countering the effect of gravity and generating an acceleration that causes a negative displacement along the “Z” axis (Figure 2.8). The Vertical Landing is performed when the sum of the *Thrust* delivered by each motor is smaller than the weight of the UAV, in this case, causing an acceleration that goes downward to the ground (Figure 2.8).

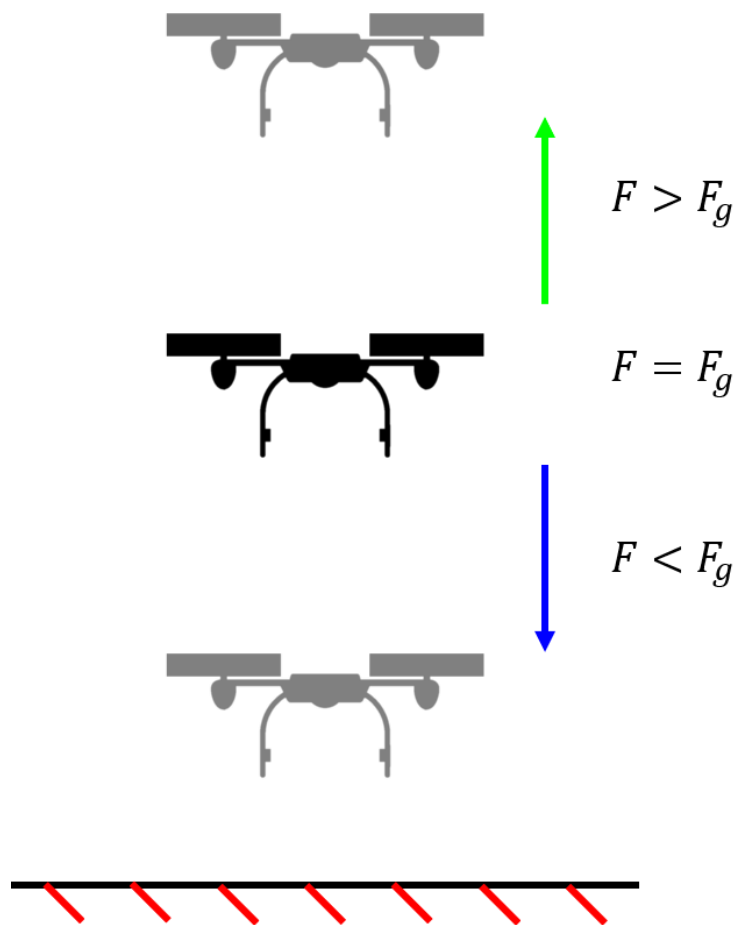


Figure 2.8 VTOL and Hover

2.3 Control equations

To achieve a stabilization of the position in the space, a control law must be applied to the vehicle. This is a series of equations that uses the error of the current state using a reference given by a monitor system, in the case of an UAV it could be a program that indicates the flight or a pilot. The Figure 2.9 illustrates a closed-loop controller diagram indicating where the equations must be located to control the next state of the plant.

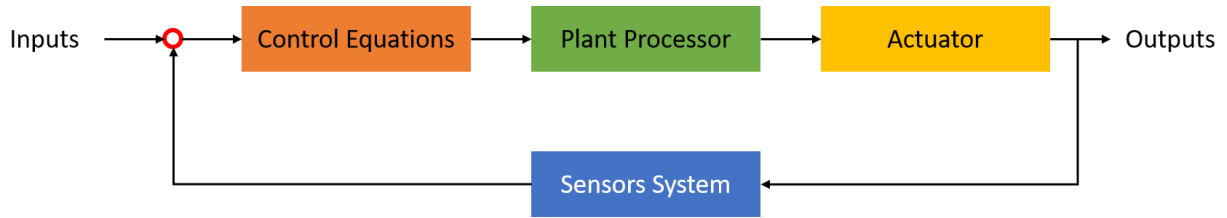


Figure 2.9 PD Controller Closed- Loop Diagram

The control law to be employed is a PD controller. This is a linear control characterized by the simplicity of the equation and at the same time by the robustness.

It consists on a proportional gain K_P that multiplies the error. This gain helps the system to pass over the stabilization point, the higher the value the faster it will reach such value, but as a disadvantage it causes higher oscillation into the system to the point that can make it unstable. The other element is the derivative gain K_D that multiplies the derivative of the error and smooths the response, the bigger the value the smoother is the response, but this causes that the time it takes to reach the stabilization value to be longer. The general equation is the next.

$$Ua = K_P \cdot e + K_D \cdot \dot{e} \quad (1)$$

Each variable to be controlled requires a control equation, having a total of four equations as can be seen on equations (2) - (5), where each controller calculates a deviation of the desired velocity for hover in steady state, this means that $\Delta\omega_F$, $\Delta\omega_\phi$, $\Delta\omega_\theta$ and $\Delta\omega_\psi$ are values of the motors velocity to produce a movement along the “Z” axis in the case of the equation (2), or to produce a *Roll*, *Pitch* or *Yaw* angles from (3) to (5) respectively

$$\Delta\omega_F = K_P \cdot e_T + K_D \cdot \dot{e}_T \quad (2)$$

$$\Delta\omega_\phi = K_P \cdot e_\phi + K_D \cdot \dot{e}_\phi \quad (3)$$

$$\Delta\omega_\theta = K_P \cdot e_\theta + K_D \cdot \dot{e}_\theta \quad (4)$$

$$\Delta\omega_\psi = K_P \cdot e_\psi + K_D \cdot \dot{e}_\psi \quad (5)$$

Using [9] as a basis, a hover constant (ω_H) must be added after the *Altitude* control equation, it is required to stabilize the drone on a hover steady state, where this constant must ensure to countereffect the gravity effect. As result, and using the matrix for quadrotors in cross configuration from [7], the equation to obtain the desired velocity of each motor can be written as follows.

$$\begin{bmatrix} \text{Height} \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta\omega_F + \omega_H \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \quad (6)$$

The hover constant is the required velocity of the motor required to perform a hover in steady state. This velocity must deliver a force equal to the weight of the vehicle to counter the effect of the gravity over the vehicle. Using the equation found in [9], this constant can be calculated using the equation (7), where k_τ is a thrust constant proper of the motors, m is the mass of the UAV and g is the standard acceleration due to gravity.

$$\omega_H = \sqrt{\frac{m \cdot g}{4 \cdot k_t}} \quad (7)$$

Chapter 3

3 Design and Manufacture

This chapter addresses the practical development of the main vehicle, first delivering the state of the project and the analysis to re-design the electronic system, focusing the second section on the addressing the information about the changes on the manufacturing process. The third section presents the design of the embedded software and the configuration required on the microcontroller. The fourth section specifies the process to find some parameters that must be loaded on the model for simulation, on the control system and the embedded software.

3.1 Electronic design

Most of the electronic development was conducted by the MSc. Edgar Espinoza on previous works at the Robotics Laboratory at ITESM [3]. The electronic work developed in this project had as main purpose of reducing the possibility of error during the manufacturing process by placing some elements in different order, reducing the number of elements that were not used anymore by the main processing unit, the communication or sensing modules or the power phase required for the control of the motors, replacement of obsolete or not adequately chosen components.

3.1.1 Electronic diagram and interactions

The interaction of the different components of the drone is made through the electronic connections between them.

The power source is a lipo battery that delivers in average 3.7 [V] of nominal voltage, having a range of a maximum of 4.2 [V] and a minimum of 3.2 [V]. This battery delivers the energy to make the drone work and is connected to the regulator and the four rotors with their respective diodes and capacitors in parallel.

The regulator is a step-up/step-down circuit that helps to maintain the voltage delivered by the lipo battery on a constant value of 3.3 [V] of output. This voltage is delivered in parallel to the MCU, the sensors and the antenna of the drone.

The altitude sensor, fed by the voltage delivered by the regulator and the common line, is connected by the communication protocol I2C to the Inertial Measurement Unit (IMU), component that is connected in parallel to the altitude sensor.

The communication module and the IMU sensor are connected in parallel to the Main Controller Unit to the Vcc and common lines, and simultaneously both are connected to

the MCU through the lines of the RS-232. Each component is connected by their respective communication lines to the main controller unit.

The MCU is connected through the PWM paths to the pull-down resistor connected to the gate pin of each one of the MOSFET that helps to amplify the output of the microcontroller to regulate the velocity of the motors. The MOSFET are connected to the motors, diodes and capacitors at the drain pins, while the source pin is connected to the common line of the circuit.

The figure 3.1 shows a basic diagram and connections of the components for energy feeding.

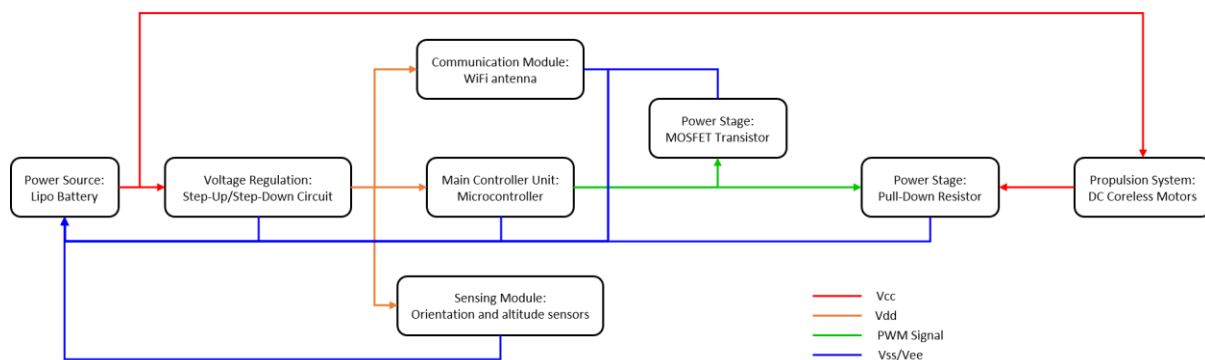


Figure 3.1 Elements and connections of the quadrotor

3.1.2 Replaced Elements

The first electronic component to be analyzed is the voltage regulator LD1117XX33 in a SOT-223 package. This element is placed stabilize and regulate the voltage supplied by the battery to the main microcontroller, WiFi module and the IMU sensor, that require a 3.3 voltage supply to work properly. This first element is located right next to the switch to turn “ON/OFF” the Quad-MAV as it can be seen on Figure 3.2. This component required replacement since, according to the datasheet, this device requires a constant feeding source of 5.3 [V]. The battery at maximum charge only delivered a maximum of 4.2 [V] for few moments for latter be stabilized as 3.7 [V] of nominal charge.

This lack of voltage, as well as unstable delivery of difference in potential, can cause unstable functionality of some elements as well as the total failure of some components as the communication module.

To replace this component, another module was used, the Step-Up/Step-Down U1V11F3. This circuit helps to regulate the voltage to 3.3 [V], working as a step-down then the voltage is higher than the required, reducing the output voltage but delivering a higher output current than the input current. When the input voltage is lower, the component

switches to a step-up functionality, where the input voltage is raised to reach the required output voltage, but reducing the input current to compensate this change, having a lower output current.

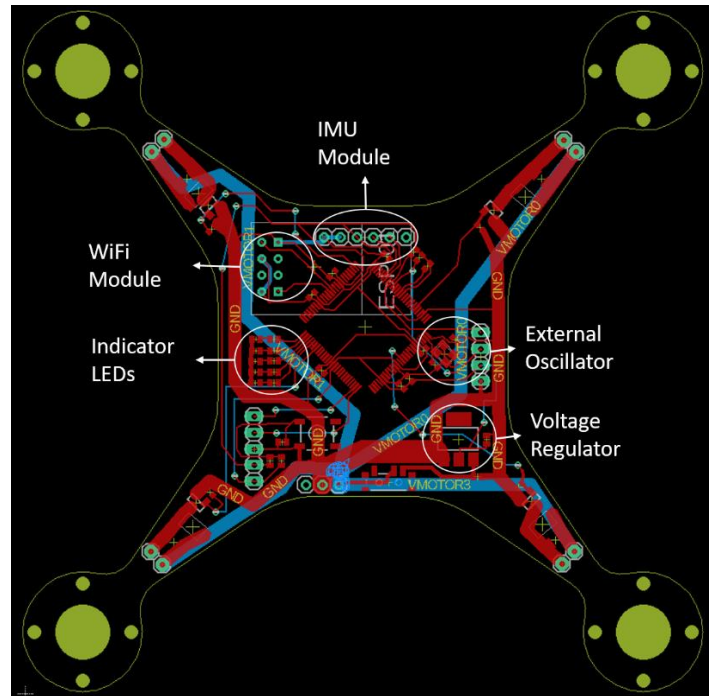


Figure 3.2 Early version of the MAV PCB

The 9-DoF (Degrees of Freedom) Razor IMU was selected since the beginning since this is a low-cost option of the sensor required to measure the attitude and orientation of the Quad-MAV at every moment. This circuit has an advantage of having an open-source philosophy, leading to a great diversity of developed programs that could be loaded to the main circuit of the component with a high performance and capability of being calibrated for the vehicle where it is going to be used.

The original model of the IMU, the SEN-10736, was taken out of the market, leading to the search of a possible new component. This, along other disadvantages lead to the model SEN-14001, a cheaper model offering with similar characteristics. The Table 3.1 compares the characteristics that led to the full replacement of the IMU model.

Even if the weight and dimensions of the replacement were not favorable as specifications, the internal components, the communication protocols, the programming method and the price were important factors to take the decision. A big advantage was also on the software development, since the SEN-14001 had implementations to measure the orientation using the Euler angles, and the firmware employed on the SEN-10736 was adapted for the new versions of the IMU module.

Table 3.1 Comparison of the different IMU boards

Characteristics	Model	
	SEN-10736	SEN-14001
Bootloading	FTDI	Direct to the PC
UART communication	Yes	Yes
SPI communication	No	Yes
I2C communication	No	Yes
Power Supply	5 [V]/3.3 [V]	3.3V
µC	ATmega328	SAMD21G18A
Base Price	\$74.95 USD	\$35.95 USD
Sensors	ITG-3200	MPU-9250
	ADXL345	
	HMC5883L	
Dimensions	24x41 [mm]	32x32 [mm]

3.1.3 Modification of the electronic layout

The previous version of the PCB board had numerous elements that were no longer required on the circuit because of the new developed embedded software. As it can be seen on the Figure 3.3, previously five LED indicators were established on the left side of the board.

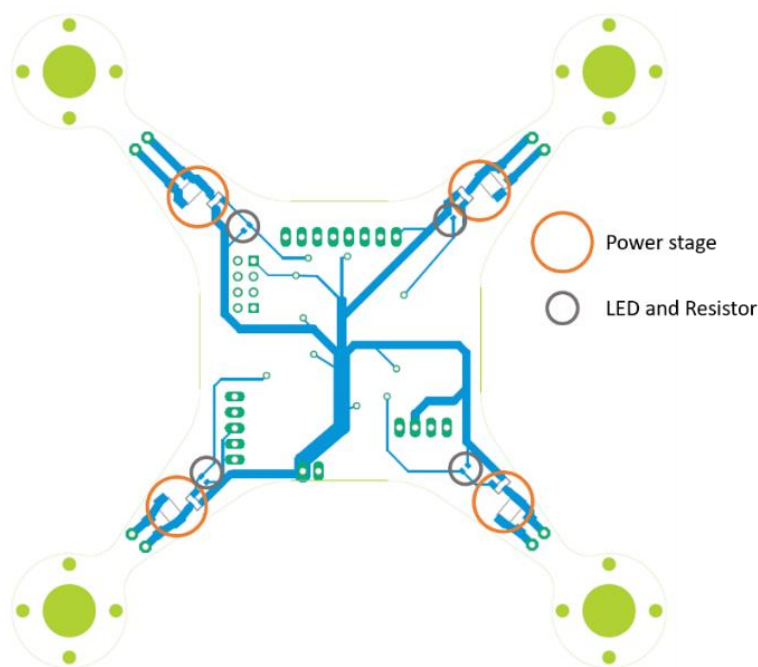


Figure 3.3 New LEDs and power phases placement

The use of this LED as indicators of the functionality of the motors in this position has no direct relationship. While in the firmware of the microcontroller the PWM could be

activated and running, it was not possible to detect if the corresponding output pin used to generate the pulse was working correctly, since it can be badly welded or the PWM module could have been damaged at some point of the manufacturing or testing process.

Using this reason as a base, 4 of the LED were placed parallel to the power phase of each motor as it can be seen on the Figure 3.3, allowing a fast response to the question “Is the PWM module working correctly?” when the corresponding motor is not working properly or not spinning with the difference of the “Duty Cycle” of the corresponding PWM output pin. The fifth LED, that gave information about the “ON/OFF” status of the microcontroller was eliminated, since the WiFi module and the IMU sensors already give feedback of that status.

Other element analyzed is the voltage regulator in a SOT-223 package. This element helps to stabilize and regulate the voltage supplied by the battery to the main microcontroller, WiFi module and the IMU unit. This element is a surface mounted device, having a simpler soldering process in comparison to the replacement that is a through-hole plate, requiring the modification of the original mounting place to add holes where the U1V11F3 is welded. The use of this new board lead to the elimination of the capacitors required by the LD1117XX33 to avoid voltage peaks on the input and output pins, simplifying the routing of the path as can be seen on the Figure 3.4.



Figure 3.4 Comparison of the input and output paths between regulators

To validate the component, it was tested using the battery employed on the Quad-MAV, the IMU, the ESP8266 and the MUC to simulate the functionality of the drone. The output voltage was constantly monitored (Figure 3.5) to ensure that even when the battery voltage is under 3.3 volts, it worked as a Step-Up regulator.

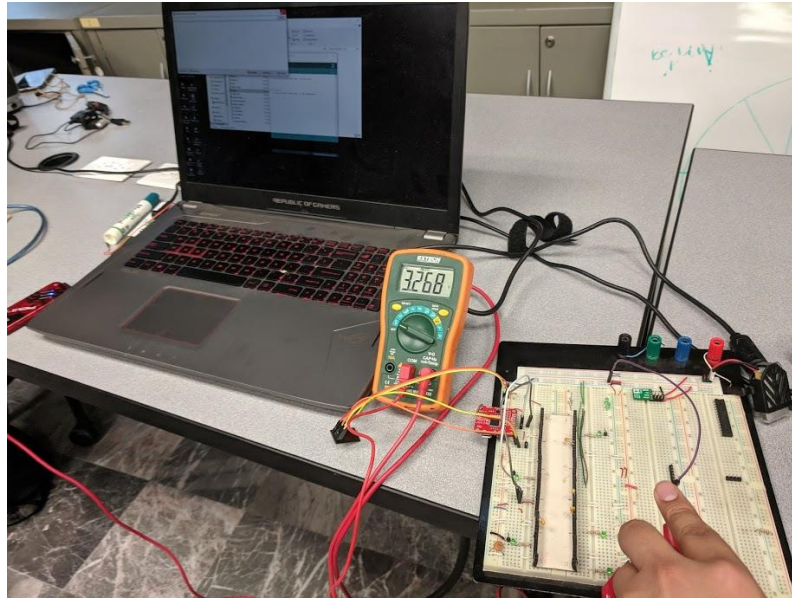


Figure 3.5 Verification of the voltage regulator.

To reduce the complexity of the paths and the number of vias, the path that delivered the voltage from the battery to the motors and from the voltage regulator to the main circuit where rerouted creating two surrounding rings, the external to deliver the voltage from the LIPO battery to each motor, and the inner ring to feed the other components with the voltage delivered by the regulator.

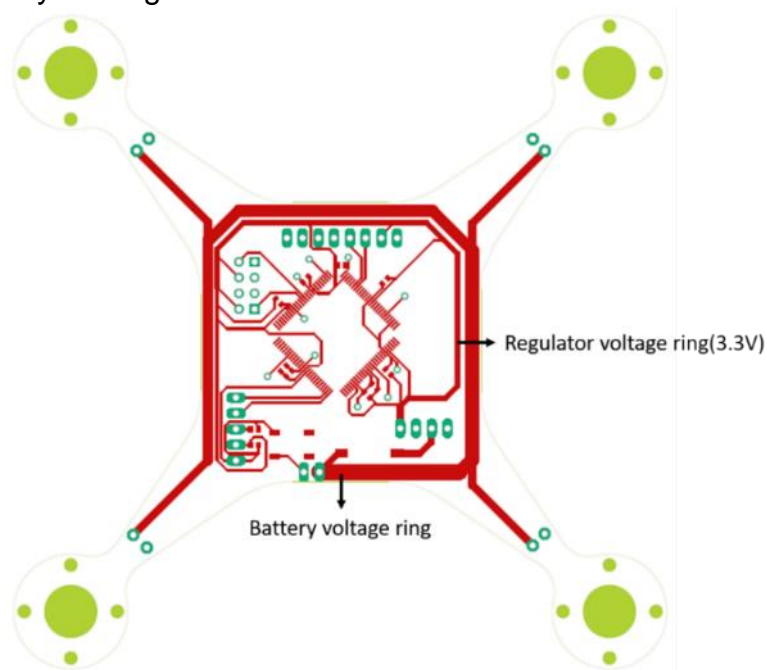


Figure 3.6 Top layout with the surrounding paths

This change on the path route conducted to move the placement of the power phases (Figure 3.2) from the top layout to the bottom layout but allowing a more direct connection to the negative pole of the battery and a more symmetrical and simplified design for a faster identification of possible electronic failures. The top layer can be seen on the Figure 3.6.

3.2 Manufacturing process

The manufacturing process in this project has suffered from a set of minor changes, mainly due the change of the layout and the reduction of components over the board. The main changes can be appreciated on the process of connecting the Top and the Bottom layout on specific places, this due the use of a new process to reduce the risk of having a bad transmission of the data or the signal from the source to the goal pin of the corresponding element.

The Table 3.2 shows the comparison of the process established by Edgar Espinoza in [3]. Different stages of the process remain equal, but the production method changed, or the order presented is different.

Table 3.2 Process comparison table

Step	Process	
	Original Method	New Method
1	Copper roughing	Copper roughing
2	Vias placing	Copper tinned
3	Copper tinned	ProConduct placing
4	Solder plaster placing	Proconduct baking process
5	Component placing	Solder tin placing
6	PCB baking process	Component welding

The Copper Roughing process consist on the removal of the excess of the copper over the board to form the paths, holes, vias connectors and pads for the through-hole components. On the first stage of the task, the machine places the fiducials over the main board to have a reference of the machine position over the board during the process.

The second stage of the Copper Roughing is the drilling of the holes where the vias and the through-hole components are welded. The third and four stages are repeated for the top and the bottom layout of the circuit. The third stages remove the copper surrounding the place where the holes were made, around the paths that connect the elements and the pads where the circuits must be welded. The fourth stage gives a smooth finish over the board removing the excess of the copper that is not used by the circuit, this to reduce the final weight of the board. After the fourth stage ends, the plate is turned over the horizontal axis and the stages 3 and 4 are repeated for the next layer.

The second process is different from the previous methodology. The tinned process is the second phase of the overall method since the ProConduct needs to be baked to get fixed to the conducting material. If the plate is baked without the tin, the paths that are thin enough could be damaged by the heat and the plate will be ruined. The tin process takes over 5 minutes to be completed. The PCB is submerged into the Liquid Tin and then is cleaned with distilled water to avoid impurities that could damage the tin cover.

The third process consists of the filling with the ProConduct of the holes where the vias and the through-hole components are placed. This chemical helps to connect and keep continuity of the board from the top to the bottom layout. Not all the holes must be filled, only the ones corresponding to the vias and where a through-hole electronic component is welded, and the main signal path is in the opposite layout where it is welded. The Figure 3.7 shows surrounded by a black circle the holes where the ProConduct must be injected. The process to place this chemical is to pour into a syringe with a very fine needle, and using pressure fulfill the holes.

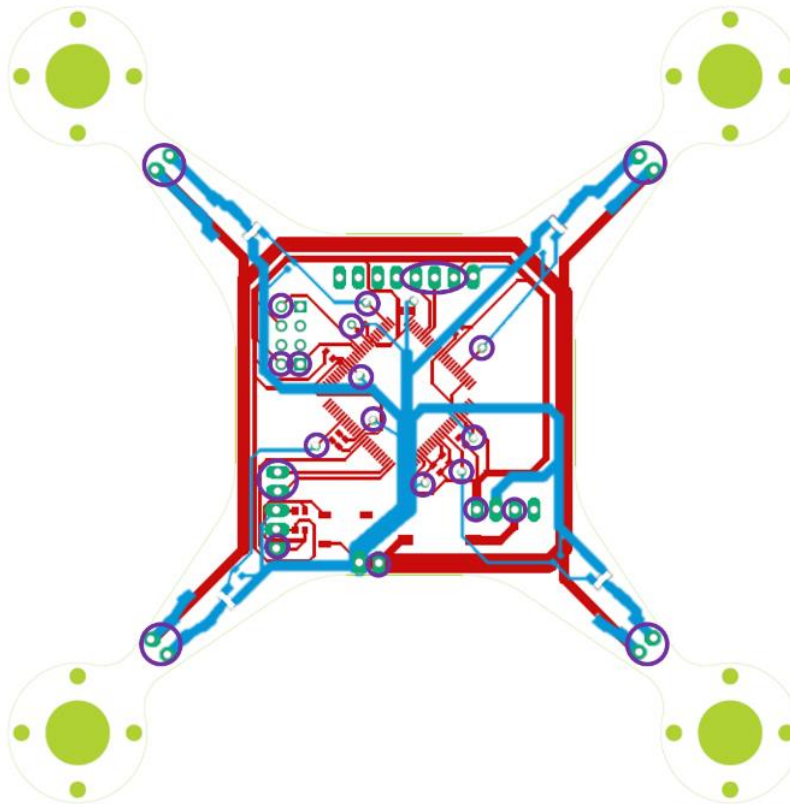


Figure 3.7 Places to fulfill with ProConduct

To fix the ProConduct into the board it is necessary to be submitted into a heat treatment. This process dries and solidifies the chemical that gets added to the wall of the holes. A continuity test is made using a multimeter to ensure that the bottom and top layers are connected (Figure 3.8). This process reduces the probability of error since the vias must

be placed using pressure, and in some cases the copper tube does not make full contact with the copper part on both sides of the plate.



Figure 3.8 Continuity verification process

The fifth process also changed since the layout was modified. In the original method in the fourth process, a soldering plaster is placed using a stencil. The replacement of some electronic components, the change of position of other components and the change of some copper paths caused that the available stencil to become obsolete. Another problem that presented this process is that the components must be placed within a range of time since the plaster dries and cannot be used in that state. In the new process a layer of flux plaster is placed and with a soldering iron the welding tin is melted and fixed over all the pads of the SMD electronic components.

To weld all the components a heat gun is used to melt the tin over the pads and with tweezers the SMD element are placed, the heat gun is removed, and the component is verified to be fixed (Figure 3.9). The through-hole devices are welded using the iron and the tin.

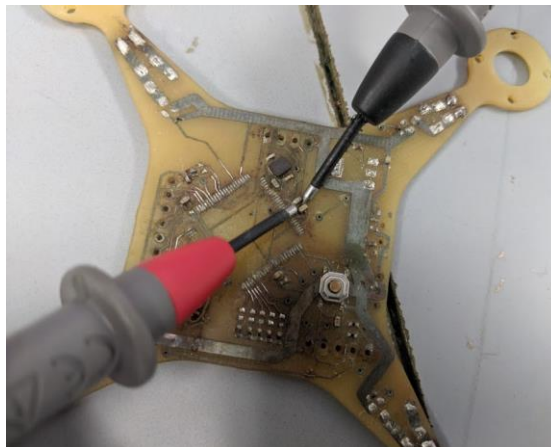


Figure 3.9 Component assembly verification step

3.3 Physical parameters of the drone

To approximate the behavior of any plant to be used and introduce the adequate data on the embedded code, it is necessary to characterize the system to model and get all the parameters that can affect the way the system may react or move in response to any environmental or systematic stimulation over the space.

The Table 3.3, extracted from [10], gives all the information about the parameters regarding the vehicle, obtained by direct and indirect measurements. This information is employed to adjust the embedded parameter of the drone on the embedded software.

Table 3.3 Parameters of the MAV

Parameter	Value	Unit
m	0.071	[kg]
k_T	2.64×10^{-8}	$\left[\frac{Ns^2}{rad^2} \right]$

3.4 Development of the embedded firmware

The embedded firmware is developed using the MPLAB® IDE as the main development environment, the compiler is the XC32 that is employed for the microcontrollers of 32 memory bits, the configuration tool MPLAB® Harmony that helps to define the input and outputs of the system as well as the kind of signal for each pin of the main processor unit, and to load the firmware to the circuit the PICkit 3 is connected to the computer with the developed program and to the board to update all the internal processes to the PIC32MZ2048ECM100. Every software employed in this task, the programmer and the microcontroller come from the Microchip company.

The electronic layout designed in [3] gives the configuration of the microcontroller pins, establishing if are inputs or outputs, communication bus or if the main function is for the control of the motor speed through the use of a PWM. Even if the electronic layout was modified for an easier interpretation and reduction of possible error during the manufacturing process, the configuration of each pin remained as it was planned from the beginning in exception for the LEDs used as indicators.

To develop an efficient firmware to be programed on the MCU, the tasks that it must do and the order of how it must be implemented was defined and ordered, designing a Finite-State Machine (FSM) to avoid collision between tasks. The use of the MPLAB® Harmony tool is convenient since the main function of the app file is presented as an FSM where more states can be added in the adequate files.

The firmware development is characterized by the adjustment of the PWM period and the change of the PWM instances order in the first subsection, and the design of the FSM is addressed in the last subsection.

3.4.1 PWM Implementation

The electronic board has a design with some components that restricts the characteristics that the PWM must have.

According to the study of the presented in [11], the optimum frequency of the PWM to control the motors using the MOSFET and the Schottky diode employed on the project is about 20 [KHz].

To adjust the PWM to that frequency it is necessary to review the datasheet of the PIC32MZ2048ECM100, where the operation standard and the adjust of the adequate parameter for the correct functionality of the microcontroller are given.

The parameter to be adjusted for the PWM frequency is the period of the timer 2, since this clock gives the counting tics to reset and adjust the signal of the pins signal. To adjust this data, it is necessary to know the main clock frequency from the MCU. According to the specifications, the it works at 200 [MHz]with the internal oscillator.

The second characteristic to stablish the period of the timer 2 is a prescale value that helps to reduce the frequency of the main oscillator. The minimum prescale value is 1, this causes that the main frequency of the internal oscillator will be divided by two, since the timer 2 switches between high and low state every time the main clock signal sends a rising signal, ignoring the falling signal. The selected value is 1.

The calculus of the timer 2 PRx parameter is given by a formula found in [12], where the $PR2$ is the period to be found and adjusted, the T_{PB} is the period of the main clock, and $TMR_{prescale\ value}$ is 1, since it helps to reach the desired value and allows a better precision on the adjustment of the PWM Duty Cycle .

$$PWM\ Period = [(PR2 + 1) \cdot T_{PB} \cdot TMR_{prescale\ value}] \quad (8)$$

To continue with the adequate order of the instance order of the PWM in the controller configuration, it is necessary to define the order of the motor numbering since it defines the equation of control later discussed in this chapter. The Figure 3.10 shows the order defined for this project on a quadrotor with cross configuration and the corresponding pins that control each motor on the board.

The MPLAB® Harmony tool has the possibility to define the quantity of the Output Compare (OC) instances that will be set and how it will work, being defined as PWM Edge

Aligned on the main configuration page. This interface numbers and identifies each PWM with a number to name each function related to that instance, each one having the format of *DRV_OCx_FunctionDescription()*, where the “x” is the number of the instance of the Output Compare.

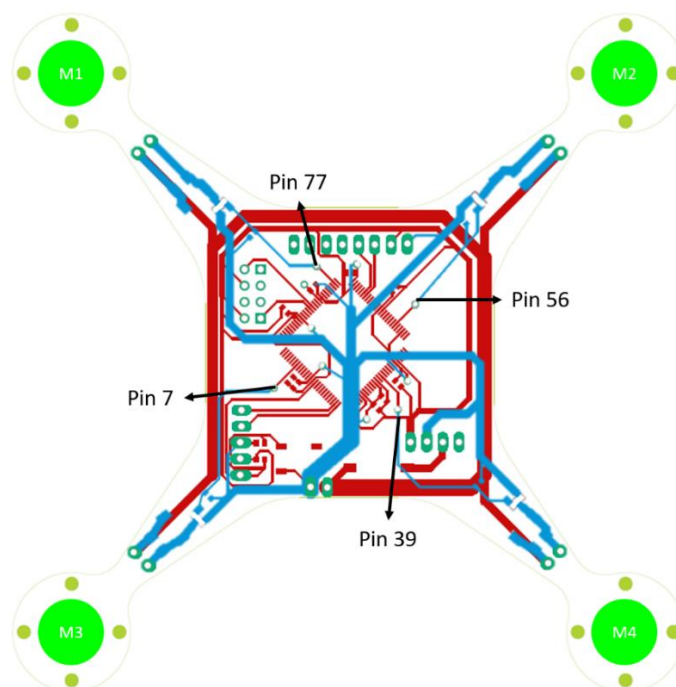


Figure 3.10 Quadrotor motor numbering and corresponding pins

At the same time, the microcontroller has some restrictions. It can control up to 9 OC modules at the same time, that can be mapped to any of the 9 OC instances. Each OC module is numbered and can only be assigned to certain pins of the MCU. This limits the modules that can be used, since the electronic board has paths assigned to each motor.

Since the control equations relate the speed of a specific motor in order of the position that it has in the main structure in relation with the center of mass of the vehicle, to avoid a wrong interpretation, the number of the instance and the motor must be related. In the Table 3.4 it can be appreciated the final relation between the pin, OC module, OC instance and motor.

Table 3.4 Assignment of the Instance, Module and Pin for each motor

Motor	OC Instance	OC Module	Microcontroller Pin
1	0	3	77
2	1	1	56
3	2	2	7
4	3	9	39

This mapping helps to define inside the final firmware the instance number and how will interact with the other instances to control the orientation and attitude of the Quad-MAV in the air.

3.4.2 FSM definition

The MPLAB® Harmony produces a series of header and application files optimizing the code generation and reducing the error margin at the time of producing the main code. In the main application file, it starts a Finite-State Machine that helps to decompose the whole set of functions that the microcontroller must accomplish into a set of simple functions that can be scheduled in a specific order to avoid a misinterpretation of the data or entering into an infinite loop inside a function since it does not accomplish the requirements to finish the task.

The first step was to identify the main functions in the most general form to be performed by the Main Controller Unit in the exact order and as follows:

- Initialize variables.
- Initialize peripheral drivers.
- Establish communication with the external modules.
- Read current state.
- Read current reference.
- Solve control equation.
- Adjust PWM width.

To each one of this set of tasks is assigned a state inside the program, and afterwards must be decomposed into more simple tasks or joining some tasks to reduce the memory employed for each state.

The initialization tasks can be merged into a simple state without altering the order listed before, this due to the influence that many variables and constants play mainly in the preestablished functions of the PWM, the communications and how this could affect the functionality of the control equations.

The communications between the peripheral modules is established in other state that must be divided into two tasks denominated handshakes. The purpose of this sub-states is to establish communication and start the process of data reception from the WiFi module and from the IMU unit. This process is necessary, since the IMU has an internal initialization process, where it needs to filter and process the sensor information to measure the initial state and later send the data in Euler angles through the UART bus for an internal calibration. At the same time, it must wait the communication from the ground station to the WiFi module. This connection must be enabled before starting the loop of send-receive data, this to avoid communication and connection issues. If it is not implemented in this way and the microcontroller starts to send the information to the WiFi

module through the data bus, the communication can't be established since the module gets saturated in the tasks and ignores the client for a handshake.

After the tasks of the communication establishment are successfully accomplished, the control equation, adjust PWM and read tasks are enabled. Each communication initialization enables the corresponding reading task while the control and PWM tasks are only enabled when both communication tasks are accomplished, since the control task requires the information of the current state provided by the IMU and the reference provided by the WiFi antenna. The read tasks are enabled after the communication is established successfully since each module sends information through the respective bus periodically. The reading tasks helps to clean the buffer of both UART modules and avoid the saturation of the buffers allowing a better performance of the Quad-MAV.

The reading tasks in both cases must be separated into three subtasks:

- Data read.
- Data storage.
- Data interpretation.

These subtasks are necessary since the data provided by the modules cannot be used by the microcontroller due to the format. The first task is to read the buffer where the data is temporary stored by the MCU into a variable that matches the data type.

The IMU module sends the Euler angles in a range of data that varies in the length, the Figure 3.11 gives the example of the longest and the shortest chain of data. The data given by the first character that sends is a “#” that helps to identify the start of the chain data and the last character is a “\n”, in ASCII code it means an end of line. After the first character, the IMU reveals the order of the data that will be sending in the form of “TYPR”, this indicates that the first value to be visualized is the altitude of the vehicle, the second value is the yaw, followed by the pitch and ending with the roll state. After the order of the data it sends a character “=” that indicates that the transmission of the values of the state will be start. The Euler angles vary in length, going from -180° to 179.99° . The shortest chain of data that indicates the value of the angle consist of only 4 characters, being only one character for the integer part, followed by a “.” and two characters for the fractional parts. An example could be the initial angle of “0.00”, where all the parts in the worst case can be appreciated. The longest chain of data consists of 7 characters, where the first character indicates a negative value of the angle, followed by 3 characters of the integer part, the “.” to separate the integer from the fractional part, and two characters of the fractional part. An example of this chain can be “-180.00”, indicating that the vehicle is flipped in a determined angle. Each angle is stored in an array of characters that has 4 rows by 8 columns. Each Euler angle and the altitude of the vehicle are separated by the previous value by a “,” that is not stored in the array, instead it is replaced by “\0”, that means the end of the data to be transformed into a floating value. The end of line is

position of the Quad-MAV, calculates the error and its derivative and solves the equations (2)-(5) in their discrete form respectively.

The control used in this project was a PD controller, where the gains are defined as universal floating constants inside the code.

This state calculates the spinning velocity required by each motor to later be transformed by a linear mathematical formula into a parameter that is used by the Adjust PWM width, where it takes an integer value and is set using the internal functions given by the syntax `DRV_OCx_width(value)`.

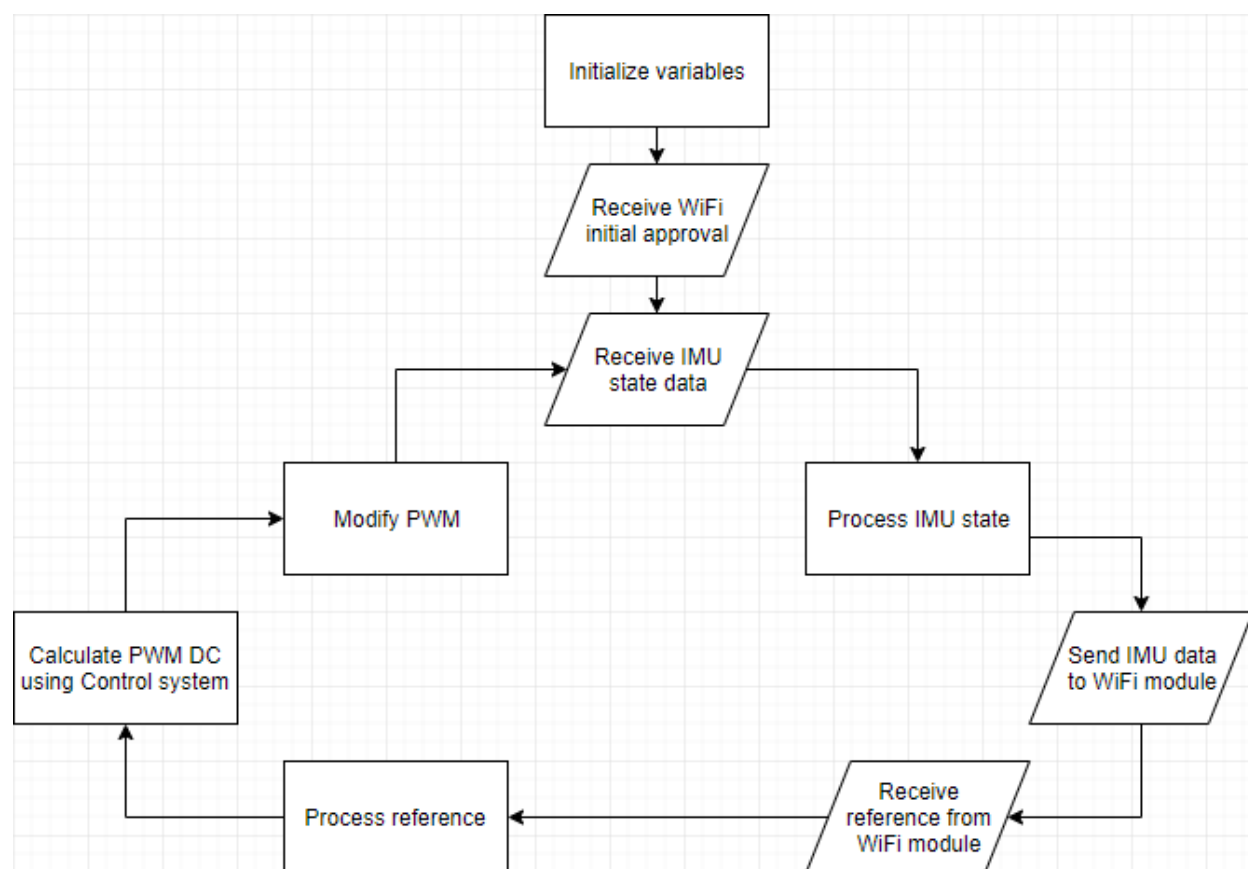


Figure 3.12 Simplified flow chart

Once defined the steps for each state, the loop must be defined. The initialization tasks and the establishment of the communications must be done only once after the Quad-MAV is turned on; the other 4 tasks must be done in a cycle of 10 [ms] due the constrains of the peripheral modules. The final cycle starts when the PIC32MZ2048ECM100 reads the value of the current stat from the IMU, proceeds to read the reference from the WiFi module received from the ground station, afterwards the control law is solved and then

the final values of the PWM width is written to adjust the motors velocity. In the Figure 3.12 can be observed the simplified final flow chart of the FMS.

3.5 Adjustment of the real model embedded parameters

Many adjustments of the parameter needed employ the real period of the microcontroller or the relationship between the duty cycle of the PWM and the maximum force that each motor can deliver to enable the basic hovering function of the drone.

3.5.1 Adjustment of the periods of the timers

The scope of this adjustment process was to adjust the duty cycle of the PWM to run each one at 20 [KHz] of frequency. These measurements where needed since the microcontroller main oscillator was the inner one.

To follow a KIS (Keep It Simple) manufacturing and firmware configuration methodology, the inner microcontroller was selected, since it represents no extra weight to the Quad-MAV and simplifies the electronic circuit and the configuration process to develop the embedded code using the MPLAB® Harmony.

A disadvantage of using the internal clock of the microcontroller is the lack of precision and stability, at high frequencies and can vary depending the environmental conditions, the temperature of the microcontroller and the board and the usage time it has. This timer needs to be adjusted along the lifetime of the MCU of the drone due it deteriorates along the time.

According to [13], the microcontroller has an inner oscillator of 200 [MHz], meaning a $T_{PB} = 5 \times 10^{-9}$ [s]. After replacing this value on the equation (9), where the timer 2 period from equation 1 is cleared, the result is equal to $PR2 = 9999$.

The second timer to be selected depends on the value of the period needed. This second timer task is to ensure the repeatability of the control task on a periodic way. Since the simulations were made every 10 [ms], then $T_{PB} = 0.01$ [s]. This value is substituted on the equation (9), giving a result of $PR4,5 = 1999999$. Since the result requires a higher resolution than 2^{16} can provide, then the timer 3 can't be used, being substituted by the timer 4 with a configuration of 2^{32} of resolution bits.

The value of the timer 2 and the resolution bits of the timer 4 are updated on the MPLAB® Harmony configurator, while the $PR4$ value selected is equal to 4,294,967,295, to allow the timer to reach the calculated value, that is defined as a constant inside the embedded code and define the period of the cycle of the FSM.

3.5.2 Adjustment of constants in control equations

As mentioned on a previous section, the control system designed and that is programmed on the Quad-MAV main controller is a PD. This controller, as mentioned before, requires a compensation to counter the effect of the acceleration due to gravity.

According to [9] the compensation must be set after the altitude control to function as a countereffect to the force produced by the Quad-MAV mass and the gravity. For a better comprehension for this adjustment on the control law it is necessary to solve the equations and observe the behavior when the error is equal to zero on every reference.

By solving the equations from (2) - (5) when all errors are equal to 0 and replacing these values on the equation (6) the following equation is stated.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \omega_H \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (9)$$

As mentioned on a previous chapter, the hover force must be equal to the weight of the drone. Solving the equation (7), using the physical parameters of the vehicle, the final value of the hovering constant to linearize the plant is equal to $2.5678 \times 10^3 \left[\frac{rad}{s} \right]$.

A second factor to consider is the direction is the orientation of the positive “Z” axis on the inertial frame. To elevate the drone over the ground level using the coordinated system proper of the aerial vehicles, the reference must be negative.

This has a consequence over the control system. The PD uses gains to reach the reference. Since the references of the *Thrust* control is on the negative coordinated axis, it requires negative gains to have a positive output on the ω_F value.

While the gains of the *Thrust* control must be negative, the gains of the *Roll* and *Pitch* controls will take the same values on the simulation. These values are the same since the behavior of the quadcopters is the same on both angles. This is possible due the symmetry that have over the “X” and “Y” axes.

Table 3.5 Control gains values.

Controlled Variable	Gain	Value
<i>Thrust</i>	Proportional	-80
	Derivative	-200
<i>Roll</i>	Proportional	5
	Derivative	5
<i>Pitch</i>	Proportional	5
	Derivative	5
<i>Yaw</i>	Proportional	4
	Derivative	5

The gains corresponding to the Yaw variable are the smaller of all. Since this angle has no effect over the rotation of the any axis and the stabilization in this spinning is easier, the gains can be small to avoid overshooting and control loose of the spinning over the “Z” axis. The final values of the gains are shown on the Table 3.5.

3.6 Discrete control equations and motor velocity calculus

The control equations used on the modeling of the system are represented on continuous time and using matrix to simplify the model and reduce the error possibility between each step since each variable has effect over different stages of the position of the drone within the space.

A limitation within the project is the programming language and the main processing unit device. The ‘C’ language and the MPLAB® Harmony provided by Microchip® have no native libraries regarding differential equations for derivatives and integration processes. At the same time the PIC32MZ2048ECM100 has a limited processing capability due the hardware characteristics. This properties of the software and hardware leads to the use of the basic definition of a derivative equation.

The basic form of a derived is of a slope where the difference of a step over the dependent variable is divided by the difference of the same step at the independent variable as shown the following equation.

$$m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x} \quad (10)$$

Substituting the variables using the concepts of the slope equation, the current state of the drone changes continuously every instant, meaning that the dependent variable is the position of the drone and the independent variable is the step of time between each measurement. Substituting these concepts on the equation (10) the it can be established:

$$\dot{e} = \frac{e_1 - e_0}{t_1 - t_0} = \frac{\Delta e}{\Delta t} \quad (11)$$

The measurement carried by the IMU module is not constant with a variation from 9 to 11 milliseconds between the transmission of the current state of the drone. To simplify the process and avoid errors on the microcontroller buffer and timesteps between the solution of the sampling the period of the FSM was set to be 10 milliseconds, meaning that the declared value is $\Delta t = 0.01$ inside the code.

Solving the control equations from (2)-(5) using the definition of the equation (11) the equations (12)-(15) are defined where the subscripts k means the current state and $k - 1$ previous state. These equations are programed into the Control state inside the FSM of the embedded code and immediately the value of e_k is transferred to e_{k-1} .

$$\Delta\omega_F = K_P \cdot e_k + K_D \cdot \frac{e_k - e_{k-1}}{0.01} \quad (12)$$

$$\Delta\omega_\phi = K_P \cdot e_k + K_D \cdot \frac{e_k - e_{k-1}}{0.01} \quad (13)$$

$$\Delta\omega_\theta = K_P \cdot e_k + K_D \cdot \frac{e_k - e_{k-1}}{0.01} \quad (14)$$

$$\Delta\omega_\psi = K_P \cdot e_k + K_D \cdot \frac{e_k - e_{k-1}}{0.01} \quad (15)$$

The values to be adjusted inside the code are the corresponding to the motor angular velocity. To obtain the value of the motors, the equation (6) is solved. Since this equation multiplies a 4×4 matrix by a column vector of 4 elements, it delivers a total of 4 equations to adjust the speed, one for each motor, as can be seen in the next set of equations.

$$\omega_1 = \Delta\omega_T + \Delta\omega_\phi - \Delta\omega_\theta + \Delta\omega_\psi + \omega_H \quad (16)$$

$$\omega_2 = \Delta\omega_T - \Delta\omega_\phi - \Delta\omega_\theta - \Delta\omega_\psi + \omega_H \quad (17)$$

$$\omega_3 = \Delta\omega_T + \Delta\omega_\phi + \Delta\omega_\theta - \Delta\omega_\psi + \omega_H \quad (18)$$

$$\omega_4 = \Delta\omega_T - \Delta\omega_\phi + \Delta\omega_\theta + \Delta\omega_\psi + \omega_H \quad (19)$$

Since there is no sensor to measure the spinning velocity of the propellers, some data must be gathered corresponding to the angular velocity it can reach and the voltages. According to the page of Ready Go, a supplier of the selected motors and similar equipment, the values of the voltage, spinning velocity and other data were obtained.

Table 3.6 Motor characteristics at 3.2 V

Characteristic	Value	Unit	Tolerance
Speed	35000	[RPM]	±15%
Current	0.48	[A]	0.58 Max
Torque	0.35	[mNm]	0.28 Min

The motors have a linear relationship voltage-torque it can deliver, having as an effect a linear relation with the force it can deliver. To find the angular velocity and Duty Cycle required on the code, the equation (20), where F is the force delivered by the motor, ω is the speed it reached and k_t is the thrust constant, was solved using the value of the speed at 3.2 [V] provided in the Table 3.5 and converted into $3665.191429 \left[\frac{rad}{s} \right]$, then the slope equation (21) was employed to find the relationship Voltage-Force of the motor, where V is the feeding voltage of the motor.

$$F = \omega^2 \cdot k_T \quad (20)$$

$$m_{FV} = \frac{F}{V} \quad (21)$$

The battery has a nominal voltage of 3.7 Volts, having a maximum of 4.2 Volts when completely charged. Since the velocity of motors is regulated using a PWM, the 100% was established as 4.2, the corresponding force was calculated using the equation (21) and later the squared velocity of the motor employing the equation (20) to develop the equation (22) to calculate the duty cycle of any motor according to the ideal model, simplifying the equation using a constant as seen on the equation (23). The values of the parameters found are on the Table 3.7.

$$DC_{x\%} = \frac{\omega_x^2 \cdot DC_{100\%}}{\omega_{4.2V}^2} \quad (22)$$

$$DC_{x\%} = \omega_x^2 \cdot m_{DC-\omega^2} \quad (23)$$

Table 3.7 Constants parameters employed to find the equation 42

Parameter	Value	Unit
$F_{3.2V}$	0.354648	[N]
m_{FV}	0.110827	$\left[\frac{N}{V} \right]$
$F_{4.2V}$	0.465475	[N]
$\omega_{4.2V}^2$	1.7631637×10^7	$\left[\frac{rad^2}{s^2} \right]$
$m_{DC-\omega^2}$	0.0002835811	$\left[\frac{Bits \cdot s^2}{rad^2} \right]$

Chapter 4

4 Experiments and analysis

This section addresses all the process for the development of the experiment and discusses the results obtained from the experimentation and how it leads for an adjustment of parameter needed for the integration of the embedded software and the electronic system. The set up for each experiment is stated at the beginning, establishing the conditions to determine if the tested section accomplishes the requirements and if the parameter previously established are adequate. If the test is not accomplished or the parameter are not adequate, it shows the procedure to solve the issue and continue the final integration of the project.

The equipment used for the experimentation was:

- Multimeter: measure continuity, capacitance and voltage.
- Oscilloscope: used to measure the PWM frequency and width of the duty cycle
- Balance: to measure the real weight of the drone and the force delivered by the motor.
- Protoboard: used to test the functionality of the electronic devices.
- Arduino Uno and FTDI: used as a *RS232* communication interface with the computer.
- IMU SEN-14001: board with the sensor for the orientation sensing.
- ESP8266: wireless compact communication module.
- U1V11F3: step-up/step-down to be used on the drone to regulate the voltage.
- Power source or drone battery: to energize the circuit for the testing. The power supply is used exclusively for the protoboard while the battery is employed for durability test on the protoboard and final tests.
- Prototype: used for the final tests of functionality
- Ground station: a computer or other device to send the reference to the drone.

The software and electronic hardware test were made in parallel. The electronic functionality hardware and software were made before the prototype electronic hardware were done to avoid reworks and wrong prototypes.

Each test presented in this chapter deliver the information in form of a small report, where the objective is presented, the list of the tools, components and equipment used, followed by an image or picture of the setup of the testing, small description of the procedure and the result. In the case where the test fails to meet the acceptance criteria, an adjustment

of parameters is done, and the test runs again into a new iteration to deliver a new judgment about the test status.

4.1 Data collection test

The data collection test is used to gather the information regarding the electronic and the embedded systems. It is necessary since gives the information about the cycle time and that the proper integration of the different modules can be achieved.

4.1.1 Data reception and transmission from between the IMU, WiFi and microcontroller

Objective. Understand the functionality of the WiFi and IMU modules, the data gathering inside by the microcontroller and ensure the data transmission from IMU to the main controller unit and from the WiFi antenna to the PIC32MZ2048ECM100 and determine the time it takes to accomplish this task to determine the time required for the FSM to complete a cycle . This test procedure was divided into three different validation stages.

Equipment:

- Protoboard
- Power source
- Ground station

First stage. The IMU module was mounted on the protoboard and connected to the testing microcontroller on the respective Rx and Tx pins. An Arduino Uno board without the ATMEGA328P was connected to another $RS232$ communication pins destined to the WiFi module to transmit the data gathered from the IMU by the microcontroller to the ground station. The protoboard was fed with 3.3 [V] directly from the power source. The Arduino IDE was used to receive the data on the computer (Figure 4.1).

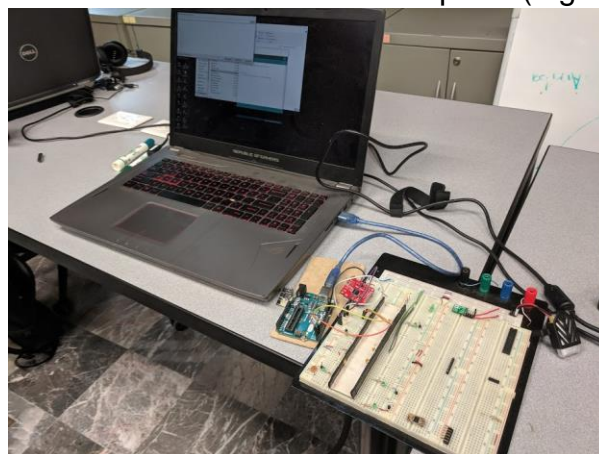


Figure 4.1 IMU-PIC32-GND station communication setup

Result. The data transmission was successful, the PIC32 received and processed the data from the IMU and resent it to the ground station (Figure 4.2).

```
#TYPR=0.00,0.00,0.00,0.00
#TYPR=0.06,0.00,0.00,0.00
#TYPR=0.12,0.00,0.00,0.00
#TYPR=0.19,0.00,0.00,0.00
#TYPR=0.25,0.00,0.00,0.00
#TYPR=0.31,0.00,0.00,0.01
#TYPR=0.38,0.00,0.00,0.01
#TYPR=0.44,0.00,0.00,0.01
#TYPR=0.50,0.00,0.00,0.01
#TYPR=0.57,0.00,0.00,0.01
#TYPR=0.63,0.00,0.00,0.02
#TYPR=0.69,0.00,0.00,0.02
#TYPR=0.76,0.00,0.00,0.02
#TYPR=0.82,0.00,0.00,0.02
#TYPR=0.88,0.00,0.00,0.02
#TYPR=0.95,0.00,0.00,0.03
#TYPR=1.01,0.00,0.00,0.03
#TYPR=1.07,0.00,0.00,0.03
#TYPR=1.14,0.00,0.00,0.03
#TYPR=1.20,0.00,0.00,0.03
#TYPR=1.26,0.00,0.00,0.04
#TYPR=1.32,0.00,0.00,0.04
#TYPR=1.38,0.00,0.00,0.04
#TYPR=1.44,0.00,0.00,0.04
#TYPR=1.50,0.00,0.00,0.04
```

Figure 4.2 Data received from the PIC32 into the computer

Second Stage. The Arduino board was replaced by the WiFi module to transmit the data to the ground station. The IDE was replaced by the PuTTY software, it is as a client for different telecommunication protocols (Figure 4.3).

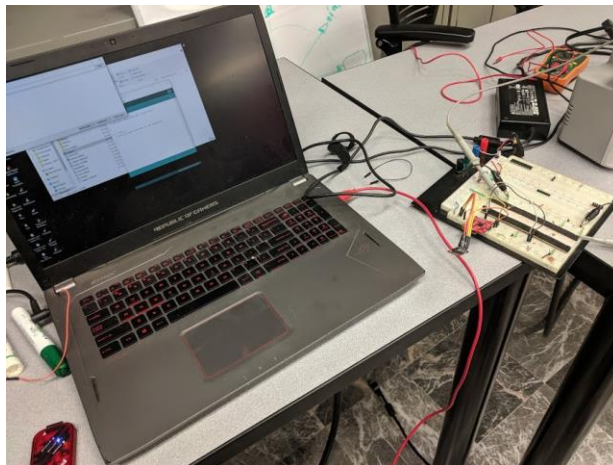


Figure 4.3 Second stage setup

Result. The procedure failed. The IMU is continuously sending data to the microcontroller that sends it to the WiFi module at the same rate it is received, saturating this module and preventing the wireless communication. To avoid this issue the firmware was modified such that the main processing unit only sends the data from the IMU under request.

New result. The communication was successful; the microcontroller receives the data from the IMU and processes it and is capable to act as a response to a command received from the WiFi module (Figure 4.4).

```
T
#TYPR=2.03,0.03,-0.12,15.4
```

Figure 4.4 Console command and response from the PIC32 using PuTTY

Third stage. Send a command through the wireless communication to modify the Duty Cycle of the four PWM or receive data. The setup of the second and the third stages are the same.

Result. The PWM of each motor could be modified. A LED and a resistor were placed after each pin regarding a motor to identify the change of the pulse width using certain preestablished commands (Figure 4.5). The main processing unit also responded to a command to send the current IMU status to the ground station (Figure 4.6).

```
T
#TYPR=3.05,0.51,-1.20,20.3
w
s
d
a
```

Figure 4.5 Adjustment of PWM with command transmission.

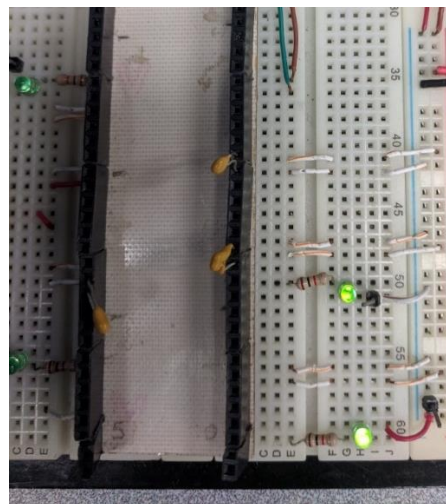


Figure 4.6 PWM regulated on the LEDs (motor pins)

Fourth stage. Determine the time it must take to the MCU unit to complete the cycle of the FSM to control all the tasks in the embedded code.

Result. The cycle of the FSM was determined using the rate of the tasks that takes more time to be completed. This task is the read of the processing and transmission of the data from the IMU to the MCU. This task varies from 9 [ms] to a value over 12 [ms]. Using the data collected and transmitted to the ground station from the MCU (Table 4.1), the data was evaluated and graphed (Figure 4.7) to obtain the following results:

- Maximum time: 12.712 [ms]
- Minimum time: 9.803 [ms]
- Average: 10.6763 [ms]
- Mode: 10.79 [ms]
- Standard deviation: 0.434 [ms]

Table 4.1 Data from the normal distribution

Ranges	Bottom Value [μ s]	Top Value [μ s]	Samples
Range 1	9803	10236.77825	470
Range 2	10236.77825	10670.5565	856
Range 3	10670.5565	11104.33475	860
Range 4	11104.33475	11538.113	385
Range 5	11538.113	11971.89125	57
Range 6	11971.89125	12405.6695	0
Range 7	12405.6695	12712	1

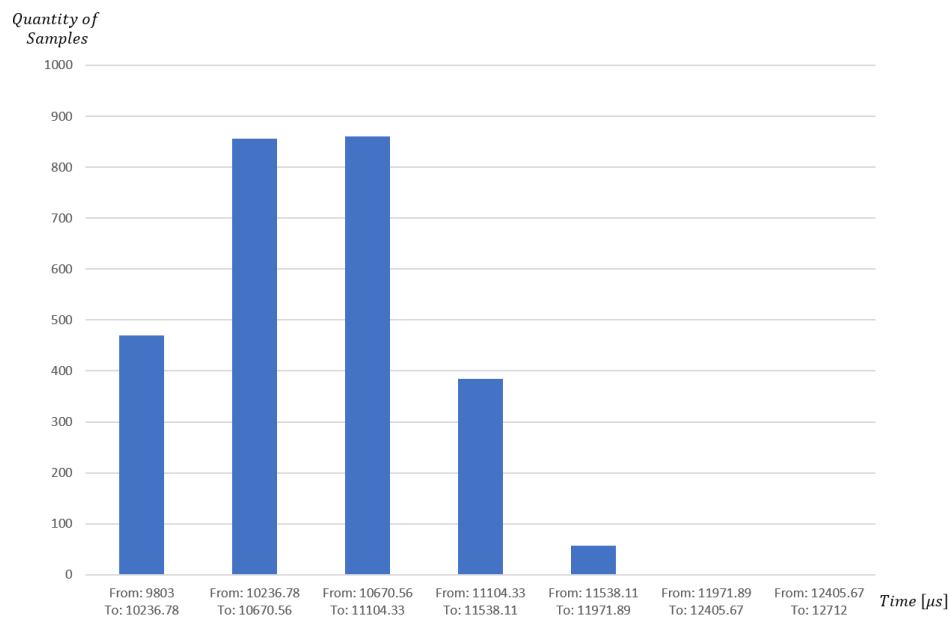


Figure 4.7 Normal Distribution of the sample of times

Using this data and simplifying the selection of the timers 4 and 5 PRx parameter avoiding the effect of the lack of stability from the internal oscillator MCU, the value of 10 milliseconds was selected.

4.2 Timers testing

The testing of the timer helps to release the most recent version of the code needed to make the drone to function properly. This test helps to adjust the frequency of the PWM and the period of the timer to ensure the functionality of the FSM.

4.2.1 PWM frequency and timer 4 period

Objective. To verify the PWM frequency delivered from the microcontroller to each motor and measure the frequency of the timer to schedule the reset of the FSM cycle in the code.

Equipment:

- Oscilloscope
- Power source
- Protoboard

Description of the testing. Supplying the circuit with the power source, it was turned on the circuit, uploaded the firmware with the setup enabling the PWM and the timers 2 and 4 of the microcontroller, a 50 [%] duty cycle, the output pins for the PWM with LEDs and the output pin for the LED to measure the period of the timer 4 to ensure the duration of the FSM loop for 10 milliseconds connecting the corresponding pins to the oscilloscope (Figure 4.8).

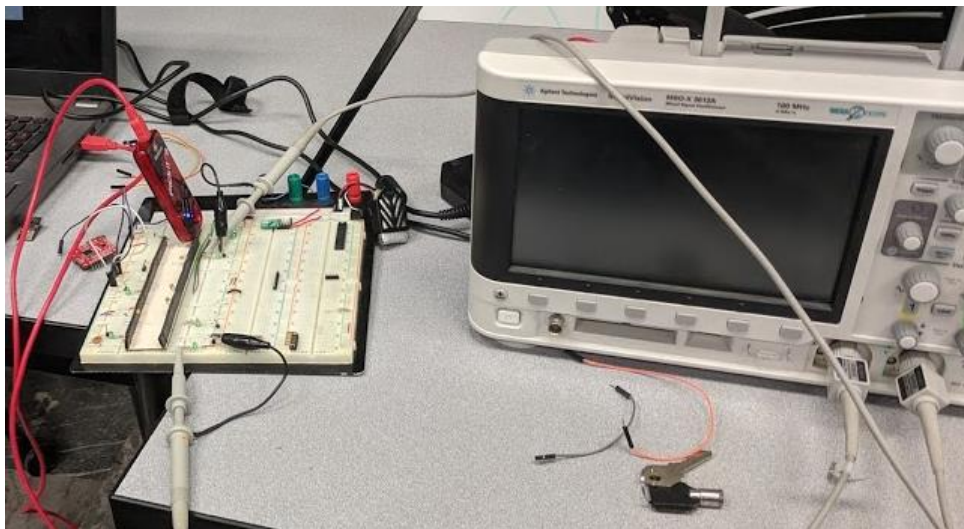


Figure 4.8 Experiment setup for frequency of the PWM and timer 4 period

Result. The test failed on this first trial. The configuration on the Harmony tool for the microcontroller was ok, but the PWM frequency was the half of the desired (Figure 4.9) and the period of the timer 4 was doubled (Figure 4.10). The reason is due the use of a wrong data. The T_{PB} used in the calculus of the PRX register of both timers is from the main oscillator, corresponding to the timer 1 from the microcontroller. The T_{PB} corresponding to the other timers is equal to the half of the period from the timer 1.

Notes. The periods marked even numbers being 9,999 and 1,999,999 for timers 2 and 4 respectively using the formula. For a more exact calculus inside the microcontroller regarding the Duty Cycle of the PWM both were replaced by 10,000 and 2,000,000.

Discussion for a new test. The period of both timers must be decreased. Since the frequency of both measurements is the half of the desired, the PRX parameter of both timers must be reduced in a 50 [%] from the original calculated value.

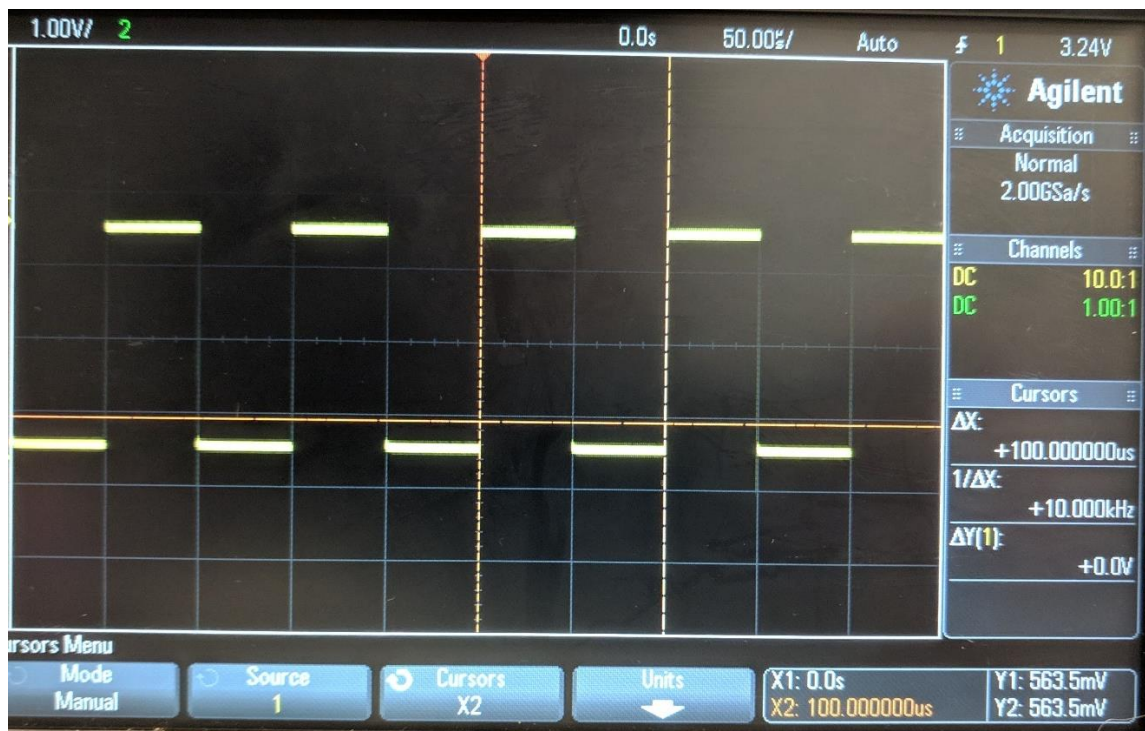


Figure 4.9 First PWM frequency measurement.

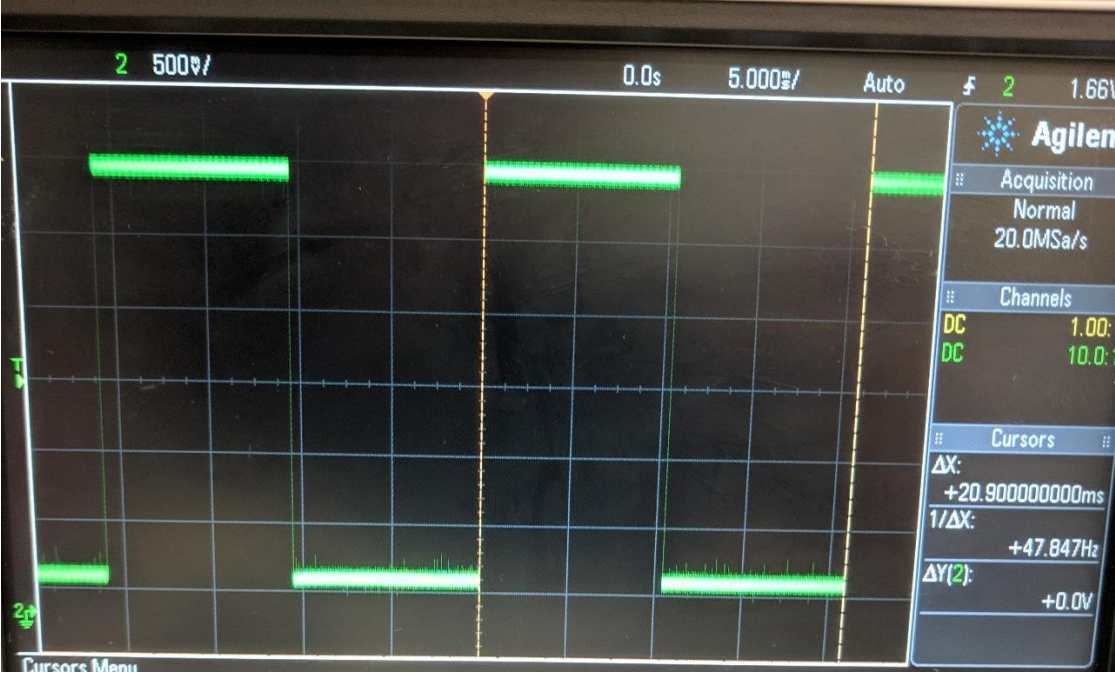


Figure 4.10 First Timer 4 frequency measurement

New result. Both, the PWM frequency was set at 20 [KHz] (Figure 4.11) and the timer 4 period met the value of 10 [ms] (Figure 4.12). Both measurements have small deviations from the desired output, but it does not affect the functionality of the vehicle.

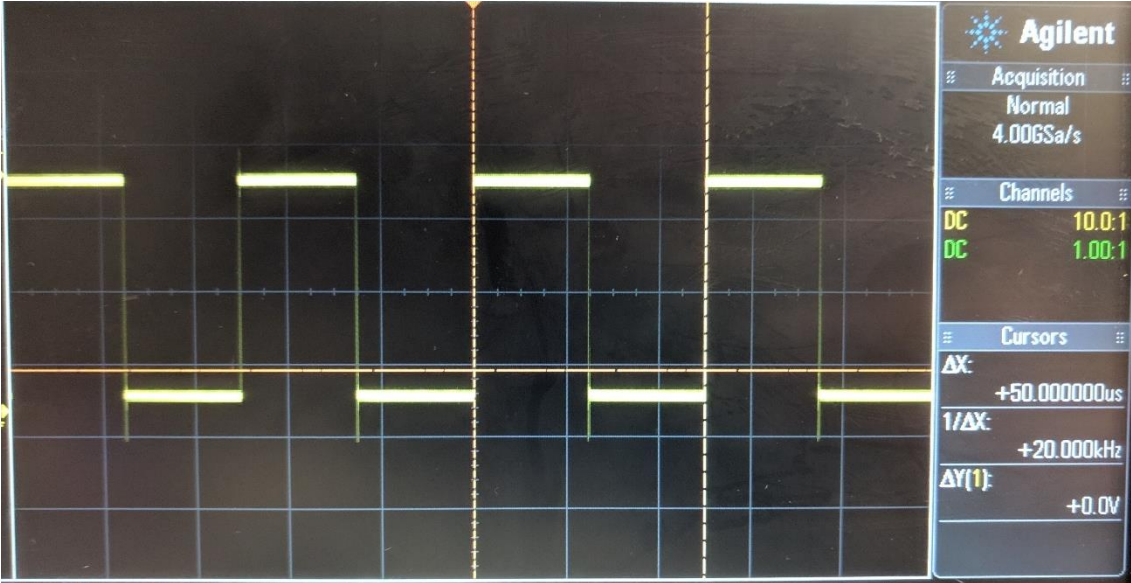


Figure 4.11 PWM frequency at 20 kHz

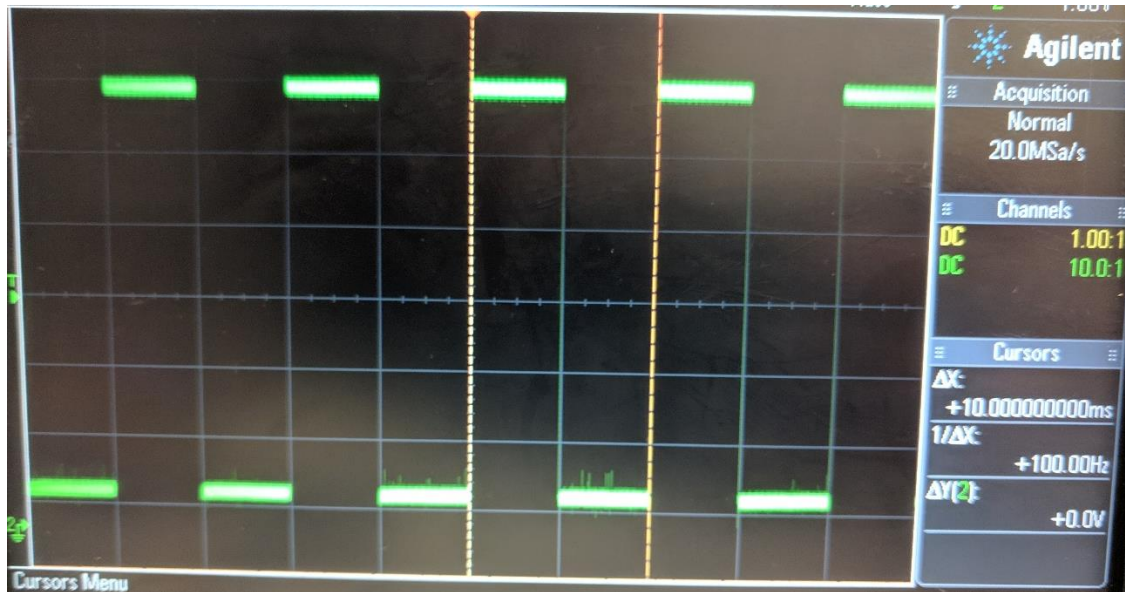


Figure 4.12 Period of the timer 4 for the FSM tasks

4.3 Prototype testing

After the release of an effective prototype is done, the prototype motors must be tested in an individual way to ensure the functionality and correct the error on the force that can be delivered and the adequate slope to transform the velocity of the motor calculated with the control equation into the information required by the main controller unit.

After the corrections are done, the flying test is carried with the prototype. The flight test was done first using a platform where the drone was anchored and then finally run a test with the drone free of movement restrictions.

4.3.1 Slope for the control of motor velocity

Objective. To find the duty cycle where each motor starts to spin and the maximum force it delivers at the battery maximum voltage.

Equipment:

- Ground station
- Prototype
- Balance

Description. The drone is placed over the balance with a foam base to avoid it to move out of the balance. Each motor has a propeller with the inverted design to lift the flight, it means that if the motor spins clockwise it has a propeller that is designed to the motor that spins counterclockwise direction, to deliver the propulsion force towards the balance.

Each motor is measured one by one. The balance is set to zero when the drone is on the board to measure the propulsion of each motor (Figure 4.13).



Figure 4.13 Testing setup to measure the force delivered by the motors

Result. Each motor showed different properties, at different Duty Cycle each motor starts to spin, while at the maximum velocity each motor similar forces. The Table 4.1 shows the different Duty Cycles for each motor. This information is used to calculate the adequate slope to transform the velocity calculated for each motor into the adequate Duty Cycle of the PWM.

Table 4.2 Values obtained from the motor force measurement

Motor	Duty Cycle Before Spinning [bits]	Maximum Force Delivered [N]	Motor velocity at maximum force [rad/s]	$m_{(DC-\omega^2)}$ [[bits·(s ²))/(ω ²)]
1	352	0.20874	2811.906503	0.002484944
2	357	0.2058	2792.034123	0.002519802
3	354	0.20874	2811.906503	0.002484691
4	344	0.21364	2844.718658	0.002428938

Using the values obtained from the measurement, the maximum payload that the quadrotor can carry with the maximum battery voltage is 0.1274 [N] or about 13 [g]. Another factor to consider is the minimum battery charge at which the vehicle can work correctly. If the battery voltage falls below 3.2 [V], the drone fails to perform the hover in steady state and gradually will start to fall to the ground.

The data obtained from these experiments allows to define new equations to transform the angular velocity into PWM Duty Cycle in Bits. The respective equations programmed on the embedded code are shown in equations (24)-(27).

$$DC_1 = \omega_1^2 \cdot m_{DC-\omega_1^2} + 352 \quad (24)$$

$$DC_2 = \omega_2^2 \cdot m_{DC-\omega_2^2} + 357 \quad (25)$$

$$DC_3 = \omega_3^2 \cdot m_{DC-\omega_3^2} + 354 \quad (26)$$

$$DC_4 = \omega_4^2 \cdot m_{DC-\omega_4^2} + 344 \quad (27)$$

4.3.2 Full prototype test

Objective. to confirm, condition or neglect the hypothesis of this thesis. Understand the behavior of the drone and define future work to improve the Quad-MAV. Define the future corrections on the prototype, find possible working failures

Equipment:

- Finished prototype
- Ground station



Figure 4.14 a) Drone anchored b) Drone free for final test

Description. The vehicle is anchored into a platform (Figure 4.14 a) to avoid damage in case of malfunction, some motor does not work properly, or bad calibration is done. If the drone can take off on the platform the ties are released, and a final flight is performed (Figure 4.14 b).

Result. The Quad-MAV was able to take off from the base. The ties destabilized it since they executed a force in multiple directions not contemplated into the model, making it non-linear with no compensation available.

After the drone was released, it hovered over the ground a small distance. It could not be measured since the flight was not completely stable. The main issue present was that the *Yaw* equations were not considered, since the values obtained were not stable when the IMU was mounted over the board.

The *Pitch* and *Roll* equations were more stable even if the drone was spinning over the “Z” axis. During the flight it was observed that the take-off angle was not lost or varied over the time.

Chapter 5

5 Conclusion

This document addressed a methodology for the design and analysis of the modeling of a Micro Aerial Vehicle type Quadrotor with a cross configuration where navigation in closed spaces, where other kind of vehicle cannot navigate or maneuver due the conditions of the terrain or the space is the purpose.

The prototype has the characteristics of being designed with low cost and commercial components with an open-source philosophy, allowing the end user to modify and adapt it to the necessities that needs to be covered at that moment.

The electronic components are of the SMD kind to optimize space in the board and reduce the weight of the vehicle, allowing it to take-off with low power motors.

The general objective regarding the design and prototype manufacturing of the low-cost vehicle was achieved. However, the vehicle electronic design and the embedded code can still be optimized.

In the case of the electronics the voltage delivered to the motors received must be stabilized to avoid issues discussed on a section of this chapter.

The embedded code can be optimized to deliver better response and reduce the cycle of the FSM programed in the Main Controller Unit firmware.

The hypothesis can be declared as truth. Some objectives as a stable flight could not be reached due time and test environment limitations.

5.1 Contributions

Part of the contributions given is an analysis of the Quad-MAV to implement control equations inside an embedded system where the use of integrals and derivatives are limited.

The design of the firmware was analyzed and implemented as a finite-state machine to make a more readable code where a state can be added or removed to adjust it to the capabilities of the code designer and to the architecture of the MCU, more centered on the PIC32 family of the Microchip® brand.

The manufacturing process and the go/nogo testing for the prototype were provided. The methodology was modified since some elements of the micro robotics laboratory were

not available for the manufacturing process at the time. Visual guides and acceptance criteria during the assembly process were defined.

The electronic system was simplified and improved with new components for a more stable power source.

5.2 Limitations

The prototype was made with the use of equipment designed for rapid prototyping. Some of the equipment like the adequate stencil for a faster placement of the SMD components was obsolete and could not be updated properly.

Most of the electronic components were adequate. A limitation of the project is the time it required to assemble a prototype. The time it takes to do only the board is approximately 2 [h], this to remove the excess of copper. The placement of the ProConduct plaster on the proper places and curing the material took 40 [min] to be completed. The soldering of the components takes over 2 [h] to be finished and tested.

5.3 Future Work

The electronic design must be improved regarding the voltage delivered to the motors. Due weight and electronic limitations a step-up/step-down to deliver constantly 5 [V] to the motors was not implemented. The lack of this element causes variations of the maximum force delivered by the motors when at the 100 [%] of the Duty-Cycle. Since the embedded code uses a slope equation where the maximum duty cycle corresponds to 4.2 [V], with the drain of the battery over the use this mathematical relationship is lost.

Some elements like a camera to transmit visual information of the vehicle is still pending and must be reviewed due the weight and electronic requirements to be met.

The embedded software code could be improved to reduce the loop time of the Finite-State Machine and improve the performance of the drone. The filters of the sensors must be improved to reduce the noise in the lectures, allowing the habilitation of the *Thrust* and *Yaw* control equations. In the case of the barometer, if the lectures cannot be stabilized, the sensor must be replaced for one of better resolution or precision.

The embedded code must be able to process video information to transmit images using a camera that must be implemented on the future. Electronic and mechanical analysis and design must be carried to install the camera.

The manufacturing process can be improved by the production of a stencil adequate to the new board model and the proper selection of the tools for the copper roughing process, especially on the selection of the end mill, that requires a tool with a diameter bigger than 4 [mm] to finish the process faster and avoid the tool wear.

Appendix A

MPLAB® Configuration and Code Functions

The development of this project was made using a Microchip PIC32MZ. To avoid compatibility and library problems the main code was made using the company's Integrated Development Environment and the current embedded tools available.

The PIC32 family requires the installation of the following software in the listed order:

- MPLAB® X IDE. The main interface to develop the firmware code to be uploaded to the MCU.
- XC32. Main compiler focused on the 32-bit microcontrollers of the brand.
- MPLAB® Harmony. Development tool used to autogenerating code and developed by Microchip. Simplifies the use of the registers and proper configuration of the pin functionality and mode, the configuration of the peripherals and internal devices of the microcontroller.

It is necessary to link all the software before starting the project. The new libraries for the development of the main firmware for the PIC32 family do not work without the use of Harmony.

MPLAB® configuration

After the proper installation of the software, it is necessary to link the IDE to the development software installed. The XC32 compiler is linked to the X IDE through the Tools menu, on the section of Options. On the emerging window it is necessary to select the tab "Build Tools" on the "Embedded" menu. The Figure A.1 shows the window where the setting is done.

To link the Harmony, it is required to first configure an initial project on the X IDE. To start an initial project the menu File must be opened and on the emerging menu select the proper options as can be seen on the Figure A.2. On the next window the path to the Harmony installation folder must be indicated on the bar indicated by the circle on the Figure A.3. It must be noted that if more than one version of Harmony is installed, the one must select the version that fits better to the project.

Once the project is created, the main window of the X IDE, as the shown in the Figure A.4, will show a list to select the most adequate configuration of the vehicle, where the peripherals, interrupt functions and other settings can be configured. At the right margin of the main window is a secondary one screen that provides basic information of the setting being configured as well as the functions and registers used in that setting and the basic use of them. On the bottom of the window is a peripheral table, where the selected peripherals are assigned to a specific pin. Each single pin can be assigned to a single peripheral each time. Once assigned the peripheral and the pin the system blocks the row and the column unless the assignation is undone.

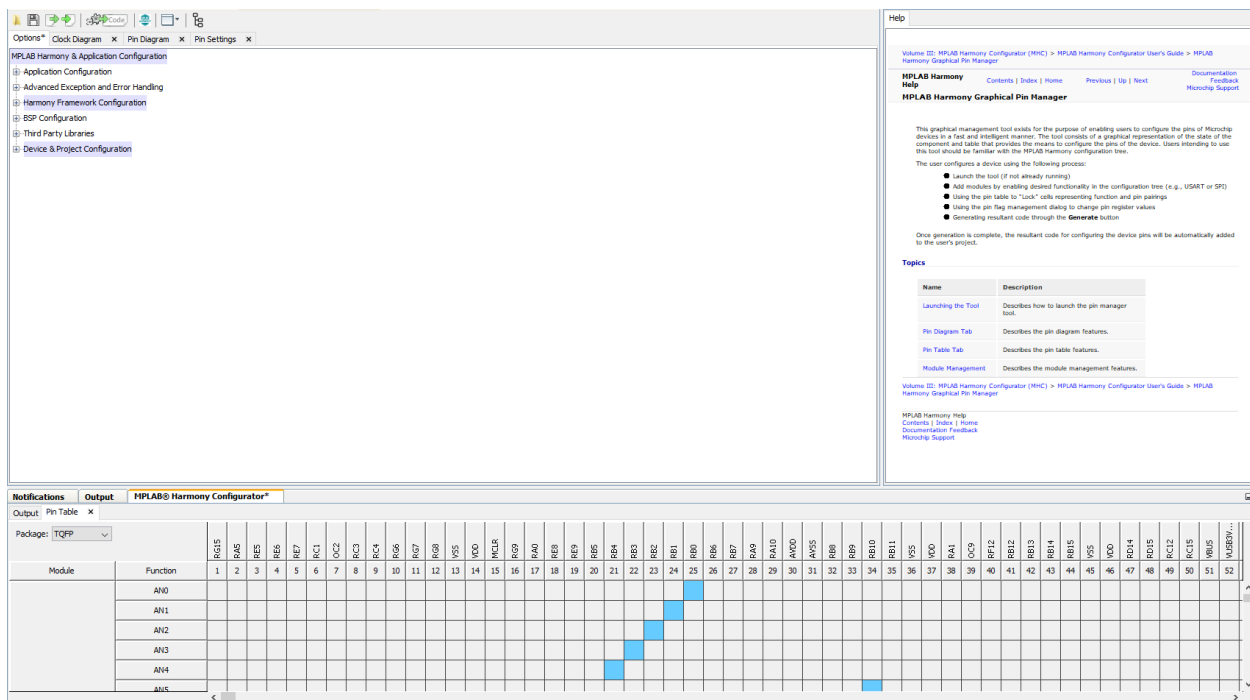


Figure A.4 Configuration of the functions and pins to be used on the project

To configure the PWM, the RS323 communication protocol and the Timers required, the “Harmony Framework” configuration must be opened, followed by the subsection of drivers, as can be seen marked on the Figure A.5.

The first driver to be modified is the Output Compare (OC). The main function of this driver is to generate the PWM function required to regulate the velocity of the motors. The interrupt mode must be disabled, since no interruption service is used in the project. On the option labeled as “Number of OC Driver” must be filled with quantity of PWMs to be used.

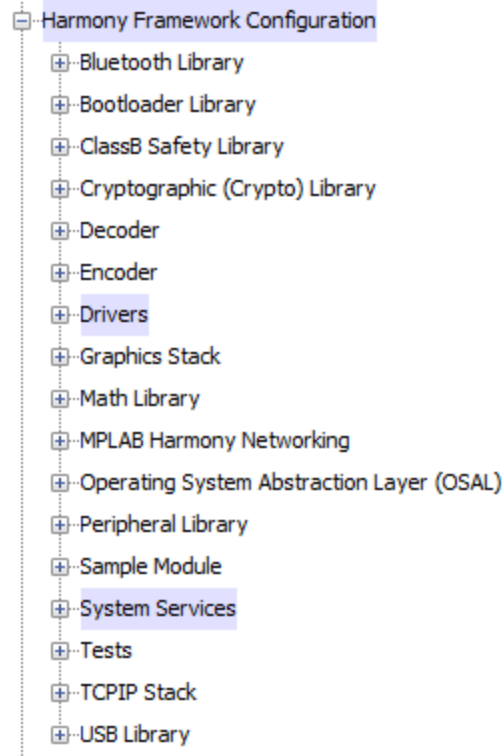


Figure A.5 Options to be modified are highlighted in blue

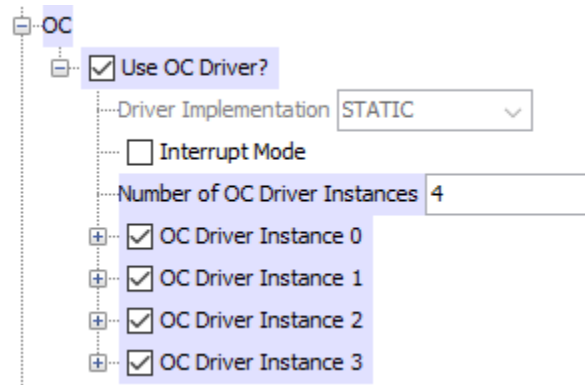


Figure A.6 Defining the number of instances

The information to be modified on each instance is shown highlighted on blue on the Figure A.6. It is important to note that the option “OC module ID” must be different on each instance to avoid errors or redefining a module with a different characteristic and initialize it in two different instances. If two instances have the same module ID, the text will be highlighted red. The option of “OC Pulse Width” must be equal to 0, this with the purpose to avoid the motor spinning when the drone is turned on. This option indicates the Duty Cycle of the PWM in bits. This configuration can be seen on the Figure A.7.

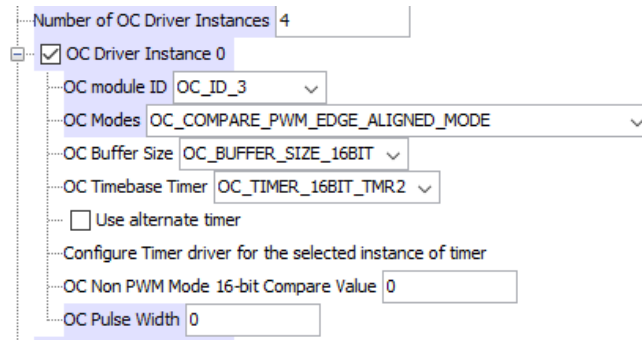


Figure A.7 Defining the OC instances and their properties

The timers option needs to be modified for two reasons. The first one is that the OC works using a defined timer as a base, in this case, the selected timer was the timer 2 as can be seen on the Figure A.8. The second reason is to have a timer counter to limit the cycle time of the FSM to have a constant execution time. The interrupt option must be disabled to avoid possible undesired cycle interruptions.

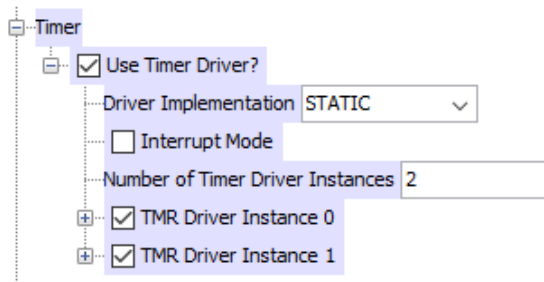


Figure A.8 Defining the number of timer instances

On the option of “Timer Module ID” the adequate timer must be selected. The prescaler can be selected to the preferred and most suitable to the function it will perform. The “Operation Mode” allows to select the maximum period the timer can reach, it can be of 16 or 32 bits (Figure A.9). For the PWM timer it can be the first option and for the second timer it must be second option, since the period of the timer will have a long period. The periods of each timer are calculated using the equation 25, located on the chapter 3.

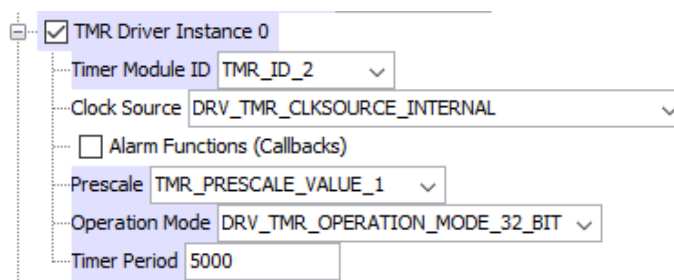


Figure A.9 Adjustment f timer Parameters

The last driver to be configured is the serial communication driver. The number of instances is 2, one used for the reception of information from the IMU and the other one used for the reception of data from the WiFi module. The configuration can be seen on the image A.10.

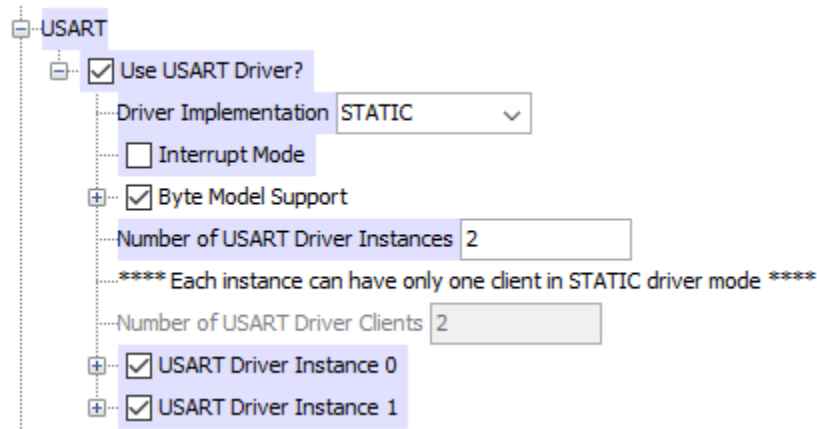


Figure A.10 Definition of the USART instances and disabling the interrupts

The two options to be modified on each instance are “USART Module ID”, where the adequate module is selected according to the defined pins for each peripheral. The second option to be modified is the “Baud Rate”, this helps to define the communication speed, on both cases the value is equal to 115200. The Figure A.11 shows the configuration of the first USART instance.

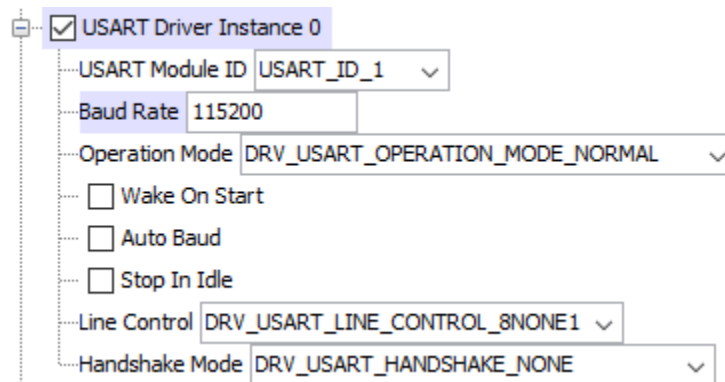


Figure A.11 Adjustment of the USART instance options

Main autogenerated functions

The MPLAB® Harmony tool helps to reduce the time expended on the configuration by the employment of a visual tool to assign the adequate parameters and selecting the input and output pins for each peripheral.

After the configuration is done the tool generates code, header files and application files with the adequate syntax and simultaneously generates a set of functions that the end user can employ for the development of the main layer of the firmware.

The Table A.1 offers the set of functions and the employment of them on the main code of the program.

Table A.1 Autogenerated functions employed

Peripheral	Function	Specific use
OC	DRV_OCx_Start ()	Initiates the function of the PWM on the respective driver pin.
		Adjust the Duty Cycle of the respective PWM, value in bits related to the Timer 2 period.
Timer	DRV_TMRx_Start ()	Initiates the function and counting of the respective timer.
	DRV_TMRx_CounterValueGet ()	Gets the counting value of the respective timer. Used to generate delays.
	DRV_TMRx_CounterClear ()	Resets the value of the respective counter. Used to restart the timer count after the FSM cycle ends.
USART	DRV_USARTx_ReadByte ()	Reads the byte found on the top of the respective serial receiving buffer. Used for handshakes and read the information on the peripherals.
	DRV_USARTx_WriteByte ()	Writes the value of the byte found on the top of the respective serial transmitting buffer. Used for handshakes.
	DRV_USARTx_ReceiverBufferIs Empty ()	Delivers a true value if no data is available on the receiving buffer of the respective serial buffer. Used to know if there is data to read on the peripherals.
	DRV_USARTx_TransmitterBufferIs Full ()	Delivers a true value if any data is available on the transmitting buffer of the respective serial buffer. Used to avoid overlap on the signals for the handshake.

Appendix B

Changes on the Manufacturing Process

The manufacturing suffered from small variations. The first one is the change of the placement of the vias for the use of the ProConduct plaster to connect the Top and the Bottom layers. The second change is the use of the soldering iron, welding plaster, heat gun and solder reel for the placement of the components on the PCB of the drone.

Copper Roughing Process

Using the CircuitPro specialized software for the manufacture using the LPKF ProtoMat S63, the Gerber files are imported to the manufacture CAM environment. The first step is to place the fiducials (Figure B.1), this to allow the machine to locate the position of the board.

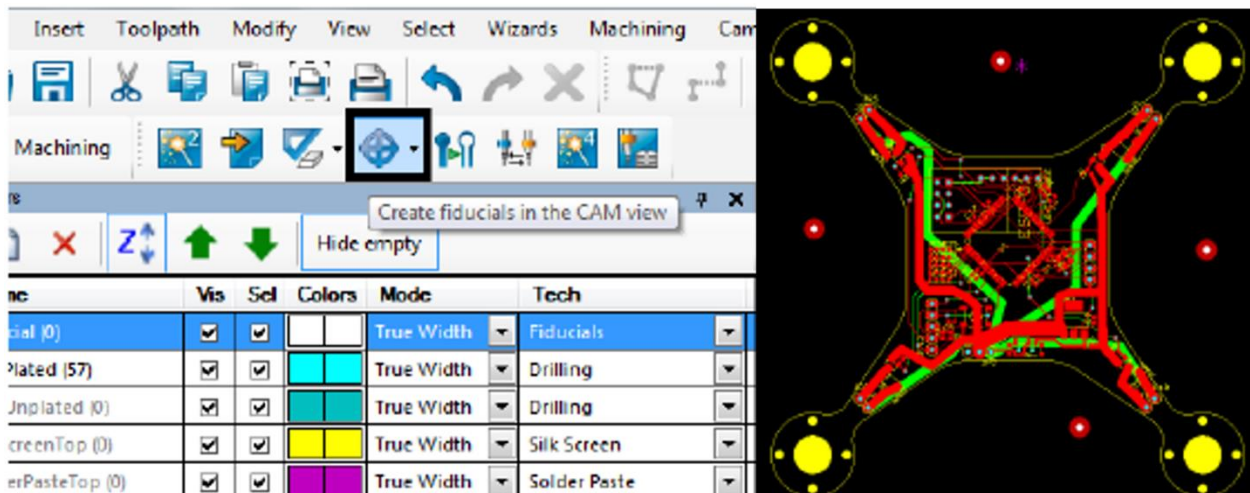


Figure B.1 Fiducials Placement

After this, the selection of the tools available must be carried. The tools are selected on the software and must be placed manually in the designed slot of the Tool Magazine.

The process continues with the selection of the Insulation Method and Contour Routing (Figure B.2). This is done in the Technology Dialog window, where the selection of the tools available on the magazine is done for each process previously mentioned.

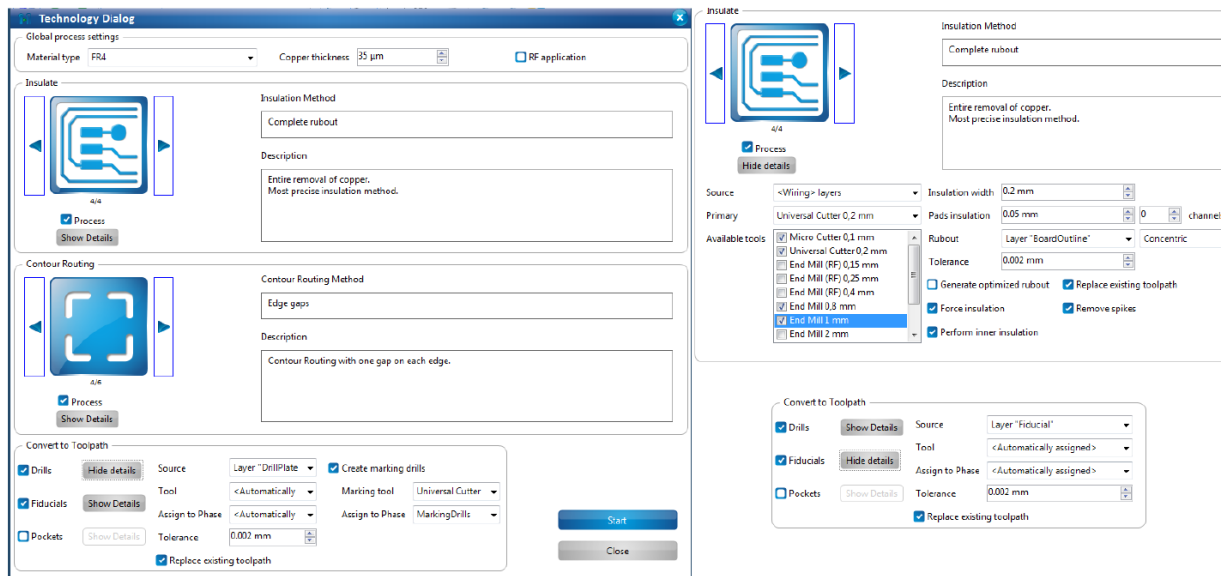


Figure B.2 Selection of the Insulation and Contour Routing

The next step is the use of the Board Production Wizard (Figure B.3), where the system guides step by step the procedure to place the board on the machine, select the quantity of PCBs to be produced (Figure B.4), and feedbacks to the user the phase of the process and the remaining time, until the process is finished. This whole set-up and production phase can take up to 2 [h].

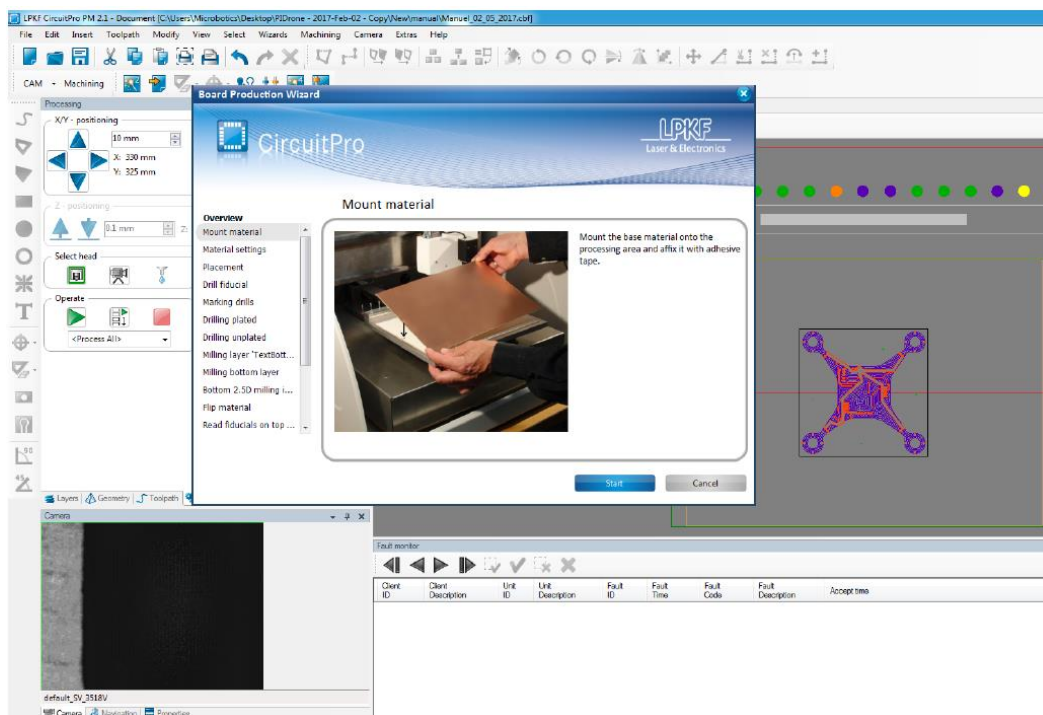


Figure B.3 Board Production Wizard starting page

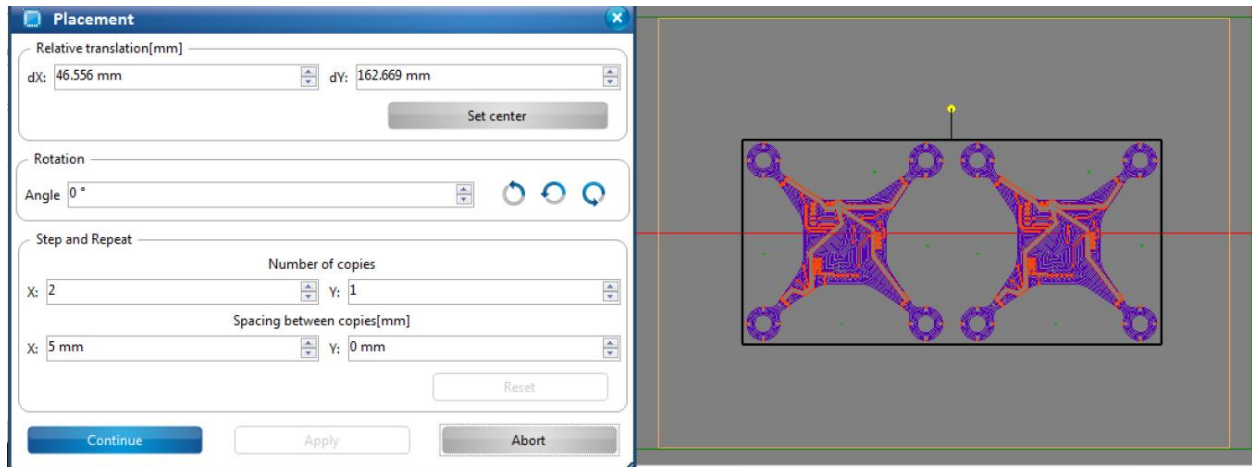


Figure B.4 Quantity of boards to be produced.

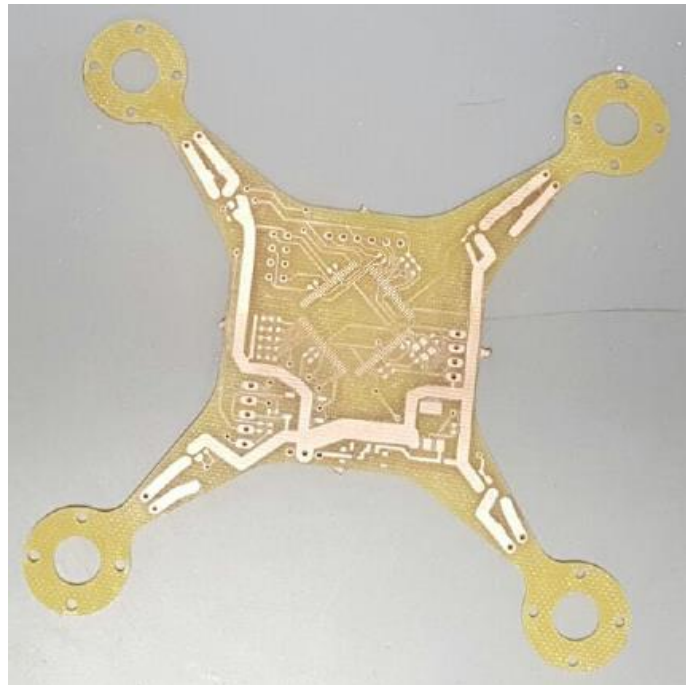


Figure B.5 Finished product

ProConduct Process.

The process of the ProConduct is done after the process of roughing of copper from the board is finished. Then the plaster is placed on the holes indicated on the Figure B.6 using a syringe and avoiding placing an excess to avoid the creation of a short on the board. The Figure B.6 show how it is placed.



Figure B.6 Placement of the ProConduct

After all the holes are filled then the plaster is cured receiving a heat treatment as in the Figure B.7. This helps to dry and solidify the plaster. The process is done in an oven for over 40 [min].



Figure B.7 Heat treatment of the ProConduct

Placement of the components

The process of placing the components was changed since the stencil required needed to be designed and manufactured due the updates on the drone design.

The new process consisted on the placement of flux on all the pads where the components were going to be mounted and then place the solder from the reel using the soldering iron as can be seen on the Figure B.8. The final product is a board with soldering on the appropriate places ready to be welded as shown on Figure B.9

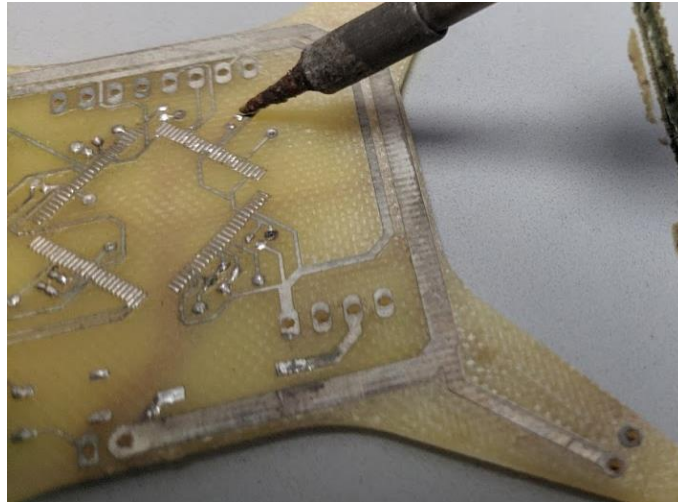


Figure B.8 Placement of the soldering

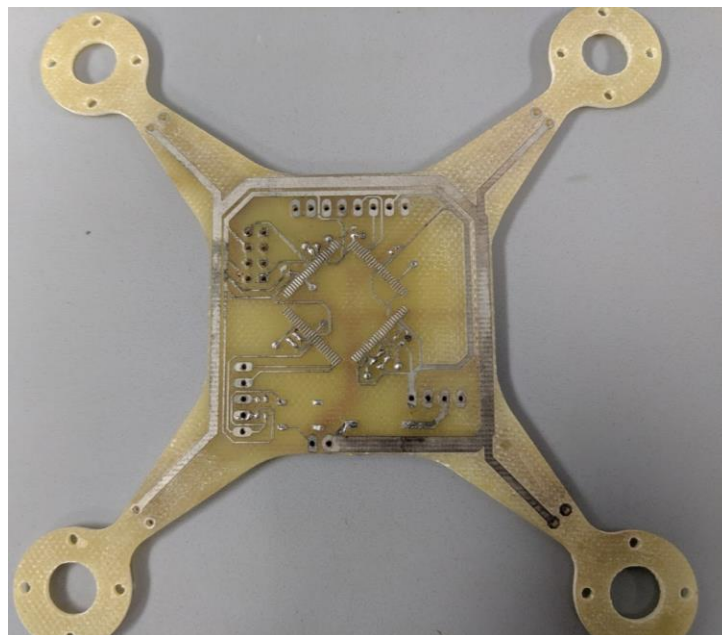


Figure B.9 Plate with all the component pads with solder

To assemble the board and have a final prototype ready for experiments the use of a heat gun is necessary along with a pair of tweezers. The heat gun helps to make the solder liquid again, allowing the placement of the component using the tweezers and fixing it to the board as show in the Figure B.10. It is important to note that the process needs to be

fast to avoid the damage of the tined path, since the heat can cause the copper to get burned and lift from the board, ruining the prototype. Doing the assembly properly takes up to 2 [h] and 30 [min]

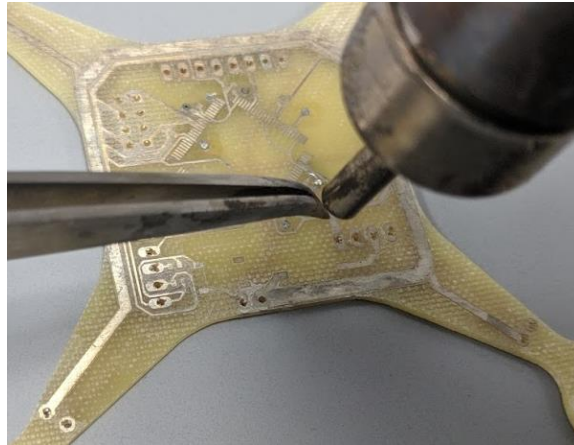


Figure B.10 Placement of the components

Appendix C

Cost of the Components.

The objective of this project is to deliver a prototype of a Quad-MAV made with low cost components. The Table C.1 delivers all the unitary cost of each component, the quantity used on the drone and the total cost of the drone. The quantities are in USD dollars.

Table C.1 Cost of the components of the drone

Component	Details	Quantity per MAV	Unitary Price	Total per MAV
Capacitors	220 [μ F]	4	\$ 0.85	\$ 3.40
	4.7 [μ F]	10	\$ 0.03	\$ 0.28
	1 [μ F]	1	\$ 0.01	\$ 0.01
	0.1 [μ F]	1	\$ 0.17	\$ 0.17
Resistors	10 [k Ω]	9	\$ 0.03	\$ 0.23
	2.4 [k Ω]	2	\$ 0.01	\$ 0.02
	1 [k Ω]	1	\$ 0.07	\$ 0.07
LEDs	Blue	2	\$ 0.33	\$ 0.66
	Green	2	\$ 0.34	\$ 0.69
Mosfet	FDN339AN	4	\$ 0.19	\$ 0.77
Schottky	DB2430700L	4	\$ 0.59	\$ 2.36
μ C	PIC32MZ2048ECM100	1	\$ 17.77	\$ 17.77
WiFi	ESP8266	1	\$ 6.95	\$ 6.95
IMU	Razor 9 DOF	1	\$ 34.95	\$ 34.95
Barometer	MPL115A2	1	\$ 14.95	\$ 14.95
Battery	Lipo, 3.7 [V], 750 [mAh], 60cc	1	\$ 7.70	\$ 7.70
Regulador	3.3 [V] U1V11F3	1	\$ 5.00	\$ 5.00
Step-Up/Down	5 [V] U1V11F5	1	\$ 5.00	\$ 5.00
Botón	Push-Button	1	\$ 0.68	\$ 0.68
Pin Header	Macho	2.3	\$ 0.18	\$ 0.41
Switch	Switch deslizante	1	\$ 1.82	\$ 1.82
Propellers	Parrot Rolling Spider Propeller	4	\$ 13.90	\$ 1.99
Motor	3.7 [V] 66000 [RPM] Coreless	4	\$ 4.40	\$ 3.52
Total Cost				\$ 109.38

Appendix D

Motor Force Measurement.

This appendix contains the graphs obtained from the measurement of the force delivered by the motors on the experiment of the subchapter 4.3.1.

This data from the Table D.1 is used to verify the linear relation between the duty cycle delivered to the MOSFET by the MCU, and the velocity of the motors. The Figures D.1 to D.4 show the difference between the estimated behavior of each motor and the real velocity as a response to a certain value of Duty Cycle

Table D.1 Measurement obtained from the motors

Duty Cycle [%]	Motor Measurement [g]			
	Motor 1	Motor 2	Motor 3	Motor 4
10	0.5	0.6	0.7	0.7
20	2.7	2.8	2.7	3
30	5	5.2	5	5.4
40	7.4	7.3	7.5	7.8
50	9.9	9.5	10	10.1
60	12.3	12	12.1	12.2
70	14.6	14.3	14.2	14.5
80	16.7	16.5	16.9	17.1
90	19	18.7	19.1	19.5
100	21.3	21	21.3	21.8

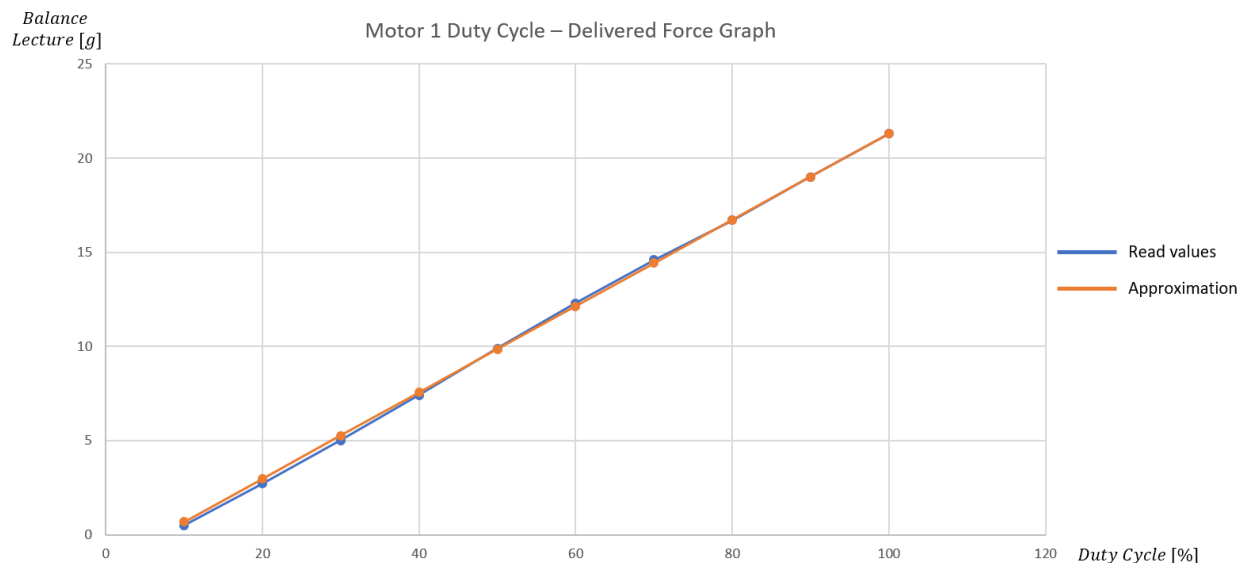


Figure D.1 Graph showing the real and ideal behavior of the Motor 1

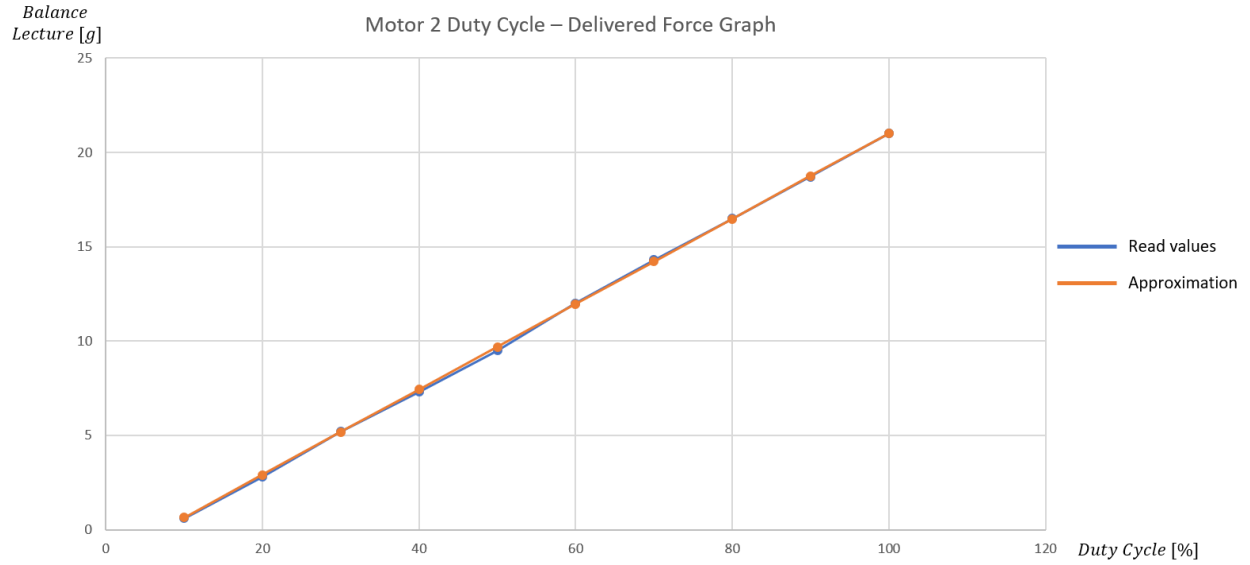


Figure D.2 Graph showing the real and ideal behavior of the Motor 2

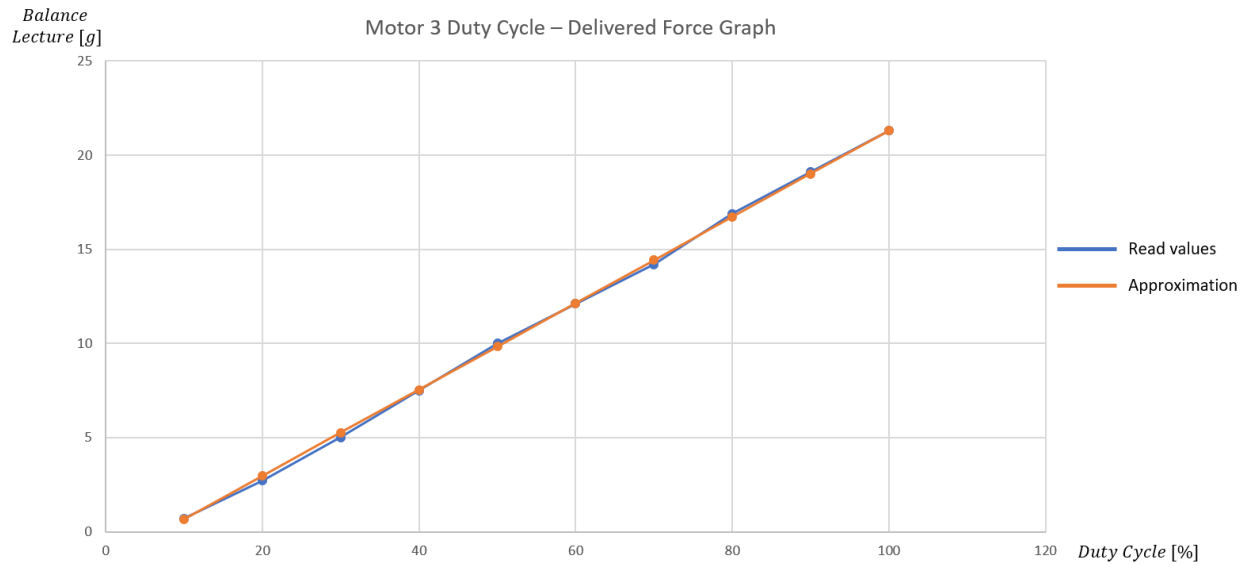


Figure D.3 Graph showing the real and ideal behavior of the Motor 3

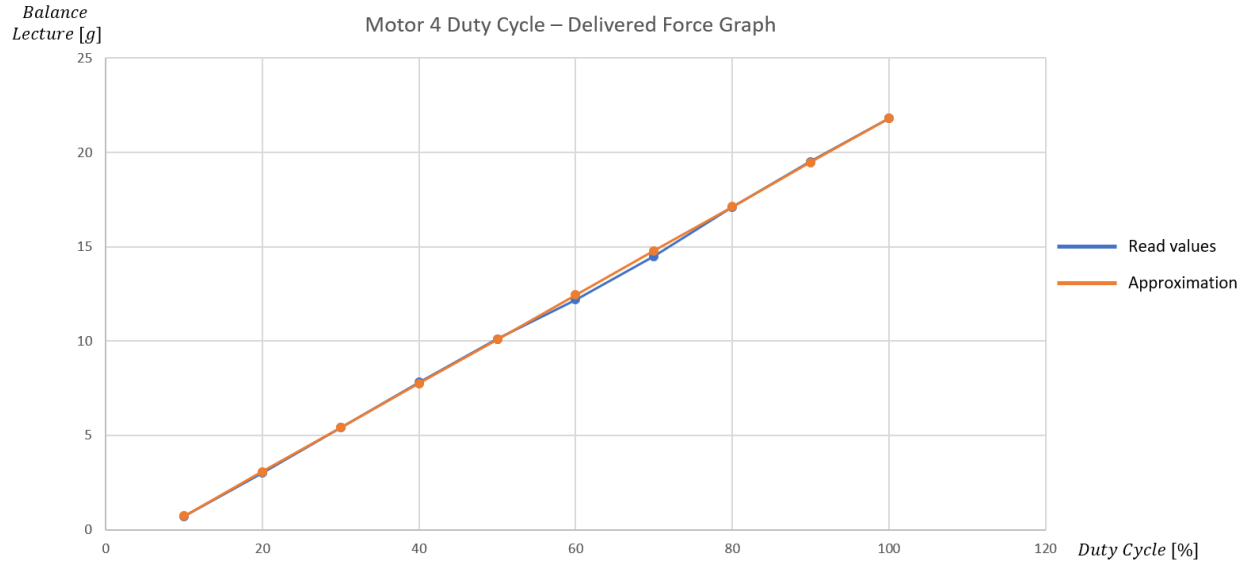


Figure D.4 Graph showing the real and ideal behavior of the Motor 4

Bibliography

- [1] G. Cai, J. Dias, and L. Seneviratne, "A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends," *Unmanned Syst.*, vol. 02, no. 02, pp. 175–199, 2014.
- [2] J. F. Keane and S. S. Carr, "A Brief History of Early Unmanned Aircraft," *John Hopkins APL Tech. Dig.*, vol. 32, no. 3, pp. 558–571, 2013.
- [3] E. A. Espinoza, "Electronic Development of a Quadrotor Micro Aerial Vehicle," College of Sciences and Engineering, M.S.c thesis, ITESM, Monterrey, N.L., 2013.
- [4] E. Capello, A. Scola, G. Guglieri, and F. Quagliotti, "Mini Quadrotor UAV: Design and Experiment," *J. Aerosp. Eng.*, vol. 25, no. 4, pp. 559–573, 2012.
- [5] S. K. Phang, K. Li, K. H. Yu, B. M. Chen, and T. H. Lee, "Systematic Design and Implementation of a Micro Unmanned Quadrotor System," *Unmanned Syst.*, vol. 02, no. 02, pp. 121–141, 2014.
- [6] C. Lehnert and P. Corke, "Design and implementation of an open source micro quadrotor," *Australas. Conf. Robot. Autom. ACRA*, Sidney, Australia, 2-4 December, pp. 2–4, 2013.
- [7] R. Niemiec and F. Gandhi, "A comparison between quadrotor flight configurations," *42nd Eur. Rotorcr. Forum 2016*, Lille, France, 5-8 September, vol. 1, pp. 186–197, 2016.
- [8] B. Siciliano and O. Khatib, *Springer handbook of robotics*, 2nd ed. Springer-Verlag New York Inc., 2013.
- [9] B. Y. N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "Experimental Evaluation of Multirobot Aerial Control Algorithms," *IEEE Robot. Autom. Mag.*, September, pp. 56–65, 2010.
- [10] H. Castañeda, J. Rodriguez, and J. L. Gordillo, "Continuous and smooth differentiator based on adaptive sliding mode control for a quad-rotor MAV," *Asian J. Control*, pp. 1–12, 2019., Doi: [10.1002/asjc.2249](https://doi.org/10.1002/asjc.2249).
- [11] H. Castaneda, L. A. Cantu, A. Leal, and J. L. Gordillo, "Guidelines for propulsion system design and implementation in a quadrotor MAV," in *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, Miami, FL., June, pp. 1302–1308, 2017.
- [12] Microchip, "PIC32 Family Reference Manual - Section 16. Output Compare," p. 42, 2011, <http://ww1.microchip.com/downloads/en/devicedoc/61111e.pdf>.

[13] Microchip, "PIC32MZ Embedded Connectivity (EC) Family Data Sheet," 2015, <http://ww1.microchip.com/downloads/en/DeviceDoc/60001191G.pdf>.

Curriculum Vitae

Pedro Antonio Guerrero Cardoso was born in Ciudad Juárez, Chihuahua, México on March 21, 1992. He earned the Bachelor in Mechatronics Engineering from the *Instituto Tecnológico y de Estudios Superiores de Monterrey*, Chihuahua Campus on June 2015. The specializations he focused during his bachelor studies were Embedded Systems and Robotic Systems.

He worked for *KIA Motors Mexico* from August 2015 to June 2016, where he performed the activities as Parts Development Specialist Jr.

From June 2016 to December 2016 he worked as temporary technician for LASER Guided Vehicles for *Siti B&T*.

He was accepted in the graduate program in Master in Sciences with Specialty in Manufacturing Systems at *Instituto Tecnológico y de Estudios Superiores de Monterrey*, Monterrey Campus, on November 2016, starting the graduate studies on January 2017, ending the program studies on December 2018.

On February 2019 he was recruited by *John Deere* company as Embedded Software Engineer, acting as a software developer and software tester for the assigned team.