BUILDING A MULTIROTOR AERIAL VEHICLE BASED ON THE PX4-PIXHAWK AUTOPILOT

Maria ZHEKOVA
University of Luxembourg
Email: maria.zhekova.001@student.uni.lu

Jose Luis SANCHEZ LOPEZ
University of Luxembourg
Email: joseluis.sanchezlopez@uni.lu

Abstract

Over the last few decades, a large variety of robots have been introduced to numerous applications, tasks and markets.
Nowadays, drones are becoming part of rapidly evolving industries that are producing applications that go far beyond military and government use. As autonomy and collision-avoidance technologies improve, so too will the ability of drones to perform increasingly complex tasks.
Proprietary drones come with a lot of advantages: very cheap, high-quality and well-integrated platforms. Nevertheless, they lack the versatility to add new hardware components and their autopilots are black boxes that cannot be accessed. These limitations suppose an important challenge when doing research and developing novel algorithms for aerial robotics.
In this paper, we present the realization of the production of drone with the use of a given template for the creation of the report [1].

1. Introduction

The technology behind quadcopters becomes more advanced and less expensive with every year. Drones can also fly autonomously through programmed waypoint navigation software and fly in any direction going from point to point. Over the last few decades the number of hobbyists who are building drones have been increasing and this task can be daunting, especially for those without a background in electronics. Nevertheless, with the right information and the right tools, building and flying a drone can be rewarding and exciting experience.
Just as a new invention could fundamentally rearchitect the ways cities work, drones have a disruptive potential that is hard to overstate. They could deliver food and medicine to people who need it, cast a watchful eye over anyone and everyone and bring the Internet to people who do not have. They could change the way people and goods are transported without having to create a new road through an arduous district and upend the way we think about distance.
The general idea of drones has been around for more than a century. They were primarily a military project, perfect for surveillance tools, small and nimble enough to avoid detection while flying over an enemy territory. Drone warfare has been hotly debated since its inception. It is both a technological debate and a moral one. With the passing years drone rose from a community of remote-control airplane fliers. The consumer drones became popular every day which accelerated the regulation in the skies of how users can fly drones and what these drones can do. In each country there are rules and as of 2020 in Europe, drone operators are required to register with national authorities. The rules will even replace existing national laws in EU member states and apply to professional operators and those flying drones for leisure [2].

2. Project description

The main objective of this Bachelor Semester Project is to design, develop, build and configure from scratch a multirotor aerial vehicle using a CUAV V5 nano and a Raspberry Pi.
The CUAV V5 nano is an open-hardware autopilot in collaboration with the PX4 team. The Raspberry Pi is a single board computer, it can run an OS (operation system) which can then run programs. To this drone we will have connected a camera and several ultrasonic sensors with which we will measure the distance

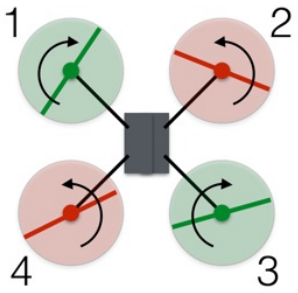
between the drone and the possible objects nearby.
Finally, the aerial platform will be evaluated with the performing of a real flight.

2.1.Domains

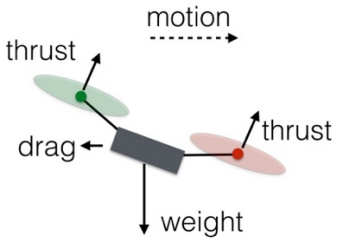
In this section the scientific and technical domain of this project will be explained.

2.1.1. Scientific

The scientific aspects covered by this project are to decide what kind of drone is wished to be built, to select the most suitable parts as frame, propellers, motors, ESCs (electronical speed controllers), battery, etc. with regard of the weight the drone can lift and its power. In addition, the capabilities of the drone will be increased with camera and several ultrasonic sensors. The camera will record a live video and it will be transmitted from the onboard computer to the user's computer via ROS (Robot Operating System), the ultrasonic sensors will be used to detect if there are nearby objects.
We won't provide many details about ROS, since it was a big part of our last project "Increasing the capabilities of multirotor aerial robots with the Raspberry Pi platform" [3].
Multirotor aircraft is controlled by adjusting the speed of the motor and in turn the speed at which the aircraft's propellers spin. It increases or decreases its height by increasing or decreasing motor speed. It flies forward by increasing the speed of the rear propellers, and vice versa, and goes left by running the right-hand propellers faster. Turns are achieved by taking advantage of the fact that two motors spin clockwise and two counterclockwise, and then precisely adjusting the speed of each individual motor.
In the figure on the right, the red rotors are rotating counterclockwise and the green rotors are rotating clockwise.



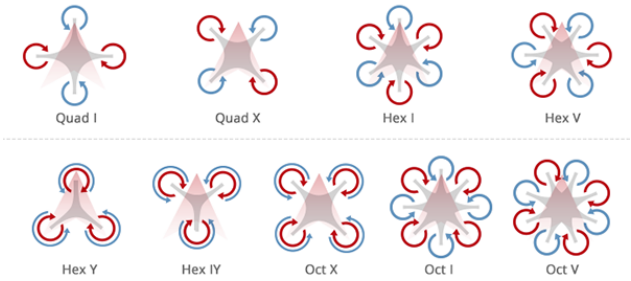
The process of flying starts with the aircraft's flight controller which receives the pilot's instructions, then translates them into digital commands which are sent to the ESCs. The ESCs interpret these instructions and increase or decrease the voltage to each individual motor.
It is important to understand the four forces of flight which are present while an object fly.
Lift – the force that takes off the flying object, pushing it up
Weight – the normal weight of any object, caused by gravity and pulling it down
Thrust – the force the flying object produces pointing in the direction of motion
Drag – the force acting on the flying object, pointing in the opposite direction of motion
These four forces combined make possible what we call flight.



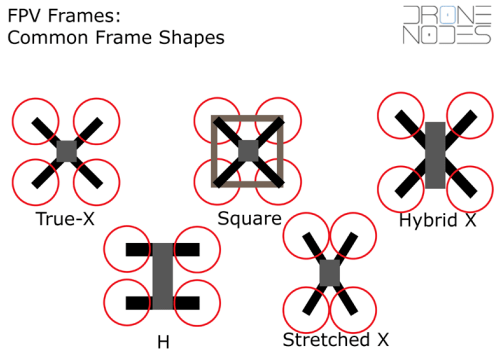
When the total thrust force remains equal to the weight, the drone stays at the same vertical level.

2.1.2. Technical

The technical aspects covered by this project are to solder if needed, to install and to configure the appropriate hardware components and to develop the required programs in order to interact with them. The onboard computer is the Raspberry Pi, it stays for the name of a series of single-board computers which are very suitable platforms for this kind of robots due to its reduced cost, small size and weight, and hardware interaction possibilities.
There are many different types of drones, based on the numbers of propellers, type of frame, size, flying range and equipment [4].



When a multirotor is designed with four blades, it becomes a quadcopter.
The quadcopter design is the most popular for several reasons, namely mechanical simplicity, quantity of motors and ESC's required for flight and their compact size. There are also multiple different styles of frame for a quadcopter.



2.2.Targeted Deliverables

2.2.1. Scientific deliverables

The Scientific deliverables of this project are several rapports and videos in which the performance and optimization are described. In order to proceed to the technical deliverables, it was crucial to understand the important components of the drone which will be explained in detail in section 4.

2.2.2. Technical deliverables

The targeted technical deliverables are to deliver a well-designed, well-built, well-configured, flying and well-equipped to remotely control drone using ROS. This drone will use as an onboard computer the Raspberry Pi platform which will be connected to the CUAV V5 nano autopilot. It will be possible to communicate with the onboard computer via ROS.

3. Pre-requisites

3.1. Scientific pre-requisites

As an important scientific knowledge, it is required to have at least a basic knowledge as a starting point of electronics in order to properly select and mount the required parts for building the drone (frame, propellers, motors, batteries, etc), considering the weight the drone could lift and the weight of the chosen parts.
It is important to have knowledge on digital and analog sensors and actuators to be able to connect them to the Raspberry Pi platform.
It is required the understanding of Linux system administration to configure and install the needed drivers. It is highly recommended to have knowledge of programming in order to develop the programs for the use of the components.

3.2. Technical pre-requisites

The technological aspects that are covered by this project consist in having knowledge of the Raspberry Pi platform and its use, in the implementation of a communication via ROS between the user's computer and the Raspberry Pi platform and the implementation of the circuit for the interfaces.
Since we will be working with Python, it was important to have knowledge of it.

4. A Scientific Deliverable 1

4.1.Requirements

The requirement that satisfies the scientific deliverable of this Bachelor Semester Project is to understand the important components of the drone in order to proceed to the technical part.

4.2.Design

Here we will explain our understandings for the different components for the build of a drone.
All drone parts and components are vital to a smooth and safe flight.

- **Frame**

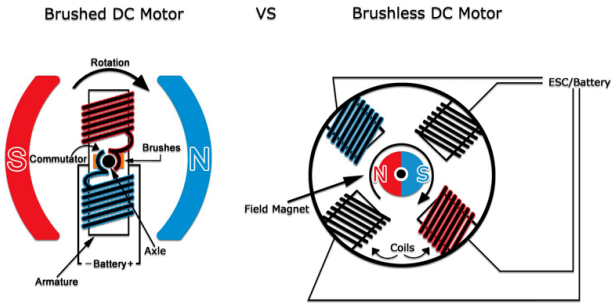
We chose a frame, cut from carbon fiber sheet. Carbon fiber is a composite material, made up of many layers of interwoven carbon fibers that have been rigidly cemented within a binding matrix of epoxy. The popularity of carbon fiber as a frame material is due to its low weight and high strength. It is important to know that carbon fiber is an electrically conductive material and must not have contact with circuits, therefore some of the parts need to be isolated.

- **ESC**

The ESC is an electronic circuit with the purpose to vary an electric motor's speed, its direction and possibly also to act as a dynamic brake. It coverts DC battery power into 3-phase AC for driving brushless motors.
The electronic speed controller is an essential component of modern multirotor, which offer high power, high frequency, high resolution 3-phase AC power to the motor in an extremely compact miniature package.

- **Motors**

Mainly there are two types of motors used in drones: brushed and brushless motors. The difference is the way they work. The most chosen one is the brushless motor because it is more powerful for its weight and it lasts longer comparing to the brushed motor. The brushed motor is more convenient to use on micro and nano drones.

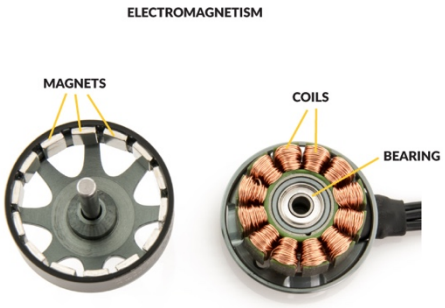


Both designs use an electromagnet, as a means of converting electrical energy into kinetic energy. When an electromagnet is electrically charged, a magnetic field is produced. This temporary magnetic field interacts with the one of the permanent magnets located within the motor. The combination of attraction and repulsion of the electromagnet or permanent magnets translates into rotational motion of the motor shaft.

Brushless motor lasts longer because there are no brushes to wear out, while the brushed motor wears out quickly.

For our project we will use the brushless motor.

The size of a brushless motor is identified by a four-digit code that details the dimensions of the stator in millimeters, for example: 2206. The first two numbers in the series determine the diameter of the stator, in this case, 22mm. The final two describe the height of the stator, the last two numbers in this series are “06” therefore the stator unit is 6mm tall.



Motors are arguably the most influential piece of equipment on a multirotor, having a considerable impact on flight characteristics relative to other components. It is essential that motors are carefully selected with adequate appropriateness for their application.

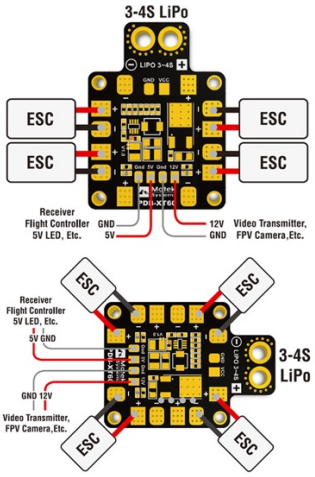
- **PDB (power distribution board)**

The PDB is a printed circuit board that is used to distribute the power from the flight battery to different components of the multirotor.

The amount of power that a voltage regulator on a PDB can supply is often limited by the number of the used components and it is very important not to exceed the maximum rated current otherwise the voltage regulator can overheat and will most likely be damaged. This is very dangerous as this can often happen while flying and the components connected to the voltage regulator will stop working, this can often lead to complete failure or loss of control.

On the following image it is displayed a PDB with both 5V and 12V UBEC on-board and a soldered on XT60 connector, very good for X-Shape FPV racers or normal H-Shape FPV racers. There are also displayed two ways to mount this board depending on the frame of the drone.

In our project we use the second one.



- **Battery**

The battery is the foundational component of a quadcopter and must be considerately selected to achieve balance between performance and flight time. Lithium batteries are the most common battery chemistry used to power quadcopters due to their high energy densities and high discharge capabilities.

Battery voltage is the potential energy difference between the positive and negative terminals. A standard lithium polymer cell has a storage voltage of 3.7V to increase the power that a single LiPO pack can deliver, these cells are grouped together in series to increase the overall battery pack voltage. LiPO packs are commonly sold in 1S, 2S, 3S, 4S, 5S or 6S configurations where the digit followed by the 'S' stands for number of cells in that specific pack. The more cells that are grouped together, the more voltage the overall battery pack will have.

The C-Rating of a battery is a unit of measurement dictating how much current a battery can continuously supply for its given charge cycle. Simply put, the higher the C-Rating of a battery, the more current the pack can continuously supply. The C-Rating can be multiplied by a batteries capacity in order to calculate a packs theoretical maximum discharge current.

With the following formula we can calculate the C-Rating:

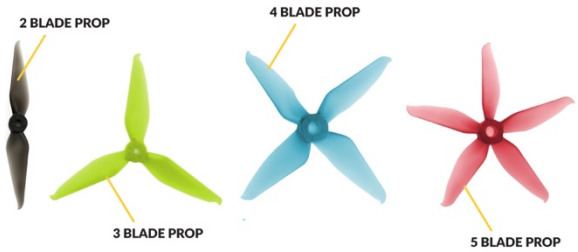
$$\frac{maximum\ safe\ current\ draw\ (mA)}{battery\ capacity\ (mAh)}$$

In Lithium batteries store large quantities of energy in a small profile and are occasionally prone to catching fire. This is a rare scenario and is most likely to occur when charging, discharging, or if a battery becomes damaged. Regardless of the odds, a LiPO fire can result in houses burning down, self-injury or damage to gear.Therefore, we must be very careful when using a LiPO battery.

- **Propellers**

For a drone, the propellers are the key component to keep it in the air. They have the most direct impact on how the multirotor flies but also the components that is most likely to be damaged and replaced often. They are commonly described using a series of number separated by a *"x"*. These numbers relate to size, pitch and number of blades. For instance:

4 x 4,5 x 3 stands for a 3-bladed propeller with the size of 4 inches and a pitch of 4,5 inches.



The pitch refers to the angle of each of the blades on the propeller.

High pitch will usually result more overall thrust and top end speed, but less low-end torque. Therefore, it will respond to an input slowly but will more power and will be efficient when the motor is moving very fast. The low pitch on other hand will result in less overall thrust and top end speed, but more low-end torque. It will respond quicker to input and use less power but will be efficient at less speed. When the wished type of the drone must change fast its direction, the low pitch is perfect for this.

The most common use is the average choice of pitch which is 4 or 4.5 inches [5].

- **Autopilot + GPS**

We are going to use the CUAV V5 nano autopilot.

The CUAV V5 nano supports both PX4 and Ardupilot, which are the two mainstream open source flight stacks to meet different users' needs and supports all types on unmanned vehicles as helicopter, multicopter, boat, VTOL and even a rover.

The PX4 is a project which provides a flexible set of tools for drone developers to share technologies to create tailored solutions for drone applications [6].

- **Remote controller**

A drone controller works by sending a radio signal from the remote control to the drone, which tells the drone what to do.
Radio signals are sent from the radio transmitter in the drone controller and received by the drone's receiver. This is why the drone controller is sometimes simply called the drone radio transmitter or the drone radio controller.
When building a drone, we must consider all the properties of the different elements. In the next section the flight time and the battery life will be explained and stated altogether with the chosen elements.

4.3. Production

Building a drone from scratch can be challenging, especially at the beginning, when it is important to choose the right elements and consider the connections between them.
We started with the idea to build the smallest possible drone which can carry the wished elements. The drone needed to have an autopilot, a camera, several sensors, and an onboard computer. It was crucial not to forget the components with which the drone fly (as its motors, propellers, battery, flight receiver, PDB and ESCs) in the calculations of the flight time, battery life, etc. We needed to have in mind also the circuit for the sensors that needed to be created and attached to the frame.
All things considered, the drone needed to be able to lift and fly with all the weight that was mounted on its frame. Therefore, several calculations needed to be done when choosing the components.
At the beginning it was difficult to find a starting point with the select of the parts, because every element in the drone is codependent and sometimes if even one of the elements is replaced, it may be necessary to change also the others. Our starting point was the dimension of the frame, since with a smaller frame we are limited at choosing elements of lightweight possible.
In this section we will discuss the chosen parts altogether with their dimensions, weight, endurance, payload, etc.

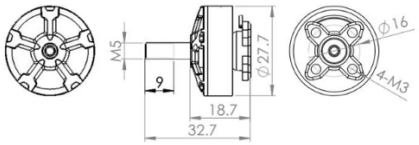
Here is a table with the main parts:

parts	weight (g)	description
Frame	141	250mm
Battery	320	4S 14.8V 3300maH 35C
Motor*4	124.6	2206 2700kV
Propellers	36	4045 x3
ESC*4	108	3-4S 30A 5V
Other components	~ 175.1	Autopilot & GPS + Receiver + PDB + Sensors + Raspi + cables…

The total weight of the drone was calculated to be 904,7g.
We will discuss the dimensions and properties of the parts.

- **Motors:**

The model of the motors is a brushless Samguk Wu 2206 with 2700kV 4S.



We will be concerned by the colored measures from the following table.

2700kv

	NO LOAD		ON LOAD				LOAD TYPE
VOLTAGE	CURRENT	SPEED	CURRENT	Pull	Power	EEP	Battery/prop
V	A	rpm	A	g	W	%	
12	1.4	32400	5.3	200	63.6	3.1	L1PoX3/4045X3
			12.9	400	154.8	2.6	
			24.8	630	297.6	2.1	
			4.8	250	57.6	4.3	L1PoX3/5040X3
			13	500	156.0	3.2	
			22.6	730	271.2	2.7	
			7.8	300	93.6	3.2	L1PoX3/5045X3
			16.5	500	198.0	2.5	
			31.0	700	372.0	1.9	
			5.2	250	62.4	4.0	L1PoX3/5040X4
			13.1	500	157.2	3.2	
			25.0	770	300.0	2.6	
			5.3	250	63.6	3.9	L1PoX3/5045
			12.6	500	151.2	3.3	
			20.6	700	247.2	2.8	
16	1.5	43200	7.2	300	115.2	2.6	L1PoX4/4045X3
			19.1	600	305.6	2.0	
			34.8	900	556.8	1.6	
			7.5	400	120.0	3.3	L1PoX4/5040X3
			20.6	800	329.6	2.4	
			33.1	1100	529.6	2.1	
			8.8	400	140.8	2.8	L1PoX4/5045X3
			27.8	800	444.8	1.8	
			43.2	1130	691.2	1.6	
			7.8	400	124.8	3.2	L1PoX4/5040X4
			22.5	800	360.0	2.2	
			35.2	1140	563.2	2.0	
			6.4	350	102.4	3.4	L1PoX4/5045
			15.8	700	252.8	2.8	
			29.9	1070	478.4	2.2	

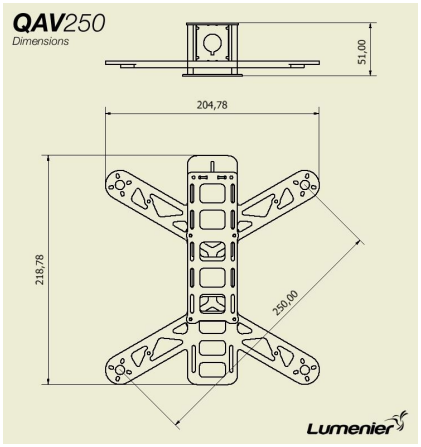
The drone will be designed for an average of 2kg.

- **ESC:**

The model of the ESC is ZTW Beatles 30A 2-4S Lipo with 5V/2A with a BEC output for brushless motors.
The dimensions are:
(L x W x H) → 43mm x 25mm x 10mm

- **Frame:**

The model of the frame is QAV250 carbon fiber with dimensions as on the following image.



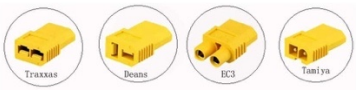
This frame is ordered and with all the received elements can be created the following frame.
The actual frame of this project is showed in section 5.

- **Propellers:**

The propellers are 3 bladed of size:
4045 – 4 x 4.5 x 3
4 inch → 101.6 mm

- **Battery:**

HRB 4S 3300mAh 14.8V 35C with XT60
(L x W x H) → 135mm x 42mm x 30mm



- **Remote controller and receiver**

The chosen model of a transmitter is FrSky Taranis X9D Plus 2019. This latest version uses the latest ACCESS communication protocol, which boasts 24 channels with a faster baud rate and lower latency with a high-speed module digital interface. It is compatible with ACCST D16 and ACCESS receivers.
The compatible receiver is a FrSky XM, Xm+ Plus

This receiver is extremely small, compared to others and weights only 1.6g
We will now discuss the calculations based on the elements before crucial for the success of the project. Most of the measures were calculated from several websites [7] [8].

Drone flight time:

```
time = capacity * discharge / AAD
```

ADD -> the average amp draw of the drone

```
AAD = AUW * P / V
```

AUW -> total weight of the drone

Battery capacity	3,300 mAh
Battery discharge	80 %
Battery voltage	14.8 V
All up weight (AUW)	904.7 g
Drone flight time	15.24 min

Thrust per motor:

Weight	
Drone weight	141 g
Battery weight	320 g
Equipment weight	443.7 g
Total weight	904.7 g
Thrust	
Power-to-weight ratio	2 :1
Number of motors	4
Total thrust	1,809.4 g
Thrust per motor	452.35 g

Regarding the propellers:

Prop Type : Choose "Custom" to enter your own values	Custom — Tk 0.9, Pk 0.82, Blades 3	
Prop Diameter :	inches 4	cm 10.2
Prop Pitch :	inches 4.5	cm 11.4
Prop Static RPM :	rev / minute 43200	
Supply Voltage & Current :	Volts 16	Amperes 1.5
Click to Calculate		
Estimated Static Thrust :	ounces 40.4	grams 1145
Supplied Power :	Horse Power 0.03	Watts 24
Prop's Absorbed Power :	Horse Power 0.92	Watts 682.6
Static Efficiency :	2844.2 %	
Static Pitch Speed :	mph 184.1	Km/h 296
Appx. Level Flight Speed :	mph 221	Km/h 356
Prop Tip MACH Speed :	Max recommended .92 — 0.676	

The **battery life**:

Battery capacity	3,300 mAh
Discharge safety	20 %
Device consumption	1.5 A
Runtime	
Battery life	105.6 min

These calculations may be different in the end because the exact weight of the drone could not be known before its development. Therefore, the results are rough and not exact. The best approach is to add additional weight into the calculations in order to avoid overloading the frame and difficulties for the lift.

4.4. Assessment

Since the requirement for the scientific deliverable was to understand the different components important for the functionality of a multirotor vehicle, the scientific deliverable satisfies fully the requirement.

5. A Technical Deliverable 1

5.1. Requirements

The technical requirements covered by this project are to develop and build from scratch a multirotor aerial vehicle with an open source autopilot on board, to install and configure the required components to interact with the necessary parts. Multiple sensors and the camera, which increase the capabilities of the drone in making it more interactive will be controlled with the Raspberry Pi platform.

For our project we need to create a compatible and functional circuit for all the interfaces we are using. In order to obtain a functional circuit, except a good wiring we need to create a script which will control the components.

Finally, we should be able to perform a flight with the developed drone.

5.2. Design

We needed to change the ordered frame in order to be able to put all the necessary parts.

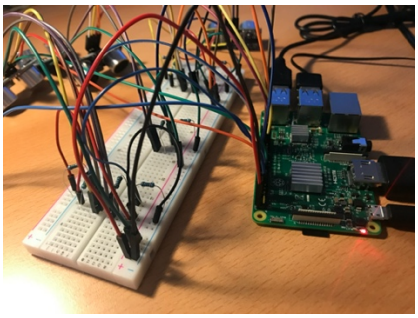
The final frame consists in three levels.

The onboard computer and the autopilot needed to be safe enough and that is the reason why they were situated on the top level. On the middle plate are situated the PDB and the circuit for the sensors. The battery is on the bottom plate.

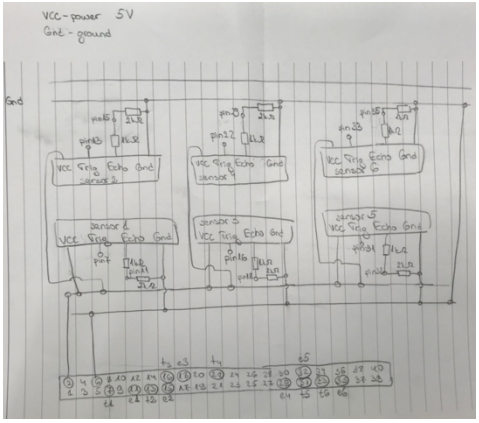
The first idea of the design was to have six sensors: one on the bottom, one on the top, two in the front – one on the left and one on the right, two in the back – one on the left and one on the right. The camera should be placed in the front.

The parts on the frame should be placed in the most symmetric way possible in order to avoid balance lost.

The circuit for the sensors needed to be designed as well. Based on knowledge from the previous Bachelor Semester Project a prototyped circuit on a breadboard was implemented.

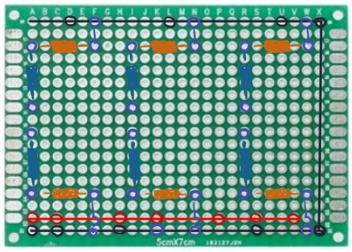


The circuit for the six sensors was based on the following created schema.



This implementation of a circuit was just a try in order to see how a connection of six sensors should work, later on a similar circuit was implemented on a solder board.

The blue rectangles are resistances of 1kΩ and the orange ones are resistances of 2kΩ. The circles are the cables that need to be soldered to the board. There are two horizontal black lines on which we see circles. They represent the Ground cables and the red line represents the Power(5V) cable (for the VCC on an ultrasonic sensor).



Since the distance from the sensors will be transmitted through ROS we needed to design the ROS workspace:

```
|-- catkin_ws (folder)
|--|-- build (folder) [created by catkin_make]
|--|--|-- ...
|--|-- devel (folder) [created by catkin_make]
|--|--|-- ...
|--|-- src (folder)
|--|-- |-- zhekova_bsp3 (folder)
|--|-- |-- |-- doc (folder)
|--|-- |-- |-- info.txt
|--|-- |-- |-- stack (folder)
|--|-- |-- |-- |-- raspicam_node (folder) [took from GitHub]
|--|-- |-- |-- |-- |-- ...
|--|-- |-- |-- |-- sensors_reader (folder)
|--|-- |-- |-- |-- |-- src (folder)
|--|-- |-- |-- |-- |-- |-- one_sensor_publisher.py
|--|-- |-- |-- |-- |-- |-- one_sensor_subscriber.py
|--|-- |-- |-- |-- |-- |-- six_sensors_publisher.py
|--|-- |-- |-- |-- |-- |-- six_sensors_subscriber.py
|--|-- |-- |-- |-- |-- |-- distance.py
|--|-- |-- |-- |-- |-- |-- distance.pyc
|--|-- |-- |-- |-- |-- CMakeLists.txt
|--|-- |-- |-- |-- |-- package.xml
|--|-- |-- |-- LICENSE
|--|-- |-- |-- README.md
|--|-- |-- CMakeLists.txt
```

The node for the camera was a node taken from GitHub [9].

We will be able to receive a live-time image from the drone on an on-ground computer altogether with the distance between an object and the drone. We will connect to the ROS master, which will be the IP of the Raspberry Pi, via *tmux*. This represents a terminal multicomplexer, which allows multiple terminal sessions to be accessed simultaneously in a single window. We will need to establish a SSH connection to the Raspberry Pi, the tmux is very useful for this purpose since it allows SSH sessions to remain active without being visible and it can run more than one command-line program at the same time.

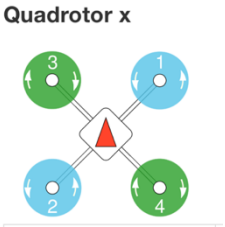
In order to connect the cables with eachother we used two types of connectors, depending on the purpose.

The cables from the ESCs to the PDB needed to be connected with a T-connector.

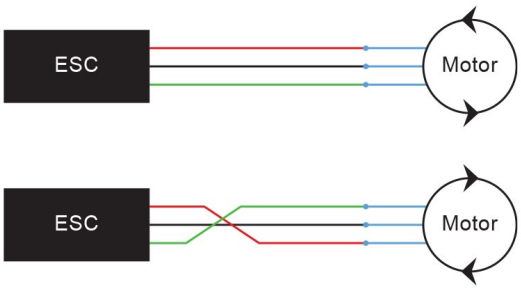
The motors needed to be connected with the other side of the ESCs with bullet connectors.

In order to be able to establish the connections it was important to learn how to solder.

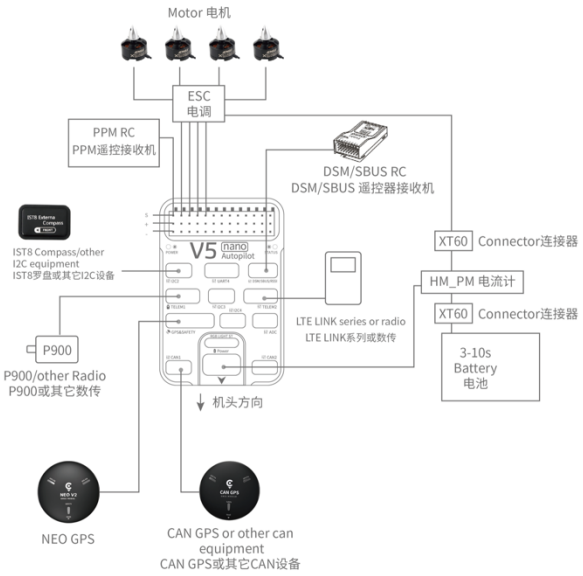
The frame is a quadrotor X type of frame and the motors are with the following direction.



As we saw before we need to establish a clockwise (CW) and counterclockwise (CCW) direction on the motor. For the CW motor the cable connection between the motor and the ESC is intuitive, the left connects to the left, the middle to the middle and the right to the right. For the CCW motor on the other hand the right and the left are inversed, but the middle stays the same.



The Raspberry Pi will be powered trough the micro-usb cable from the PDB and the autopilot needs to be connected to the other components as on the following schema [10].



As we see on the schema the power module between the battery and the PDB is also connected to the autopilot.

In our frame the first motor is the one on the right in the front, then the second is the one on the left in the back, the third the one on the left in the front and the fourth, the one on the right on the back.

Before placing the autopilot on the frame, we need to configure and calibrate it with QGroundControl. First, we need to update the Firmware and Airframe with selecting out type

of frame.

Then, to perform the mandatory setup calibrations of the sensor orientation, compass, accelerometer, level horizon calibration, radio setup and flight modes.

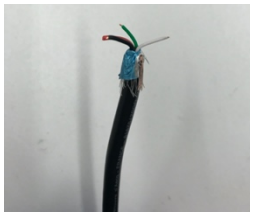
5.3.Production

The production of the technical part was the most challenging part of this project. We had few issues during the semester and the build of the quadcopter started in the last weeks. The biggest issue was the delay of receiving the motors. They were ordered three times from different websites, the first two orders ended with lost of the package on its way to Luxembourg, fortunately the last order was a success.

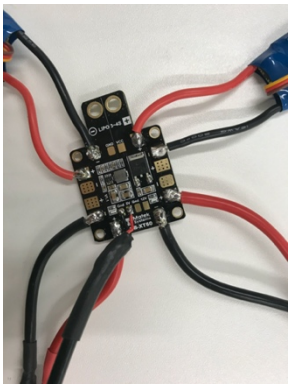
First, we will cover the build of the drone and then we will explain the production of the script for ROS.

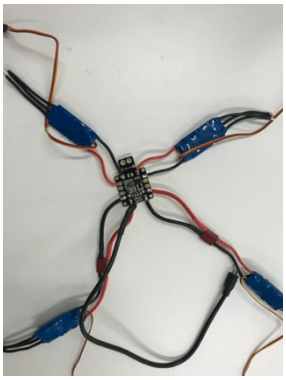
- **PDB, ESCs, micro usb**

The first parts to solder were the ESCs to the PDB. Keeping in mind the image form section [4.2] of the design of the PDB, the frame of this drone was small and the PDB was supposed to be situated on the back of the frame, we could connect only the front ESCs with a T-connector connection since it makes the cables difficult to twist. After soldering the cables, it was important to put heat shrink tubes so that the possibility of shortcut is decreased. To the breadboard was needed to be soldered also a cable to power the Raspberry Pi. The Raspberry Pi can be powered trough a micro-usb port. We cut the other side of a micro-usb cable. In this cable we find 4 more smaller cables, from which we will need only the red one (power 5V) and the black one (Ground). We used a heat shrink and we soldered the two wires to the destinated places on the PDB.



The final result of the soldered elements to the PDB gave us the following.





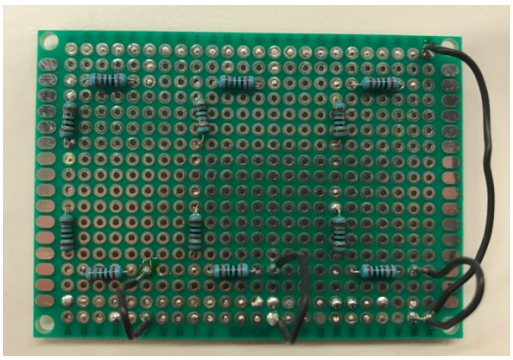
We needed to solder a XT60 connector to the PDB in order to be able to connect the battery.

- **Sensors circuit**

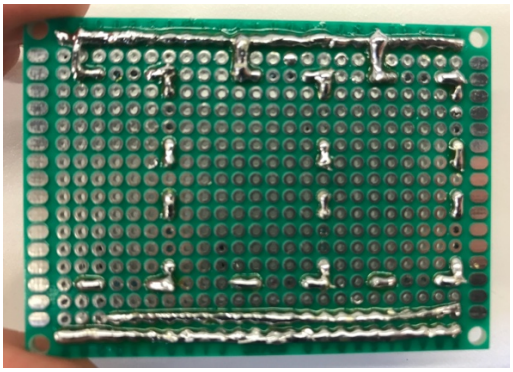
Next we needed to find a way to implement the prototyped circuit to a solder board circuit.

We used a universal 7cm-5cm solder board.

Since on a solder board is difficult to merge two holes only with solder (this property is preventing the user of getting an unwanted solder) we needed to improvise. We stripped a wire and soldered it on the board so that the holes on the three horizontal rows are connected to each other. Then we soldered the resistances and the cables for the connections on the board and we obtained the following.

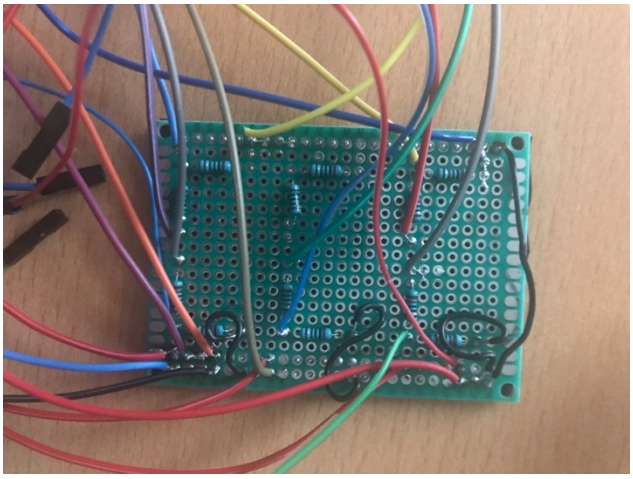


The back of the board looks like this.



The board is made with resistances for six sensors. The resistances are for the maximum number of sensors in case a desire for an improvement of the drone.

After the rest of the cables are added we obtained the following.

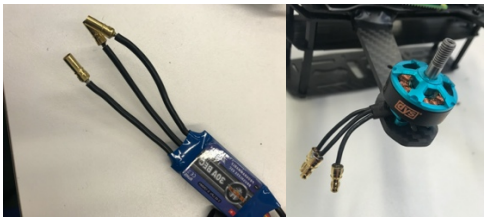


Since we are using only one sensor for this project the cables were put together and attached to the arms of the frame.

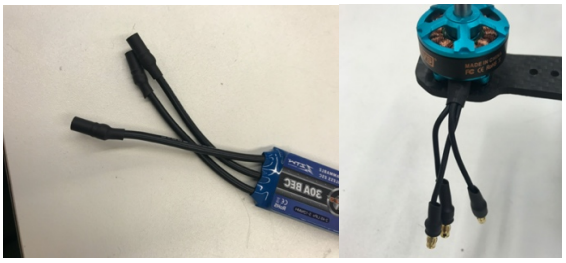
- **Motors and ESCs**

The final soldering part was the connections between the motors and the ESCs.

First, we needed to cut the wires of the motors, to solder male bullet connectors to the motors and female bullet connector to the ESCs. We must be careful not to inverse the type of connector because it can result in a creation of a shortcut, the female always goes on the active side and the male on the passive side.

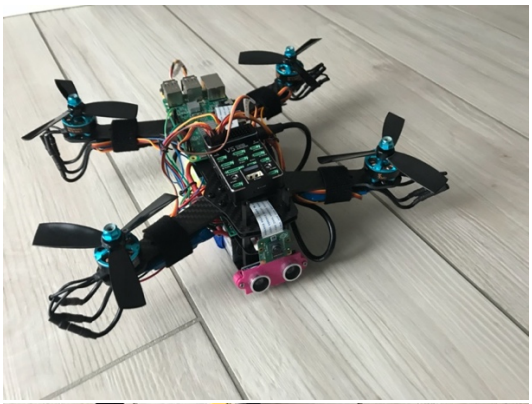


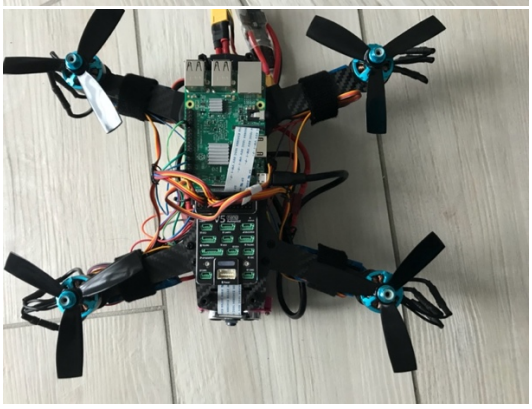
Then we covered the connectors with heat shrink tubes.



Before placing the components on the frame, the top two levels were isolated in order to prevent shortcut.

Finally, the build of the drone was finished.





The weight of the drone is 1.2kg. We calculated again with the help of the website, how much thrust we will have per motor with 2:1 power-to-weight ratio and we obtained 600g.

Total weight	1.2 kg
Thrust	
Power-to-weight ratio	2 :1
Number of motors	4
Total thrust	2,400 g
Thrust per motor	600 g

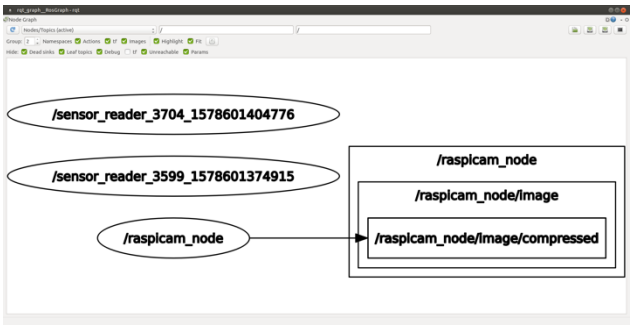
The result turns out to be good enough regarding the table for the motors which we displayed in section 4.3.

- **ROS**

We were familiar enough with the ROS environment from the previous project and we won't get into any details on how it works.

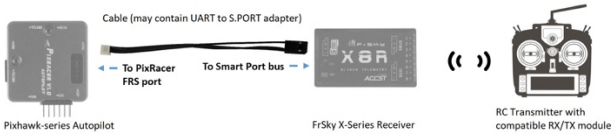
First, we created a workspace for this project. In this workspace we have two nodes; one for our sensors and the other for the camera. The created node is added to a GitHub repository [11]. We are working only in the one for the sensors. For this project we created 2 publishers and 2 subscribers, therefore 2 topics, from which we are going to use only one. The reason is

because one of the topics is for six sensors and the other only for one sensor. The one that is going to be used is the topic for one sensor.
We wanted to keep the possibility of having multiple sensors on the drone for a possible follow-up project.
The publishers of ROS use an imported script *distance.py (Figure 1)*.
This script consists in a function which takes the situated pins on the Raspberry Pi of the Trigger and the Echo of the ultrasonic sensor, calculates the distance in-between the sensor and an object in front of it and returns the result.
The publisher for the only one sensor is called *one_sensor_publisher.py* (*Figure 2)*. It uses a Float32 message to transfer in-between its topic.
For this publisher we have a subscriber, *one_sensor_subscriber.py* which can be run on the on-ground computer (*Figure 3)*.
Similar to these publisher and subscriber, we have created scripts in case one day we wish to have six sensors on board. (*Figure 3, 4 and 5)*. The difference is that this topic uses a list of Float32 numbers as a message to transfer in-between.
Here we can see all the active topics.



- **Transmitter and Receiver**

A similar connection as the following needed to be established in our project.



We needed to upgrade the transmitter and to flash a compatible framework to the receiver. During the configuration a lot of issues were encountered but, in the end, we succeeded to establish a connection and to configure the remote controller.

5.4. Assessment

The challenge to place more than one sensor could not be achieved for this project because of the lack of space on the frame. Nevertheless, everything needed for the communication and powering the sensors is considered as achieved and if in the future a possibility to attach more sensors to the frame is open, we will have the recourses to do it.
Despite the encountered issues during this project we stayed positive till the end and we learned that a solution can be found for every problem.

Acknowledgment

We would like to thank the University of Luxembourg for allowing us to spend a semester working on the project for the development of a multirotor from scratch.
We are grateful to the Technical support team of the University of Luxembourg for their kindness of lending us equipment, producing the additional plate, 3d printing the cases for the sensors and responding to every question we had regarding the connections.
A special gratitude goes to the project academic tutor, Jose Luis Sanchez Lopez, whose contribution and encouragement, helped to coordinate the project.
Last but not least we would like to thank two colleagues, Alessandro Giffra and Thierry Zigrand, who contributed when help and clarification was needed.

6. Conclusion

During the built of the drone a lot of improvisation has been made regarding the way to attach the components and we needed to be creative and willing to consider new ideas.
We learned how we can solder elements and how to choose the right parts if we wish to build a drone from scratch. We learned that we needed to be really careful and to stay sharp when working with electronics, because every step is important.
We managed to arrive at the end of this project, the drone was configured and ready for the final state in which a perform of a flight is needed.

Unfortunately, we could not proceed to this state, since the time for the submission has come. Nevertheless, this task will be achieved shortly after the submission of the project.

Appendix

```python
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

def distance(trigger,echo):

    PIN_TRIGGER = trigger
    PIN_ECHO = echo

    GPIO.output(PIN_TRIGGER, True)

    time.sleep(0.00001)
    GPIO.output(PIN_TRIGGER,False)

    startTime = time.time()
    stopTime = time.time()

    while GPIO.input(PIN_ECHO)==0:
        startTime = time.time()

    while GPIO.input(PIN_ECHO)==1:
        stopTime = time.time()

    duration = stopTime - startTime

    dist = (duration * 34300)/2

    return dist
```

Figure 1 - from section 5.3
distance.py

```python
#!/usr/bin/env python
import rospy
import distance
from std_msgs.msg import Float32
import RPi.GPIO as GPIO

def talker():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD) # mode--> pin

    #front sensor:
    trig=15
    echo=32

    GPIO.setup(trig,GPIO.OUT)
    GPIO.setup(echo, GPIO.IN)

    pub = rospy.Publisher('distance_from_1_sensor',Float32, queue_size=10)
    rospy.init_node('sensor_reader', anonymous=True)
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
        dist=distance.distance(trig,echo)
        rospy.loginfo(dist)
        pub.publish(dist)
        rate.sleep()

    GPIO.cleanup()


if __name__=='__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        rospy.loginfo('Measurement stopped by user.')
```

Figure 2 – from section 5.3
one_sensor_publisher.py

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import Float32
import RPi.GPIO as GPIO

def callback(msg):
    rospy.loginfo(str(msg.data))

def listener():
    rospy.init_node('one_sensor_subscriber', anonymous=True)
    rospy.Subscriber('distance_from_1_sensor',Float32, callback)

    rospy.spin()

if __name__=='__main__':
    try:
        listener()
    except rospy.ROSInterruptException:
        rospy.loginfo('Measurement stopped by user.')
```

Figure 2 - from section 5.3
one_sensor_subscriber.py

```python
#!/usr/bin/env python
import rospy
import distance
from std_msgs.msg import Float32MultiArray
import RPi.GPIO as GPIO

def talker():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD) # mode--> pin

    #Sensor 1 - front:
    trig1=15
    echo1=32
    first=(trig1,echo1)

    #Sensor 2 - bottom:
    trig1=33
    echo1=11
    second=(trig2,echo2)

    #Sensor 3 - back right:
    trig3=16
    echo3=18
    third=(trig3,echo3)

    #Sensor 4 - top:
    trig4= 22
    echo4= 39
    fourth=(trig4,echo4)

    #Sensor 5 - front right:
    trig5=31
    echo5=32
    fift=(trig5,echo5)

    #Sensor 6 - front left:
    trig6=7
    echo6=35
    sixth=(trig6,echo6)

    dataset=(first,second,third,fourth,fifth,sixth)
```

Figure 4 – from section 5.3
first part of the publisher for 6 sensors

```
    pub = rospy.Publisher('distance_from_6_sensors',Float32MultiArray, queue_size=10)
    rospy.init_node('sensor_reader', anonymous=True)
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
        measures = []
        for i in dataset:
            x=list(i)
            GPIO.setup(x[0], GPIO.OUT)
            GPIO.setup(x[1], GPIO.IN)
            dist=str(round(distance.distance(x[0],x[1]),2))
            measures.append(dist)

        mes=Float32MultiArray(data=measures)
        print(mes.data)
        rospy.loginfo(mes)
        pub.publish(mes)
        rate.sleep()
    GPIO.cleanup()


if __name__=='__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        rospy.loginfo('Measurement stopped by user.')
```

Figure 5 – from section 5.3
second part of the publisher for 6 sensors

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Float32MultiArray
import RPi.GPIO as GPIO

def callback(msg):
    rospy.loginfo(str(msg.data))

def listener():
    rospy.init_node'six_sensors_subscriber', anonymous=True)
    rospy.Subscriber('distance_from_6_sensors',Float32MultiArray, callback)

    rospy.spin()

if __name__=='__main__':
    try:
        listener()
    except rospy.ROSInterruptException:
        rospy.loginfo('Measurement stopped by user.')
```

Figure 6 – from section 5.3
the subscriber for 6 sensors

References

[1] N. Guelfi, “BiCS Bachelor Semester Project Report Template”,2020. [Online]. Available: https://github.com/nicolasguelfi/lu.uni.course.bics.global/blob/master/lu.uni.course.bics.all.projects.template.reports/main.pdf [Accessed 5th of January 2020]

[2] Julian Turner, “Euro vision: unravelling the new pan-European drone regulations”,2019. [Online]. Available: https://www.airport-technology.com/features/new-eu-drone-rules/ [Accessed 5th of January 2020

[3] M. Zhekova & J.L. Sanchez Lopez, “Increasing the capabilities of multirotor aerial robots with the Raspberry Pi platform“, 2019. [Online]. Available : [Accessed 10th of January 2020]https://github.com/mzhekova97/zhekova_bsp2

[4] Liza Brown, “Types of Drones: Explore Different Types of Drones”,2019. [Online]. Available: https://filmora.wondershare.com/drones/types-of-drones.html [Accessed 5th of January 2020]

[5] GetFPV LLC, “All about Multirotor Drone FPV Propellers”,2018. [Online]. Available: https://www.getfpv.com/learn/new-to-fpv/all-about-multirotor-fpv-drone-propellers/ [Accessed 5th of January 2020]

[6] PX4 Dev. Team, “PX4 Autopilot User Guide (master)”,2020. [Online]. Available: https://docs.px4.io/master/en/index.html [Accessed 5th of January 2020]

[7] Omni Calc., “Drone Motor Calculator”. [Online]. Available: https://www.omnicalculator.com/other/drone-motor [Accessed 9th of January 2020]

[8] “Estimate Propeller’s Static Thrust”, 2017. [Online]. Available: https://rcplanes.online/calc_thrust.htm [Accessed 9th of January 2020]

[9] UbiquityRobotics, “raspicam_node“, 2019. [Online]. Available: https://github.com/UbiquityRobotics/raspicam_node/tree/195694afee514370aaf28712e1e09c48bdaf2af7 [Accessed 9th of January 2020]

[10] PX4 Dev Team, “CUAV V5 nano Wiring Wuick Start”, 2020. [Online]. Available: https://docs.px4.io/master/en/assembly/quick_start_cuav_v5_nano.html [Accessed 9th of January 2020]

[11] M. Zhekova & J.L. Sanchez Lopez, “ROS-node for one or multiple sensors”, 2020. [Online]. Available: https://github.com/mzhekova97/zhekova_bsp3 [Accessed 9th of January 2020]