POLITECNICO DI TORINO

Master of Science Degree in ELECTRONIC ENGINEERING

MASTER THESIS

Optimization of an Ultralight Autonomous Drone for Service Robotics

Supervisor:
prof. Marcello CHIABERGE
Thesis advisor:
dott. ing. Gianluca DARA

Candidate:
Simone SILVESTRO
s243872

ACADEMIC YEAR 2018 - 2019

Abstract

Drone industry is constantly growing and evolving in time. Federal Aviation Administration Aerospace Forecast predicts that drone market volume will be about three times the actual one in 2023. This because UAVs are becoming more and more fundamental in the most various application like agriculture, emergency response, urban planning and maintenance, entertainment, security and so many others. For all these tasks, in the near future a large number of drones will fly over our heads. From here the necessity to build safe, lightweight and autonomous UAV for the drones colonization of the very lower part of the Earth atmosphere. Furthermore EASA fixed to 250 grams the limit of the C0 open UAS, category with quite few restrictions to fly.

PIC4SeR (PoliTO Interdepartmental Centre for Service Robotics) wants to keep up with this incoming demand and enter in this market, deciding to invest time and resources on the optimization of an ultralight autonomous drone, with the future intention of full customization with the needs of the end-user for different and specific cases such as service robotics, smart city search and rescue and precision agriculture that are the four fields of action of the interdepartmental centre where this thesis took place.

With the base of the first flying prototype and a deep research on the State Of Art literature on drones, in particular lightweight and autonomous ones, the work done for this thesis was to optimize the drone with special attention to hardware and weight, changing the usual construction method from carbon fiber frame to a newborn design using Printed Circuit Board and 3D printed plastic, and finally to make it intelligent and usable by non-expert users with basic knowledge. The improvements made and overall stability of the new prototype are satisfactory, a solid point from which to develop many ultralight drone based technologies.

Contents

1 Introduction **1**
- 1.1 Objective of the thesis . . . 1
- 1.2 Organization of the thesis . . . 2

2 SOA **3**
- 2.1 Autonomous drones . . . 3
- 2.2 Ultralight drones . . . 5
- 2.3 Drones regulation . . . 8
- 2.4 Autopilots . . . 9

3 Previous build **11**
- 3.1 Parts List . . . 11
- 3.2 Frame . . . 13

4 Current build **15**
- 4.1 Parts List . . . 15
- 4.2 Frame . . . 16
  - 4.2.1 PCB main frame . . . 16
    - Design . . . 16
  - 4.2.2 Mechanical simulations . . . 21
  - 4.2.3 3D printed main frame support and legs . . . 23
  - 4.2.4 3D printed secondary frame . . . 23
- 4.3 Final assembly . . . 24

5 Testing **25**
- 5.1 SITL . . . 25
- 5.2 HITL . . . 27
  - 5.2.1 Camera trigger . . . 28
  - 5.2.2 Mission Plan . . . 29
- 5.3 Flight Tests . . . 32
- 5.4 Flight Analysis . . . 32
  - 5.4.1 PX4 flight analysis . . . 33
    - Vibrations . . . 33
  - 5.4.2 Power . . . 34

5.4.3 Magnetic field 35
5.4.4 GPS 36
5.4.5 Matlab 38
5.5 Image post processing 39
5.5.1 Sharp image extraction 40
5.5.2 3D reconstruction 40

6 Troubleshooting 42
6.1 ESC 42
6.2 Sensors 46
6.3 GPS 46

7 Comparison between builds 51
7.1 Frame 51
7.1.1 Material 51
7.1.2 Vibration analysis 52
7.2 Dimensions 53
7.3 Weight 54
7.4 Time Of Flight 56
7.5 Image capture 56

8 Conclusion and future development 58

Bibliography 60

Glossary

ABS Acrylonitrile Butadiene Styrene.

CAD Computer-Aided Drafting.

DOP Diluition Of Position.

EASA European Aviation Safety Agency.
EKF Extended Kalman Filter.
ENAC Ente Nazionale Aviazione Civile.
ESC Electronic Speed Control.

FC Flight Controller.
FOV Field Of View.
FR-4 A composite material made of fiberglass cloth glued with epoxy resin, the name came from a NEMA designation.

Gazebo A powerful environment for 3D simulations, PX4 Autopilot and ROS compatible.
Geo-fence Virtual perimeter used to delimit the flight area, usually a circumference around a point or a simple polygonal shape.
GNSS Global Navigation Satellite Systems.
GPS Global Positioning System.

HITL Hardware In The Loop.

IC Integrated Circuit.
IMU Inertial Measurement Unit.

KiCad EDA An Open Source and Cross Platform Electronics Design Automation Suite, a completely integrated environment for both schematic and PCB layout design..

MAMSL Metres Above Mean Sea Level.
MAVROS MAVlink on ROS.
MTOW Maximum takeoff weight.

Ninja A small build system focused on speed.
NSH NuttSHell.

NuttX A RTOS compatible with microconstoller from 8 to 32 bit, like PixRacer and PixHawk.

PCB Printed Circuit Board.
PLA PolyLactic Acid.
PWM Pulse Width Modulation.
PX4 Autopilot A professional autopilot flight stack for industries and academia, suitable for many kind of vehicles from drones to racing to ground vehicle and even submersibles.

QGC QGroundControl.

Rally Point Point different from the home position where vehicle can land in case of a failsafe Return To Land, generally in strategic positions along the mission path.
ROS Robot Operating System.
RTK Real Time Kinematic.
RTL Return To Land.
RTOS Real-Time Operating System.

SITL Software In The Loop.
SLAM Simultaneous Localization and Mapping.
SOA State Of Art.

TOF Time Of Flight.
TTFF Time To First Fix.

UAS Unmanned Aerial System.
UAV Unmanned Aerial Vehicle.
UGV Unmanned Ground Vehicle.
UTM Universal Transverse Mercator.
UWB Ultra Wide Band.

Waypoint Intermediate point of the path plan.

List of Figures

2.1 Small drones 6
2.2 Crazyflie project by University of Utah 8

3.1 Parts list previous built 13
3.2 Previous built frame 14

4.1 Parts changed in current built 16
4.2 Frame Shapes 17
4.3 Unibody or Separate arms 17
4.4 Previous middle frame 18
4.5 KiCad frame design 19
4.6 SolidWorks static simulations 22
4.7 Frame support tests 23
4.8 Upper frames 24
4.9 Final Assembly 24

5.1 SITL simulation 26
5.2 HITL simulation overview 27
5.3 Vibration-related graphs 34
5.4 Power plot 35
5.5 Thrust and magnetic Field 36
5.6 Diluition Of Position[11] 36
5.7 GPS-related graphs 37
5.8 Flight path analysis 39
5.9 Hold flight analysis 39
5.10 Reconstruction softwares 41

6.1 ESC three phases under oscilloscope 42
6.2 BLHeli Suite Interface 43
6.3 BLHeli ATMEL/SILABS selection 44
6.4 QGC Generic Quadrotor X frame configuration 44
6.5 BLHeli final configuration 45
6.6 Holes in GNSS position 47
6.7 previous GPS u-center visualization 49

6.8 tiny GPS u-center visualization 49

7.1 Vibration comparison 53
7.2 Dimensions comparison 53
7.3 Propeller spacing 54
7.4 eCalc TOF evaluation 56
7.5 700TVL analog vs 1080p digital image capture 57

Chapter 1

Introduction

1.1 Objective of the thesis

in the last couple years many research discipline and working fields have massively requested the usage of flying vehicles to monitor buildings, cities, crops from behind and also new technological work areas such as precision agriculture started growing. From here the market has grown, number of multicopters in the air increased and it will multiply rapidly in next years. the easy way to produce drones capable of the most various task is to overshoot in builds hardware, with large MTOW of at least 1 kg load, using heavy materials not optimized in any aspect except stability, but this imply to let massive drones fly over the ground also where smaller and less dangerous drones will be a better solution for safety. The need for smaller and lighter drones is immediately evident.

The goal of the thesis is the optimization of a first ultralight autonomous prototype. This available starting build was dimensioned from power management to motors and propellers, then built and flight only in manual mode using the radio controller by an operator. The work starts from here, initially with the study and the design of a newborn frame made with a totally new approach using PCB and 3D printing, and in parallel the autonomous mission flights have been

1.2 Organization of the thesis

The thesis is divided into eight chapters, here summarized for a for a quick understanding of the whole structure:

Chapter 1: very short introduction, focusing the thesis goal and valence.

Chapter 2: State Of Art presenting actual market and technologies, with a specific focus on autonomous and ultra lightweight drones, with a final outline on drones regulation.

Chapter 3: an overview of the first prototype use as the base to be optimized, part list and a depth view on the carbon fiber frame.

Chapter 4: detailed description of the newborn build, from part list to an in-depth view of the design process for PCB frame and 3D printed supports.

Chapter 5: testing phase is here explained in its various declination, from simulations as SITL and HITL to flight tests on facility and flight field to test the quadcopter piloted in manual first and then in autonomous missions. Important aspect of the testing is the flight analysis, method to prove the quality of the frame, actuation, and to find and fix all kind of errors involving sensors and control in general. The last part of the testing chapter is dedicated to a base of image post-processing done to verify the suitability of the on board camera.

Chapter 6: some of the major problems encountered and the troubleshooting process followed to fix them are here presented, Electronic Speed Control broken during flight, IMU and GNSS module incorrect behavior in software.

Chapter 7: here a comparison between the old and the new prototype if treated, divided in the main subsection which are frame, dimensions, weight, Time Of Flight and image capture.

Chapter 8: the thesis end with some conclusion of the work done and ideas for future possible development.

This said, welcome into my thesis.

Chapter 2

SOA

State Of Art is the was the first big step of the thesis, necessary to have a closer look into the field of drones and autonomous driving that were two new concepts for me. On this chapter I will go through some of the many papers and research that I made in first months, to see what other universities or researcher have made in the world, to validate the purpose of my work and then to search and select the best components for the build. For this reason, the subsections are divided into fields of research, starting from Autonomous multicopters, then a view on ultra lightweight drones to the union of the two categories which is the goal of this thesis.

2.1 Autonomous drones

For autonomous driving in Unmanned Aerial Vehicle sector is intended the capability of the system to know the position and in certain case also sensing the environment in order to move with little or completely no pilot inputs. This means that the system have to rely on onboard sensors of various genre to compute the absolute or relative position in space and then to take decision on the path to follow to reach a certain goal or waypoint.
Many sensors can be used to accomplish the task, for example:

- Distance sensors: could be laser or ultrasonic, a light or sound impulse is send from the trasmitting apparatus to the outside of the build in the desired direction, and the Time Of Flight of the signal bouncing back on the eventual barrier surface determines the distance from the object to be avoided.

- LIDAR: acronym of Light Detection and Ranging o Laser Imaging Detection and Ranging, is a planar laser distance sensor, a prism rotating 360° by means of a DC motor takes the distance of objects on the LIDAR plane on N points on the complete rotations, where the number of samples is determined by manufacturer and then by the software

- Bluetooth: in indoor applications some kind of low power bluetooth module called beacon are used for relative positioning, these methodology need anchors with known positions to perform the location estimation.
- Ultra Wide Band: a recent wireless technology used similarly to the bluetooth one but with better precision, in the next years this protocol will came out also builtin with consumer smartphones and for these reason more and more applications will appear.
- Global Navigation Satellite Systems: is the standard in outdoor applications, used worldwide with five constellations of satellites by USA (GPS), Europe(GALILEO), China (BEIDU), Japan (QZSS) and Russia(GLONASS), standard receivers could measure the position with meters-precision but using particular RTK antennas the uncertainty of the measure drop down to the cm-mm precision.
- Camera systems: real time image processing can be used to track the vehicle movements with SLAM techniques and to detect objects to avoid, the most various type of cameras can be used from standard RGB ones, to depth cameras or stereocameras, the requirement for this type of environment analysis is a board with good computational power, at least a microprocessor with some GBs of ram and eventually some graphic accelerator specially if some kind of neural networks are used.

All of these sensors have already been used by many researcher around the globe to accomplish the most various tasks, and the literature confirm that. I will now present some of the most relevant papers and projects of recent autonomous applications found in my investigation.

First project, written by researchers from Korea Advanced Institute of Science & Technology, is entitled "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning"[1] available on IEEE website, is a research application for automatic gate detection and guidance using neural networks. The goal of this work was to implement some open source, cheap and off-the-shelf solution for indoor managing of multirotors. The hearth of the project is an INVIDIA Jetson TX2, prototyping embedded computer with 32 GB ROM and 8GB RAM onboard, with a very good computational power suitable to run on it complex neural networks to accomplish the guidance task. Then the main sensor used is a camera, then the image stream passes in a Convolutional Neural Network for the gate detection, computing the Line Of SIght from the drone to the center of the gate itself, and adjusting the drone velocities to reach and pass the portal itself. These algorithms were based on You Only Look Once model , tested and validated, the code is also available on a Github repository that can be found at the bottom of the paper.
This work is powerful with 90% detection rate at 15fps and the access to the firmware is interesting, but the Jetson TX2 itself weights 85 grams, more than 1/3 of the total

MTOW of the 250 available, not suitable for the application needed.

Another paper found on the International Research Journal of Engineering and Technology (IRJET), entitled "Development of an Autonomous Drone for Surveillance Application"[2] from MIT Mysore researchers, propose a solution using a Raspberry pi with it's camera module, PixHawk autopilot, and OpenCV libraries. The autopilot is set to sense the attitude of the quadcopter using onboard IMUs and compass, read GNSS coordinates and then interface with the Raspberry. This second embedded board is used instead to read stream from the Pi camera and using Open Computer Vision (OpenCV) python libraries to detect targets with the camera to move the setpoint proportionally to the position inframe of the desired shape/object to reach, for example a landing spot or a target to follow. The design include also GPS since in outdoor surveillance applications the latitude and longitude position of the aircraft is a must-have data to work on, merged with visual algorithm make the system more robust and precise.
This project is similar to the one that we want implement, but again the weight is not taken as project specification, Raspberry is 42 grams and PixHawk 38 grams, same as the previous case. To be taken into account the possibility of video stream from the drone itself to a control station and image processing off-board, can be a solution to reduce the load of a companion PC on-board if some kind of visual handling is needed.

As can be clearly noticed, the problem of this type of approaches is the total mass of the complete UAV, completely out of the needed range of the "ultralight" category, which is a restrictive requirement.

2.2 Ultralight drones

In this section, a little introduction on ultra lightweight drones available in commercial solutions and also in research projects available in the literature. Small and lightweight drones are a category approached in total different way with respect to other with more payload, usually constituted of just the microcontroller and radio command transmitting antennas, DC motors and a plastic frame to take all the pieces together, but most of the times they are just small toys good to fly indoor in manual mode with a very basic control in stabilized mode. An evolution of these drones are called Whoop, now being more and more appreciated into racing/cinematic drone enthusiast for the ability to flight in very small spaces like houses without the risk of damage something thanks to the blades protections. This type of drones have a built-in analog camera with video transmission module able to stream live video to goggles or analog screens. Also in this case, the controller is basic and cannot be used for any kind of autonomous missions due to the lack of GPS sensors and the flight time is around 3-4 min. The category just over this one is called Cinewhoop, they can carry also an action cam for high quality footages but the weight increase around 500 grams only with the drone parts, double with respect to the project goal. One possibility could be take one of these drones, change the needed parts, a camera recorder, a GPS module, but this cannot be accomplish because they run on proprietary flight stack or implements only the actuation using radio commands with the only possibility to calibrate stuffs and setup some modes on the controller, but not to add peripherals such as GNSS module. Major commercial drones manufacturers are moving in the direction of ultralight drones with quality cameras and excellent attitude control thanks to the very high levels of technology both in hardware and software development, for example a well known drone producer just release a mini version of other well established drones for video shooting with 2.7k 3 axes stabilized camera, front and bottom distance sensing, GPS and a bunch of automatized flight modes for specific shootings, with an insane flight time of around 25 min, all of these characteristics in 249 grams. The counter of this wonderful specs is that the software is completely closed, and there's no possibility to interface with the drone outside of the tools give by the manufacturer itself, certain not suitaable in fields like research.

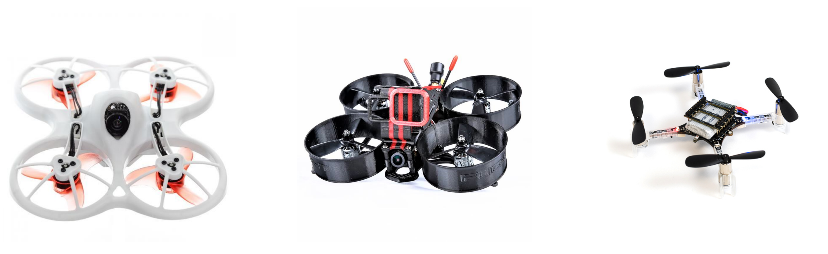

(A) Whoop (B) Cinewhoop (C) Crazyflie

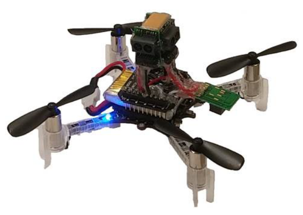
FIGURE 2.1: Small drones

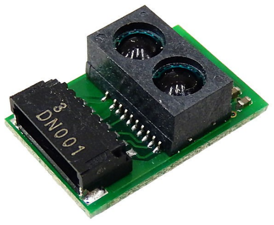
Exiting from the commercial solutions, a research on open source, papers and projects by universities and research centers around the world, some other interesting projects have been developed. The first big development kit that stands out is called Crazyflie, a very small, open source, ultra lightweight quadcopter by Bitcraze, a startup born in 2013 as result of a crowdfunding campaign for the very first prototype of this interesting open source platform. This tiny drone includes:

- weights only 27 grams.
- 2.4GHz radio communication.
- Bluetooth Low Energy.
- micro USB connection for battery charging.
- 9 axes IMU with gyroscope, accelerometer, compass.
- High precision pressure sensor.

- Expansion decks to expand the drone capabilities, from LED ring to wireless charging system, to optical flow sensors.

As another pro of this micro quadcopter, a lot of organizations used these technology, for example Microsoft, NASA, MIT or universities of the caliber of ETH Zurich.
For all these reasons, the project was strongly taken into consideration in a preliminary phase, but contextualizing it in the field of application the PIC4SeR need that are principally outdoors, the lack of a GNSS module is kind of an issue, specially without some powerful computational hardware to compute correct position for example trough camera, and also the weight is too low, making the quadcopter very susceptible to environmental factors and in particular case wind. In fact the applications of this drone are indoor, and for the localization most of the time developers rely on a Vicon tracking systems composed of many cameras positioned in the room with calibrated positions and a bunch of reflectors on the tiny multicopters to compute the drone position relative to anchors, and from this measure the loop is closed from camera images to the control of the drones positions, in some cases also with swarm configurations, but the control is delocalized on companion computers off the fleet of quadcopters, so again not a good solution when the flight distance is more than couple meters indoor.
In fact, the best solution for outdoor applications is a vehicle still ultralight but closer as possible to the limit of the 250 grams in order to make it the most possible robust to the wind wich will in any case influence the stability of the drone, but at least is important to minimize the external factors influence on the drone controllability.
A very interesting application using this tiny drone is entitled "Autonomous Drone"[3] by a group of Electronic and Informatics researcher of the University of Utah, they developed a modded version of the Crazyflie with come onboard sensors. The goal of their work was to build a fully indoor autonomous microcopter able to avoid walls and land on a red marker by using image recognition. The sensors to avoid obstacles are small infrared module by SHARP capable of sensing from 4 to 50cm, this limited range is a tradeoff between the weight and the sensors measuring distance, but are ok if used just for the avoidance maneuver and not to real time mapping the room. Looking at some graphs in the paper itself, the role of the ultrasonic sensors is the trigger of an avoidance maneuver acting on roll/pitch angle to move the drone away from the detected barrier. Then, for input/output problems, the camera detection was moved to the companion PC, detecting a selected target and sending the land command to the tiny quadcopter when the landing spot is detected with a certain grade of accuracy.
The considerations that emerged from this researcher activity are a very short flight time and battery levels instability, about 30 seconds of flights before landing and changing batteries, drone tent to gain speed too fast and the controllability is weak, glitches on infrared and gyroscope sensors, and from all of this the general thoughts there is plenty of room to improve this kind of project, but more importantly that if it doesn't already work well in a room, it certainly could not be the solution for outdoor ones.


(A) Crazyflie Mod



(B) Infrared module

FIGURE 2.2: Crazyflie project by University of Utah

Summing up the research carried out, emerges that autonomous solutions with navigation under GNSS are usually over the 250 grams weight limit but still with open software, were ultra lightweignt solutions are generally applied only in indoor application and cannot carry as payload a decent camera. The goal of this thesis to merge the two characteristics is so valuable, taking the best aspect of the two, a good payload capacity for camera and maybe sensors but with total reduction of the MTOW.

2.3 Drones regulation

The European Aviation Safety Agency is the organ in power to legislate in aircraft regulation in Europe, in Italy every drone operator appeals to ENAC, but this one has the same exact rules of the european organization. Last Implementing Regulation of the 24 May 2019, states what is summarized in Table 2.1, wich confirms the C0 class as the only one with no need to have registered pilot to fly the drones. UAV part of this category must in any case be subject to a series of rules, have a maximum velocity of $19\frac{m}{s}$, limit the max height to 120 meters over the takeoff point, be safely controllable in therms of stability and data link, have power supply under 24V DC maximum and have a clear and complete user manual.

UAS Class (MTOW)	Operations	Operator registration
C0 (<250g)	Over involved people, not over crowds	No
C1 (<900g)	Over involved people, not over crowds	Yes
C2 (<4kg)	Safe distance from uninvolved people	Yes

TABLE 2.1: EASA drones categories

However, some new rules will came out in 2020, they can be found in the Unnamed Drones tab in the timeline and state that from June drones less than 250 grams equipped with sensors able to acquire personal data must fly with mandatory registration of the operator/pilot, and that can fly only in areas where is expected that no uninvolved people is overflown by the vehicle. This operation is probably due to the launch on the market of some interesting drones ultralight but with hi-end cameras, to limit incorrect usage by privates, with the drawback of more of more papers to be compiled by technicians, but the light drone still have not to be registered.

2.4 Autopilots

The other important check was the state of the art in therms of available open autopilots, both from software and hardware fields.
From the software point of view, the two open solutions are:

- **Ardupilot**: A common software born in 2009 initially used coupled mainly with some Arduino boards as controller, then upgraded for other platforms based on STM32 microcontroller like the PixHawk family. The suite is also completed by some ground station programs like APM Planner or Mission Planner that can be used to change autopilot settings or to manage plan files, missions and data link. it supports a huge variety of frames and vehicles, from UAV to UGV and even submarines. The License of the Ardupilot stack is the General Public License v3 (GPLv3), it means that every development of the software must then be shared with the entire community, the most open source license available.

- **PX4**: the Aurdupilot alternative, an open professional autopilot complete environment, with a flight stack on the microcontroller based on a real time operating system called NuttX then interfaced with PX4 middleware, communicating with MAVLink protocol (a standard on this type of applications) that can also be interfaced with ROS environment with MAVROS communication node. As for Ardupilot, run on the most various type of aerial and ground vehicle, the companion desktop application is called QGroundController and works for calibrations, configurations, analysis and mission planning. The license in this case is the BSD 3-clause license, still open but the development can also not be redistributed, so is a better solution for applications where there's the possibility to build some new application that could be patented or sell, in the innovation field.

As can be noticed from the little summary above, Ardupilot is the hobbyist solution, where PX4 is more professional and the possibility to close the software is also a good road to take open for future massive improvements, and more important the community is more structured and bigger, with the Dronecode platform and its various designations on Github, Slack and also the PX4 forum where troubleshooting on both hardware and software issues is followed by hobbyist, professionals and also repository maintainers. The code is more complete than Ardupilot, and for this reason also more complex, with more than 2000 total files and about 300k lines of code on the Firmware repository in Github.

Chosen the software, the research passed on the available autopilots board on the market. The alternatives are principally some kind of STM32 microcontroller based solutions, raspberry modified with shells or other with microprocessors usually used in smartphone manufacturing. The last 2 solutions have clearly more computational power, but at the same time the weight is way more than the simple microcontroller one, which is still valuable for autonomous flight thanks to its 32 bit floating point architecture, lack of power only when some kind of image processing is necessary, but perfect for attitude control only under GNSS, IMU and distance sensors.

Name	Hardware based	weight[g]
PixRacer	STM32F	10
mRo x2.1 Rev 2	STM32F	10
PixHawk Mini	STM32F	15.8
Snapdragon Flight	Snapdragon 801	20
BeagleBone Blue	ARM Cortex-A8	36
PixHawk	STM32F	38
Navio2	Raspberry Pi	73
CUAV v5	STM32F	90

TABLE 2.2: Some small PX4 autopilots on the market

In the Table 2.2 a list of some of the most popular PX4 autopilots are reported in weight order from lightest to heaviest. The first positions are occupied by microcontroller boards, since the count of peripherals need to use this kind of integrated circuits is lower than the microprocessor case. Other solutions are more powerful and can also manage camera stream locally, with the tradeoff on the payload. Since the PixRacer solution is one of the lightest ones and already used in previous build, the specification of the project doesn't necessary states some real time camera management, so I decided to stick with the previous decision for this autopilot, also because is the small brother of the PixHawk which is largely used and so supported by a large number of users in the PX4 community.

Chapter 3

Previous build

The constraints for the previous built were the following:

- **Autopilot drone**: the quadcopter have to run an autonomous flight stack able to perform autonomous movements (trajectory management, height maintenance, mission flights) without being constantly piloted by an operator.
- **Open platform**: The complete system have to rely on open standards, supported by a community, constantly up to date with the new technologies available on the market.
- **ROS compatible**: These constraint came with the idea to integrate in the future the drone in a fleet of different robots such as other UAVs or UGVs.
- **Ultralight**: to be classified as an A1C0 category aircraft, the Maximum takeoff weight of the drone must be under 250 grams. This constraint is the key of the whole project, all component must be the lightest ones, without without sacrificing performance when possible.

3.1 Parts List

All components used in previous built with a brief description are here reported:

- **PixRacer Autopilot**: small Flight Controller capable to run complete autopilot stack firmware like PX4 and run complex mission task with precise waypoints and fully autonomous navigation. Equipped with 32-bit STM32F427 microcontroller, MPU-9250 9-axis IMU, ST LIS3MDL magnetometer, MEAS MS5611 barometer, microSD slot, many standard protocols like I2C, SPI, UART, PPM, PWM and others, ESP8266 WiFi module included to exchange data with companion computer, all in just 10.56g of weight.
- **ESP8266**: tiny WiFi module to create an access point on the drone to exchange data with companion computer.

- **Tattu 800mAh 2S LiPo Battery**: 7.4V, 45C rated, the right compromise between weight and a total Time Of Flight around 10 min.
- **ACSP5 Power module**: Up to 42V voltage input (10S LiPo batteries), Up to 80A current sensing with built in INA1x9 High-Side Current Monitor, 17 mm x 17 mm package, JST-GH connector.
- **BLHeli 4-in-1 ESC**: Integrate BusyBee2 controller running Up to 48MHz, FD6288 MOSFET Drivers and Dual N-Channel 30V 12A MOSFET for phase legs. The 4-in-1 solution is common for space saving, without compromise performance in the case of a small build.
- **FrSky X4R Telemetry receiver**: 16 channel S-BUS and 3 channels PWM
- **T-Motor MT-1306-10 3100KV** : tiny and efficient motors, only 11.2 g each, capable of a max thrust of 218 g each with 6020 props (872 g total), more than the double of the total mass of the drone, so enough for this application.
- **T-Motor 6020 carbon fiber propellers** : 2-blades lightweight propellers, better than 3-blades solutions when goal is not thrust but Time Of Flight
- **mRo NEO-M8 Dual compass GPS**: GPS + GLONASS + GALILEO constellations, 5dBi gain, dual magnetometer integrateD (LIS3MDL and IST8310) for a robust compass estimation.
- **Runcam Racer Micro FPV camera**: 4:3 analog camera with fast 6 ms latency, in 5.5 grams.
- **ImmersionRC 5.8GHz Video Tx**: 600mW output power, SMA connector for external antenna, 7 bands.
- **Triumph 5.8GHz FPV Antenna Set**: TBS circular antenna, SMA connector, 11.5 g.

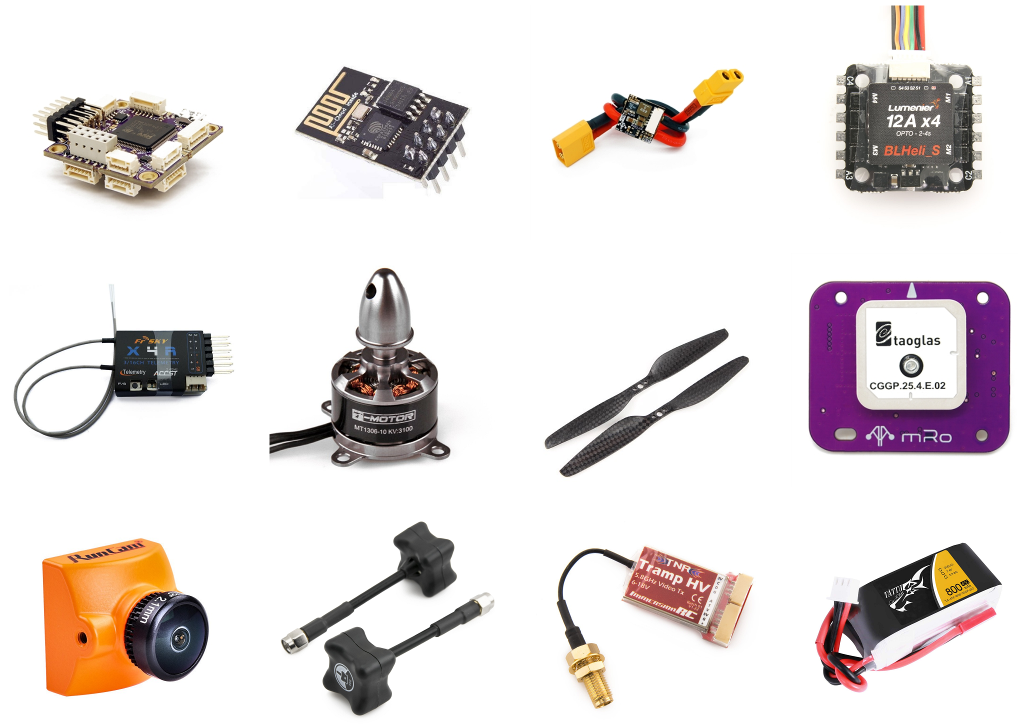


FIGURE 3.1: Parts list previous built

3.2 Frame

The frame of the old prototype was developed in a mechanical master thesis, it consist of a carbon fiber layer cutted with CNC technique, reinforced with 4 flanges perpendicular to main frame to stabilize and better distribute forces along the principal axes. It uses a Wide X configuration with the "X" a little stretched to make frontal space for the camera in the case of frontal video capture (example in structure scan when the lateral walls of a building have to be inspected). Both on the central rectangle and in the arms lot of material have been removed using basic shapes such as rectangles, circles and triangles for weight reduction. For the arms, several section shapes have been taken into account (rectangular, circular hollow, N and T) and the selected one was the T shape, the most efficient in therms of trade-off between transversal forces resistance, total mass per length and ease of production. The final result was a good base for the needed application.

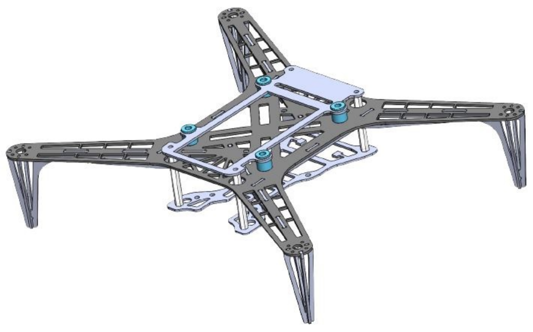


FIGURE 3.2: Previous built frame

From the Figure 3.2 is clear that the configuration chosen was a wide X one, the best configuration in the case of a drone moving in stabilized mode (most of the time with roll/pitch angles small with peaks around 20°) to make frontal room for the camera view. With this arrangement is possible to use camera in different missions, for example survey mode with the position of the camera perpendicular to the ground, but also in scan mode for buildings with the camera pointing the horizon.
The pieces chosen for this first build are ok, specially power management and propulsion system, propellers are made out of carbon fiber so lightweight, and with two-blades which are considered the best to maximise TOF compared with three-blades or more, motors also are a good tradeoff between power and dimensions, other smaller are used only for quad of 50-60 grams max MTOW and larger ones will be too heavy. Autopilot and telemetry transmission system also are ok, some improvements can be done on GNSS module and camera, listed in Chapter 4.

Chapter 4

Current build

The new build is similar in components to the previous one, but introduces a new project paradigm, more integrable thanks to the PCB frame introduced and explained in Subsection 4.2.1 but still with the same modularity of the old one because only off-the shelf components are used.

4.1 Parts List

Most of the components of the new build are the same, and so only listed here, GNSS module and camera changed and are so also explained.
Here the full part list:

- **PixRacer autopilot**
- **ESP8266 WiFi module**
- **Tattu 800mAh 2S LiPo Battery**
- **BLHeli 4-in-1 ESC**
- **FrSky X4R Telemetry receiver**
- **T-Motor MT-1306-10 3100KV**
- **T-Motor 6020 carbon fiber propellers**
- **mRo uGPS ublox SAM M8Q + Compass**: micro GPS, supports GPS and GLONASS constellations with built-in LIS3MDL compass, standard JST-GH connectors PixRacer compatible.
- **Caddx Turtle V2 HD Camera**: 1080p 60fps mp4 High Definition video recording, 155° Field Of View angle on the horizontal axis, microSD card slot up to 64GB, Input power accepted from 4.5 to 20 V, 20x20 cm standard mounting holes, just 12 grams camera plus recording system.

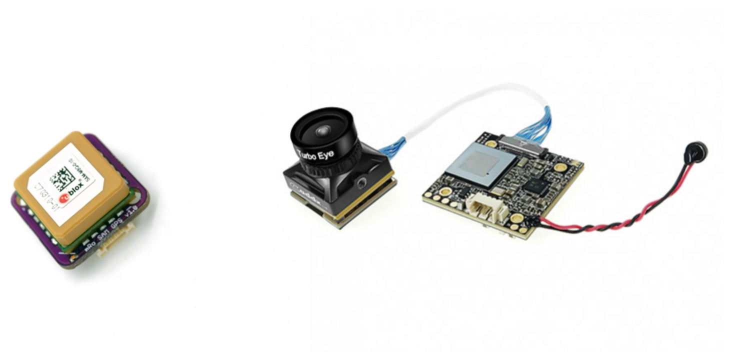


FIGURE 4.1: Parts changed in current built

4.2 Frame

Mechanical support for all the necessary components of the build is the first purpose of these piece of hardware, and historically they have been made out of plastic, carbon fiber or other composite materials, standard among the most part of manufacturers. The new frame came from a totally different approach. In drone builds, cable management is a very important aspect, especially in small frames. Cables can be divided in two, digital signal buses (for UART, I2C and SPI communication) and power transfer. Particular attention must be paid to three-phase current wires going from Electronic Speed Control towards brushless motors, these are the highest alternating currents moving along the drone, and the induced magnetic field can strongly affect both communication and sensing, specially in the case of compass sensor where the magnetic field is the measured quantity. It is strongly recommended to keep sensors reasonably far away from power lines or/and shield sources of strong oscillating signals. As stated by the the self-definition itself "A printed circuit board (PCB) is an electronic circuit used in devices to provide mechanical support and a pathway to its electronic components" [4], perfect match of two fields converging into the dual purpose of the PCB.

4.2.1 PCB main frame

Design

Preliminary researches on drone body frames have been conducted to verify that the previous work is still useful for the the current case study. Summarizing the main aspects taken into consideration:

- **Shape**: quadrotors most common configurations are True X, Stretch X, Wide X, Dead Cat and HX illustrated in Figure 4.2 for brevity. First one is the symmetrical on every axes so good in the case of equal type of movements on both pitch and roll, Stretch one have a major distance between the center and the position of the rear motors, commonly used in drone racing builds when it's fundamental to maximize pitch forces to increase maximum fly velocity, clearly not this case. Wide X is the same shape as the previous one, but rotated 90°, so frontal and rear motors couple are more spaced, expanding the possible view for a camera facing forward. Dead Cat is an hybrid between the frontal part of a Wide X and the rear of a Stretch X, an interesting configuration, but discharged because more asymmetrical, the advantages offered did not justify the choice. HX is basically a Wide X with little more space in the center to make room for electronics, and is the right choice for the design.

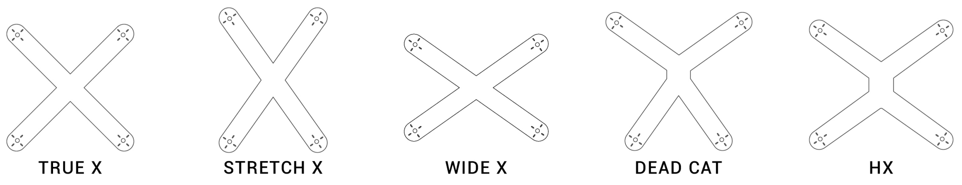


FIGURE 4.2: Frame shapes [5]

- **Unibody or Separate arms**: difference from the two approaches can be observed in the Figure 4.3. A single body frame is more easy to construct, gives more rigidity to the design, but in case of crash have to be replaced completely. Separated arms is more modular, if in a crash just one of the four arms will break, the operator have just to change the broken one with decreased cost of the operation. Another benefit coming from the separate arms configuration is a better mechanical damping from vibrations, but the connection of the four separated arms imply the need of extra plates, nuts and bolts, and so an increase of the total weight of the frame. For all these reasons, the final choice fell on the unibody solution.

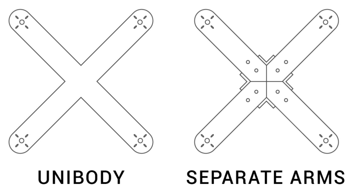


FIGURE 4.3: Unibody or Separate arms [5]

These two characteristics correspond to the description of the previous structure (Figure 4.4, and since the previous shape is the result of a mechatronic master thesis accompanied by static and dynamic mechanical simulations, it represents a really good starting point for the new design.

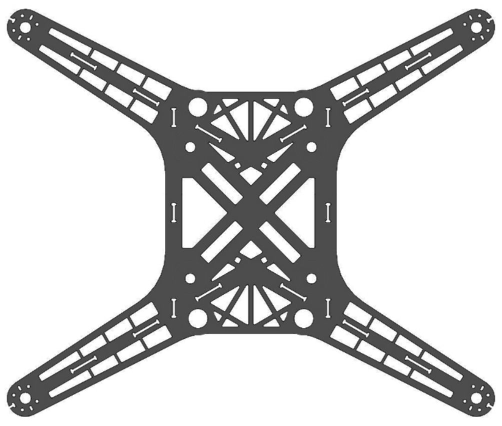


FIGURE 4.4: Previous middle frame

As can be seen by the Figure 4.4, frame is constituted of a central body made out of a rectangular shape to accommodate ESC, Power module, telemetry module and battery, and four arms for motors, longer than the carbon fiber propellers length. Several functional holes are then present in the frame, on the arms central axes and in the central body sides, small rectangular slots with curved extremities are engraved as housing for support flanges, glued perpendicular to the main frame to harden the structure with respect to forces applied perpendicular to the frame plane (the force of gravity, rotating propellers propulsion and hard landings). Four holes are also cutted off to insert rubber dampers, gum cylinders used to mechanically isolate the upper frame supporting the autopilot and IMU sensors
Then some factors have been taken into account in therms of dimension, weight and electrical considerations:

- ESC and Power module can be integrated into the PCB itself, the same thing it's not recommended also for other parts like the autopilot or the GPS module, the first one for simplicity and the second one because GPS module need to be placed on top of the build to better receive GPS signal from the atmosphere.

- Some optimization in therms of total area can be gained from the previous frame, in the order of cm, so the shape can be shrinked in both width and length.

- Explore different options for the thickness of the board is another main actor to reach the final design, a thicker PCB is heavier but supports higher mechanical forces and stress, a thinner one instead is weaker but also lighter, a trade off between robustness and lightness have to be set.

- integration of electronic components on the PCB must not affect modularity of the system, since the drone is a prototype and not a final product and specially in testing different functionalities something can go wrong, so one of the constraint imposed was to use easy replaceable electronics.

With these list clear in mind, the 2D shape of the already available frame need to be imported in an electronic CAD environment, the one chosen was **KiCad EDA** for many reasons:

- Free license.
- Open source, slightly different versions of the software available online in many repositories with increased potentiality.
- Cross platform, works both in Windows and Linux operating systems, fully compatible.
- Large community support of technicians, makers and hobbyist.
- Footprints available in main electronics online shops such as RC-electronic or Mouser.
- Very intuitive and already in-use by other researchers in PIC4SeR labs, so better to use the same software to give continuity to the project.

Imported the previous frame, measurements have been done with the dimensions of all the components needed, some space can be reduced in both length and width, so the central rectangle was exploded, arms moved closer by some cm. Then, continuing with the idea of a modular design, the footprints of the three components associated directly to the power system (ACSP5 buck-boost regulator, Electronic Speed Control and and motors) have been measured. Footprints of these pieces of hardware have been added to the project library, and positioned on the layout. Motors obviously placed at the extremity of each arm, the ESC in such a way that all the lines going from each pin to the motor relative phase have possibly the same length for the best actuation of the alternate three phase control, so it was positioned in the exact center of the figure.

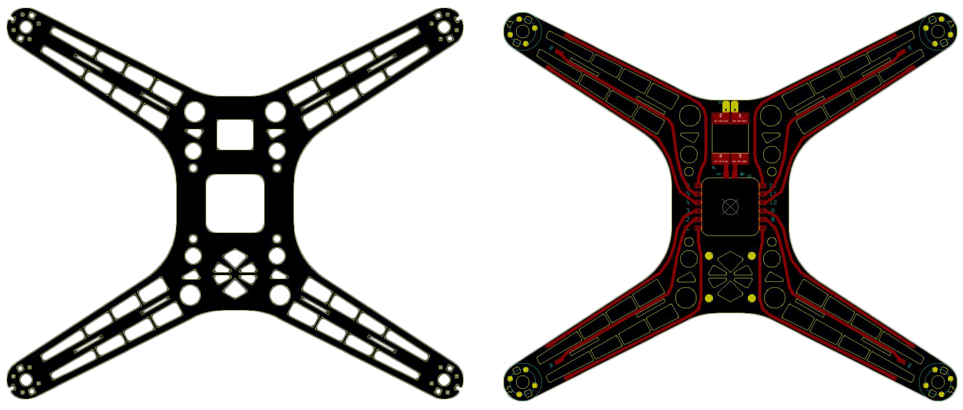


FIGURE 4.5: KiCad frame design

The only traces to be routed in the PCB are the power ones, some researches have been done on the routing dimensions to have a correct power transfer, without reflections, power loss and high increase in temperature According to IPC-2221A, the generic standard for Printed Board Design, in the section 6.2 relative to Conductive Material Requirements, the current formula is set to determine the maximum current admitted with respect to the trace dimensions and a certain temperature rise.

$$\mathrm{I} = \mathrm{k}\Delta\mathrm{T}^{0.44}\mathrm{A}^{0.725} \tag{4.1}$$

Rearranging the formula and setting the copper thickness, the minimum copper width can be evaluated as follows:

$$W = \frac{A}{H \cdot 1.378[\frac{mils}{oz}]} \tag{4.2}$$

$$A = (\frac{I}{k \cdot \Delta T^{0.44}})^{1/0.725} \tag{4.3}$$

Putting together 4.2 and 4.3:

$$W = \frac{(\frac{I}{k \cdot \Delta T^{0.44}})^{1/0.725}}{H \cdot 1.378} \tag{4.4}$$

The quantities in the equation are:

- $I = current\ [A]$
- $A = cross\ section\ [mils^2]$
- $\Delta T = temperature\ rise\ [°C]$
- $k = 0.048\ for\ outer\ layers\ k = 0.024\ for\ inner\ layers$
- $H = thickness\ [oz]$

The ESC of the build can drain a total of 12 Amps, divided by the 4 motors result in $I = 3\ A$, copper thickness to $H = 2\ oz$ and a very low temperature increase of $\Delta T = 15°C$ (also taking in mind that drones propellers will constantly blow air on the frame and so the routes will be constantly cooled, so this temperature rise is a lot overestimated) the corresponding required trace width results $W = 1.39\ mm$ at least. The width of the traces going from ESC to motors has been set to 1.5 mm, the maximum possible dimension to respect manufacturer constraints in such a drilled PCB.

As can be seen in Figure **??** all the tracks have been routed using rounded shapes to minimize changing in route's impedances and reduce reflections, this procedure does not come with the standard KiCad software, a little limitation bypassed with a build of the same software found on Github [6], run smoothly after couple try on Ubuntu 16.04, not compatible with Windows environment, so at least last steps of the PCB layout was completed on Linux OS. From here the 2D shape of the frame is considered defined.

4.2.2 Mechanical simulations

The technical drawing have been imported in SolidWorks, extruded with different heights, some static simulation have been analyzed. Two kind of forces needed for the simulation:

- **Complete drone Maximum takeoff weight**: in all these kind of simulation, to have a good view of the real bending of the frame, the mass used is *double* the drone total mass, so $250 * 2 = 500\ grams$ of weight, corresponding to $0.5kg * 9.8\frac{m}{s^2} = 4.9\ N$. These rule of thumb is good to easily analyze the static case taking into account the extreme displacement like hard changing in direction and movement on the vertical axis, for example when drone is free falling and suddenly the motors start spin at high thrust to regain altitude.
- **Maximum motor-propeller force**: staying at Table 4.1 relative to Tiger Motor MT1306-10 3100KV brushless motors total thrust in different configurations in therms of power supply and dimension of the propellers, the maximum thrust in grams achievable with 6030 propellers (very similar to 6020 actually used in the build) is a 218[g] thrust. This is the force applied each of four cylinders placed on the extremity of the arms, same position of the actual motors and connected on the top of the frame as can be seen in Figure 4.6

Tested with T-motor 6 A ESC

Prop	Voltage [V]	Current [A]	Power [W]	Thrust [g]	Efficiency [g/W]
4025	7	3.2	22.4	123	5.49
	7.4	3.4	25.16	131	5.21
5030	7	4.8	33.6	171	5.09
	7.4	5.2	38.48	179	4.65
6030	7	5.7	39.9	205	5.14
	7.4	6.2	45.88	218	4.75

TABLE 4.1: Tiger Motor ® MT-1306-10 3100 KV thrust table

Standard PCB manufacturer height are from 0.8 to 1.6 mm with 0.2 mm step, so simulation on every of these quantities have been run. Total mass of the frame evaluate for each thickness is store in the Table 4.2. Clearly the more the Height increase, less the bend of the material will be, but the result in weight is doubled from the first to the last one tested.

Frame height [mm]	0.8	1.0	1.2	1.4	1.6
Frame weight [g]	33.9	42.3	50.9	59.26	67.7

TABLE 4.2: Simulated thickness

From there, two approaches have been screened, one is to emulate the carbon frame of the previous built using this mainframe coupled with PCB flanges in a T shape, the second one is to 3D print the support. In the PCB flange hypothesis, frame and flanges can be soldered together, but soldering moisture is not the best material for mechanical stresses, with the 3D print support and legs, some kind of glue like epoxy can be used resulting in a more suitable option. Other consideration on the weight have to be observed, and looking at material densities in Table 4.3 with the same shape in the two cases, ABS or PLA solution result in a lighter component, or the shape can be better modeled with more volume and more optimized than a simple flange. The second approach is the one chosen, so that the main frame could remain the same when the support can easily mutate in time just printing different sketches, also to accommodate maybe different batteries or other small changes in camera position and stuff like this.

Material	Density [g/cm^3]
FR-4	1.85
PLA	1.21 – 1.43
ABS	0.90 – 1.53

Table 4.3: Flange materials densities

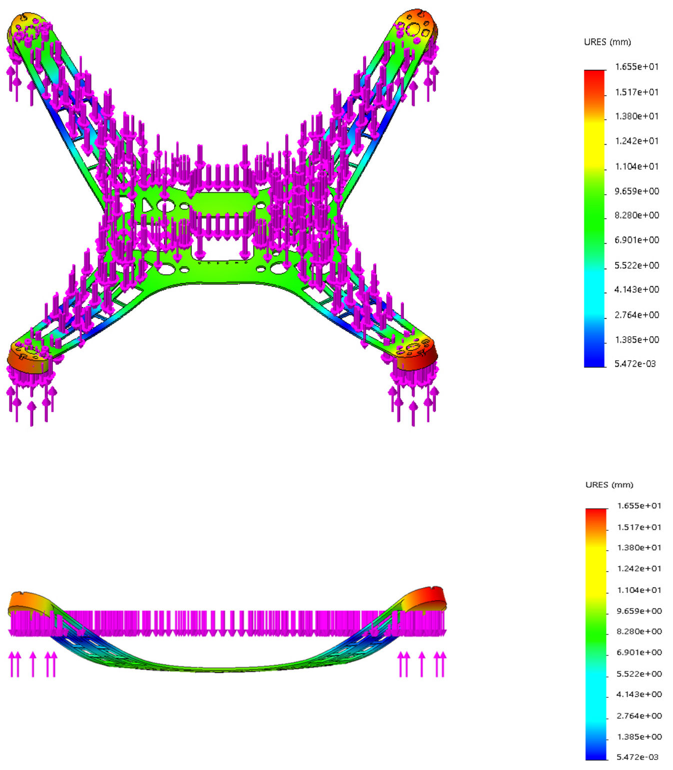


Figure 4.6: SolidWorks static simulations

4.2.3 3D printed main frame support and legs

The design of the frame support started from same considerations done for the frame, since the first one is a single body, the same approach is also adopted for flanges. Some preliminary 3D printed prototypes have been produced for fast testing on the real strength of the resulting assembly of PCB and PLA/ABS. The tests sample printed are reported in Figure 4.7, first one is just a single leg designed in a very basic rectangular shape, this to test if the height of the flange is enough resistant, with a fixed height of 8 mm and width of 5 mm which if the same as the central rectangle of the arms. The test was good, for a second prototype with two arms integrated in just one model, with legs optimized in a x shape internally engraved, and all the parallelepipeds rounded in the corners to save some material. Also this one results in a solid and compliant piece, so the last and definitive model was arranged, including all the four arms and legs in one single design, with 4 round rings to accommodate damping cylinders.

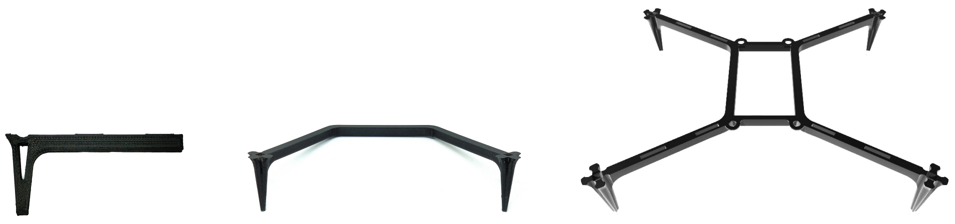


FIGURE 4.7: Frame support tests

4.2.4 3D printed secondary frame

Another part 3D printed is the support for the autopilot and couple other components, in Figure 4.8 the two different configuration developed are reported, one permits the mounting of a big GPS like the one of the old build in a frontal position, the other one is otherwise studied for small GPS like the one used in the current configuration. A third possibility is to use th smaller one with old GPS mounted on top of the autopilot, so in a central position forming a sort of tower. Frontally, both shape have a little protrusion where the camera can be mounted on the damped structure to record more stable video.

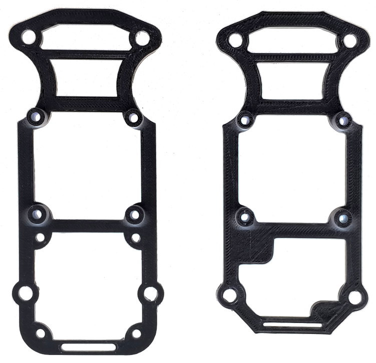


Figure 4.8: Upper frames

4.3 Final assembly

The Figure 4.9 is how the complete assembly look like. The result of the build was pretty satisfying for some aspect in particular, primarily in the reduction in weight, with a total Maximum takeoff weight of just 214 grams all-included, but also in the general clean of the build in therms of cable management, some regard just on the central body where wires to connect the autopilot peripherals and sensors are are still essential. ESC and Power module directly soldered on the frame-pcb look and work pretty well also at high motor throttle, no heating or strong vibration detected, the hover flight seem more stable than previous built. Also component fit good on both main and damped frames. A particular consideration have to be done to the possibility of a bigger battery with the consequent increase of the Time Of Flight, passing from a 800mAH to a 1300mAh (maximum possible capacity considering battery weight) result into 60% more time flight, exceeding widely the 10 min barrier. A complete comparison with the previous built will be treated in Chapter 7.

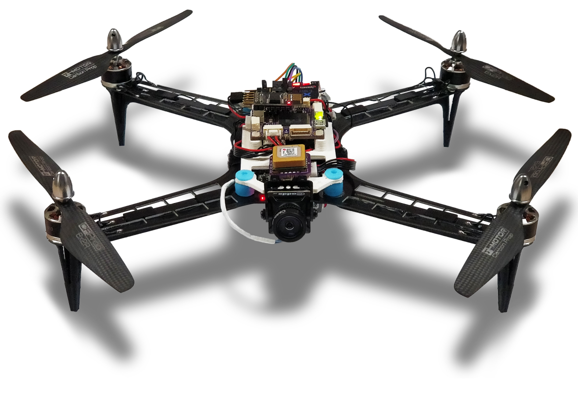


Figure 4.9: Final Assembly

Chapter 5

Testing

Testing was the main activity performed during thesis period, under various aspects mixing Hardware and Software fields. In this chapter, the intention is to summarize the activities and unfold the purpose of each tests method adopted. Every testing section require complete toolchain installation, every instruction can be found at *dev.px4.io > Getting Started > Toolchain Installation*, software involved are Ninja, MAVROS, NuttX QGroundControl, Gazebo, everyone of these will be better explained in the section in which they will be treated.

5.1 SITL

Software In The Loop is a simulation of the entire flight stack running on computer, and can (and usually do) interface with a virtual environment where the robot/drone can move in 3D space. PX4 stack is fully compatible with many simulators, the one used for this project is Gazebo, a very powerful robot simulator for test algorithms in realistic scenarios based on solid physics engine all into a well-designed simulator. Software In The Loop simulations are used mainly to test bug in the code itself such as infinite cycles, wrong routines, overflows and the general stability of the whole software, preliminary step recommended when the code heavily modified or upgraded with new feature build on a previously stable version, that can lead to an instable behavior. As said, the code run entirely on a PC, so the firmware have initially to be selected and built with bash command here reported:

Listing 5.1: Run SITL on Linux

```
cd ~/src/Firmware

export PX4_HOME_LAT=45.043010
export PX4_HOME_LON=7.539431
export PX4_SIM_SPEED_FACTOR=2
make posix gazebo

make px4_sitl_default gazebo --verbose static_map_plugin.world
```

Some options have been added:

- **export PX4_HOME_LAT=[home latitude]** : set the initial latitude of the quadrotor in the simulated world.
- **export PX4_HOME_LON=[home longitude]** : similarly to previous command, for longitude.
- **export PX4_SIM_SPEED_FACTOR=2** : used to double the simulation speed, time saving and will not compromise the correct behavior of a flight stack usually running on 32 bit microcontroller, now running on a 64 bit processor with multiple core, the simulation can be speeded up more proportionally to the PC computational power.

Once the Gazebo environment have correctly loaded and the quadrotor is shown in the 3D space created, QGroundControl can be launched in another window. If the procedure goes right, a red arrow appears in the QGC map in the home position, meaning that the two programs are correctly communicating each other. now the quadcopter can fly in with different options, one is to directly send commands like *pxh> commander takeoff* in the prompt where gazebo is running, in this case the result is the drone taking off and hover waiting for a command. Another way to pilot the drone is directly send commands trough QGroundControl in the exact same way the user do in the case of a real flight, launching same *commander* instruction that can be used in Gazebo but trough NuttSHell prompt (*QGC > Analyze tab > Mavlink Console*) or planning any type of mission in *QGC > Plan tab* and launching the mission start.
For the purpose of this thesis, SITL simulation have just been run to check the environment prior to other type of tests, but for future firmware upgrade it will become a necessary step in the workflow.

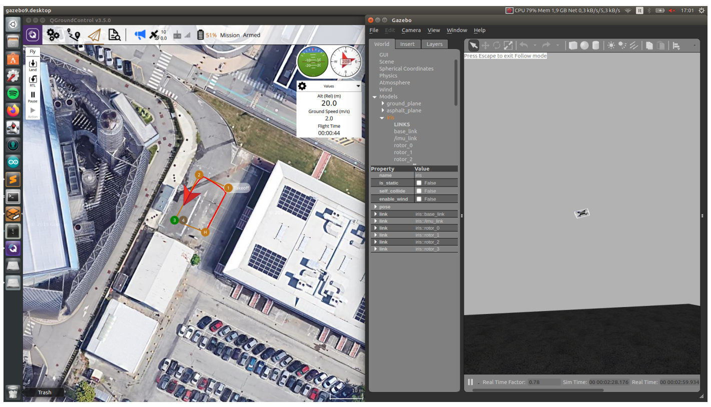


FIGURE 5.1: SITL simulation overview

5.2 HITL

Hardware In The Loop is a simulation used to run PX4 firmware directly on the autopilot hardware, in our case on the PixRacer. This kind of test is aimed at testing if the total load of the code can run smoothly on microcontroller, and also if any pin/peripherals respond correctly to commands/actuations.
HITL run on the same environment of SITL, with the flight controller connected to a development computer via USB/UART.
Figure 5.2 show the general architecture of the simulation environment.

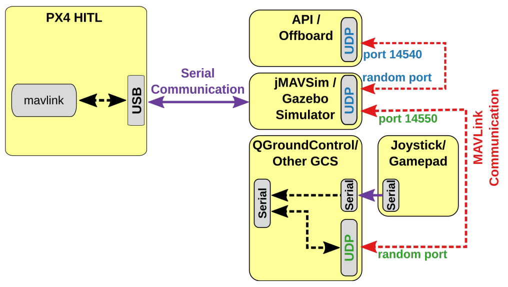


FIGURE 5.2: HITL simulation overview [7]

HITL need QGroundControl and Gazebo setup, here reported the step needed for the first one:

1. Connect PixRacer via USB.
2. go to *QGC > Vehicle setup > Safety* and enable HITL mode switching the drop-down menu from *Disabled* to *Enabled*.
3. from *QGC > Vehicle setup > Airframe* select *Simulation (Copter)* with extra option *HIL Quadcopter*
4. if needed, calibrate RC and/or Joystick
5. in *QGC > Application settings > General*, under the item *"Autoconnect to the following devices"*, uncheck all boxes except for UDP.
6. if you need, configure Failsafe and Joystick options changing *COM_RC_IN_MODE* and *NAV_RCL_ACT* parameters in *QGC > Vehicle setup > Parameters* tab.

To correctly setup Gazebo you need to follow these steps:

1. At least the first time, need to build PX4 with gazebo option to build also Gazebo plugins, with the cmd lines:

```
cd <Firmware_clone>
DONT_RUN=1 make px4_sitl_default gazebo
```

2. Open the .sdf file of the vehicle's model for example *Tools/sitl_gazebo/models/iris/iris.sdf*, search for *plugin name='mavlink_interface'* and change parameters *<serialEnabled>* and *<hil_mode>* to **true**.

3. in the same file, if necessary, change the serial device port of the PixRacer attached, to check the device id, run in another terminal the command:

```
dmesg | grep "tty"
```

   This command result in the list of the attached devices in order they are plugged-in, so disconnecting the USB cable, reconnecting it and launching the command give at the last line the name of the device, usually switch from ttyACM0 to ttyACM1

4. Set up environment variables:

```
source Tools/setup_gazebo.bash $(pwd) $(pwd)/build/px4_sitl_default
```

5. open Gazebo with the correct world, with the parameters change it will run in HITL mode.

```
gazebo Tools/sitl_gazebo/worlds/iris.world
```

After these steps, the user will be able to run QGroundControl and run commands and missions.

5.2.1 Camera trigger

One of the drone requirements is the autonomous acquisition of high definition video/photo. For this purpose, a lightweight FPV camera is used, and in its package, came with a switch to power on/off the camera with a long press, and to start/stop the video recording, unfortunately for geolocalization matter, the camera does not permit photo capture, which is pretty common in this type of drone cameras, only bigger ones have this possibility. First, some parameters are required to correctly setup camera trigger on the autopilot. Parameters to change are the following:

- **TRIG_PINS = 6**: set camera trigger pin, can be set also for multiple pins if needed.
- **TRIG_INTERFACE = 1**: pushbutton work pulling down the trigger line, so the pin must be set to GPIO mode, always on high/low state, changing state when triggered.
- **TRIG_POLARITY = 0**: set trigger polarity, 0 means Active low, 1 Active high. This parameter cannot be changed into QGC
- **TRIG_ACT_TIME = 1000**: set active trigger time at 1s (1000ms) since manufacturer of the Caddx HD camera declare 1 sec press on the button to start

video recording. item **TRIG_MODE = 1**: this set the camera trigger mode to *Time based, on command*, it means that the command will be triggered on command and not time based, since video capture must begin at the first waypoint and terminate just before landing maneuver, indipendently on the effective flight time.

After a reboot to set all parameters on the Flight Controller, the correct behavior of the trigger pin has been tested. To do this, the "trigger command" can be manually activated from *QGC > Vehicle Setup > Camera* with the "Trigger Camera" button, or sending on the command line in *QGC > Analyze > Mavlink Console* the command *camera_trigger test*. The correct pull-down of the sixth pin of the PixRacer have been verified by the use of an oscilloscope, and everything worked as expected.
The next step was to incorporate the trigger command into the mission flight plan, this will treated in next subsection.

5.2.2 Mission Plan

Detailed Plan File Format can be found on the attached *link*. Basically .plan file is a JSON string containing very basic information on vehicle and firmware running, and mission informations such as Home position, Rally Points, Geo-fence, Waypoints and mission commands.

LISTING 5.2: Plan File example

```
{
    "fileType": "Plan",
    "geoFence": {
        "circles": [
        ],
        "polygons": [
        ],
        "version": 2
    },
    "groundStation": "QGroundControl",
    "mission": {
        "cruiseSpeed": 15,
        "firmwareType": 12,
        "hoverSpeed": 1,
        "items": [
            {"autoContinue": true,"command": 530,"doJumpId": 1,
               "frame": 2,"params": [0, 0, null, null, null,
               null, null],"type": "SimpleItem"
            },
            {"autoContinue": true,"command": 2000,"doJumpId": 2
               ,"frame": 2,"params": [0, 0, 1, null, null, null,
                null ],"type": "SimpleItem"
            },
            { "AMSLAltAboveTerrain": null, "Altitude": 5, "
               AltitudeMode": 1, "autoContinue": true, "command"
               : 22, "doJumpId": 3, "frame": 3, "params": [ 15,
               0, 0, null, 45.061850921636996, 7.662915040275976
               , 5], "type": "SimpleItem"
            },
            { "autoContinue": true, "command": 203, "doJumpId":
                4, "frame": 2, "params": [ 0, 0, 0, 0, 1, 0, 0 ]
               , "type": "SimpleItem"
            },
            { "autoContinue": true, "command": 93, "doJumpId":
               5, "frame": 2, "params": [ 20, 0, 0, 0, 0, 0, 0 ]
               , "type": "SimpleItem"
            },
            { "autoContinue": true, "command": 203, "doJumpId":
                6, "frame": 2, "params": [ 0, 0, 0, 0, 1, 0, 0 ]
               , "type": "SimpleItem"
            },
            { "autoContinue": true, "command": 20, "doJumpId":
               7, "frame": 2, "params": [ 0, 0, 0, 0, 0, 0, 0 ],
                "type": "SimpleItem"
            }
        ],
        "plannedHomePosition": [45.06184801299343, 7.6629065558
           804825, 247],
        "vehicleType": 2,
        "version": 2
    },
    "rallyPoints": {
        "points": [],
        "version": 2
    },
    "version": 1
}
```

In the Listing 5.2 is reported a really simple mission file, that will now be explained in detail. Proceeding in order of appearance, the sections of the file are:

- **Geo-fence**: define the eventual boundaries in therms of circles and/or polygons.
- **Mission**: here the real mission plan is defined, with SimpleItems (basically points or timing where commands have to be executed) or ComplexItems (such as Surveys, Corridor Scan, Structure scan, automatized by the ground station). SimpleItems are divided into two main fields:

  - **Waypoints**: points of the path between the takeoff and the land used one by one as position setpoints by the vehicle running the mission.
  - **Commands**: actions to be executed during the mission, like maintaining hover for a certain amount of seconds or trigger a camera for example.

  At the end of the mission string is present the position in therms of latitude, longitude and heights in MAMSL.

- **Rally Points**: optional safe points where vehicle can land in case of failsafe before the completion of the mission.

In the mission 5.2, first 2 mission items are just configurations, then the *"command" = 22* correspond to the quadcopter takeoff, followed by the MAVLink command 203 which correspond to VEHICLE_CMD_DO_DIGICAM_CONTROL, same command used internally by the camera_trigger function as can be seen from code 5.3.

LISTING 5.3: camera trigger test code

```
CameraTrigger::test()
{
        vehicle_command_s vcmd = {};
        vcmd.timestamp = hrt_absolute_time();
        vcmd.param5 = 1.0;
        vcmd.command = vehicle_command_s::VEHICLE_CMD_DO_DIGICAM_CONTROL;

        orb_advertise_queue(ORB_ID(vehicle_command),
                                                    &vcmd,
                                                    vehicle_command_s::ORB_QUEUE_LENGTH);
}
```

Then the vehicle is forced to hold position for 20 seconds with *"command" = 93* to have a small footage of 20 second flight, and then again the camera control to stop the recording just before the Return to Land command send with the mavlink message number 20. This skeleton can be used for every kind of mission, just changing home position and putting as many waypoints needed or complex items (such as corridor scan or perimeter survey) replacing the hold command. QGroundControl (QGC) is able to automatically generate automatized path just delimiting a perimeter and setting some camera and image parameters, but this works only with devices capable of taking photos, the actual camera instead is only capable of video recording, so every camera trigger will result into the start/stop recording actual clip. This can be in the future a way to better geolocalize portions of the video clips instead of have single cut from begin to the end of the mission, but for example in terrain reconstruction or 3D buildings reconstruction a single cut is a correct input file, and so the plan file 5.3 will be taken as the drone basic mission routine. Complete list of the available commands is available in the Messages section on the MAVLink website

5.3 Flight Tests

Different type of flight tests have been conducted to test different functionalities gradually. Test are divided into 3 typologies:

- **Manual**: The simplest flight possible, all in manual mode guided with the remote controller by drone pilot, preliminary step to verify the correct response to controls in therms of Throttle, Roll, Pitch, Yaw and correct communication between drone, radio and control station trough WiFi to see the flight in real time with the QGroundControl window.
- **Hover**: This type is to verify the correct handling of GPS position and barometric height. Can be tested with the takeoff command on the MAVLink terminal internal to QGC, or more simply taking the drone in the air with a manual takeoff with the radio controller and then modify the flight mode changing the state of the relative radio switch (have to be set previous to flight using the ground control interface or drone parameters). A correct hover flight correspond to correct data fusion between GPS, barometric sensor and magnetometer, in case parameters or sensors orientation are wrong, drone will diverge instead of remaining on a well defined 3D air perimeter (with a good GPS precision will remain roughly on a 2 meters radius sphere, see 5.4.5 for some analysis).
- **Mission**: Once other 2 type of test flight have been correctly executed, a proper mission with some waypoints can be run safely. Here the check is on the drone flight path, that have to match the planned one with a good uncertainty, max two meters error in latitude, longitude and MAMSL.

Every one of these tests have not only the purpose to check the correct run of the flight, but also different target analysis, that will be discussed in the next section.

5.4 Flight Analysis

Every flight is registered and saved as a log file on the microSD mounted on the Pixracer. The PX4 log file came in the .ulg file format, a propretary one that can be managed in different ways:

- **pyulog**: a set of python scripts made available directly from PX4 Github repository permits (after correct installation and configuration) to convert .ulg files into more common and manageable ones, for example the function *ulog2csv* extract the data contained into the Ulog file into many sub-folders defined by the representing datas (IMU, Actuator controls, Position, Power management etc), each folder containing a set of .csv files, that are essentially formatted tables, very handy when have to be loaded and analyzed by any kind of scripts, for example Matlab.

- **Online PX4 Flight Review**: This online tool permits to upload directly the .ulg flight log and after couple seconds visualize various Graphs representing almost all the parameters necessary to a good analysis of the flight and drone attitude behavior. Flights remains also online until deleted manually, and so are accessible from every terminal with an internet connection. Further details can be found on the next section 5.4.1
- **Third parts analyzers**: there are very different log analyzer free and also under subscription on the market, they have been taken into consideration but then discarded because don't needed or less easy to use.

Most of the logs collected came from the old built but with all new components mounted, this was done to rely on a stable and already tested frame, so this factor is taken for good and only autopilot and peripheral are the object of tests. The behavior of the two builds in fact did not change at all, since components are the very exact same and frame dimensions and shape is more or less the same, in depth comparison will be treated in Chapter 7.

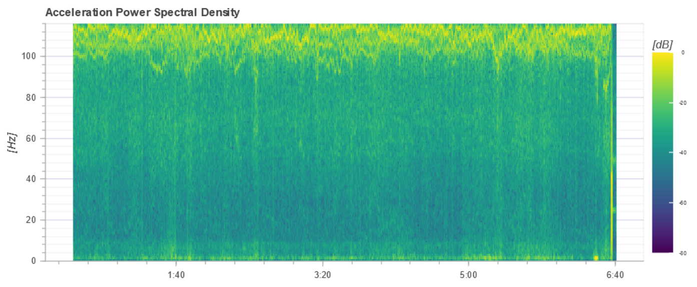
5.4.1 PX4 flight analysis

This amazing tool by PX4 team itself is available at https://review.px4.io/, and include almost all the plots to check most relevant drone indicator of a good flight. Graphs can be read from the top to the bottom of the page, in order of appearance, here i will reorder them, selecting first the basic one, the quantities that can directly influence the one following, and then more depth parameters still important. Not all the parameters will be treated here, for additional info the reference webpage can be consulted at *docs.px4.io > Flight Log Analysis > Log Analysis using Flight Review*. Every graph illustrated from now on this chapter is related to an hover flight test, the best test to find with ease problems in build or control, since the flight must be smooth and power consumption is constant over time.

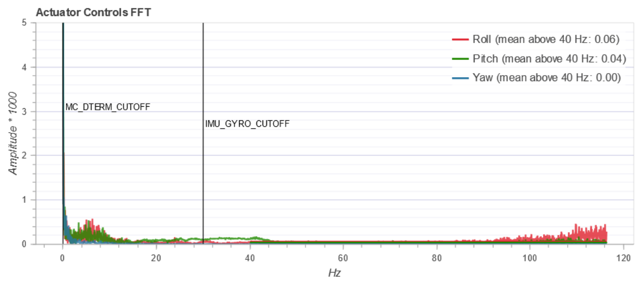
Vibrations

Different drone frames and builds in general result in different mechanical properties, this factor is amplified when using non-standard frames as in this thesis. Weak materials or loose screws can lead into strong vibrations, one of the main antagonist of a bad control since IMU (in particular accelerometer and gyroscope) measure quantities strongly influenced by oscillation, so this strong oscillations result as noise in attitude measures. In the figure 5.3a, the Acceleration Power Spectral Density is represented, weak vibrations are represented in blue, strong vibrations in yellow. A good spectral density to safely fly have just some marked yellow lines near the motors spinning frequency (more or less 100Hz), and the rest of the plot tending to blue. As can be seen from Figure 5.3a, the plot represent perfectly the just illustrated behavior, so vibrations are considered good.

For the actuation graph instead, the represented figure must have one spike on on 0Hz with some kind of slope, then lower until around 20Hz and remain as flat as possible, every other spikes means overshoots on control actuators and have then to be tuned or at least well filtered changing parameters such as IMU_GYRO_CUTOFF, which is basically the frequency used to filter IMU and in particular case gyroscope. Another time, the Figure 5.3 perfectly represent the illustrated attitude, and for this reason taken as good one.



(A) Acceleration PSC



(B) Actuators FFT

FIGURE 5.3: Vibration-related graphs

5.4.2 Power

The other main aspect is Power consumption:

- Battery voltage and Battery remainning must decrease constantly with the same trend, just on two different scales since battery voltage is expressed in Volt when battery remaining is indicated in integer values between 0 and 100.
- BAttery current absorption must remain more or less constant on a hover flight, if peaks revealed, check bad wirings specially in the branch from battery to motors.
- 5V must remain constant, this is the Autopilot power supply, drops on this parameters lead to bad controls or in the worst scenario to the complete poweroff of the flight controller with the consequence of a almost certain crash.

Another time, the plot is just the expected one.

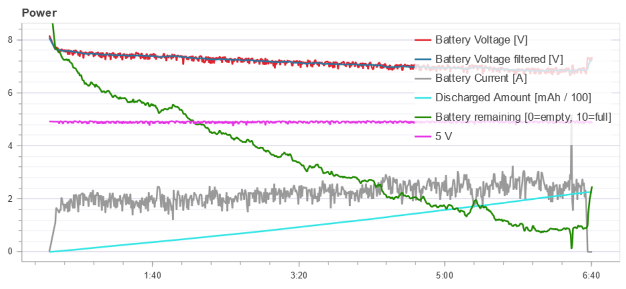


FIGURE 5.4: Power plot

5.4.3 Magnetic field

The compass sensors is with the GPS fundamental to evaluate the position of the quadcopter in the air, since GPS result just as a dot with the proper coordinates on the map, when compass give the pointing vector of the front of the vehicle, and from here the correct actuation in therms of velocity direction setpoint can be evaluate both to hold position or to follow a certain path previously planned, these consideration are not so important in manual flight however in this application have to be monitored for a good stability of the overall system. Magnetometer sense magnetic fields in the 3 cartesian axes and is so strongly influenced by external magnets or magnetic source such as other electronic or power lines. Magnetism laws states that a wire carrying currents generate magnetic fields concentric to the wire itself with strength proportional to the current passing with the law $B = \frac{\mu_0 I}{2\pi r}$. for this reason, magnetometer sensor on the Build have to be taken as far as possible at least to ESC and three-phase wires going towards the motors. Since almost all the GPS modules have an integrated magnetometer, a good practice is ti place the GPS far from the units just mentioned, in certain builds also placing it on a 5-10 cm pole just over the drone frame. As mentioned, higher the current, higher the induced magnetic fields, and in a drone the factor that increases the power currents is the motors throttle. In the graph 5.5 the magnetic fields and the motors thrust are plotted together, a good response can be appreciated when the two graphs are uncorrelated each other, contrary, a magnetic fields that became higher increasing the throttle it is a symptom of coupling between these two quantities and will lead to system weakness. In our case, the two parameters are affected just on the takeoff phase, but this is considered normal in a compact build like our case, but during flight time they are uncorrelated and so considered fine.

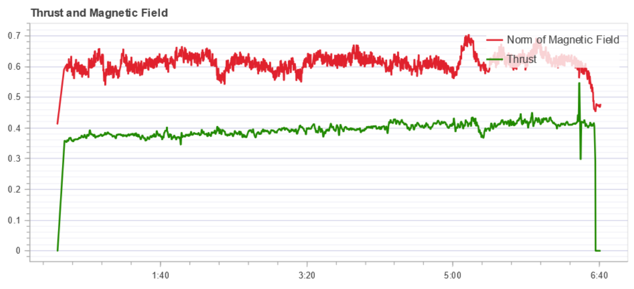


Figure 5.5: Thrust and magnetic Field

5.4.4 GPS

Global Positioning System precision is fundamental to the correct positioning in space of the aerial vehicle, where no camera tracking and SLAM is used the only way to move the quadcopter in the horizontal plane, the height on the vertical axes are also evaluated with a rough approximation from the GPS is instead evaluated with the relative pressure measured by the on-board barometer using as reference the bar of the home position taken just before takeoff. In the first graph we can see four plots, now explained:

- Number of satellites used: is the counts of the number of satellites with a good signal to noise ratio, usually above 30dB, the ones then used to triangulate the current position as the intersection of all "n" the sphere with the "n" used satellites as center. This is the standard practice, more satellites are usually in the view, but some are covered from the horizon line or shielded and attenuated by trees, buildings or clouds and are stated as non-reliable for a correct measurement.

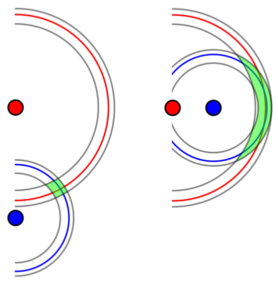
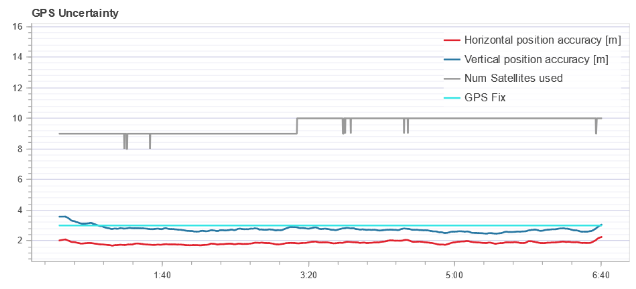


Figure 5.6: Diluition Of Position

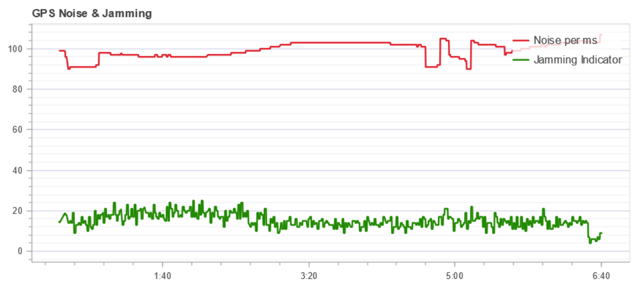
- Horizontal position accuracy: this metric is more technically called Horizontal Diluition Of Position, directly derive from the Geometric DOP, an indication of the good positioning of the used satellites, looking at the Figure 5.6 the dots represent satellites, blue and red semi arcs the exact measure of the distance and the other arcs the boundaries of the measure uncertainty. As can be noticed without changing measure errors, in the first case the resulting portion of space where receiver can be (green area) is lot smaller that in the second case. In practice, more the satellites are close each other, higher the DOP, less the accuracy.

- Vertical position accuracy: similarly to the just mentioned but on the Vertical axes, these value increase if satellites are close together but also forming a straight line, less in-line and more the Vertical accuracy. In fact all the Diluition Of Position are mathematical indicators of the good or bad distribution in space of the satellites with respect to a precise three-dimensional criterion.

- GPS fix: indicate if the fix is 2D or 3D, first case is unusable in our application since gives a very rough approximation of the positioning, and as can be seen from Figure 5.7a the value remain constantly 3, showing a stable three-dimensional GPS mode.



(A) GPS Uncertainty



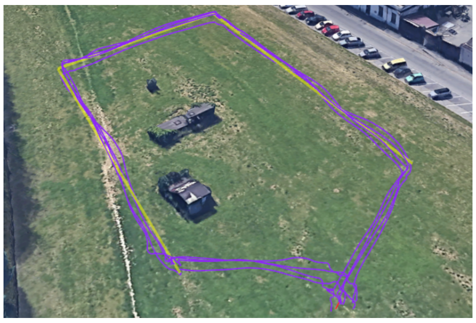
(B) GPS noise & jamming

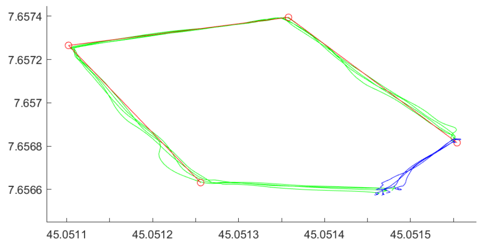
FIGURE 5.7: GPS-related graphs

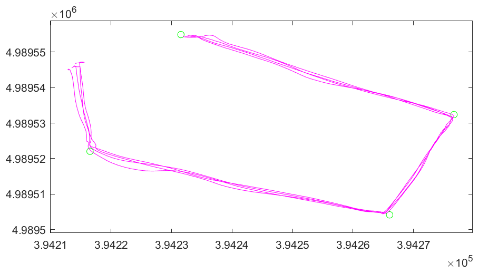
Other consideration on the GPS quality are noise and jamming indicators, first one gives a parameters on the cleanliness of the signal (like signal to noise ratio) but published on the log file as "noise per ms", which is at the moment still to clarify what does it mean in practice, but the PX4 analysis guide state that a good noise indicators remains always greater or equal to 80, and as can be seen in Figure 5.7b is constant around 100. The Jamming parameters instead relates to the possibility of other sources via cable or transmitting in the same frequency range of the GPS that can disturb the receiving of the GPS signal. The jamming indicator should remain under 40, in our case is near or below 20.

5.4.5 Matlab

To prove the overall stability of the system, some analysis on the mission path have been conducted. One of the mission flight with four waypoints has been taken as reference for this purpose (Figure 5.8a), a simple polygon with a total fly distance of around 150 meters without considering takeoff and landing. The quadcopter flied four time the exact same mission to see also the repeatability of the mission itself and compute some statistics. The log files have been collected, passed trough the Flight Review online, and exported for simplicity into .kml files, a format compatible with Google Earth (Figure 5.8a) and very handy to manage in Matlab. Imported the kml files and converted into Matlab structure with the *kml2struct.m* script, the mission plan is recorded as 4 waypoints, flights are instead a number of points forming the route flown (Figure 5.8b). This data are give in Latitude and Longitude coordinates in degree, so have been converted into Universal Transverse Mercator coordinates which result into a local metric expression, this by using *deg2utm.m* script. Then the distance between each of the four flights and the planned path have to be computed, since the path is represented as a series of points, the easy method is to compute the distance of the flight from each line passing from 2 points (so with 4 waypoints 3 lines are used), and save it into three array called *dist1*, *dist2*, *dist3*, and then save the complete distance from path to flight as min(dist1, dist2, dist3). This can be done not in all cases, but is perfect in the shape selected, because after leaving waypoint 1 and so the first line, the distance tends always to increase, and so the second line is used for the min distance from here, and so on. The plots of the distances are shown in Figure 5.8d, and indicates that Drone have always been under the 2 meters threshold, with a mean distance of 40 centimeters.


(A) Google Earth


(B) Matlab lat-lon


(C) Matlab UTM

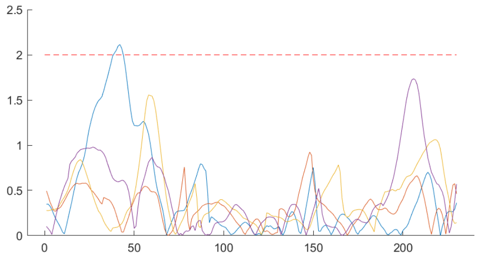
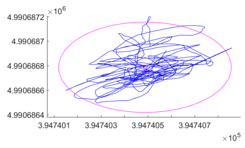
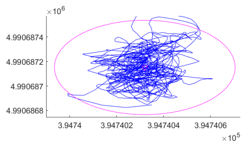

(D) Distance error

FIGURE 5.8: Flight path analysis

Same type of analysis have been conducted on Hold flight mode, with the drone takeoff at 5 metres followed by around 6 min maintaining its position and then land when battery reaches 5% charge. As can be seen in figure 5.9 all the three paths remain on the purple circle delimiting the 2 meters area around the mission waypoint.


(A)


(B)

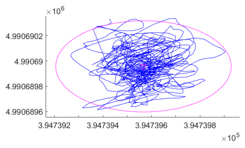

(C)

FIGURE 5.9: Hold flight analysis

5.5 Image post processing

One other kind of test have been made on the .mp4 video shot with the HD on-board camera, the aim of this test is to prove that the selected recording system despite being only used to show footage if the flight for entertainment, can be also used for different and engineering-relevant tasks such as 3D image reconstruction. Most programs accept as imput both photos and videos, but is not always clear how the extraction of the frames is made, so the steps used for testing the validity of the camera was the following:

- Frames extraction from video into jpg images.
- Sharp image selection, taking the less blurred images.

- 3D reconstruction with Pix4D or Agisoft Metashape.

The other important operation to ensure a good result is the camera calibration, wich is done by inserting sensod data into the model parameters, listed in Table 5.1. These are used for lens correction based on geometrical properties, to transform the form of the image to a linear representation, specially in fisheye camera like the one used in our case which have a huge 170° of recording FOV, capturing images very distorted in moving to the sides of the frame. In fact, in the reconstruction programs, there's an option for fisheye cameras, where a massive perspective correction is necessary.

Property	Value
Sensor Width [mm]	5.37
Sensor Height [mm]	4.04
Image Width [pixel]	1920
Image Height [pixel]	1080
Pixel Size [μm]	3.0
Focal Length [mm]	1.8

TABLE 5.1: Camera parameters

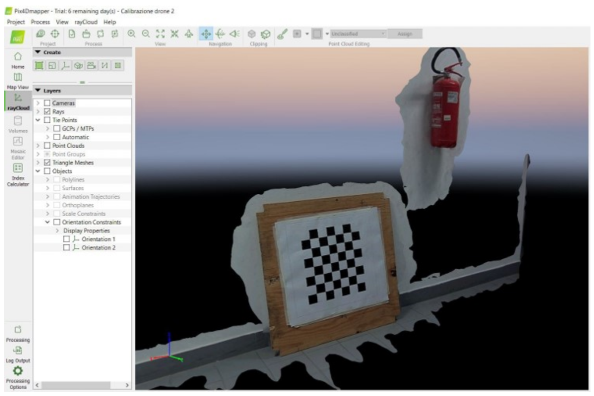
5.5.1 Sharp image extraction

As said before, the image extraction done in the reconstruction programs is sometimes unknown, so a pre-selection of the best images to be used in 3D modeling is a good practice. Firstly, all the frames from the video are saved in a folder as jpg files without any control on the quality of each frame, then the folder is set as input of a python script that select the less blurred images. In the script the cv2 OpenCV library is used, the program require to insert the path of the directory containing all the photos to be analyzed and how many frame consider to find the less blurred image, a sort of "reduction factor" to select one photo every n with this number defined by the user, the best value change primarily on the relative velocity of the vehicle recording the video, slower the speed and higher the number of frames.
This operation is needed specially when the object to be modeled is near to the camera, because of relative velocity and distance are inversely proportional, and also to attenuate the rolling shutter, the method used to acquire frames not all pixels in a single time but scanning rapidly the sensor in a predefined direction (usually from top to bottom or vice versa), technique used to reduce sensor overheat and more simple to manage from a temporization point of view, only hi-end cameras have global shutter. Selected the images, they can be processed with dedicated softwares.

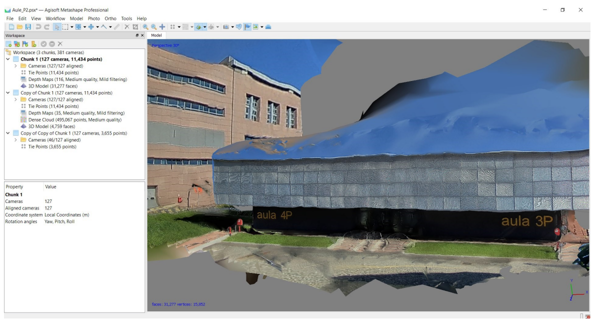
5.5.2 3D reconstruction

Different software for structure scan postprocessing are available on the market, we tried two already used internally in the PIC4SeR in other projects and they are called

Pix4D and Agisoft Metashape. First one is a powerful Computer Vision Lab developed starting from 2011 by the École Polytechnique Fédérale de Lausanne and then spin-off in Pix4D company, it includes a suite of sub-products for different task, for example Pix4Dmapper, Pix4Dsurvey, or Pix4Dmodel. A free 15 day trial was used for the tests, then the subscription 260$\frac{€}{month}$ or one-time charge of 3990 €, quite expensive. The other solution presented by Metashape is a 179 $ Standard edition including photogrammetric triangulation, generation and editing of dense point cloud , generation and texturing of 3D model and fisheye camera support, which are the features used for the tests, and a second Professional Edition with lot more features using georeference, markers, targets detection, multispectral processing, scripting and many many other options for 3499 $. Pro version of both softwares is more or less the same in both features and pricing, for very basic applications the possibility of the Standard edition is a plus on Metashape, and for this reason more tests have been computed with this second one.
In Figure 6.6a a perfect reconstruction of an chessboard and a fire extinguisher made in the lab, where Figure 6.6b is a reconstruction of a prefabricated just behind the Polito drones facility cage. The results as can be seen are satisfying, so the camera is considered fully usable for this kind of applications.



(A) Chessboard reconstruction



(B) Structure reconstruction

FIGURE 5.10: Reconstruction softwares

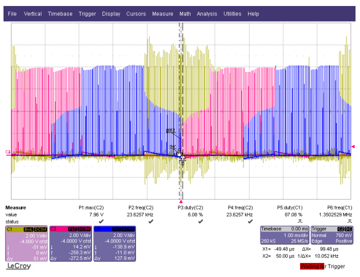
Chapter 6

Troubleshooting

Several problems came out in the project, both hardware and software. In this chapter some of the many encountered will be described.

6.1 ESC

During one of the first flight tests on the flight field a crash happened. one of the motors seem not to work properly, the drone start rotating on itself and loosing quote. From the Log file the actuators tried to regain quite putting the throttle to maximum value, without any result. At this point was confident that the problem could just be in the hardware. One of the power mosfet on the phase-leg burned, so the three phase current could not be correctly produced for the brushless motor. Only another ESC was available in laboratory, but unfortunately also in this one one of the four motors does not work, and these was also verified with the oscilloscope as can be seen from Figure 6.1 one of the 3 phase has an incorrect behavior, with several overshoot both in positive and negative.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
C3
gate
sw
LeCroy

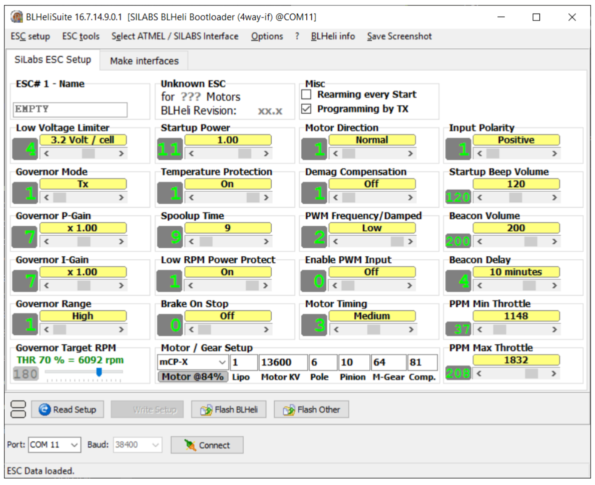
```
Measure   P1:max(C2)  P2:freq(C2)  P3:duty(C2)  P4:freq(C2)  P5:duty(C1)  P6:freq(C1)
value     7.96 V      23.6257 kHz  6.08 %       23.6257 kHz  87.08 %      1.3502529 MHz
status    ✓           ✓            ✓            ✓

C1 BwL DC1M        C2 FLT DC1M        C3 FLT DC1M         Timebase   0.00 ms   Trigger   C2 DC
     2.00 V/div         2.00 V/div         2.00 V/div              1.00 ms/div   Normal    760 mV
   -4.000 V ofst     -4.0000 V ofst     -4.0000 V ofst     250 kS   25 MS/s      Edge    Positive
↓     -51 mV       ↓     14.2 mV      ↓   -138.9 mV        X1=  -49.48 µs    ΔX=    99.48 µs
↑       0 mV       ↑   -258.3 mV      ↑    -11.0 mV        X2=   50.00 µs  1/ΔX=   10.052 kHz
Δy     51 mV       Δy  -272.5 mV      Δy   127.9 mV
                                                           Waiting for Trigger
```

FIGURE 6.1: ESC three phases under oscilloscope

Testing the circuit with a multimeter in diode mode no shorts to Val or GND were found, so mosfets are ok on this one. ESC are basically made of three main components:

- **EFM8 Busy Bee 2**: some little and specialized 8-bit microcontroller translates the four PWM input coming from the Flight Controllereach one into three signals to drive power mosfets.
- **Fortior Tech FD6288**: Mosfet drivers, active Integrated Circuit capable of converting control signals of the microcontroller into voltages suitable to switch on/off the pullup/pulldown power mosfets.
- **Fairchild Dual PowerTrench MOSFETs**: for each motors there are three phase legs, that are basically two power mosfets, one toward Val and the other to GND, that are used to make the square waveform with a 33% duty cycle realizing the alternate current that effectively drives the motors.

At this point the error could be caused both by microcontrollers or by drivers. A possible solution is to re-flash an updated version of the firmware on the BB2 micro-controllers, operation that can be easily accomplish trough standard Flight Controller and BetaFlight desktop app used in common drone builds, but not compatible with the PixRacer used in this case. So the dedicated program for the ESC has been used, called BLHeliSuite with source codes available on GitHub page. Once downloaded and unzipped, the .exe can be launched and the following windows appears:



ESC# 1 - Name: EMPTY
Unknown ESC
for ??? Motors
BLHeli Revision: xx.x
Misc: ☐ Rearming every Start; ☑ Programming by TX

Low Voltage Limiter: 4, 3.2 Volt / cell	Startup Power: 11, 1.00	Motor Direction: 1, Normal	Input Polarity: 1, Positive
Governor Mode: 1, Tx	Temperature Protection: 1, On	Demag Compensation: 1, Off	Startup Beep Volume: 120, 120
Governor P-Gain: 7, x 1.00	Spoolup Time: 9, 9	PWM Frequency/Damped: 2, Low	Beacon Volume: 200, 200
Governor I-Gain: 7, x 1.00	Low RPM Power Protect: 1, On	Enable PWM Input: 0, Off	Beacon Delay: 4, 10 minutes
Governor Range: 1, High	Brake On Stop: 0, Off	Motor Timing: 3, Medium	PPM Min Throttle: 37, 1148
Governor Target RPM: THR 70 % = 6092 rpm, 180	Motor / Gear Setup: mCP-X, Motor @84%		PPM Max Throttle: 208, 1832

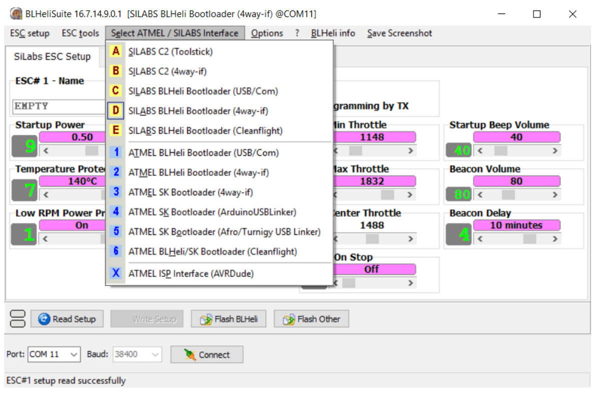
Lipo	Motor KV	Pole	Pinion	M-Gear	Comp.
1	13600	6	10	64	81

Port: COM 11 Baud: 38400

ESC Data loaded.

Figure 6.2: BLHeli Suite Interface

An Arduino Nano was used to program the ESC, so after connecting the Arduino, select in the bottom the correct COM Port, then click Connect, if successfully connected go to the *Make Interface* tab, then in the bottom right *Arduino 4way-interface*, Then click ok in the first screen that will pop-up, in the next window select $4wArduino\_Nano\_\_\_16\_MULTIv2000$ or something similar, depends on the version, but need to have MULTI in the name which stay for multi-motor. Click Open, then Yes, wait the writing, then OK. Now the Arduino is ready, click on the tab bar *Select ATMEL/SILABS Interface* and select *SILABS BLHeli Bootloader (4way-if)*.


BLHeliSuite 16.7.14.9.0.1 [SILABS BLHeli Bootloader (4way-if) @COM11]
ESC setup
ESC tools
Select ATMEL / SILABS Interface
Options
?
BLHeli info
Save Screenshot
A SILABS C2 (Toolstick)
B SILABS C2 (4way-if)
C SILABS BLHeli Bootloader (USB/Com)
D SILABS BLHeli Bootloader (4way-if)
E SILABS BLHeli Bootloader (Cleanflight)
1 ATMEL BLHeli Bootloader (USB/Com)
2 ATMEL BLHeli Bootloader (4way-if)
3 ATMEL SK Bootloader (4way-if)
4 ATMEL SK Bootloader (ArduinoUSBLinker)
5 ATMEL SK Bootloader (Afro/Turnigy USB Linker)
6 ATMEL BLHeli/SK Bootloader (Cleanflight)
X ATMEL ISP Interface (AVRDude)
SiLabs ESC Setup
Read Setup
Write Setup
Flash BLHeli
Flash Other
Connect

ESC# 1 - Name: EMPTY
Startup Power: 0.50 (9)
Temperature Prote: 140°C (7)
Low RPM Power Pr: On (1)
gramming by TX
in Throttle: 1148
ax Throttle: 1832
enter Throttle: 1488
On Stop: Off
Startup Beep Volume: 40 (40)
Beacon Volume: 80 (80)
Beacon Delay: 10 minutes (4)
Port: COM 11 Baud: 38400
ESC#1 setup read successfully

Figure 6.3: BLHeli ATMEL/SILABS selection

Then connect the PWM wires of the ESC to the Arduino digital pins as follows:

Device	Pins			
Arduino Nano	D3	D4	D5	D6
BLHeli 4in1 ESC	S1	S2	S3	S4

Table 6.1: Arduino Nano to BLHeli 4in1 ESC connections

These connections are not restrictive, but the ESC config file i saved to reproduce exactly these procedure respects this order. After connecting all the PWM pins, connect the battery to the ESC, then in the *SiLabs ESC Setup* click on Read Setup, on the bottom left of the window. Under *Multiple ESC* Number 1234 must appear, if not, some esc is not recognized. Maybe reconnecting the PWM cables from the ESC to the Arduino could solve the problem, or replacing it with other ones, if nothing solve unrecognition of one or more ESC could be a problem in the communication part or directly on the BB2 chip, malfunctioning or burned this was not the case. Once the 4 ESCs have been read successfully, the actual setup can be seen on the *ESC overview* tab. Another thing is to set the Correct rotation of the motors, 2 must rotate clockwise and 2 anticlockwise, in my motor configuration, the ESC to be set in inverse rotation are 2 and 4 ones, corresponding in the PX4 generic multirotor frame configuration to motors to the 3 and 4 as in the figure 6.4

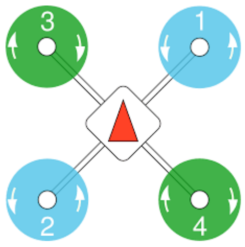
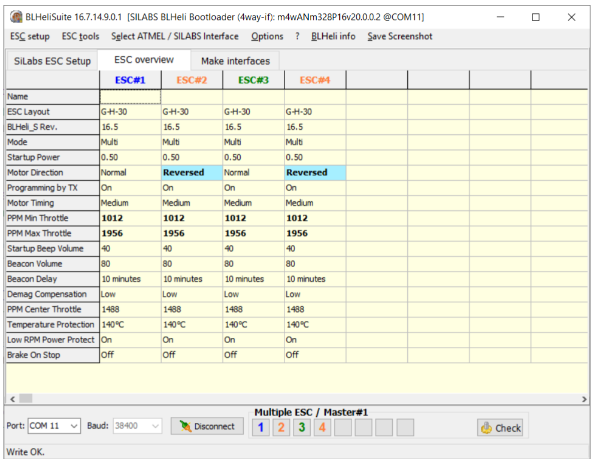


Figure 6.4: QGC Generic Quadrotor X frame configuration

The final configuration is the one in figure 6.5.



	ESC#1	ESC#2	ESC#3	ESC#4
Name				
ESC Layout	G-H-30	G-H-30	G-H-30	G-H-30
BLHeli_S Rev.	16.5	16.5	16.5	16.5
Mode	Multi	Multi	Multi	Multi
Startup Power	0.50	0.50	0.50	0.50
Motor Direction	Normal	**Reversed**	Normal	**Reversed**
Programming by TX	On	On	On	On
Motor Timing	Medium	Medium	Medium	Medium
PPM Min Throttle	**1012**	**1012**	**1012**	**1012**
PPM Max Throttle	**1956**	**1956**	**1956**	**1956**
Startup Beep Volume	40	40	40	40
Beacon Volume	80	80	80	80
Beacon Delay	10 minutes	10 minutes	10 minutes	10 minutes
Demag Compensation	Low	Low	Low	Low
PPM Center Throttle	1488	1488	1488	1488
Temperature Protection	140°C	140°C	140°C	140°C
Low RPM Power Protect	On	On	On	On
Brake On Stop	Off	Off	Off	Off

Port: COM 11 Baud: 38400

Write OK.

FIGURE 6.5: BLHeli final configuration

After these settings have been completed correctly, connect the ESC to the PixRacer as follow:

Device	Pins			
PixRacer	1	2	3	4
BLHeli 4in1 ESC	S3	S1	S2	S4

TABLE 6.2: PixRacer to BLHeli 4in1 ESC connections

Now the motors must spin correctly according to the setup described, it's recommended to perform an ESC calibration inside the QGC desktop app, that will change the PPM Min/Max parameters of each motors with respect to the real performance of the single ESC (so if one is a little defected, the settings will be automatically set to keep up with the others).
After this procedure, unfortunately, the motor still doesn't work, so probably the Mosfet driver is burned. At this point, the procedure could be the replacement of the driver or buy a new ESC. The first solution is a little less expensive but the component is not so easy to find commercially (also datasheet found was only in chinese), and the procedure could burn other components on the ESC, so the decision was to buy another one. The new component arrived, again do not worked as it is, so was again attached to the Arduino Nano and to the PC for other configurations. After trying the steps already done for the first ESC, searching deeply online, i found that could be the wrong setting of *PPM Min/Max throttle*, Min too high and Max too low, such that the Autopilot could not arm the motors. The configuration working for this case was 1012 as PPM Min value and 1956 as PPM Max, with these value the Autopilot can correctly arm the motors. After that, all the 4 motors worked smoothly and in the correct clockwise/anticlockwise direction, so the ESC troubleshooting was completed. From here on, fortunately, no other problems were encountered on this component.

6.2 Sensors

Several errors on all type of sensors have been detected on testing the hardware. Errors on the IMU sensors are divided into the three sub-unit composing these IC: gyroscope, accelerometer magnetometer.
Gyro sensor several time gives the error *Preflight Fail: Accels inconsistent - Check Cal*, but as the message advice, a more accurate calibration of the sensor always solved the error, in fact gyroscope is the more stable of the three and after a good tuning works fine.
Accelerometer was more insidious, with the message *Critical: Preflight Fail: High Accelerometer Bias* appearing very often. Digging into the Firmware folder, this error is printed in ekf2Check.cpp where checks all the sensors used by the Extended Kalman Filter algorithm to estimate vehicle states in therms of position, velocity, IMU estimations. Here the "pass" variable is initially set to true, then changed into false if at least one test fails. Accelerometer uncertainty is evaluated by a computation of a test_uncertainty, then added to the $COM\_ARM\_EKF\_AB$ which is the "maximum value of EKF accelerometer delta velocity bias estimate that will allow arming"[8], and this parameter must be less than $EKF2\_ABL\_LIM * FILTER\_UPDATE\_PERIOD\_MS * 0.001$. In this case, the solution without changing internally the Kalman filter and the control was to high the value to the maximum accepted by the autopilot, so change from 0.4 to 0.8 the $EKF2\_ABL\_LIM$ value. With this operation the problem did show in next flights.
Also Compass sensor result in several different Bias and Drift errors. As for the previous case, searching in the px4 stack, the COM_ARM_MAG variable was responsible of the preflight check failed, so the same approach was used, taking to 2.5 Gauss the maximum inconsistency magnetic field between different units. This problem was found also in other build with the same autopilot on the PX4 forum, and the only solution working was the one already tried. Since magnetometer is fundamental to complete the GPS position with the direction of the drone orientation with respect to the north, other test were conducted. First thing was to use only the external magnetometer sitting on the GPS module, procedure recommended in case of high magnetic interferences coming from ESC compartment, but results in a very weak control, so it was discarded for the final setup.

6.3 GPS

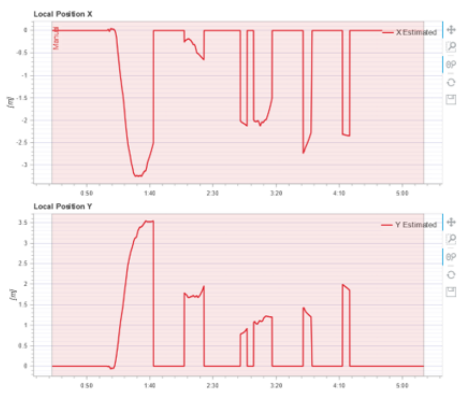
The Global Positioning System module gives other many problems. three different GPS modules have been tested:

1. mRo GPS u-Blox Neo-M8N Dual Compass LIS3MDL + IST8310, the module used in previous build.

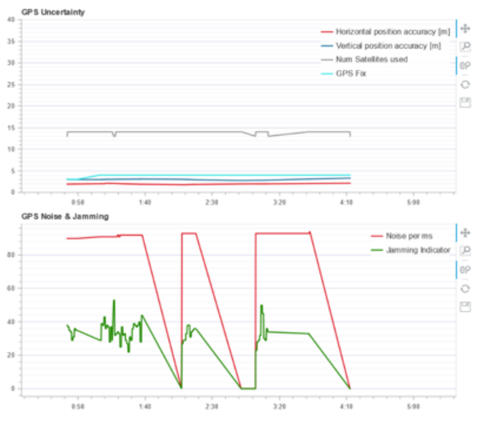
2. mRo uGPS ublox SAM M8Q + Compass LIS3MDL, the smaller module listed for the new build.

3. 3DR uBlox GPS with compass kit, another GPS available in laboratory.

The three candidates have been tested in the order listed above because of the availability, the second arrived very late and the last one was already used with other drones configuration and always worked, so taken as reference. Every preliminary flight test have been done in the PoliTO drone facility, a 14 x 8 x 10 meters safety cage, to test the correct vehicle positioning on the QGC interface, then all the real flight tests have been done in an affiliated flight field out of Turin. First module has been attached directly to the drone and tested as it is. No correct flight have been done with this module, every time it presented errors in the GPS Vertical and Horizontal accuracy. Lot of different stuff have been tested, changing connections to the autopilot, position the GNSSmodule in other position on the frame, shield the bottom of the module with grounded aluminium foil, change parameters inside the autopilot, tried with other autopilots, nothing worked. The problem is still unknown, especially because the unit works correctly on u-center, with good satellites count and signal to noise ratio.
Then the selected GPS module arrived, so the first one was replaced. This initially, the GPS was plug and play, once connected it worked immediately on the facility, GNSS position was precise in QGroundControl. So the quadcopter was calibrated and moved to the flight field to plan and run a mission: the result was a crash. Analyzing log and inspecting the drone hardware, the problem was a motor MOSFET burned in-flight, so not a positioning problem, but the drone could no longer fly until the arrival of the new ESC, about 3 weeks later. Then, again taken to the flight field, a mission with 2 waypoints was planned, drone takes off, reaches first one correctly, then started to jitter on the throttle command, the drone result going up and down about every second, after 4-5 of these maneuver i change the drone mode into manual and landed the vehicle to avoid other crashes.



(A) Chessboard reconstruction

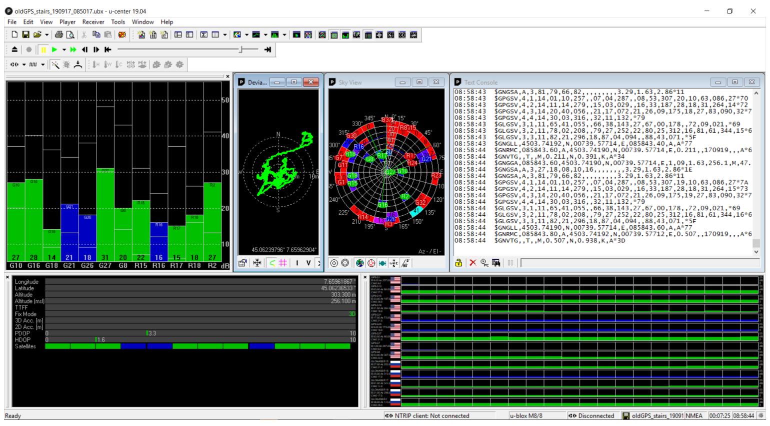


(B) Structure reconstruction

FIGURE 6.6: Holes in GNSS position

Log files were analyzed and the cause was a dropout on the GPS position. This condition continued to appear also in successive tests, apparently just after reaching a waypoint, when the drone turn and start pitching to gain linear velocity to move towards next target, so the initial diagnosis was a drop on the GPS power supply causing the shutdown of the peripheral. This behavior was not confirmed by the log file, where no high current peaks or important lowering of the 5V voltage were present in the datas. Another test to prove that this was not the cause of the lost of positioning, the on-board camera was placed on the drone pointing directly on the GPS with the led indicating the correct functioning of the unit clearly visible. Then several hold flight tests were performed on the cage, the led always continues to flash at the correct frequency indicating that the system runs ok, but once in a while holes in the GPS position plot continue to appear (see Figure 6.6), also in hold mode where all the actuators return gentle waveforms, with no impulses on the throttle command and peak currents required. The power supply hypothesis was rejected.
At this point, third and last GPS module was mounted and tested, after a good cold start and sensors calibration, it immediately worked smoothly, enabling to accomplish the first autonomous mission.

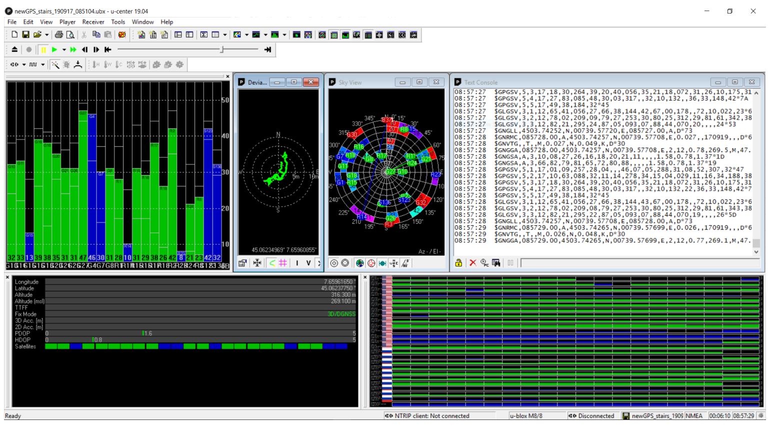
Since the goal of the thesis was the optimization of the quadcopter, and the working GPS module is older and heavier then the one selected for the new build, other investigations have been carried out, with a deep comparison of the three available modules using the u-center desktop application. In Figure 6.7 and 6.8 the basic interface of the software is shown, on this powerful program is possible to read and save .ubx proprietary log files, to plot all mind of statistics and interface with unit configurations. Modules were connected in the same conditions and a cold start was performed: this operation duration is about 30 min long, time used to completely read the almanac from satellites, which is the complete list of all the available satellites orbiting, then stored in the module memory to shorten the Time To First Fix at next power on of the unit. At a first look, the performances of the new small peripheral was better than the others, with a higher GPS counts and consequently lower Horizontal and Vertical Diluition Of Position.
Looking at Figures 6.7 and 6.8, the histogram in the left of the picture represents the signal power of each satellite seen from the GPS module, can be easily noticed that the new one see lot more satellites and also with a better signal to noise ratio. The screen of the third one is not reported since is very similar to 6.7. Searching both on datasheets of the two antennas, Ublox modules and online, the factory configuration of the three different units were found, then validated in the configuration window inside ucenter, the values of interest are listed in Table 6.3



```
08:58:43  $GNGSA,A,3,81,79,66,82,,,,,,,,,3.29,1.63,2.86*11
08:58:43  $GPGSV,4,1,14,01,10,257,,07,04,287,,08,53,307,20,10,63,086,27*70
08:58:43  $GPGSV,4,2,14,11,14,279,,15,03,029,,16,33,187,28,18,31,264,14*72
08:58:43  $GPGSV,4,3,14,20,40,056,,21,17,072,21,26,09,175,18,27,83,090,32*7
08:58:43  $GPGSV,4,4,14,30,03,316,,32,11,132,*79
08:58:43  $GLGSV,3,1,11,65,41,055,,66,38,143,27,67,00,178,,72,09,021,*69
08:58:43  $GLGSV,3,2,11,78,02,208,,79,27,252,22,80,25,312,16,81,61,344,15*6
08:58:43  $GLGSV,3,3,11,82,21,296,18,87,04,094,,88,43,071,*5F
08:58:43  $GNGLL,4503.74192,N,00739.57714,E,085843.40,A,A*77
08:58:44  $GNRMC,085843.60,A,4503.74190,N,00739.57714,E,0.211,,170919,,,A*6
08:58:44  $GNVTG,,T,,M,0.211,N,0.391,K,A*34
08:58:44  $GNGGA,085843.60,4503.74190,N,00739.57714,E,1,09,1.63,256.1,M,47.
08:58:44  $GNGSA,A,3,27,18,08,10,16,,,,,,,,3.29,1.63,2.86*1E
08:58:44  $GNGSA,A,3,81,79,66,82,,,,,,,,,3.29,1.63,2.86*11
08:58:44  $GPGSV,4,1,14,01,10,257,,07,04,287,,08,53,307,19,10,63,086,27*7A
08:58:44  $GPGSV,4,2,14,11,14,279,,15,03,029,,16,33,187,28,18,31,264,15*73
08:58:44  $GPGSV,4,3,14,20,40,056,,21,17,072,21,26,09,175,19,27,83,090,32*7
08:58:44  $GPGSV,4,4,14,30,03,316,,32,11,132,*79
08:58:44  $GLGSV,3,1,11,65,41,055,,66,38,143,27,67,00,178,,72,09,021,*69
08:58:44  $GLGSV,3,2,11,78,02,208,,79,27,252,22,80,25,312,16,81,61,344,16*6
08:58:44  $GLGSV,3,3,11,82,21,296,18,87,04,094,,88,43,071,*5F
08:58:44  $GNGLL,4503.74190,N,00739.57714,E,085843.60,A,A*77
08:58:44  $GNRMC,085843.80,A,4503.74192,N,00739.57712,E,0.507,,170919,,,A*6
08:58:44  $GNVTG,,T,,M,0.507,N,0.938,K,A*3D
```

oldGPS_stairs_190917_085017.ubx - u-center 19.04
File Edit View Player Receiver Tools Window Help
Text Console
Sky View
Ready
NTRIP client: Not connected
u-blox M8/8
Disconnected
oldGPS_stairs_19091 NMEA
00:07:25 08:58:44

FIGURE 6.7: previous GPS u-center visualization



```
08:57:27  $GPGSV,5,3,17,18,30,264,39,20,40,056,35,21,18,072,31,26,10,175,31
08:57:27  $GPGSV,5,4,17,27,83,085,48,30,03,317,,32,10,132,,36,33,148,42*7A
08:57:27  $GPGSV,5,5,17,49,38,184,32*45
08:57:27  $GLGSV,3,1,12,65,41,056,27,66,38,144,42,67,00,178,,72,10,022,23*6
08:57:27  $GLGSV,3,2,12,78,02,209,09,79,27,253,30,80,25,312,29,81,61,342,38
08:57:27  $GLGSV,3,3,12,82,21,295,24,87,05,093,07,88,44,070,20,,,,24*53
08:57:27  $GNGLL,4503.74252,N,00739.57720,E,085727.00,A,D*73
08:57:28  $GNRMC,085728.00,A,4503.74257,N,00739.57708,E,0.027,,170919,,,D*6
08:57:28  $GNVTG,,T,,M,0.027,N,0.049,K,D*30
08:57:28  $GNGGA,085728.00,4503.74257,N,00739.57708,E,2,12,0.78,269.5,M,47.
08:57:28  $GNGSA,A,3,10,08,27,26,16,18,20,21,11,,,,1.58,0.78,1.37*1D
08:57:28  $GNGSA,A,3,66,82,79,81,65,72,80,88,,,,,1.58,0.78,1.37*19
08:57:28  $GPGSV,5,1,17,01,09,257,28,04,,,46,07,05,288,31,08,52,307,32*47
08:57:28  $GPGSV,5,2,17,10,63,088,32,11,14,278,34,15,04,029,11,16,34,188,38
08:57:28  $GPGSV,5,3,17,18,30,264,39,20,40,056,35,21,18,072,31,26,10,175,31
08:57:28  $GPGSV,5,4,17,27,83,085,48,30,03,317,,32,10,132,22,36,33,148,42*7
08:57:28  $GPGSV,5,5,17,49,38,184,32*45
08:57:28  $GLGSV,3,1,12,65,41,056,27,66,38,144,43,67,00,178,,72,10,022,23*6
08:57:28  $GLGSV,3,2,12,78,02,209,08,79,27,253,30,80,25,312,29,81,61,343,38
08:57:28  $GLGSV,3,3,12,82,21,295,22,87,05,093,07,88,44,070,19,,,,26*5D
08:57:28  $GNGLL,4503.74257,N,00739.57708,E,085728.00,A,D*73
08:57:29  $GNRMC,085729.00,A,4503.74265,N,00739.57699,E,0.026,,170919,,,D*6
08:57:29  $GNVTG,,T,,M,0.026,N,0.048,K,D*30
08:57:29  $GNGGA,085729.00,4503.74265,N,00739.57699,E,2,12,0.77,269.1,M,47.
```

newGPS_stairs_190917_085104.ubx - u-center 19.04
File Edit View Player Receiver Tools Window Help
Text Console
Sky View
Ready
NTRIP client: Not connected
u-blox M8/8
Disconnected
newGPS_stairs_1909 NMEA
00:06:10 08:57:29

FIGURE 6.8: tiny GPS u-center visualization

Module	Baudrate [bps]	Measurement Frequency [Hz]
uBlox Neo-M8N	115200	5
uBlox SAM M8Q	9600	1
uBlox 3DR GPS	34800	3

TABLE 6.3: GPS modules config comparison

As can be clearly seen on the table, the most usable GPS found is the one with middle characteristics in therms of both baudrate and measurement period, and that the new tiny GPS uses by default very slow serial and GPS cycle, and this could be the cause of the malfunction of the position on the autopilot. For this reason, these two parameters were changed into 34800 bit per second for the baudrate and to 3Hz for the cycle on the new unit, and after another cold start, checking on u-center, the signals are acquired correctly. Just to confirm that, I tried to higher the measurement frequency up to 10Hz with the transmission rate untouched, resulting in one good sample followed by two void ones, because the data rate on the serial port can't keep up with the fast acquisition of the GPS signals. With this tiny GPS module configured, the drone flight missions perfectly, so the troubleshooting is considered completed.

Chapter 7

Comparison between builds

7.1 Frame

As said in chapters 3 and 4, the second frame differ from the first one and result in a very similar shape with a completely different approach. In next sub-chapters the main aspects of the 2 designs will be explained and all the points of strengths and weaknesses highlighted, starting from construction Material, to shape and finally weight.

7.1.1 Material

The first aspect considered is the strength of the materials to produce the frame. Previous build is based on carbon fiber, a very common one in mechanical lightweight projects in all kind of applications, from aerospace to bikes, from automotive to wind turbine blades. For this reason, all the mechanical properties of these material are very well known (reported in Table 7.1a) and tested, also if changing the manifacturing methods these can vary but only in specific application. In particular drone frames are made of standard thicknesses, in this particular application the carbon fiber sheet needed is thinner for mass reduction. For these reasons, a non-standard cut of the frame is necessary, and this is reflected directly into a price increase of the manufacturing cost of the hardware piece, with an estimated bill of around 100€ considering materials and handwork to the main frame and the flanges (3 different shapes for a total of 7 pieces) and the precise bonding of the 6 flanges.
The new approach is contrarily more modular, easy to reproduce and cheaper. Starting from the Printed Circuit Board, with just the gerber files the same exact design can be produced by any manufacturer, in our case using pcbway.com the cost result in just 80€ for 5 frames. The supports then are just 3D printed with 21 grams of ABS that can be buy for around 20€ 1kg, so the material cost is around 50 cents, with a total of less than 20€, and more importantly the plastic frame can very easily replaced since 3D printers are now very common in prototyping, even by hobbyist or in small factories. According to the data in tables 7.1 the new built must be a lot weaker than the carbon fiber one, but the strength of the composite came from the union of glued with the ABS

using epoxy, the same resin used in PCB manufacture. This multimaterial architecture result at least at a first inspection very similar in therms of rigidity compared with the previous one, but no mechanical bench tests have been made apart from flight tests.

Carbon Fiber mechanical properties		
Density	1.6	[g/cm^3]
Young's modules	255-500	[GPa]
Tangent Modulus	96-192	[GPa]
Yeld Stress	200	[MPa]
Tensile Stress	1600	[MPa]

(A)

FR-4 mechanical properties		
Density	1.85	[g/cm^3]
Young's modules	24	[GPa]
Tangent Modulus	10.5	[GPa]
Yeld Stress	65	[MPa]
Tensile Stress	70	[MPa]

(B)

PLA mechanical properties		
Density	1.2–1.4	[g/cm^3]
Young's modules	3.75	[GPa]
Yeld Stress	70	[MPa]
Tensile Stress	59	[MPa]

(C)

ABS mechanical properties		
Density	1.2	[g/cm^3]
Young's modules	2.5-4.1	[GPa]
Yeld Stress	115	[MPa]
Tensile Stress	70	[MPa]

(D)

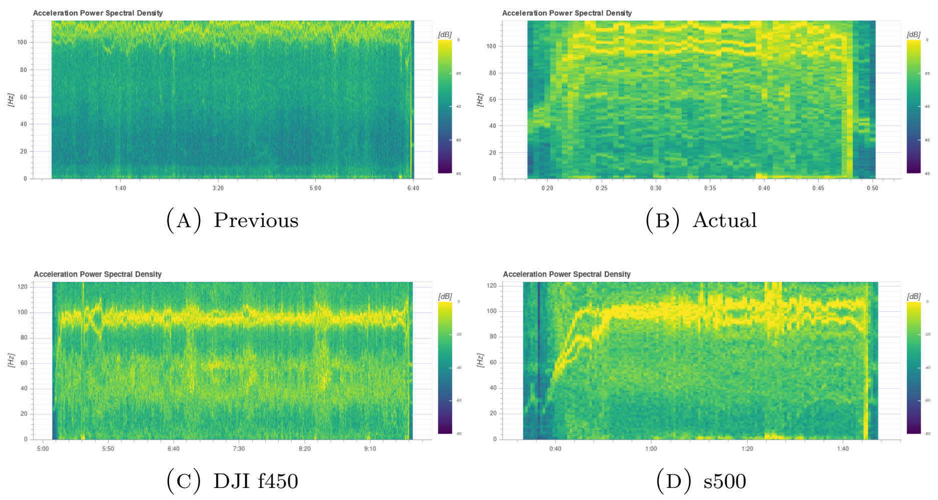
TABLE 7.1: Frame materials mechanical properties

The most important aspect of this design, is the possibility to directly integrate in the Printed Circuit Board several functional units of the quadcopter, like the power managing module with buck-boost converter and current sensing, as long with all the Electronic Speed Control electronics. Then other hypotheses have also been endorsed, like the integration of wireless positioning system Integrated Circuits, sensors very useful for communication inter-vehicle to exchange telemetry and relative positions also in view of swarm flight or cooperating with other Unmanned Ground Vehicles. Patch antennas properly sized for the applications can also sit on the PCB design, both for WiFi or analog video transmission.

7.1.2 Vibration analysis

The other important aspect in quadcopter frames are the evaluation of the vibrations perceived from the sensors, result of the spins of the motors at a certain frequency and the resonance of the while structure under these stresses. As can be seen in Figure 7.1 the spectrum of the accelerations is similar, with peaks in three bands near 100Hz and some resonant ones coming down the frequency axes, very similar to the DJI F450 frame spectra, considered with good vibration by the PX4 team itself.
This result are really important and give the confirm that this type of approach to frame design is compliant with the static and dynamic mechanical stresses of an ultralight

drone. More mechanical analysis specially dynamic in frequency domain will be very helpful in future to to further reduce the noises in accelerations.



(A) Previous (B) Actual

(C) DJI f450 (D) s500

FIGURE 7.1: Vibration comparison

7.2 Dimensions

One other important aspect in ultralight quadcopters is the dimension of the vehicle, for weight reduction but also to make the drone look more innocent specially in the perspective of swarm flights where at least 2 drones will fly together in a fleet. for this reason the initial shape was shrinked down by 3 cm in width and 2cm in length with a resulting 210x180 mm frame dimension, 320x350 mm of maximum footprint considering also propellers. In Figure7.2 can clearly seen that the new prototype is more streamlined in particular on the central section.

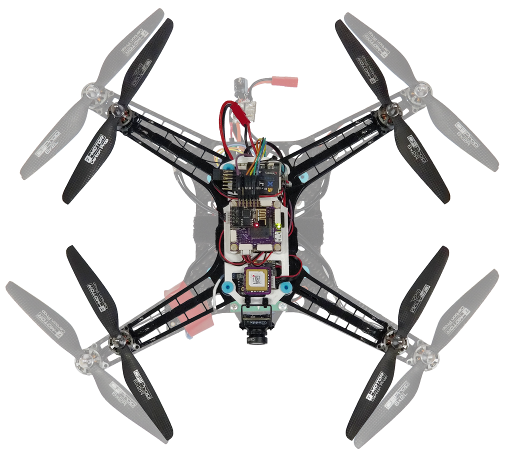
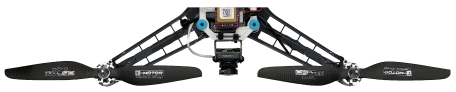
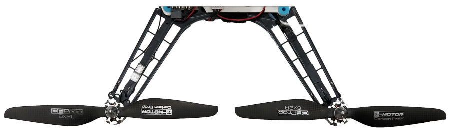


FIGURE 7.2: Dimensions comparison

As can be clearly seen from Figure 7.3 there's not other space left for optimization since the propellers are really near each other in the lateral view and really near the camera on the frontal one, in fact the preferred mode for shooting is with the camera tilted at least 45° degree down to remove the rotating propellers from the video.



(A) Front



(B) Lateral

Figure 7.3: Propeller spacing

7.3 Weight

The Table 7.2 contain the part lists of the two builds, highlighting in blue the elements where a weight optimization has been optained, in red the previous weight of the optimized hardware. For the GPS module and camera system the load reduction is made by the choice of lightweight hardware more adapt for the small quadcopter, with the Electronic Speed Control and Power module the weight reduction came from the removal of power cables connecting the two units and three-phase wirings from ESC to the motors.

	New build		**Previous build**	
Component	quantity	weight [g]	quantity	weight [g]
Autopilot	1	11.6	1	11.6
ESP8266	1	1.5	1	1.5
GPS module	1	16.6	1	8
Power Module ACSP5	1	5	1	2.3
Capacitor	1	1.6	1	1.6
Motors + washers and screws	4	53.2	4	53.2
Carbon Fiber propellers	4	7.2	4	7.2
ESC 4 in 1	1	6.8	1	4.8
Telemetry rx + cable	1	7.2	1	7.2
Battery LiPo 2S	1	43.1	1	43.1
Camera system	1	19	1	7.1
JST 6 pin 16 cm	2	14.2	2	14.2
Rubber damper	4	2.2	4	2.2
Battery strap	1	1	1	1
PCB	0	0	1	13.7
3D support	0	0	1	23
Epoxy	0	0	1	7
Upperframe (PLA)	1	7.1	1	8
Carbon Fiber Frame	1	39.7	0	0
TOTAL WEIGHT		249		216.7

TABLE 7.2: Weight comparison

The decreasing of the total mass in this application is a key point to customization since extra payload can be carry, with other kind of peripheral/sensors or batteries with more mAh, the grams of surplus leave space to work on different applications but always using the same base architecture.
Some of the idea for future development are:

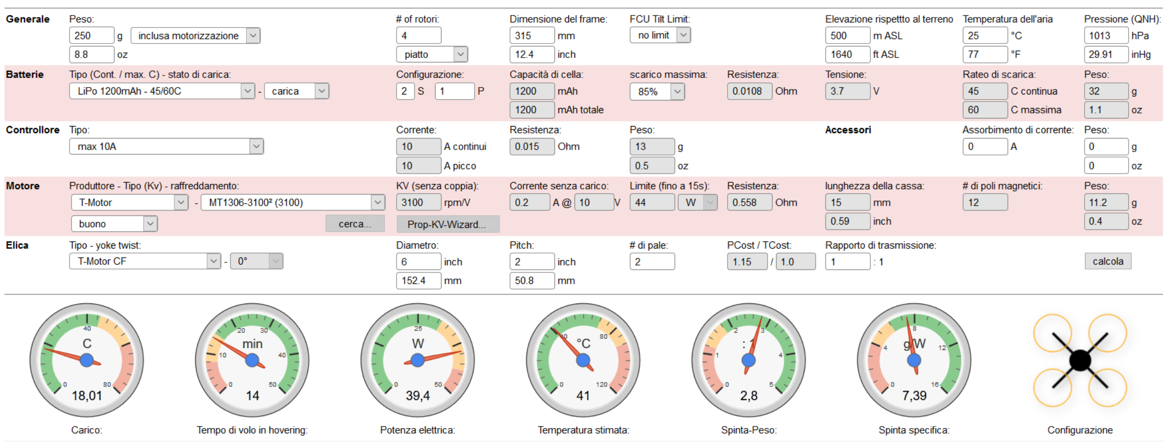
- Battery upgrade: increasing TOF (see Section 7.4)
- Camera pan/tilt: some tiny servomotors with a total weight of just 2 grams each have been buy to have the possibility to remotely pan and tilt the camera and to stabilize the video on 2 axes
- Ultra Wide Band integration: these wireless technology can be incorporated in the design allowing intra-vehicle relative positioning.
- Ultrasonic/laser distance sensing, for obstacle avoidance or precise height measure.

7.4 Time Of Flight

Depending on application, TOF can be relevant. The extra 34 grams to reach 250 limit can be spent in more battery capacity, for example with a 1300mAh 2S LiPo instead of the actual 800mAh one, resulting in a longer flight time, that can be computed as follow:

$$TOF = \frac{\text{Battery Capacity * Discharge}}{\text{Current Drain}}$$

Since the correct evaluation of the flight time is more complex than the formula above and there are tools to compute it with all the need data on motors efficiency and propeller profiles, i used eCalc[**ecalc**] tool available online. In fact, this tool was proved to be reliable by other researcher at PIC4SeR. Inserting all the needed datas as shown in Figure , with a weight of 250 grams so more than the actual load, the result is 14 min flight time (in the second tachometer from the right), almost double with respect to actual one wich is around 6-7 effective minutes.



Generale — Peso: 250 g, inclusa motorizzazione; 8.8 oz — # of rotori: 4, piatto — Dimensione del frame: 315 mm; 12.4 inch — FCU Tilt Limit: no limit — Elevazione rispettto al terreno: 500 m ASL; 1640 ft ASL — Temperatura dell'aria: 25 °C; 77 °F — Pressione (QNH): 1013 hPa; 29.91 inHg

Batterie — Tipo (Cont. / max. C) - stato di carica: LiPo 1200mAh - 45/60C - carica — Configurazione: 2 S 1 P — Capacità di cella: 1200 mAh; 1200 mAh totale — scarico massima: 85% — Resistenza: 0.0108 Ohm — Tensione: 3.7 V — Rateo di scarica: 45 C continua; 60 C massima — Peso: 32 g; 1.1 oz

Controllore — Tipo: max 10A — Corrente: 10 A continui; 10 A picco — Resistenza: 0.015 Ohm — Peso: 13 g; 0.5 oz — **Accessori** — Assorbimento di corrente: 0 A — Peso: 0 g; 0 oz

Motore — Produttore - Tipo (Kv) - raffreddamento: T-Motor - MT1306-3100² (3100); buono — KV (senza coppia): 3100 rpm/V — Corrente senza carico: 0.2 A @ 10 V — Limite (fino a 15s): 44 W — Resistenza: 0.558 Ohm — lunghezza della cassa: 15 mm; 0.59 inch — # di poli magnetici: 12 — Peso: 11.2 g; 0.4 oz

Elica — Tipo - yoke twist: T-Motor CF - 0° — Diametro: 6 inch; 152.4 mm — Pitch: 2 inch; 50.8 mm — # di pale: 2 — PCost / TCost: 1.15 / 1.0 — Rapporto di trasmissione: 1 : 1

Carico: 18,01 C
Tempo di volo in hovering: 14 min
Potenza elettrica: 39,4 W
Temperatura stimata: 41 °C
Spinta-Peso: 2,8 :1
Spinta specifica: 7,39 g/W
Configurazione

FIGURE 7.4: eCalc TOF evaluation

7.5 Image capture

For the camera section, the switch from analog to digital is a step back for real time visualization, but the video quality increase a lot. In Figure 7.5 a comparison between 700TVL analog video capture and 1080p HD digital video, that are the 2 most common resolutions for FVP applications in drone, the car on the right has more details than the left one which seems more washed instead, this can be clearly noticed on the billboard's writings. Analog signal than have to be saved directly on the drone to eliminate the noise due to the transmission of the signal from the vehicle to the receiver, noises that increase proportionally to the transmitting distance between antennas and the presence of external disturbs such as barriers of every genres and materials, in particular armed walls or every kind of metal structures.

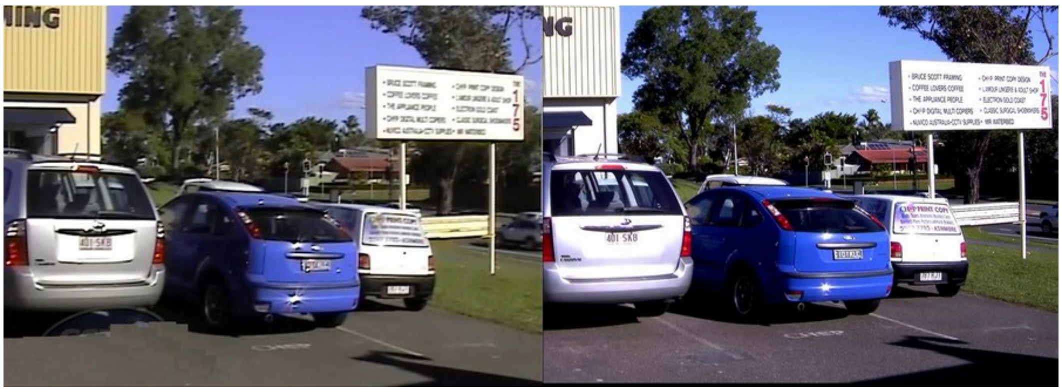


FIGURE 7.5: 700TVL analog vs 1080p digital image capture

This because the analog video is a sum of different values such as chrominance and luminance, read in horizontal lines at a certain frequency, and then modulated in a continuous analog waveform, which have a maximum operating slope, this is the main cause of the "washed" resulting view. Digital video saved from the Caddx Turtle V2 camera instead is the representation in 1920*1080 pixel of the image in .mp4 format, very easily manageable, compressed, elaborated.
The video file open also the possibility to extract frames as .jpg images and use them for the most various application, from image recognition, to 3D reconstruction of buildings and structures and many other. The digital recorder module chose, if need, have also some dedicated pins for composite video signals, that can set as input of a Video Transmitter with his own antenna to also have the analog live video stream on the Ground station, this solution is a trade-off, since the VTX and antenna will add around 10 grams to the build. In general, the real-time video stream is necessary only in scenarios where celerity is a big actor, such as emergencies or when the drone have to be remotely piloted by an operator which have to see where the quadcopter is going, in all the other automated mission, is ok to just record the entire flight and post-process it in a second time.

Chapter 8

Conclusion and future development

The thesis work is considered concluded, as listed in 7 many optimizations have been taken onto the new design both from hardware and software point of wiew, leaving a good prototype ready to work on various applications, some that can be accomplished immediately as mission flights and image capture.
Many ideas for future development came to my mind thanks to researches and to the stimulating ambient that can be breath at PIC4SeR with the contamination from other projects developed in this interdepartmental research center, here some ideas:

- **Swarm flight**: starting building other quadcopters with the same structure, work on the iteration intra-drone and the ground station with WiFi link.
- **UWB integration**: Ultra Wide Band module can be integrated in the in the PCB for the relative positioning of the drone fleet, also with UGV working on the ground for example in a vineyard.
- **Video stream integration**: if needed, real time video stream can be integrated in the design thanks to the extra payload gained with the new frame design, analog video trough standard VTX or digital frames via WiFi using another camera found on the market called Caddx Tarsier.
- **Camera stabilization**: some very little servomotor has been already selected for its weight of just 2.2 grams, the tilt operation can be very handy specially for structure scan, with a second servo compensating the roll axes could be used for image stabilization on 2 axes resulting in a more qualitative video, more stable and so also less blurred.
- **Electronic design on PCB**: if the design will require optimizations on the Power management system (DC-DC and current sensing) or on the Electronic Speed Controls, the new custom design could me integrated in the PCB frame directly, and this will also cut the costs of this two components, collapsing into a frame/power/ESCs 3-in-1 solution.

With these thesis, the new ultra lightweight autonomous drone prototype of the PIC4SeR was born, a 216 grams quadcopter capable of automatic flights, fully modular with open off-the-shelf components.
It's name is **PIC**COLO.

Bibliography

[1] S. Jung et al. "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning". In: *IEEE Robotics and Automation Letters* 3.3 (July 2018), pp. 2539–2544. ISSN: 2377-3774. DOI: 10.1109/LRA.2018.2808368.

[2] MA Dinesh et al. "Development of an Autonomous Drone for Surveillance Application". In: (2018).

[3] Joseph Helland et al. "Autonomous Drone". In: ().

[4] *Printed Circuit Board (PCB).* URL: https://www.techopedia.com/definition/2267/printed-circuit-board-pcb.

[5] *All About Multirotor Drone FPV Frames.* URL: https://www.getfpv.com/learn/new-to-fpv/all-about-multirotor-fpv-drone-frame/.

[6] *KiCad build with rounded tracks corners option.* URL: https://github.com/heikkipu/kicad-devel.

[7] *Hardware in the Loop Simulation (HITL).* URL: https://dev.px4.io/v1.9.0/en/simulation/hitl.html.

[8] *Parameter Reference.* URL: https://docs.px4.io/v1.9.0/en/advanced_config/parameter_reference.html.

[9] *Implementation of an Ultralight Autopilot Drone for Service Robotics.* URL: https://webthesis.biblio.polito.it/9538/.

[10] *FAA Aerospace Forecast.* URL: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2019-39_FAA_Aerospace_Forecast.pdf.

[11] *Own work.* URL: https://commons.wikimedia.org/w/index.php?curid=26640401.