Master of Aerospace Engineering

Research Project

Autonomous Robotic Aerial Vehicle

S2 Project Report

Author :
Anton
Sambalov

Tutor :
RaghuVamsi
Deepthimahanthi

Due date of report: 30/Jun/2019

Actual submission date: 24/Jun/2019

Starting date of project: 29/Jan/2018

Contents

List of Figures

List of symbols

UAV *Unmanned Aerial Vehicle*

ATOL *Automatic Take-off and Landing*

ESC *Electronic Speed Controller*

LiPo *Lithium Polymer*

GPS *Global Positioning System*

GPS *Global Positioning System*

(C)CW *(Counter-)Clockwise*

LED *Light-Emitting Diode*

GCS *Ground Control Station*

QGC *QGroundControl*

RC *Radio Controller*

RPi *Raspberry Pi*

Tx *Transmit*

Rx *Receive*

PWM *Pulse Width Modulation*

Declaration of Authenticity

This assignment is entirely my own work. Quotations from literature are properly indicated with appropriated references in the text. All literature used in this piece of work is indicated in the bibliography placed at the end. I confirm that no sources have been used other than those stated.

I understand that plagiarism (copy without mentioning the reference) is a serious examinations offence that may result in disciplinary action being taken.

Date: 25.06.2019

Signature: Anton Sambalov

Abstract

Main goal of this project is to design, assemble and program a vehicle, which is able to autonomously move on the ground or in the air. Decision on the way to move should depend on the type of an encountered by the vehicle obstacle. In this way combination of quadcopter and rover is going to have a positive influence on the power consumption and time-efficiency of the vehicle.

Initial prototype of the vehicle has been designed and assembled by two MAE2 students: Sakshi Chaudhary and Nandhini Raghunathan. As the prototype was not in the working condition, the project has started with finalisation of the vehicle. During the second semester aerial module has been equipped with additional components and its performance has been analysed using log file. As behavior of the system has been satisfactory, next step includes flight test, which was not performed only due to the absence of the powerful enough battery. Ground module has been connected and motors controlled using a python script. In the following part of the project obstacle avoidance algorithm is going to be implemented onto the vehicle.

Keywords: **Quadrotor**, **Rover**, **Pixhawk**, **Raspberry Pi**, **QGroundControl**, **Obstacle Avoidance**

Introduction

Over the past years there can be noticed a strong tendency towards the development of autonomous vehicles. In the larger cities self-driving cars are being tested on the streets, autonomous delivery robots are cruising on the sidewalks and autonomous Unmanned Aerial Vehicles are available on the market for anyone who is interested. Those are just several of many examples. It is not surprising, as such technologies have extensive range of applications.

Rovers are one of the commonly used robotic vehicles. Rover is a wheel based vehicle used in order to perform tasks in the hardly accessible areas. Due to its high efficiency and ability to move across the rough terrain, rovers are especially known for their missions in the space exploration sector.

Unmanned Aerial Vehicle or UAV is another example of a robotic vehicle which has a large demand. UAV, as it's name says, is an aerial vehicle which does not have a pilot on board. UAV can be controlled from the ground by the operator or fly autonomously using the pre-programmed on-board computer. As a consequence of their comparatively moderate pricing and ease of use, there are many different models optimized for aerial photography, scientific, military and other applications.

The objective of this project is to construct and program an Autonomous Robotic Aerial Vehicle which could be used on Earth as well as on the other planets. The robot should be able to move on the ground and fly above it whenever it is necessary.

This report is written in order to describe the main goals and tasks of the project, the work which has been performed in the second semester and the assignments which are going to be performed through the duration of the project. The report is structured as follows. Description of the project is discussed in details chapter 1. Information on the construction of the vehicle can be found in chapter 2. Main results and their analysis is contained in chapter 3.

Chapter 1

Project definition

1.1 Goal of the project

In the scope of this project an Autonomous Robotic Aerial Vehicle is assembled and programmed. Such vehicle can be seen as a combination of two interconnected modules: ground module (rover) and aerial module (quadcopter). Such integration allows the vehicle to freely move in 3D space. High power efficiency of the rover combined with an ability of quadcopter to bypass any obstacle in a short time results in a favorable design.

In order to be independent of the human operator the robot should be able to autonomously take decisions and move from the initial point to the defined destination point. In order to do this, there is a need to implement sensors which will allow the robot to detect obstacles on its way. If an obstacle is detected, robot should be able to choose the most appropriate way to pass it: either go around it on the ground or cross it in the air using the quadcopter module. Choice of the module to be used depends on the three main parameters:

- **Power**: The aerial module should be chosen to fly over the obstacle in case the ground module has not enough power to pass it (for instance a hill, slope of which is too steep).

- **Energy**: As the rover has better efficiency in terms of the power consumption, use of the ground module is preferred to save energy as long as obstacle can be avoided on the ground.

- **Time**: If the mission has strict time constraints, the time it takes to avoid an obstacle could be minimised by the use of the aerial module.

1.2 Project issues

The main issue of the project is communication within the robotic vehicle. Vehicle should be able to switch between the ground and air modules whenever it is necessary.

In order to perform the task as described in section 1.1, the path planning algorithm should be implemented together with the obstacle avoidance. In order for them to be functional, decisions have to be made on the sensors to be installed on the prototype.

Steps that have to be made in order to make the vehicle autonomous include implementation of the path planning and obstacle avoidance algorithms. In order to move from initial point to the final one efficiently, robot will have to find the shortest path between two of those points. This is done by breaking a complete path into shorter segments, which are connected by waypoints. Robot should be able to read coordinates of the waypoints and move from one to another until the final destination is reached.

On the way from one waypoint to another robot must detect obstacles with the use of a sensor. The exact sensor is going to be selected on the later stages of the project. As the obstacle is detected, either ground or aerial module can be used in order to avoid it. Avoidance on the ground is done by finding an alternative path around the obstacle and includes automatic creation of intermediate waypoints. If the obstacle cannot be avoided by moving on the ground or the time is of great importance at this specific moment, aerial module is used in order to avoid an obstacle. In this case vehicle is going to use ATOL (Automatic Take-off and Landing) algorithm. Path planning together with obstacle avoidance is shown if fig. 1.1.

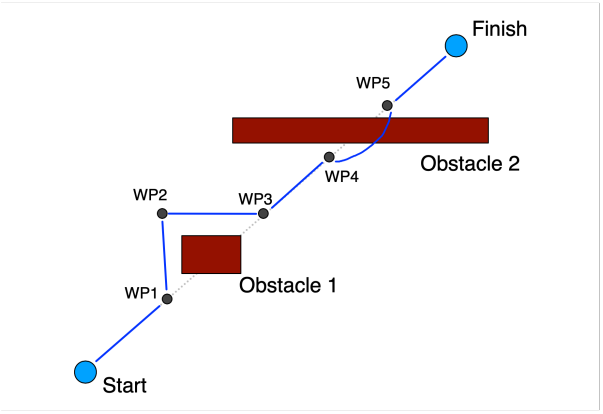


Figure 1.1: Path planning and obstacle avoidance algorithms

Each waypoint has its coordinates, where the z-axis is perpendicular to the ground and

has a value of zero on the intersection with it. Vehicle moves on the ground as long as the z-coordinate value of the waypoint is zero. If the z-coordinate has a value higher than zero, aerial module becomes activated and the vehicle is going to take-off.

1.3 State of the Art

Several rovers have already been exploring other planets. The most famous ones include NASAs Opportunity and Curiosity rovers, which have been used in order to explore Mars. Chinese rover Yutu has been exploring the Moon until the August 2016 and a new version with upgraded reliability called Yutu-2 has recently landed on the far side of the Moon [2]. In the year 2020 NASA is planning to send the first flying vehicle to the Mars: Mars Helicopter [3]. While there has been a continuous upgrade in the power systems, structures and autonomy, none of those vehicles has been designed to move both on the surface as well as above it. Therefore, it can be stated that the autonomous robotic aerial vehicle is a unique project, as it concentrates on entirely autonomous combination of UAV and a rover.

Concept of having a vehicle capable of both flying and driving has been a popular theme of sci-fi stories for a long time, however the first genuine attempts to develop such vehicle have been done in a relatively recent time. Most of the pioneers in this sector are young companies working on their first prototypes. Those include The Transition by Terrafugia and The Urban Aeronautics X-Hawk.

1.4 Potential usability

Autonomous robotic aerial vehicles can potentially be used on the Earth as well as on the other planets. On the Earth possible missions include the search and rescue in case of a disaster, as vehicle can autonomously pass through a rough terrain and detect objects or human beings. As well vehicle could be efficiently used for delivery, medical help, surveillance and other demanding tasks. On other planets, for example on Mars, vehicle could be used in order to collect and analyse samples for scientific reasons to explore the hardly reachable parts of the planet.

Chapter 2

Robotic Aerial Vehicle

The project has been on-going since beginning of the year 2018. MAE2 students Sakshi Chaudhary and Nandhini Raghunathan have concluded their work with the assembly of the initial prototype [4]. The prototype, as described in section 1.1, consists of the two modules: ground and aerial. During the second semester vehicle has been further upgraded in order to achieve the main goal in the later stage.

This chapter concentrates on the construction of the vehicle (section 2.1) as well as implementations done on its hardware (section 2.2, section 2.3 and section 2.4). Discussion on the sensors to be installed during further stages of this project can be found in section 2.5. Fully assembled vehicle can be seen in fig. 2.1.

2.1 Vehicle Components

2.1.1 Air Module

Air module is responsible for the flight part of the mission. It should not only be able to lift its own weight, but as well take-off with the ground module as a payload. The air module of a vehicle consists of the components listed below.

Frame

Frame or chassis of the quadcopter is essentially its basis: all of the other components, such as motors and ESC's, are fixed to it. Frame should be sized in accordance with propellers, as there should be enough space for them not to touch each other. As well frame should be strong enough to support the payload.

X-type frame with 450mm wheelbase is used for this project.

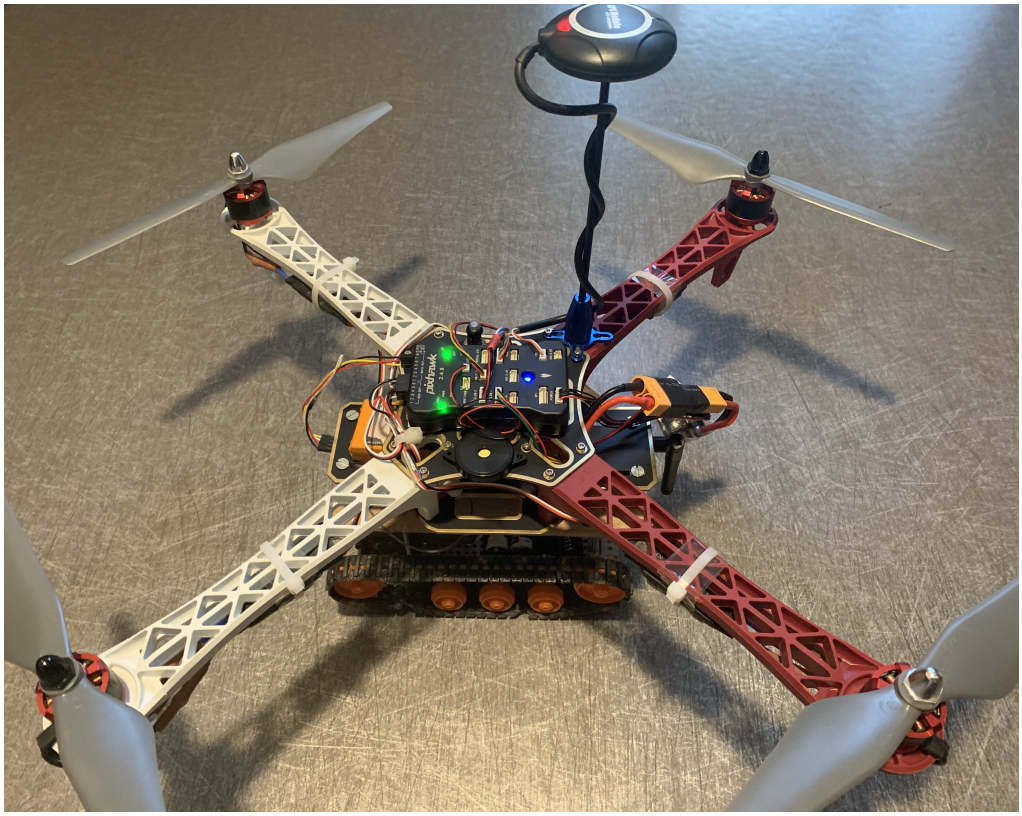


Figure 2.1: Fully assembled vehicle

Power Distribution Board (PDB)

Power distribution board is a circuit board used to distribute power from the battery to other components (motors, flight controller, companion computer) with the required voltage.

Motors

Motors are rated using Kv units, which corresponds to number of revolutions per minute motor can perform when supplied a current of 1v. Higher Kv defines a faster spinning motor, while lower value corresponds to a higher produced torque.

Vehicle has four of the 2212 920kV brushless DC motors, meaning they have a stator width of 22mm and height of 12mm.

Propellers

Propellers have a large influence on the speed, total amount of thrust produced and maneuverability of the quadcopter. Higher pitch of the propeller in general is beneficial for the maximal velocity quadcopter can achieve. Bigger propellers have a larger airflow around them and therefore more lift, however they take more time to spin-up and are not efficient. Meanwhile use of the smaller propellers allows to change their rotation faster, and therefore improve the maneuverability.

Propellers of type 9450 are used (9.4 inch diameter, 5.0 inch pitch).

Electronic Speed Controllers (ESC)

ESC receive signal from the flight controller, containing the information about direction and speed with which motors should rotate. As they are connected to the battery, necessary power can be taken from it. Four of the 30A ESCs are used in this project.

Battery

Typical power source for a quadcopter is a Lithium Polymer (LiPo) battery. Important parameters of the battery are nominal voltage, amount of cells (3.7v per one cell or 1s), capacity (in mAh) and discharge rate. 3S battery is required in order to make the vehicle lift of the ground.

Flight Controller

Flight controller might be regarded as the brain of a quadcopter. The vehicle is equipped with a Pixhawk, which serves as a flight controller. While it has internal gyroscope, compass and other components, many additional elements such as telemetry and GPS can be connected directly to it. More information considering Pixhawk can be found in section 2.2.

2.1.2 Ground Module

Rover has a base similar to one of a tank: continuous track is used in order to achieve maximum control on a rough terrain and prevent the vehicle from getting stuck. Following equipment is used:

- **Frame**: Tamiya universal plate set is used as a frame.

- **Tracks and Wheels**: Ground module is based on Tamiya 70100 track and wheel set.
- **Gearbox**: Tamiya 70097 twin-motor gearbox consists of the gearboxes and two independent brushed DC motors, which are used to drive two shafts. Motors run on 3-6V.
- **Power System**: Battery box allows to install up to 4 batteries in series. At the moment two AA batteries are used to supply power to the motors.
- **Motor Driver**: The TB6612FNG motor driver is able to control speed and direction of the two motors. Having a supply range of 2.5V to 13.5V, it suits our motors. More on the connections of the motor driver can be found in section 2.4.
- **Computer**: In order to let the motor driver know when and which way it should run the motors, as well as to communicate with the air module, there is a need for the on-board computer. Raspberry Pi is used for those purposes. Information about Raspberry Pi and its pins can be found in section 2.3.

2.2 Pixhawk

Pixhawk is one of the most popular general purpose flight controllers. Pixhawk has a processor and a failsafe co-processor. It is equipped with internal gyroscope, accelerometer, magnetometer and barometer, allowing it to efficiently monitor the flight. Additionally there are 14 ports (see fig. 2.2), which are used to connect extra equipment.

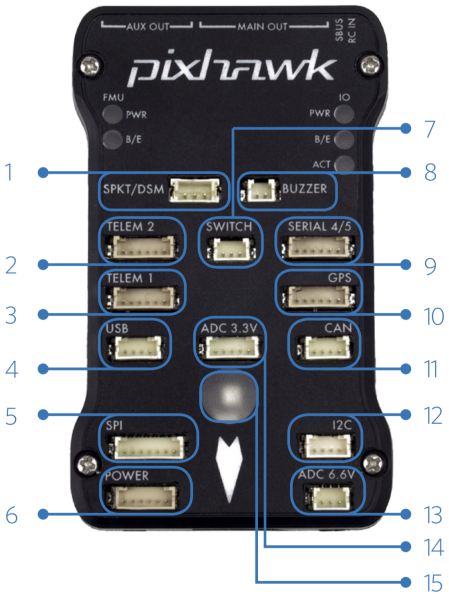
SD card can be installed for high-rate logging. Pins used for radio control input and for outputs can be seen in fig. 2.3. Connections made to Pixhawk and their use are described in section 2.2.1. Steps performed in order to bring Pixhawk to the working condition can be seen in section 2.2.2

2.2.1 Connections to the Pixhawk

Every connected to the Pixhawk component is listed below.

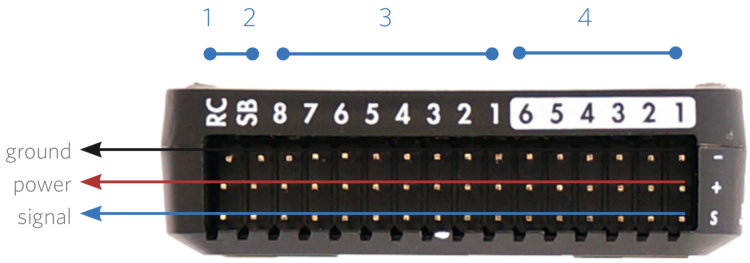
ESCs

Four of the ESCs are connected to the main outputs numbered 1 to 4 (see fig. 2.3). Columns 1 and 2 are used to connect counter-clockwise (CCW) rotating propeller, while columns 3 and 4 are used for the clockwise (CW) rotating propeller [5]. Each ESC is connected using 3 wires: one connected to the ground pin, second to the power and third to the signal.



1 Spektrum DSM receiver
2 Telemetry (radio telemetry)
3 Telemetry (on-screen display)
4 USB
5 SPI (serial peripheral interface) bus
6 Power module
7 Safety switch button
8 Buzzer
9 Serial
10 GPS module
11 CAN (controller area network) bus
12 I²C splitter or compass module
13 Analog to digital converter 6.6 V
14 Analog to digital converter 3.3 V
15 LED indicator

Figure 2.2: Ports of the Pixhawk



1 Radio control receiver input
2 S.Bus output
3 Main outputs
4 Auxiliary outputs

Figure 2.3: Pins of the Pixhawk

Receiver

The goal of the project is to develop a fully autonomous robotic vehicle, however for the first stages of the project it is obligatory to use a radio controller in order to make sure the system works in a satisfactory way. Turnigy TGY-i6S proportional radio control system is used for this reason. Turnigy TGY-iA6C receiver is connected to the RC column pins of the Pixhawk, allowing it to receive radio signals.

Battery

3DR power module is connected to the Pixhawk POWER port (number 6 on fig. 2.2) using the 6-wire cable. Now limited power from the LiPo battery can power up the flight controller.

Buzzer

Buzzer is responsible for playing sounds, for example during the arming. It is connected to the Pixhawk BUZZER port using 2-wire cable.

Safety Switch

Safety Switch is used to arm or disarm the vehicle (enable or disable output to the motors). It is attached to SWITCH port via 3-wire cable.

LED of the safety switch has following indications [6]:

- Constant blinking: system is initializing.
- Intermittent blinking: system is ready and safety switch can be pressed to enable arming.
- Solid: Arming is enabled.

Telemetry

Telemetry is used in order to receive data and communicate with the quadcopter wirelessly. It is composed of the ground module which is connected via USB to the ground station, and the air module is connected using a 4-wire cable to the Pixhawk. YKS 3DR Radio Telemetry Kit is used, as it is light and able to support air data rates up to 250kbps.

GPS Module with Compass

Pixhawk supports use of GPS and up to 4 external magnetometers. However, only the best one of them is chosen by the system and used at any point in time. Priority is given to the external magnetometer. In case of a failure of an external magnetometer, internal one is going to be used for heading.

In order to reduce electromagnetic interference, GPS and compass should be mounted as far away as possible from the motor power supply lines to avoid the interferance. This has been done by fixing a GPS module support platform to the frame. Module has been

fixed 14cm above the frame. GPS has been connected with 4-wire cable to the GPS module port, while compass has been connected with 2-wire cable to the I2C splitter.

Solid red LED shows that module is powered. Flashing blue LED shows that GPS lock has been acquired.

SD card

4GB micro-SD card is plugged into the SD-card slot at the top part of the board. It is going to be used to safe logs during the flight and analyze them. More information on the log analysis can be found in section 3.1.1.

Pixhawk with all of the components connected to it can be seen in fig. 2.4.

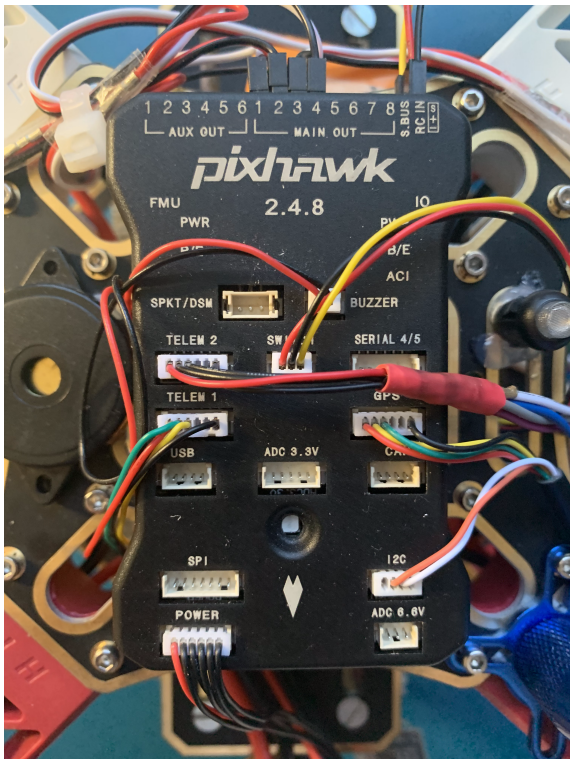


Figure 2.4: Pixhawk with all of the connections

2.2.2 Setup of the Pixhawk

Once all of the equipment has been connected, it was the time to configure and calibrate the Pixhawk. Ground Control Station (GCS) is used for this purpose. GCS is a software which allows to wirelessly (in case the telemetry is installed) connect to the vehicle and display its position and attitude in real-time. As well parameters of the vehicle can be

configured in order to suit needs of a user. When the vehicle is set-up, GCS can be used to control UAV in flight.

While there are multiple available ground stations, QGroundControl (QGC) has been selected as a GCS. First step was to download the software and connect the Pixhawk to computer via USB. QGC recognizes Pixhawk and gives 2 options for the flight stack to be installed: PX4 or ArduPilot. It was chosen to install PX4 flight stack due to its modular and extensible architecture, companion development tools and flexible flight modes [7].

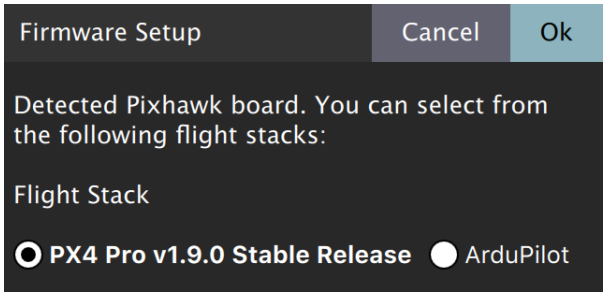


Figure 2.5: Firmware setup using QGC

After the flight stack has been successfully installed and the vehicle has rebooted, USB could be disconnected and process of calibration could be done using telemetry. Prior to the calibration vehicle type has been selected as a "Generic Quadrotor x". Following list describes the calibration process.

Sensors

- **Compass**: Compass calibration can be done once for both internal and external magnetometers. ROTATION_NONE parameter should be chosen prior to calibration, as front edge of the Pixhawk coincides with the front of the quadcopter. Vehicle should be placed in one of the orientations displayed by the software and rotated around a specified axis. After rotation around every single orientation has been finished, compass calibration is completed and display is as on fig. 2.6.

- **Gyroscope**: Gyroscope calibration is performed by leaving a vehicle on a flat surface.

- **Accelerometer**: Calibration of accelerometer is similar to one of the compass, except no rotation around the axis is required.

- **Level Horizon**: Level Horizon calibration is done in order to setup the flight view icon of the QGC. Vehicle should be left in its level flight orientation on a flat surface. For quadcopter it is a hover orientation.

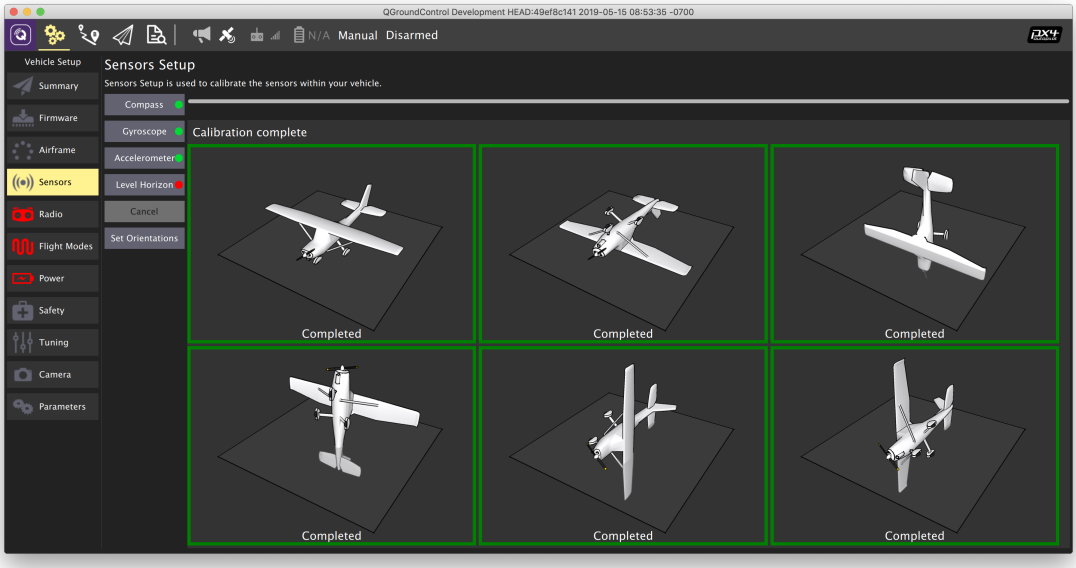


Figure 2.6: Completed compass calibration using QGC

Radio

The radio setup is used to map the control stick input to roll, pitch, yaw and throttle. As well minimum, maximum, trim and reverse settings are calibrated for all of the RC channels.

Calibration is started be selecting the mode in which radio controller operates. In our case it is Mode 1, meaning that throttle is controlled with the right stick. After that user is requested to move sticks to the shown position untill the calibration is finalised. Radio calibration process can be seen on fig. 2.7.

Flight Modes

Flight modes are divided into two types: autopilot-assisted modes and fully autonomous ones. In the autopilot-assisted mode pilot has control over the vehicle via RC, while its behavior is partially managed by the flight controller. Fully autonomous flight modes do not require any further input from the pilot and follow the preprogrammed task. Flight modes can be changed using the channels of RC or directly from the ground station. Some of the most common autopilot-assisted modes supported by PX4 are [8]:

- Stabilized - vehicle levels when sticks are set to the middle position and is almost impossible to be flipped.
- Position - once sticks are released, vehicle keeps its current position.

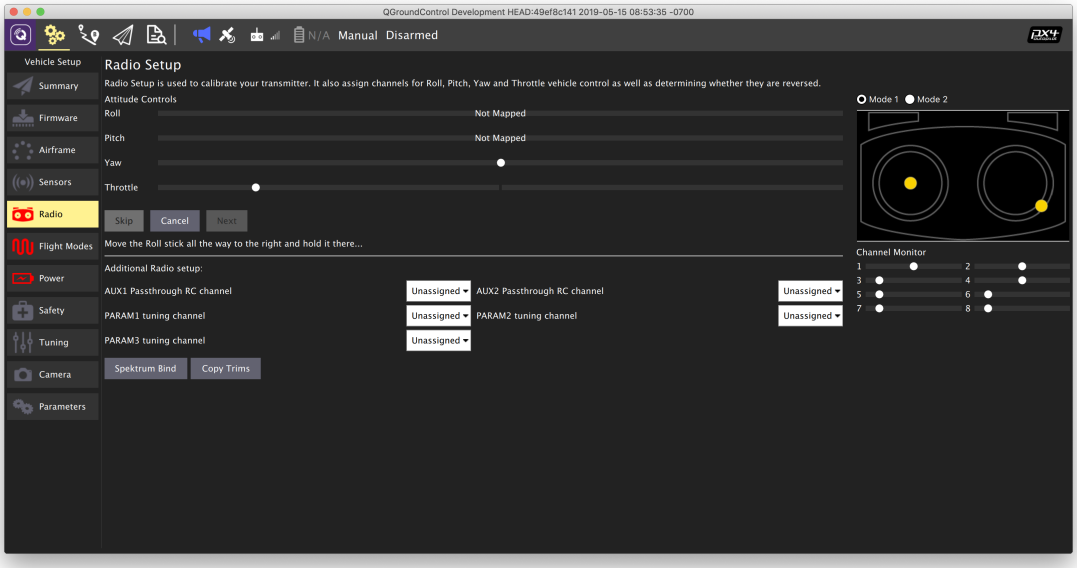


Figure 2.7: Radio setup using QGC

- Altitude - climb and decent rates are controlled to have maximum rate.

Common autonomous flight modes are:

- Takeoff/Land - vehicle takes-off to the preset altitude.
- Land - vehicle lands at the spot where the mode has been turned on.
- Hold - vehicle hovers at the current position.
- Return - vehicle gains safe altitude, returns to the take-off position and lands.
- Mission - vehicle executes a pre-programmed flight plan.
- Offboard - vehicle follows points provided by the companion computer over MAVLink.

For the initial configuration stabilized, position and altitude modes have been set up using a single-channel selection mode (up to 6 modes, each corresponding to the different PWM value of the single channel).

Power

Power calibration is used in order to see a remaining battery power percentage through the ground control station. In this case damage due to a too low charge and crash of the vehicle can be prevented.

Number of cells in series in battery should be selected, in our case it is 3 cells or 3S. Full voltage per cell parameter sets the voltage, at which battery is going to be considered fully charged. While 4.2V is a nominal value for a LiPo battery, value of full voltage is set to be 4.05 in order to account for aging of the battery. Empty voltage per cell sets a minimal voltage at which battery can operate without damaging itself. 3.5V value is set to be empty.

2.3 Raspberry Pi

The Raspberry Pi (RPi) is a credit card size single-board computer, which can be used for many different applications. In the scope of this project Raspberry Pi1 model A+ V1.1 is used in order to control the rover movement and to implement the obstacle avoidance algorithm.

Raspberry Pi1 model A+ has 40 GPIO pins, micro SD card slot, HDMI port, USB port and a micro USB power supply. GPIO pins layout can be seen in fig. 2.8.

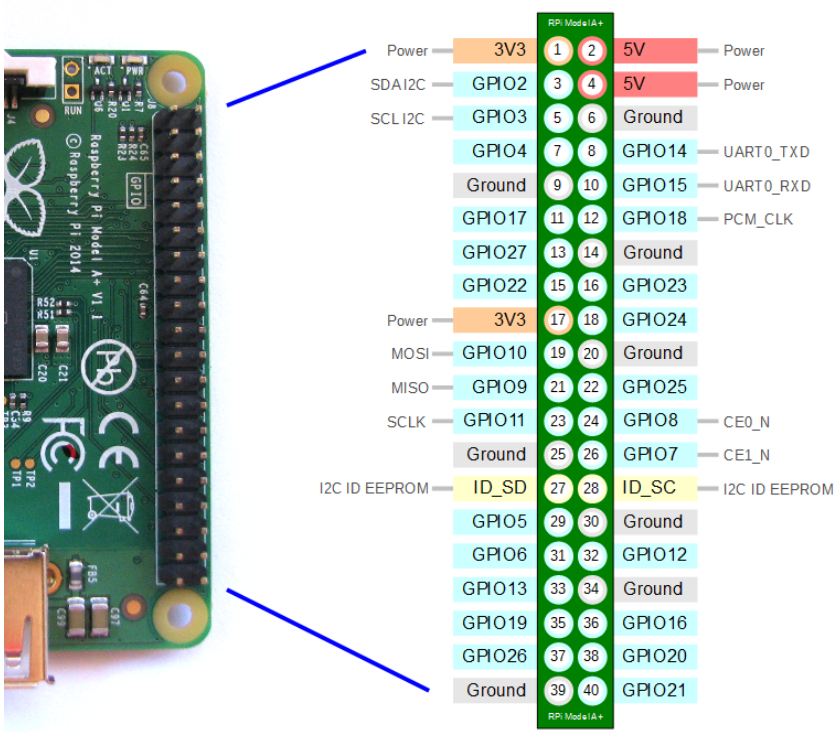


Figure 2.8: GPIO pins setup on the Raspberry Pi

2.3.1 Initial Setup of the Raspberry Pi

This section describes the steps performed in order to bring Raspberry Pi to the working condition. As the first step micro SD card has been connected to the computer and formatted using SD Formatter software, erasing all of the previous data from it. Second step was to download the Raspbian operating system onto the SD card. For this NOOBS (New Out Of The Box Software) zip file has been downloaded from the Raspberry Pi website, and files have been extracted and copied to the card. Once files have been transfered, SD card could be removed from PC and plugged into the micro SD card slot of the Raspberry Pi.

Afterwards RPi has been configured as follows: monitor is connected to the RPi through HTML port, USB hub is plugged into the USB slot, and keyboard with mouse are plugged into the USB hub. Now RPi can be powered by a micro USB power supply. Connected Raspberry can be seen in fig. 2.9.

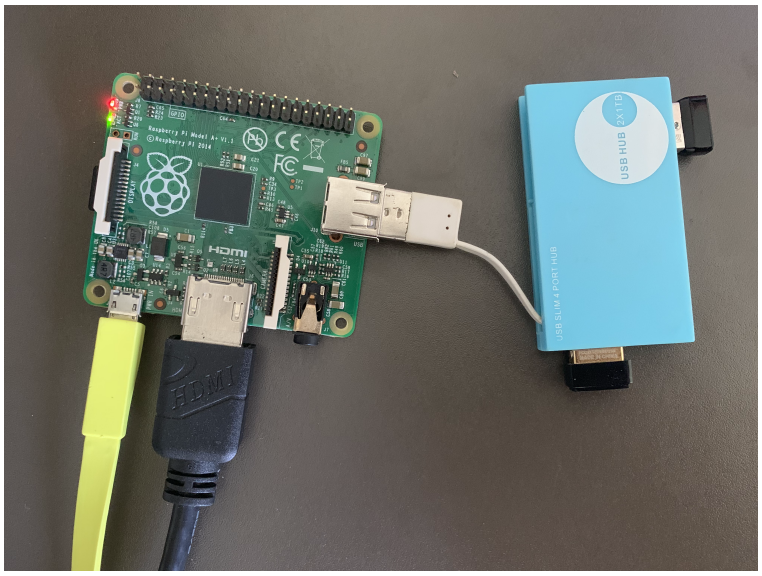


Figure 2.9: Raspberry Pi connected to the power, monitor, keyboard and mouse

Red LED indicates that the RPi is now connected and soon the monitor turns on. As the first step user is given choice to install the software. Raspbian Full has been installed. Once installation is completed, Welcome to Raspberry Pi setup page turns on, from where one can set country, language, timezone, change the standard password and connect to the wifi (with the use of the wifi dongle). Now Raspbian can be updated to the latest version and RPi rebooted.

2.3.2 SSH

Once the RPi is operational, it is convenient to connect RPi to the PC using SSH connection. In this way requirement to use monitor, keyboard and a mouse can be avoided during the further steps. First of all, SSH should be enabled through the RPi terminal. This was done by typing "sudo raspi-config" command, selecting "Interfacing Option" and choosing "SSH enable". The laptop running macOS has been connected to the same wifi as RPi and terminal on PC is opened. Command "ssh pi@172.20.10.3" has been typed in, where second part corresponds to the ip address of the Raspberry Pi. After agreeing to connect (only the first connection) and entering the password terminal connects to the RPi remotely. This procedure can be seen in fig. 2.10.

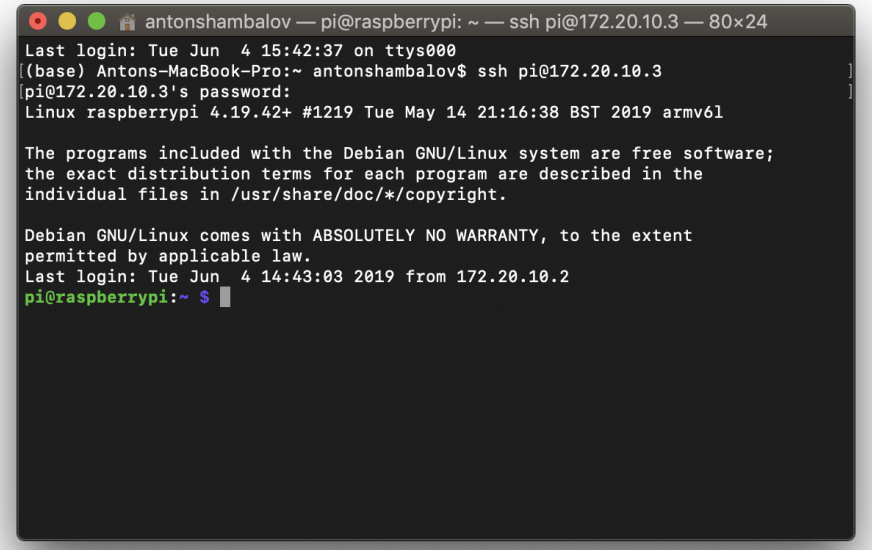


Figure 2.10: Raspberry Pi connected to computer using SSH

2.3.3 Connection between RPi and Pixhawk

In order to change between ground and air module, there should be a communication link between them. RPi is connected to the Pixhawk using four wires. TELEM2 port of the Pixhawk is connect to RPi's 5V (pin 2), Ground (pin 6), Tx (pin 8) and Rx (pin 10) pins. For clear representation see fig. 2.11. It should be noted, that RPi model 3 has same pin layout as model 1. Communication is done using the MAVLink protocol [1].

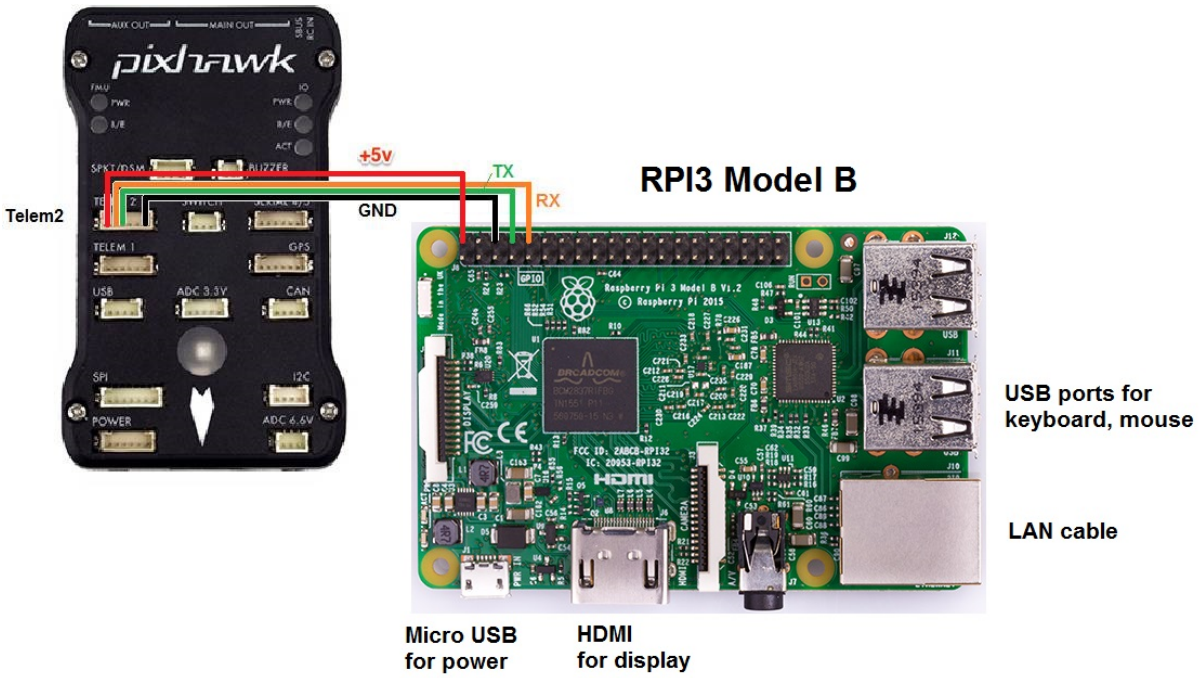


Figure 2.11: Raspberry Pi connection to the Pixhawk [1]

2.4 Motor Driver

In order to operate the rover motors, motor driver TB6612FNG mentioned in section 2.1.2 had to be connected to the motors, power supply and Raspberry Pi [9]. 3V coming from the batteries are directed to the motors using VM (Motor Voltage) pin. VCC (Logic Voltage) port is used to power the chip itself and can be connected to both 3.3V or 5V of the Raspberry Pi. For the moment is was chosen to use 3.3V pin. AIN and BIN pins determine the direction of the rotation, while PWM pins control the speed. All of the connections are described in table 2.1. Connected equipment is depicted in fig. 2.12.

2.5 Sensors

In order to implement obstacle avoidance on further steps of the project, decision has to be made on the type and amount of sensors to be installed onto the vehicle. Sensors are used to detect obstacles and distance to them on the path.

Ultrasonic sensors are measuring distance to an object by emitting an ultrasonic wave and receiving back its reflection from the obstacle. Calculation of distance is performed by measuring the time from transmission to receipt of the wave.

HC-SR04 Ultrasonic Sensor is going to be installed on the front part of the vehicle.

TB6612FNG pins	Connection
VM	Battery Box (V)
GND	Battery Box (GND)
A01	Motor A (+)
A02	Motor A (-)
B01	Motor B (+)
B02	Motor B (-)
GND	RPi GND pin 14
VCC	RPi 3.3v pin 1
PWMA	RPi pin 12
AIN1	RPi pin 16
AIN2	RPi pin 18
STBY	RPi pin 22
BIN1	RPi pin 15
BIN2	RPi pin 13
PWMB	RPi pin 11

Table 2.1: Connection of the motor control board pins

Those sensors are able to measure distance between 2cm and 400cm, while having a window of 30 degrees. Such parameters are sufficient to detect an obstacle at the safe distance and make a decision if to pass it on the ground or in the air. Once the system is proven to work well, another 5 HC-SR04 sensors can be installed: 1 in the back, 1 on each side and 2 attached below front propellers, pointing to the front. Placement of the sensors can be seen in fig. 2.13. Such design would allow to avoid rotation about own axis to look for potential obstacles.

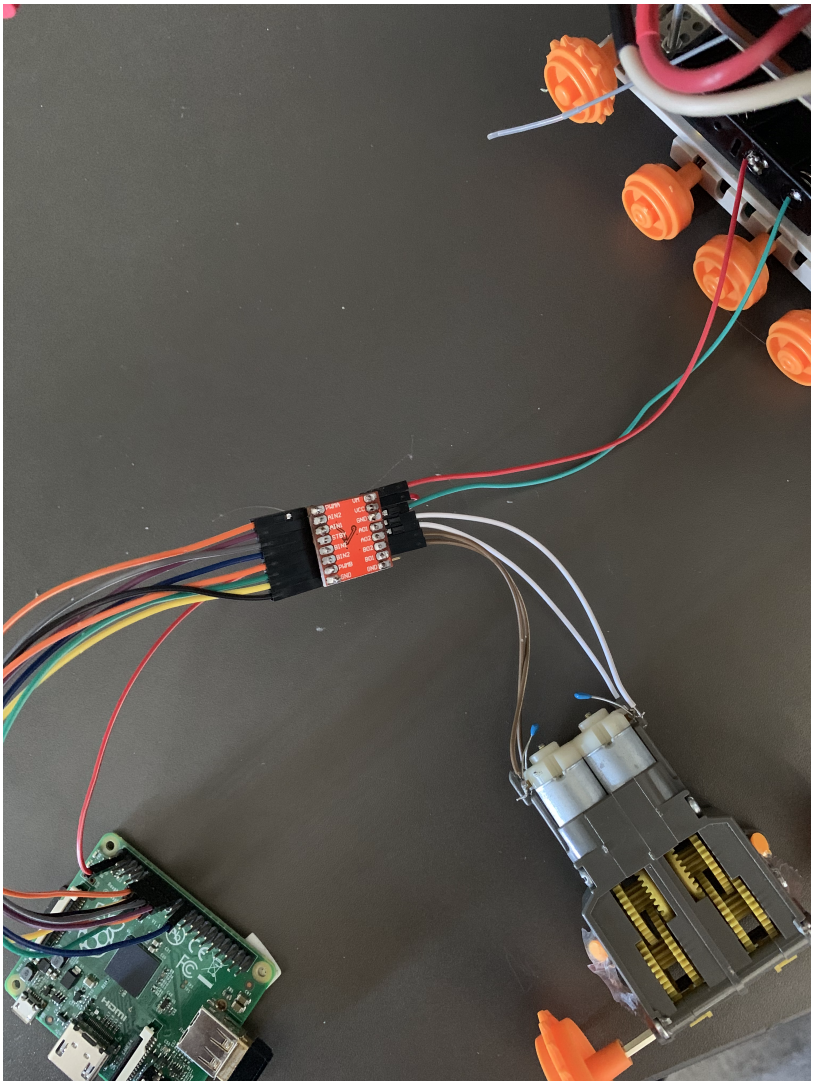


Figure 2.12: Connection of the motor control board pins

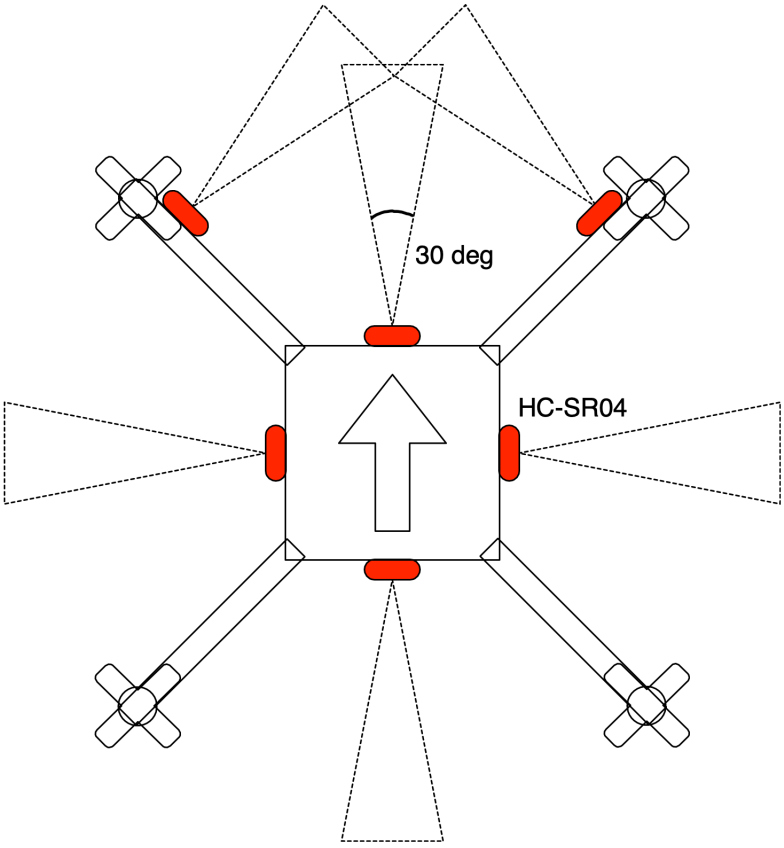


Figure 2.13: Placement of the sensors on the vehicle

Chapter 3

Results and Analysis

Once the vehicle has been built and setup as described in chapter 2, it was the time to test its abilities and analyse the performance.

3.1 Air module ground test

The purpose of initial test was to see if all of the equipment is correctly connected and motors respond to the input from RC. For those reasons propellers were not attached to the drone. Once telemetry has been connected and QGC opened on PC, LiPo battery was connected to the vehicle powering it up. LED lights have shown that equipment is powered and ready to be used. GPS has locked, compass has been showing correct orientation. Gyroscopes responded to the manual inclination of the vehicle. QGC screen at this point can be seen on fig. 3.1.

Now safety switch has been pressed to enable arming and vehicle has been armed by moving left-side stick into the right-bottom corner. Motors have been given input in 100% thrust, -1 to 1 yaw and roll, and disarmed afterwards. Motors responded to inputs and all of the equipment was working accordingly. Further analysis was performed using logs and is described in section 3.1.1.

3.1.1 Flight log analysis

Data from the sensors together with the drone state are being logged onto the micro SD card by PX4. This data can be used in order to analyse performance of the vehicle. Recording of the log is started every time the vehicle is being armed.

Log files are saved into the "log" folder of the SD card and separated by the date of a flight and its time. Flight Review [10] online tool is used to draw plots from the .ulg format

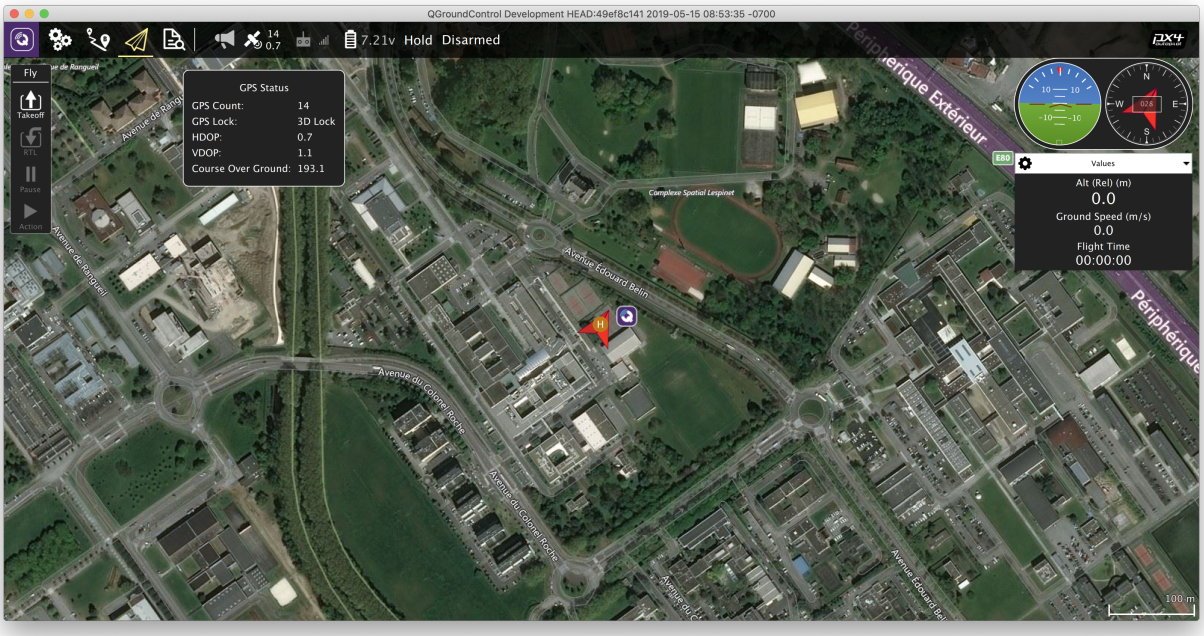


Figure 3.1: QGroundControl main window

data file. Some of the useful information drawn after the test described in section 3.1 can be seen below. PX4 guide is used to compare graphs with the common values [11].

GPS Uncertainty

GPS uncertainty shows information about the GPS signal and can be seen in fig. 3.2. Signal is considered to be good when number of satellites used is around 12. As can be seen on average GPS has been using more than 12 satellites. Both horizontal and vertical accuracy are around 1m, which is fine for our applications. GPS Fix value of 3 shows, that 3D GPS fix is achieved.

GPS Noise and Jamming

GPS signal sometimes might be disturbed by equipment operating in the same frequency as GPS. Plot seen on fig. 3.3 is used to analyse if GPS signal was interfered. Jamming indicator should be below 40 (it is at 20 in our case). Noise per ms below 120 is negligible. As a conclusion, the GPS operates well.

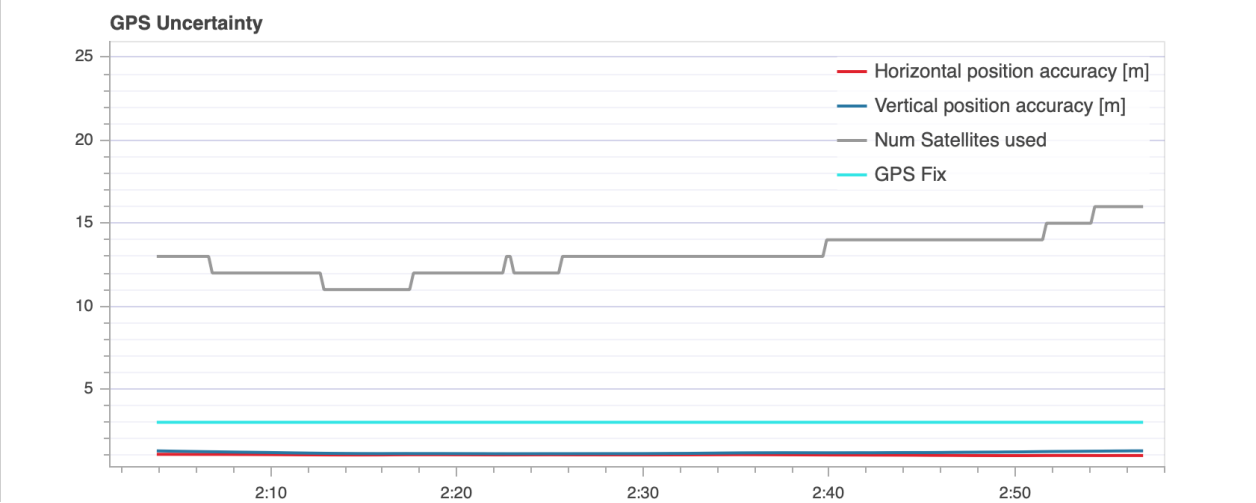


Figure 3.2: GPS Uncertainty analysis

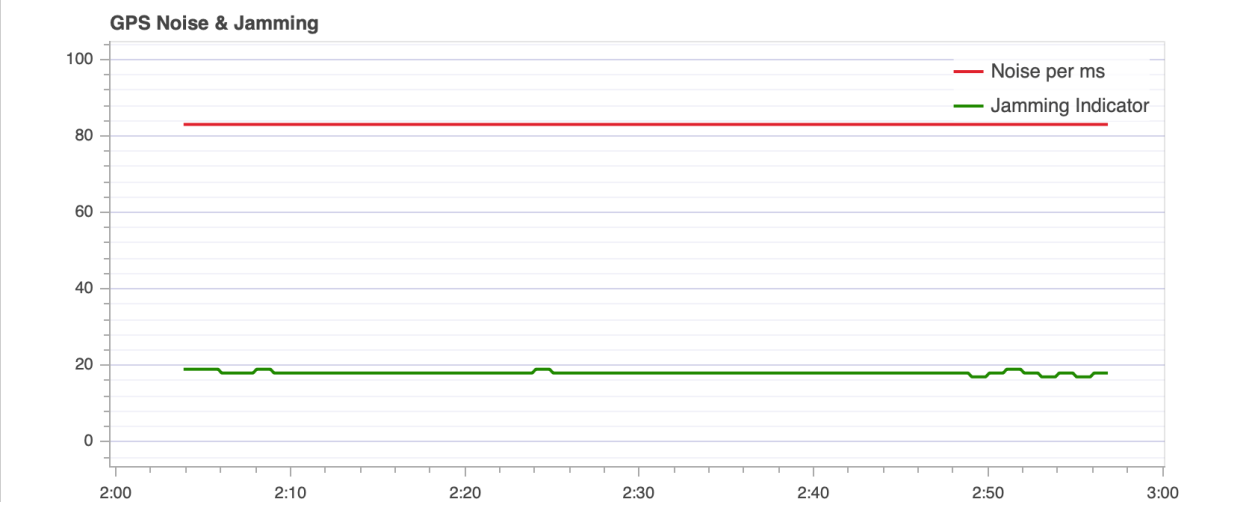


Figure 3.3: GPS noise and jamming analysis

Thrust and Magnetic Field

Thrust and the norm of the magnetic sensor measurement vector can be seen in fig. 3.4. Magnetic field should be constant and independent of the thrust. Otherwise compass might be disturbed by the current going to motors. As it can be seen, compass is well calibrated and located far enough from the wires.

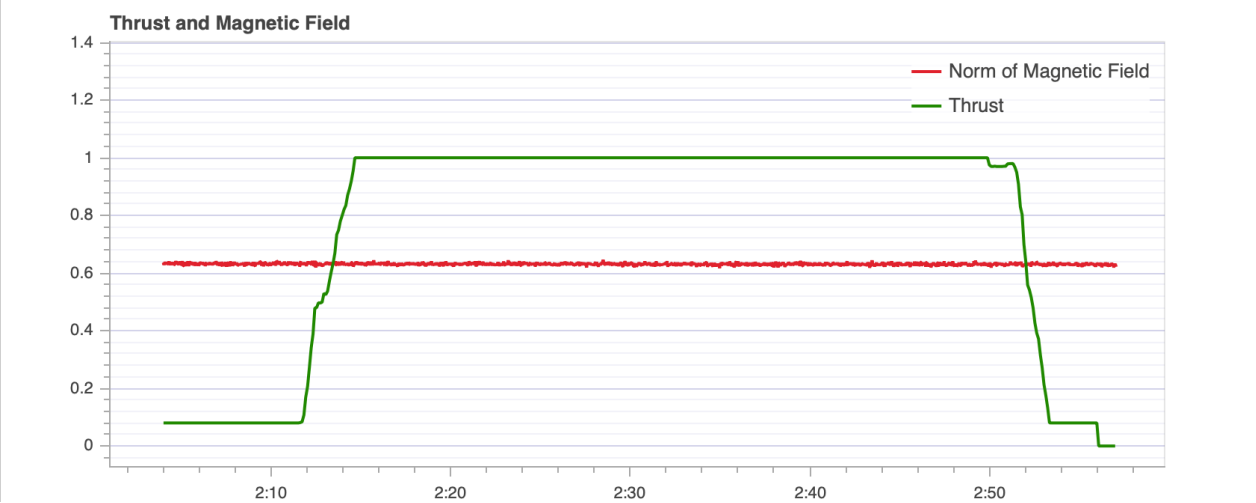


Figure 3.4: Thrust and Magnetic Field analysis

Estimator Watchdog

Estimator watchdog monitors health of the estimator. If it has a non-zero value, there is a problem with sensors. Estimator Watchdog plot had a constant zero value, indicating that there are no problems.

Sampling Regularity of Sensor Data

Sampling Regularity of Sensor Data (fig. 3.5) analysis is used to check performance of the SD card. As the data is published at 250Hz, the delta t value should stay close to 4ms.

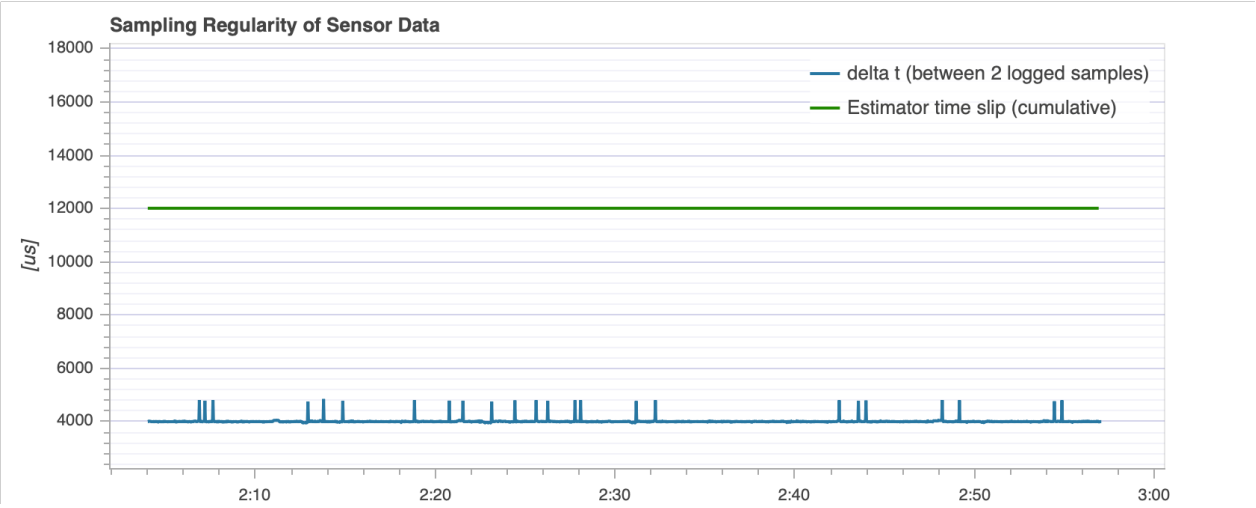


Figure 3.5: Sampling Regularity of Sensor Data analysis

Manual Control Inputs

Manual Control Inputs graph (fig. 3.6) shows inputs received by flight controller from the RC. It can be seen that inputs described in section 3.1 have been received as expected and the flight mode is set to Stabilized as it should be.

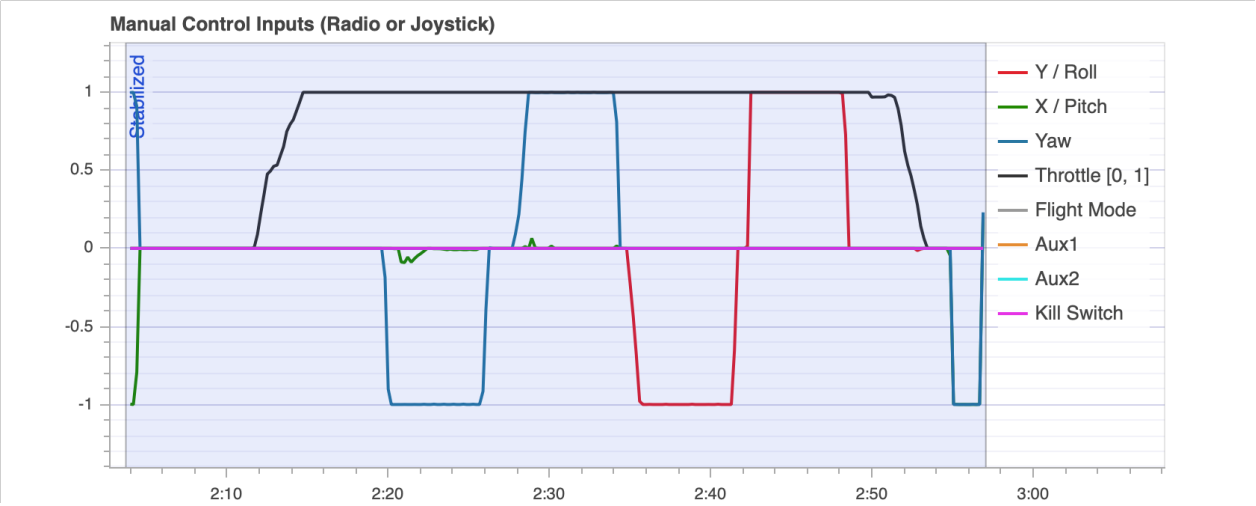


Figure 3.6: Manual Control Inputs analysis

3.2 Ground movement

After motor control board, motors, battery box and Raspberry Pi have been successfully connected, it was the time to make the gear spin. First step was to connect to the RPi using SSH as described in section 2.3.2. Python v3.5.3 has been installed onto RPi and used for creating a script. File testSpin.py has been created in the Documents folder by using command "nano testSpin.py".

First part of the code (fig. 3.7) initializes necessary modules and setups pins according to the connection between the RPi and the motor control board. Warnings are ignored in order to make the system run continuously. PWM is set and initialized in order to control speed of the motor.

Second step was to define functions which can be used to drive vehicle to the front, backward, or turn it into desired rotation. Function runMotor takes 3 inputs: motor (A or B), speed (0 to 100) and direction (CW or CCW). By changing those parameters, desired direction and speed can be achieved. Command GPIO.HIGH supplies 3.3V to indicated pin, while GPIO.LOW gives 0V. Functions can be seen in fig. 3.8.

For the demonstration repeating loop has been set as the main function (fig. 3.9). Loop consists of spinning the motors in every possible direction for 2 seconds. Main function

```
import RPi.GPIO as GPIO #Import standard GPIO Module
from time import sleep

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False);

# PWM Frequency
pwmFreq = 100

#Setup Pins for motor controller
GPIO.setup(12, GPIO.OUT) #PWMA
GPIO.setup(18, GPIO.OUT) #AIN2
GPIO.setup(16, GPIO.OUT) #AIN1
GPIO.setup(22, GPIO.OUT) #STBY
GPIO.setup(15, GPIO.OUT) #BIN1
GPIO.setup(13, GPIO.OUT) #BIN2
GPIO.setup(11, GPIO.OUT) #PWMB

pwma = GPIO.PWM(12, pwmFreq) #pin 18 to PWM
pwmb = GPIO.PWM(11, pwmFreq) #pin 13 to PWM
pwma.start(100)
pwmb.start(100)
```

Figure 3.7: Code to run motors: Setup of the pins

has been executed and it was proven that motors are working as they should. Next step includes installation of the track and movement of complete vehicle on the ground.

```
## Functions

def forward(spd):
        print("Forward")
        runMotor(0, spd, 0)
        runMotor(1, spd, 0)

def reverse(spd):
        print("Reverse")
        runMotor(0, spd, 1)
        runMotor(1, spd, 1)

def turnLeft(spd):
        print("Left Turn")
        runMotor(0, spd, 0)
        runMotor(1, spd, 1)

def turnRight(spd):
        print("Right Turn")
        runMotor(0, spd, 1)
        runMotor(1, spd, 0)

def runMotor(motor, spd, direction):
        print("Run Motor")
        GPIO.output(22, GPIO.HIGH);
        in1 = GPIO.HIGH
        in2 = GPIO.LOW

        if(direction == 1):
                in1 = GPIO.LOW
                in2 = GPIO.HIGH

        if(motor == 0):
                GPIO.output(16, in1)
                GPIO.output(18, in2)
                pwma.ChangeDutyCycle(spd)
        elif(motor == 1):
                GPIO.output(15, in1)
                GPIO.output(13, in2)
                pwmb.ChangeDutyCycle(spd)

def motorStop():
        print("Motor Stop")
        GPIO.output(22, GPIO.LOW)
```

Figure 3.8: Code to run motors: Functions

```
def main(args=None):
        while True:
                print("Main")
                forward(50)
                sleep(2)
                motorStop()
                sleep(1)

                reverse(50)
                sleep(2)
                motorStop()
                sleep(1)

                turnLeft(50)
                sleep(2)
                motorStop()
                sleep(1)

                turnRight(50)
                sleep(2)
                motorStop()
                sleep(10)

if __name__ == "__main__":
    """ This is executed when run from the command line """
    main()
```

Figure 3.9: Code to run motors: Main

Conclusion and Perspectives

Project has been started with bringing the previously assembled prototype to the working state and performing tests in order to check its abilities. Air module has been equipped with GPS module, external magnetometer and telemetry. More powerful battery has been ordered. Equipment has been tested without propellers, by utilizing QGroundControl, logs saved into SD card and Flight Review online tool. It was concluded that equipment of the aerial module works well and is ready to be tested in flight.

Ground module has been accomplished with the motor driver and all of the connections between electrical components have been established. Setup of the Raspberry Pi has been performed, and Python coding language has been used in order to give command to motors. It was checked that the gearbox is capable of driving the vehicle to the front, backwards and performing turns. However, due to the stiffness of the track, it was not possible to make the vehicle move on ground. Necessary details are going to be ordered and assembled in beginning of the new semester.

After the vehicle is going to be capable of driving and flying, autonomous navigation from one waypoint to another is going to be implemented on both modules. Once vehicle can navigate by itself between the waypoints, two modules are going to be connected with an algorithm, which enables to switch between the ground and air modules. In the final part of the project vehicle should be capable of avoiding obstacles on its path, while choosing if to pass it on the ground or in the air. HC-SR04 Ultrasonic Sensors are going to be used in order to measure the distance to an obstacle and its dimensions. The change of the mode of movement should depend on the time-efficiency and energy consumption.

Bibliography

[1] "Communicating with raspberry pi via mavlink." http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html, 2019.

[2] "Change-4 powers down for second lunar night." https://spacenews.com/change-4-powers-down-for-second-lunar-night/, Feb 2019.

[3] NASA, "Mars helicopter to fly on nasa's next red planet rover mission." https://www.nasa.gov/press-release/mars-helicopter-to-fly-on-nasa-s-next-red-planet-rover-mission, May 2018.

[4] S. CHAUDHARY and N. RAGHUNATHAN, "Autonomous robotic aerial vehicle: S3 project report," March 2019.

[5] "Connect escs and motors (to the pixhawk)." http://ardupilot.org/copter/docs/connect-escs-and-motors.html, 2019.

[6] "Safety switch." http://ardupilot.org/copter/docs/common-safety-switch-pixhawk.html, 2019.

[7] "Px4." https://px4.io, 2018.

[8] "Flight modes." https://docs.px4.io/en/flight_modes/?q=, 2019.

[9] M. Digest, "How to connect the tb6612fng dual h-bridge motor dirver to an arduino and raspberry pi." https://www.youtube.com/watch?v=3LBiyBTnt7g, February 2019.

[10] "Flight review." https://logs.px4.io, 2019.

[11] "Log analysis using flight review." https://docs.px4.io/en/log/flight_review.html, 2019.