Master in Applications and Technologies for
Unmanned Aircraft Systems (Drones)

Transversal project guide

2021-2022

1. Presentation

The transversal project consists in developing an application based on drones, in all its phases, including:

1. Assembly and tuning of the drone platform
2. Payload integration
3. Mission planning
4. Execution of the mission
5. Exploitation of collected data

In this project, which takes place during the first semester of the Master, many of the concepts that will be introduced in the different subjects throughout the semester will be put into practice.

The project is developed in teams of two or three people and is organized in 4 phases:

Phase 1: Assembly, configuration and first flight tests of the platform (3 sessions during the Taking-off month)

Phase 2: Application of concepts developed in the different subjects taught during the first semester, some of whose practices will use the drone platform

Phase 3: Demonstration of the basic mission in the DroneLab (during the month of January)

Phase 4: Implementation of the advanced mission, which delves into some specific aspect of the basic mission (January and February)

The project must be documented by means of a blog that compiles the description of the tasks carried out and the results obtained, including graphic material (photos and videos) and reflections on the learning obtained with the project.

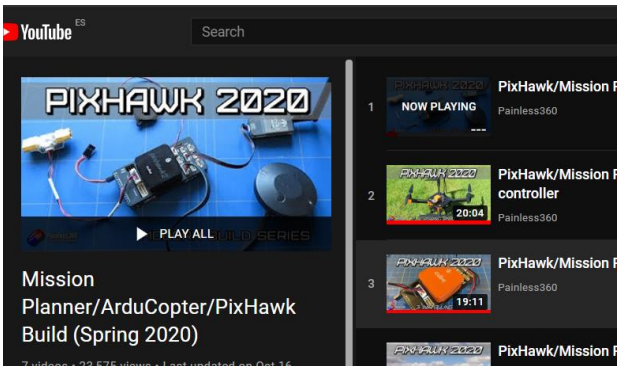
2. The kit

At the beginning of the project, each team will receive a kit with all the necessary components. Figure 1 shows the kit as delivered and the drone platform at the end of phase 1.

Figure 1: The kit as delivered and the drone platform after phase 1

At the end of the project, each team must return the kit in exactly the same condition in which it was received.

The different phases of the assembly and set-up of the drone platform are magnificently documented in a collection of videos, which will be a fundamental material during the project. That collection of videos can be found here:



https://www.youtube.com/watch?v=zr37vSsx7ug&list=PLYsWjANuAm4qJ8Lko-3xiYNGb2Hh22rDw

3. The mission

The drone platform must be ready to perform a basic and advanced missions, that will be demonstrated in the Drone Lab in January/February. The definition of the basic mission is:

“The drone must perform a scan pattern like the one shown in figure 2 to locate the landing pad that you have in your kit. When located, the drone must send its location to the ground station“

Figure 2: Scan pattern for the basic mission

The advanced mission will allow you to delve into specific aspects of your interest. The following are just some examples of possible advanced missions. During the course, we will propose you more possibilities.

Example 1: Instead of placing a single landing pad, we will place three in different locations on the DroneLab. The drone must perform a scan pattern, locate the three landing pads and modify the flight plan to visit each of the detected landing pads at a lower height (emphasis in some contents of the subject Integration in RPAS).

Example 2: Integrate an Intel RealSense 435 or D435i depth camera on the front of the drone to perform the Obstacle Avoidance (emphasis in some contents of the subject Unmaned Aircraft).

Example 3: Perform a landing pad detection using artificial intelligence algorithms instead of image processing techniques (emphasis in some contents of the subject Payload).

4. The blog

To document your work in the transversal project every team will develop a blog using WordPress. Have a look to this example to get an idea of our expectations:

https://uas.upc.edu/g42020/

You will update your blog frequently, as you advance in the project. You will present four versions of the blog, connected with the four phases of the project and the final version in which you will take into account comments and feedback from other classmates, in a peer assessment process. Academic staff will review the four first versions of the blog and give you feedback that will help you to improve. They also will grade the final version and the result will be used to compute the final grade of subjects: Unmanned Aircraft, Payload, System Integration in RPAS and Airspace Structure and Regulation.

Table 1 indicates the dates and expectations for each of these versions. In each version you have to include also your reflections on your learning and on teamwork efficacy.

Version 1	October 18	Your work in phase 1, including assembly, set up and flight test.
Version 2	January 10	Your work in phase 2, applying the concepts developed in the different subjects taught during the first semester.
Version 3	January 31	Results of basic mission demonstration in the drone lab.
Version 4	March 4	Results of the advanced mission
Final version	March 25	Improvements coming from peer assessment

Table 1: The different versions of the blog

5. Activities

To develop the transversal project (specifically phase 1) we have designed a series of activities that are briefly describes in table 2. For each activity you will find a guide in this document, with basic information that will help you during the activity.

			Pag.
1	The very very basics of a drone and its components	A collection of videos to get a wide picture of the elements involved in the project.	6
2	Unboxing	Delivery of the kit and review of its contents	8
3	Radio control	Configuration of the ratio for remote control of the different devices of the drone	11
4	Frame assembly (part 1)	Assembly of the main body, arms and motors of the platform.	18
5	The autopilot	Configuration and first calibrations of the autopilot.	20
6	Frame assembly (part 2)	Installation of different devices in the frame (GPS, radio receivers, buzzer, optical sensor, etc.).	23
7	More calibrations	Final set up, calibration of ESC and motor test.	29
8	Flight tests	First flight test of the platform	31
9	Log analysis	Recovering and analysis of telemetry logs coming from the first flight tests.	32
10	Phase 1 review		33
11	Static flight plans	How to create and simulate flight plans using Mission Planner	34
12	Dynamic flight plans	How to control the mission from a Python script	36
13	Interacting with the real autopilot	Python script that gets some data from the autopilot	40
14	Raspberry Pi basic configuration	Prepare the RPi to connect it with the ground station and the autopilot	45
15	Controlling the button and LEDs	Configure the RPi and Python scripts to control the button and the LEDs	50
16	Installing and using the camera and the OpenCV library	Python scripts to take pictures and basic image processing with OpenCV	53
17	Connecting the RPi with the autopilot	Python scripts running in the RPi to control the autopilot	58

Table 2: Activities to develop the transversal project

Activity #1

The very very basics of a drone and its components

Aim

In this activity you will learn the very very basics of a drone system and its components. At the end, you will have a simple but wide picture before starting the hard work in the transversal project.

Work to do

Watch this introductory short video, which covers the whole picture and introduces the basic components of a drone system.

https://youtu.be/rBuwot8A7ac

The videos in the following table will allow you to delve into most of the concepts and components mentioned in the introductory video. You do not need to watch them right know. But keep in mind that you have here a good collection of introductory videos that may help you during the transversal project.

Radio control	
https://www.youtube.com/watch?v=LyPtzv0y5DE	
ESC	
https://www.youtube.com/watch?v=OZNxbxL7cdc	
Brushless motors	
https://www.youtube.com/watch?v=uLutMoh4Ttg	

Radio control connections

https://www.youtube.com/watch?v=GbUMC8h2uhU&t=8s

Radio configuration

https://www.youtube.com/watch?v=azs1wmr4c7w

Flight controllers vs flight computers

https://www.youtube.com/watch?v=PlKeFj5teo4

Mission Planner

https://www.youtube.com/watch?v=z_sZUOrnfY8&t=3s

Pixhawk autopilot

https://www.youtube.com/watch?v=m7THu-W-kjE&list=PLYsWjANuAm4rSLU7PQczfpQ9B4KbcPWsI&index=1

Flight modes

https://www.youtube.com/watch?v=mSSbjo9pYgM

Activity #2

Unboxing

The kit with all the components and tools that are needed to develop the transversal practice is delivered in a box like the one shown in figure 1. As can be seen, the box contains 13 objects. Figure 2 contains a collection of images that allow each of these 13 objects to be identified. Verify that they are all in your box.

At the end of the transversal project, you must return the kit in exactly the same condition in which you received it, with an explanatory document in the event that any of the components have been lost or damaged. To that purpose, we recommend you to take pictures of the box and its components so that at the end you can check that everything is as it was at the beginning,

The kit contains two more items that are not included in the box, because their use is dangerous. They will be delivered to the teams, under the control of the academic staff, when they are needed. They are shown in figure 3.

One task that you must carry out throughout the project is to build a table that describes each of the components that are part of the kit, briefly indicating its function, how and where it can be purchased and its price, in such a way that using this table anyone could easily acquire what is necessary to develop the project autonomoulsy. This table should be included in your project blog with all the comments you deem appropriate.

Figure 1: The kit as it is delivered to every team

Figure 2: The 13 objects included in the kit

Figure 3: Two additional objects that belongs to the kit but are not included in the box.

Activity #3

Radio control

Aim

In this activity you will learn to configure the radio to control the speed of a motor and the position of a servo. To do this, you will need the components indicated in table 1.

Do not forget to collect material (images or videos) to document your work on the blog.

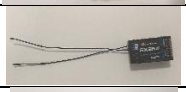
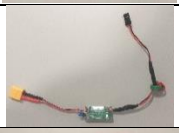
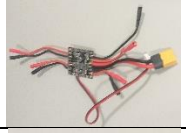
The radio	
The radio receiver	
The cable to supply power to the receiver with voltage level adapter (UBEC)	
Two motors with their speed controlers (ESC)	
A servo	
Power distribution board	
Power battery	

Tabla 1: The components needed for this activity

Basic concepts

The radio can generate radio signals through up to 24 channels. The different channels are multiplexed using the flexible division multiplexing technique.

The signal that is emitted by a certain channel is controlled from one of the radio's actuators: sticks, switches and knobs (S1 and S2 in figure 1). For example, moving a stick can vary the signal that is emitted by a certain channel.

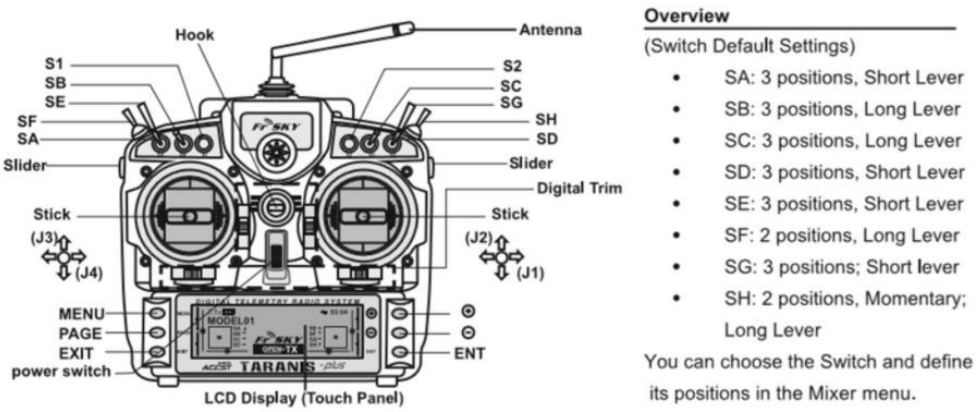


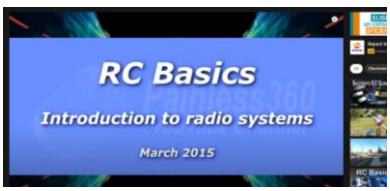
Figura 1: The different actuators of the radio

The multiplexed signal received by the receiver is separated into the different channels and each of the resulting signals is converted into a PWM signal, which is emitted through the receiver output corresponding to the associated channel (up to 8 different channels). See figure 2.

Figure 2: Output channels of the receiver, that emit PWM signals

Each of the PWM signals generated by the receiver can control a device. For example, depending on the width of the pulses of the PWM signal (which will depend on the radiofrequency signal received by the corresponding channel), the motor connected to that channel will rotate at a higher or lower speed, or the servomotor arm will change its position.

To complement this basic explanation, we recommend you to watch this video:



https://www.youtube.com/watch?v=LyPtzv0y5DE

Connecting the devices

In this activity we will connect two motors (on channels 1 and 2) and a servo (on channel 3) to the radio receiver. The necessary connections can be seen in figure 3.

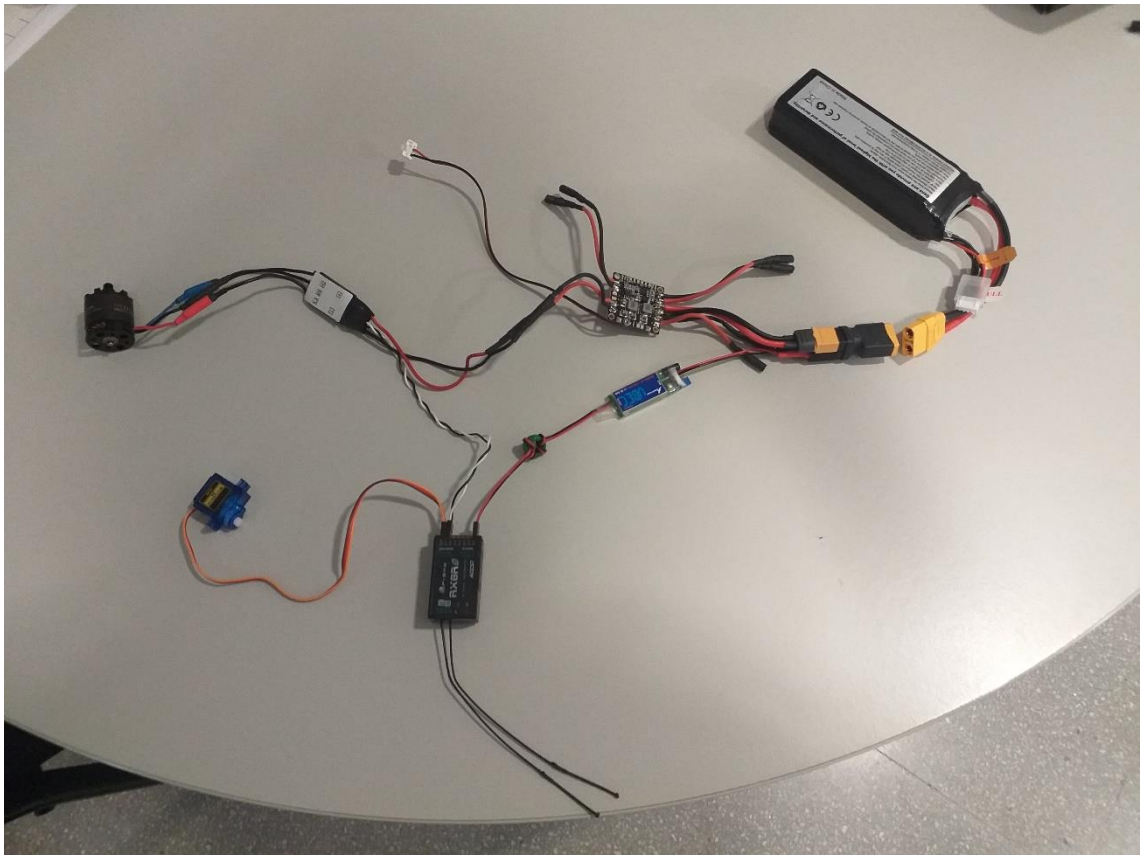


Figure 3: Connection of the different devices involved in this activity

The power distribution board is connected to the battery and provides power to the ESC that controls the motor (only one motor is shown, but you have to connect two). The ESC is connected to the corresponding channel of the receiver, through which it will receive the PWM signal that controls the speed of rotation.

The servo is connected to another of the receiver channels. In this case, the connecting cable, in addition to the PWM signal, also provides power to the servo.

A special cable is used to power the receiver, which is also connected to the battery. In the assembly of the drone frame we will not use this cable since the receiver's power will come from the autopilot.

Radio configuration

Radio station setup involves:

- Associate each of the channels to one of the actuators (lever, switch or knob)
- Determine the shape of the signal to be output, which will determine, for example, the range of motor rotation speeds or the number of different positions that the servo arm can take.

This configuration can be done directly in the radio itself, through the small screen it has, or through the computer using the Open TX Companion software. We will use this second option, which is more comfortable.

After installing the software on your computer, you have to connect the radio to the computer via USB and put the radio in USB mode. Figure 4 indicates how to do this.

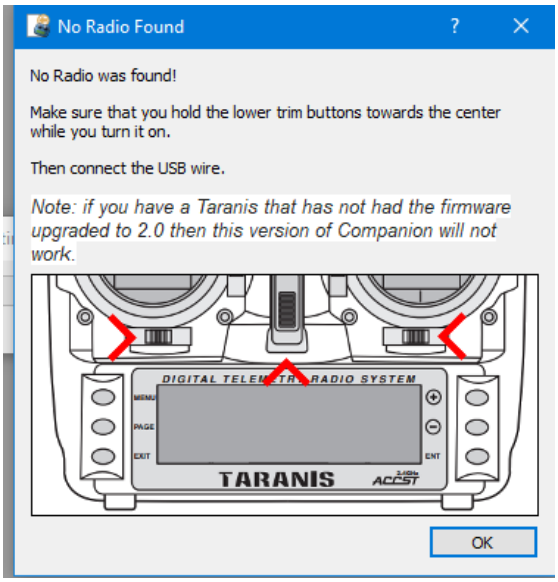


Figure 4: How to put the radio in USB mode

It is important to note that in Open TX companion (and in other tools), although many of the radio's actuators are identified with the same names as in the figure 1, the names indicated in table 2 are also used.

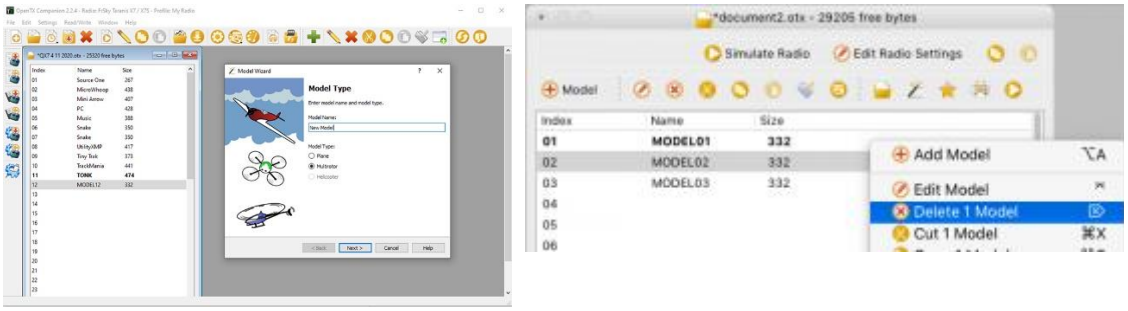
Thr	Vertical movement of left-side stick
Ail	Horizontal movement of right-side stick
Ele	Vertical movement of right-side stick
Rud	Horizontal movement of left-side stick

Table 2: Names used to identify some actuators

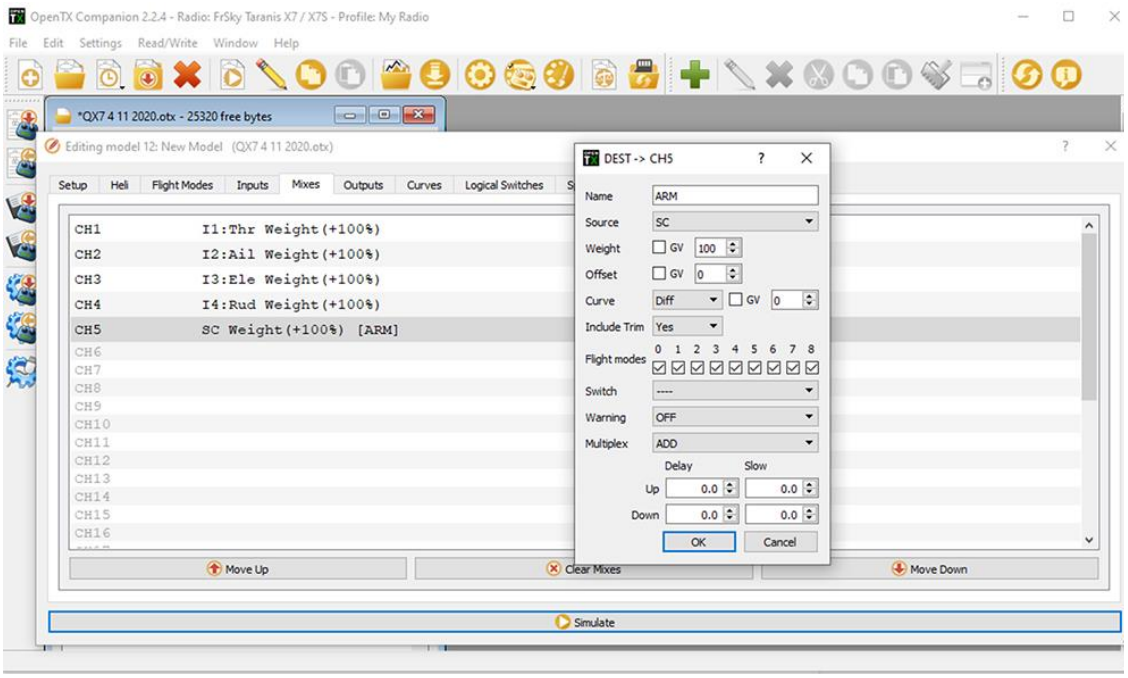
Using Open TX companion it is easy to perform the following operations:

1. Create a new model that we will call (Radio control activity) and that we will use to control the two motors and the servo. At the end of this activity you will create a more complete model to control the drone.
2. Associate the Thr actuator to channel 1
3. Associate the Ail actuator to channel 2
4. Associate the SB switch (which has 3 positions) to channel 3
5. Save the model created into the radio.

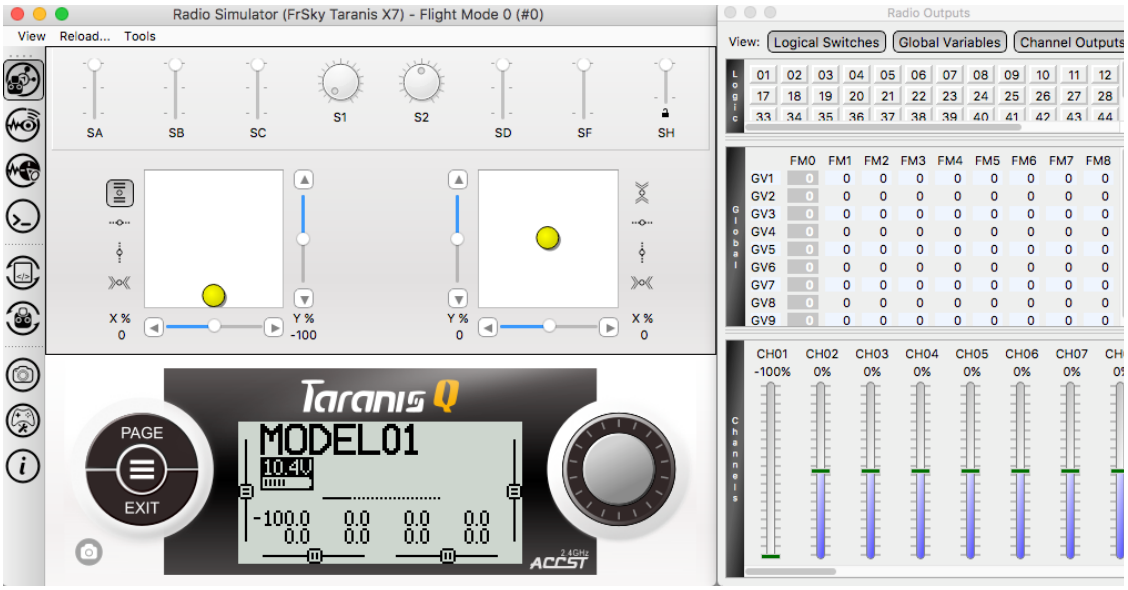
The images shown in figure 5 will help you follow this process.



Creating and editing models



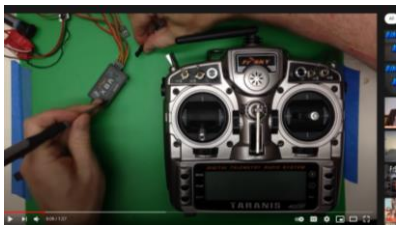
Associate channels to actuators



Simulation to verify configuration

Figure 5: Different steps for radio configuration

Finally, it is necessary to bind the transmitter (with the created model) to the receiver. To do this, follow the instructions that you will see in this video.



https://www.youtube.com/watch?v=YTneUV2TdZM

From that moment on, you can control the motors and the servo from the radio's actuators that have been configured.

Check that the devices respond correctly to the actions you take from the radio. You can experiment with different values of the Weight and Offset parameters of the channels (see figure 5) to see the impact that these changes have on the rotational speeds of the motors and in the positions to which the servo motor moves. Perform all the tests you need until you have a good understanding of how to decide the Weight and Offset values to achieve the desired effect on the controlled devices.

Keep in mind that, even if you change the configuration of any of the elements of the model, it is not necessary to redo the binding between the transmitter and the receiver (which you do have to do if you want to use another model).

Also check if the motors rotate in the direction indicated on the motor itself (with red arrows). If they do not rotate in the correct direction, you will have to change the order of connections between the motor and the ESC.

Configuring a channel with mixes

We are going to make a change in the configuration of channel 3, which controls the servo motor. In the model created, channel 3 is associated with switch SB, which has 3 positions. For that reason, we can only make the servo have three possible positions. We are going to make a change in the configuration of channel 3 to get the servo have 6 positions. For this we have to combine the actuation of 2 switches, for example, the SB and the SF (which has 2 positions). Each combination of positions of those 2 switches (6 combinations in total) will correspond to a different position of the servo.

Figure 6 shows an example of how channel 5 is configured for the described purpose. Two mixes have been associated with that channel. In the first, we specify the output of this channel when actuating on SB. The second indicates the change of these outputs (REPLACE) when the position of the SF switch is changed. In the simulator it can be verified that the different combinations of these two switches generate 6 outputs that cover the range between -100 and +100.

This mechanism will be used to control up to 6 different flight modes from the radio.

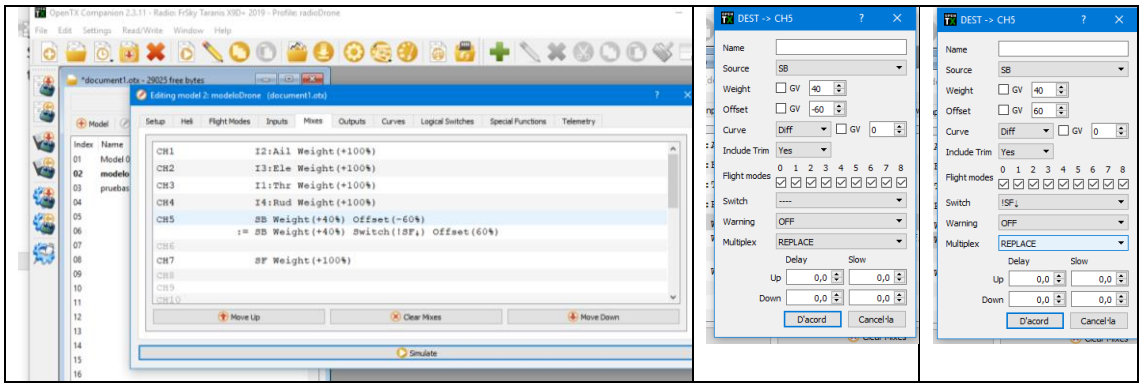


Figure 6: Configuration of a cannel with two mixes

Configuring a model to control the drone

Finally, you must create a new model, which you can call “Our drone”, which will be the one that you will use to control the drone during the transversal project. In that model:

- Channel 1 must be associated with Ail
- Channel 2 must be associated with Ele
- Channel 3 must be associated with Thr
- Channel 4 must be associated with Rud
- In channel 5 a mix must be configured to be able to select between 4 different flight modes, using the SA and SB switches.

Activity #4

Frame assembly (part 1)

Aim

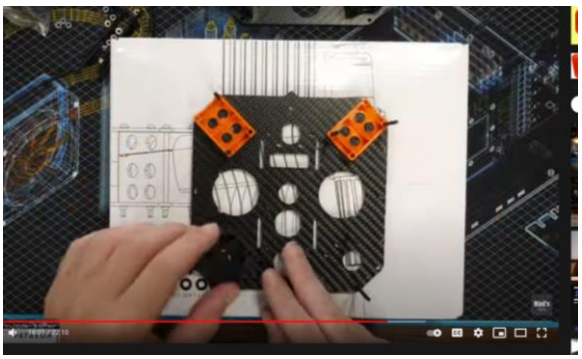
This activity is the first phase of assembly of the frame, which includes the central body (with the power distribution board), and the four arms with the motors. Figure 1 shows the final result of this assembly phase.

Later, in phase 2 of the frame assembly, you will add other elements such as the autopilot, the radio receiver or the GPS receiver.

Figure 1: Result of the frame assemply (phase 1)

The task is quite simple. In the kit you will find all the necessary components (including the set of tools you will need) and a sheet with simple assembly instructions.

In addition, the following video shows (from the middle of the video on) how to make the assembly.



https://www.youtube.com/watch?v=0duMCPiPpWg

Images in figure 2, which correspond to different assembly phases, will also help you.

Finally, look at the first picture in figure 2. Don't forget to add the twisted cable extension to the ESC of each motor. This extension is not shown in the rest of photos and videos, but it will greatly facilitate some operations in phase 2 of the assembly of the frame.

Do not forget to collect material (images or videos) to document your work on the blog.

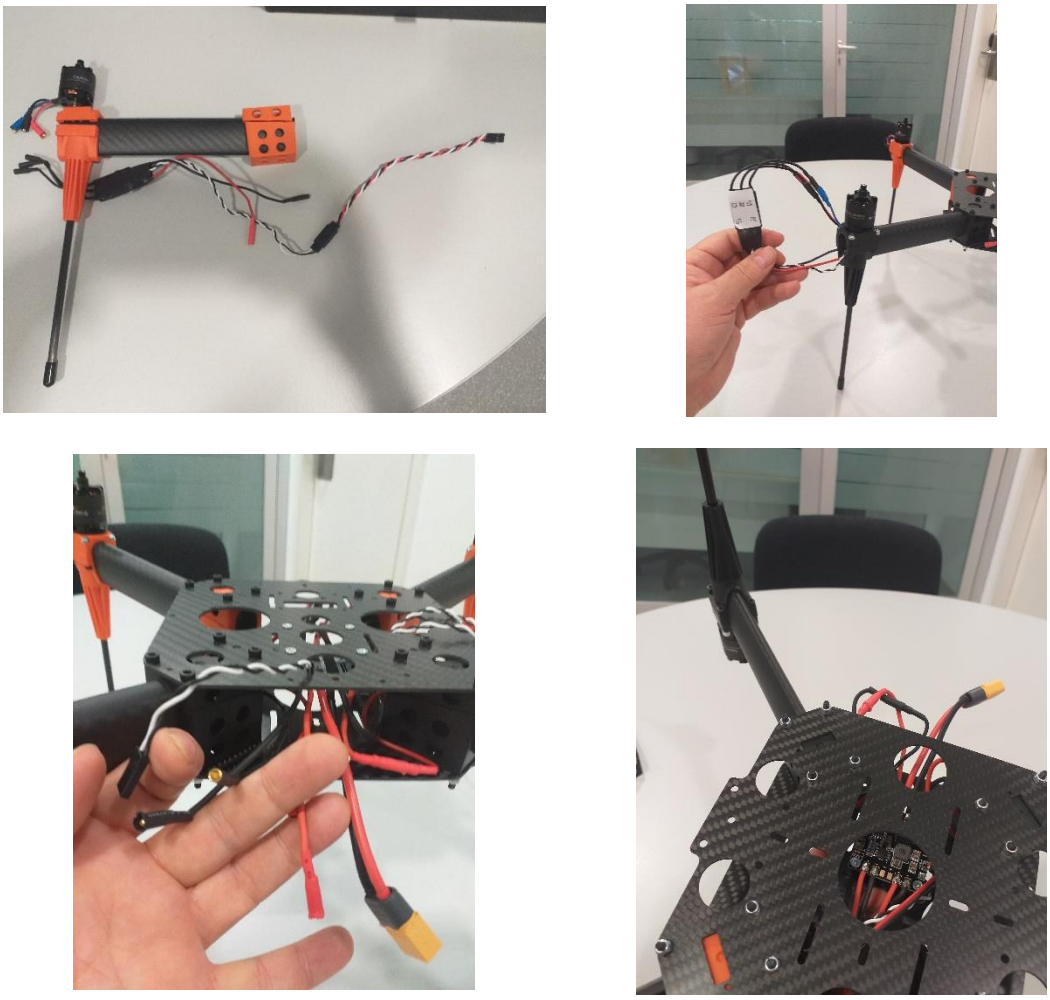


Figure 2: Different pictures corresponding to the assembly process

Activity #5

Autopilot

Aim

In this activity you will learn some basic aspects of the autopilot that we will use in the transversal project. In addition, you will perform some configuration and calibration operations.

The autopilot we're going to use is the Pixhawk Orange Cube, shown in figure 1.

Figure 1: Pixhawk Orange Cube

The different steps for autopilot setup and first calibrations can be seen in this video:

https://www.youtube.com/watch?v=0zTS_ZuHLb0&list=PLYsWjANuAm4qJ8Lko-3xiYNGb2Hh22rDw&index=2

Do not forget to collect material (images or videos) to document your work on the blog.

Basic concepts

The autopilot is a really sophisticated piece of hardware. It is capable of generating the set of PWM signals necessary to perform basic flight operations such as: take off, land, move forward, move left/right or rotate. To do this, it is connected to the radio receiver that will send to the autopilot the orders it receives from the radio station. This connection is shown in figure 2. The radio receiver has an output called SBUS OUT through which it multiplexes the signals received by all the channels (instead of generating a different PWM signal for each channel, as we saw in the a previous activity). That output connects to the RCIN input of the autopilot.

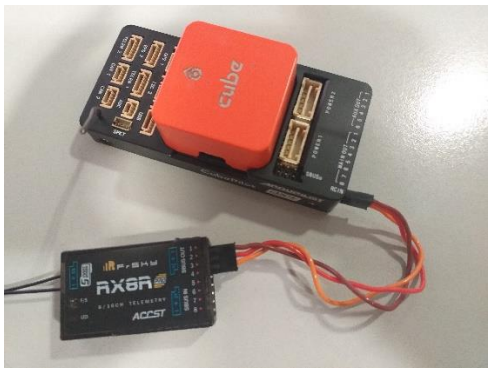


Figure 2: Connecting the autopilot to the radio receiver

Naturally, the orders that we will now give from the radio will not be simple operations like the ones we have seen in activity 2 (for example, accelerate a particular motor by moving one of the sticks). Now, moving the sticks we will indicate to the autopilot more complex operations such as those indicated above (take off, advance, turn, etc.). Specifically, remember that at the end of activity 3 you prepared a model called “Our drone” in which, for example, you associated channel 1 to the Ail actuator. The signals received by the autopilot on channel 1 will be used to control the movement of Role (left/right movement) of the drone. The same happens with the other channels, as indicated in table 1.

Channel	Actuator	Movement
1	Ail	Roll (left/right)
2	Ele	Pitch (forward/back))
3	Thr	Thro (elevation)
4	Rud	Yaw (turn left/right)

Table 1: Configuration of the radio channels to control basic drone movements

The autopilot is also capable of generating the motor control signals to maintain the stability of the drone, depending on the flight mode that has been chosen. For example, in Alt Hold flight mode the drone will keep its height constant as long as the drone operator does not modify it from the radio. To do this, the autopilot will use the information it obtains from internal sensors (such as the barometer, accelerometer, gyroscope) or external sensors (such as GPS). Naturally, at some point it will be necessary to tell the autopilot which flight mode to use, so that it can make the necessary decisions in each case.

Remember that the "Our drone" model, which you configured in a previous activity, allows you to configure up to 4 different modes, acting on the SA and SB switches. In the transversal project we will use the following modes:

Stabilize: Stabilizes in pitch and roll, but the altitude has to be controlled by the operator

Alt Hold: Stabilizes in altitude (and also in pitch and roll) using the internal barometer

Loiter: Stabilizes in altitude (and also in pitch and roll), but using GPS, so you have to fly in sites with GPS coverage.

The autopilot can also execute a flight plan autonomously. In this case, it will be necessary to have a ground station where the flight plan can be configured, sent to the autopilot and then collect the telemetry data that that the autopilot has collected during the flight.

In the transversal project you will use your laptop as a ground station where you will install Mission Planner, which is the most popular ground station software.

Step for the activity

The steps described in table 2 will guide you in this activity. For each step you have the time in the video where you can see how to do the step.

1	Connect the radio receiver, GPS and Buzzer to the autopilot	04:45
2	Install the Mission Planner in your laptop	08:10
3	Install in the autopilot the required firmware	08:30
4	Configure the parameters. After doing that, go to the main menú of Mission Planner and reboot: DATA -> Actions -> PREFLIGHT REBOOT_SHOOT If everything is ok you will see the leds on and ear the buzzer	10:55
5	Preparar el GPS para su calibración Prepare the Compass GPS for calibration	12:42
6	Calibrate the accelerometer	13:25
7	Calibrate the Compass GPS	15:07
8	Calibrate the radio	17:35
9	Configure the flight modes (chose: Stabilizer, Alt Hold y Loiter)	18:40

Table 2: Steps for configuration and first calibrations of autopilot

Activity #6

Frame assembly (part 2)

Aim

In this activity you will finish the assembly of the frame, incorporating (among other elements) the radio receiver, the autopilot to which you will connect the motors, the GPS, optical sensor, the telemetry radio, etc.

Figure 1 shows the frame after this part 2 of the assembly (although the propellers will only be assembled at the time of flight tests).

Figure 1: The frame after part 2

To facilitate the process, we propose in the following section a series of steps to follow. Some of them include images that may be helpful.

In the following video you can find very useful information about some of the steps.

https://www.youtube.com/watch?v=mwJNhkAG2bE&list=PLYsWjANuAm4qJ8Lko-3xiYNGb2Hh22rDw&index=3

Most of the devices will have to be connected to the autopilot. Figure 2 shows the different connection slots of the autopilot, which will be referred to in the description of the steps to follow.

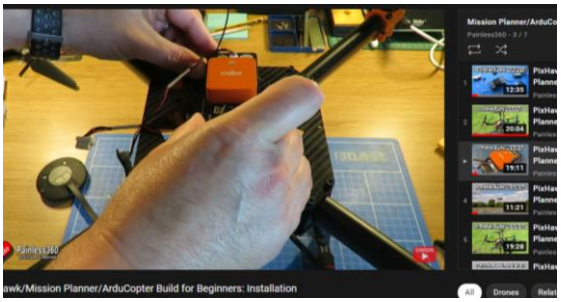
Figure 2: Autopilot connection slots

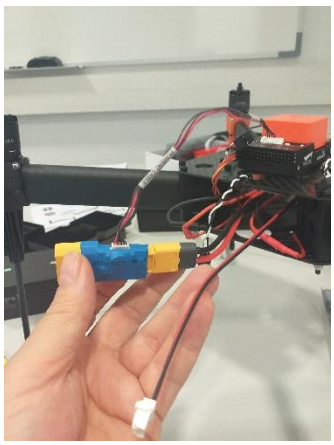
Do not forget to collect the necessary information (photos and/or videos) to incorporate it into your project blog.

The steps

1. Fix the autopilot into the frame

For this step we recommend you to watch the video from minute 4:30.



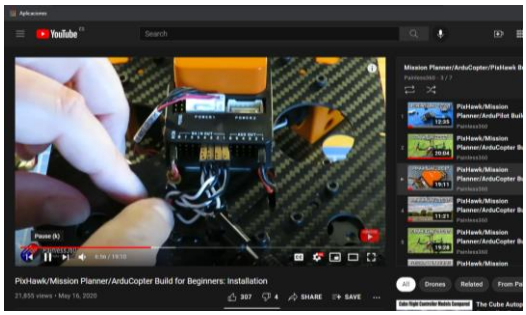


2. Connect the autopilot and the receiver

Choose the place where the radio receiver is placed and connect the SBUS OUT to the RCIN input of the autopilot. Through this connection the autopilot itself provides power to the receiver.

3. Connect the four motors to the autopilot

For this step we recommend you to watch the video from minute 6:25.



It is important to consult this web page to see in which output of the autopilot each motor must be connected.

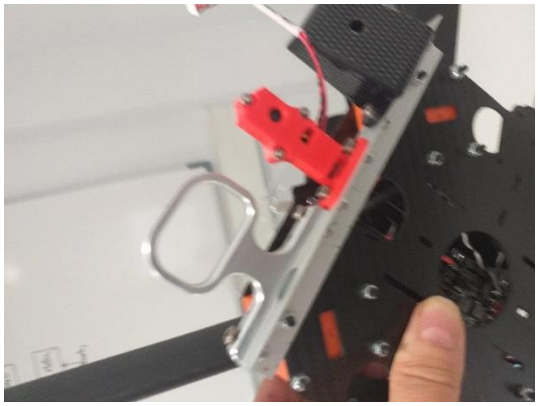
https://ardupilot.org/copter/docs/connect-escs-and-motors.html

4. Mount the transparent plate and the GPS

The transparent plate and the GPS bracket should be mounted together, as shown in the images.

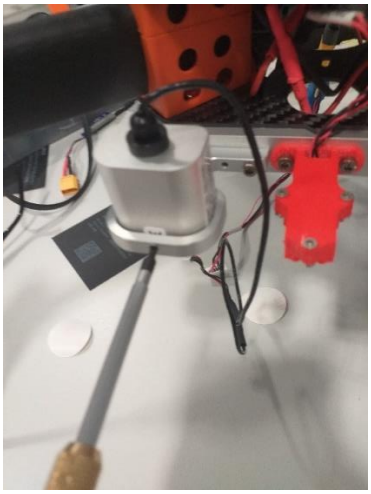
Attention because the GPS must be connected to the CAN 1 input of the autopilot.

4. Install the structure that contains the altimeter



5. Install the optical sensor

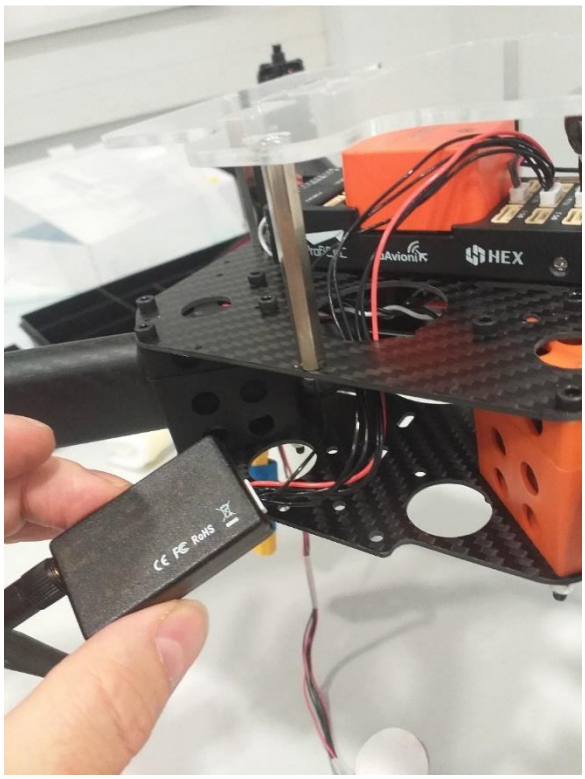
It must be connected to slot I2C 2 of the autopilot.



6. Install the telemetry radio

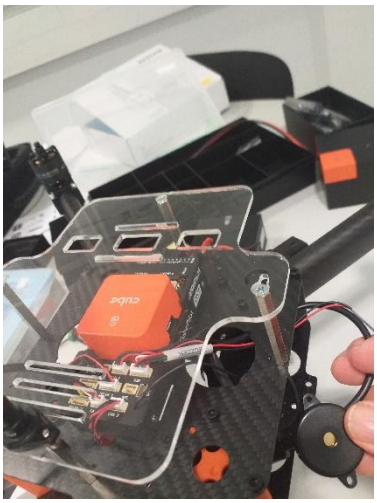
This second mechanism of communication via radio, which we call telemetry radio, will allow the autopilot to communicate with Mission Planner, for example, to receive flight plans or to send telemetry data. Remember that Mission Planner and the autopilot can also be connected via USB, but obviously it will not always be possible to use that means of communication (for example, when the drone is flying).

The telemetry radio should be connected to the Telem 1 input of the autopilot.

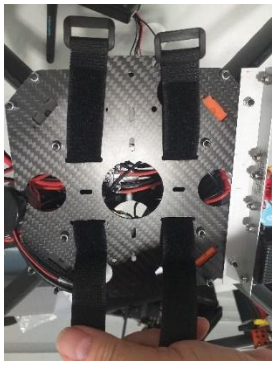


7. Install the buzzer

It must be connected to slot USB of the autopilot.

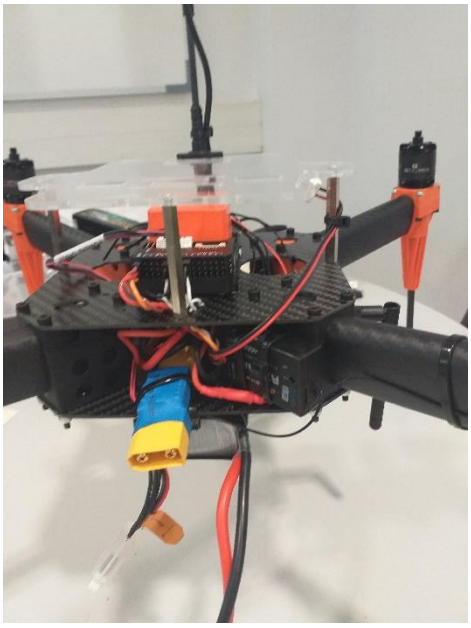


8. Incorporate the straps that will hold the battery



9. Fix all the elements

It is important to secure as best as possible all the elements that have been incorporated into the frame, using velcro tape and zip ties.



Activity #7

More Calibrations

Aim

In this activity you will complete the set up of the platform. Among others, you will calibrate the ESC and test the motors, configure the flight modes and load default parameters for your frame. At the end, the platform will be ready for the first fly test.

Again, this video will help you during the process.

https://www.youtube.com/watch?v=mwJNhkAG2bE&list=PLYsWjANuAm4qJ8Lko-3xiYNGb2Hh22rDw&index=3

Do not forget to collect the necessary information (photos and/or videos) to incorporate it into your project blog.

Step for the activity

The steps described in table 2 will guide you in this activity. For each step you have the time in the video where you can see how to do the step. Look at the clarification notes for some of the steps.

1	ESC calibration (follow the steps exactly as shown in the video)	08:45
2	Configure the flight modes	Note 1
3	Configure safeFail (assign RTL mode)	11:23 Note 2
4	Configure the arm/disarm operation	Note 3
5	Test the motors	13:55 Note 4
6	Configure battery monitor	15:15
7	Load default parameters for our frame	16:05
8	Install the props	17:33

Table 2: Steps for configuration and first calibrations of autopilot

Note 1

Remember that in activity 3 you configured the radio to manage up to 6 different flight modes using switches SA and SB. In this step you tell Mission Planner that you will use three flight modes: Stabilize, AltHold and Loite.

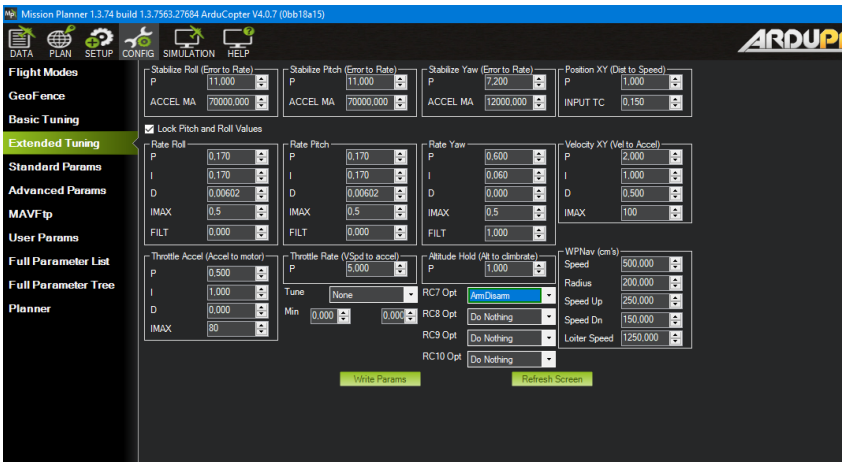
Note 2

safeFail parameter specify what should do the autopilot if the system fails. RTL mode indicates that the dron must return to origin.

Note 3

Arming the drone allows the motors to start spinning. You will have to arm the drone before start flying it. Disarming will cause the motors to stop spinning.

You will not configure the Arm/disarm operation as indicated in the video. We prefer a different procedure. You must modify your model “Our drone” to assign channel 7 of the radio to switch SF, that will be used to Arm/disarm. Then, go to Mission Planner, Config -> Extended tunning to configure RC/ opt -> ArmDisarm, as shown in the figure.



Note 4

This web page will help you in this process.

https://ardupilot.org/copter/docs/connect-escs-and-motors.html

Activity #8

Flight tests

The drone platform resulting from phase 1 of the transversal project is ready. Now, it is time to verify that it can fly correctly.

Each member of the team will fly the drone individually in a small controlled space in the Campus (see figure 1). The flight plan for the test is the following:

1. Elevate the drone 2 meters approx.
2. Move forward 2 meters approx.
3. Move left 2 meters approx.
4. Turn left 180 degrees
5. Elevate the drone 2 additional meters approx.
6. Move forward 2 meters approx.
7. Turn right 90 degrees
8. Go down 2 meters approx.
9. Move backward 2 meters approx.
10. Turn right 90 degrees
11. Land

Do not forget to collect the necessary information (photos and/or videos) to incorporate it into your project blog.

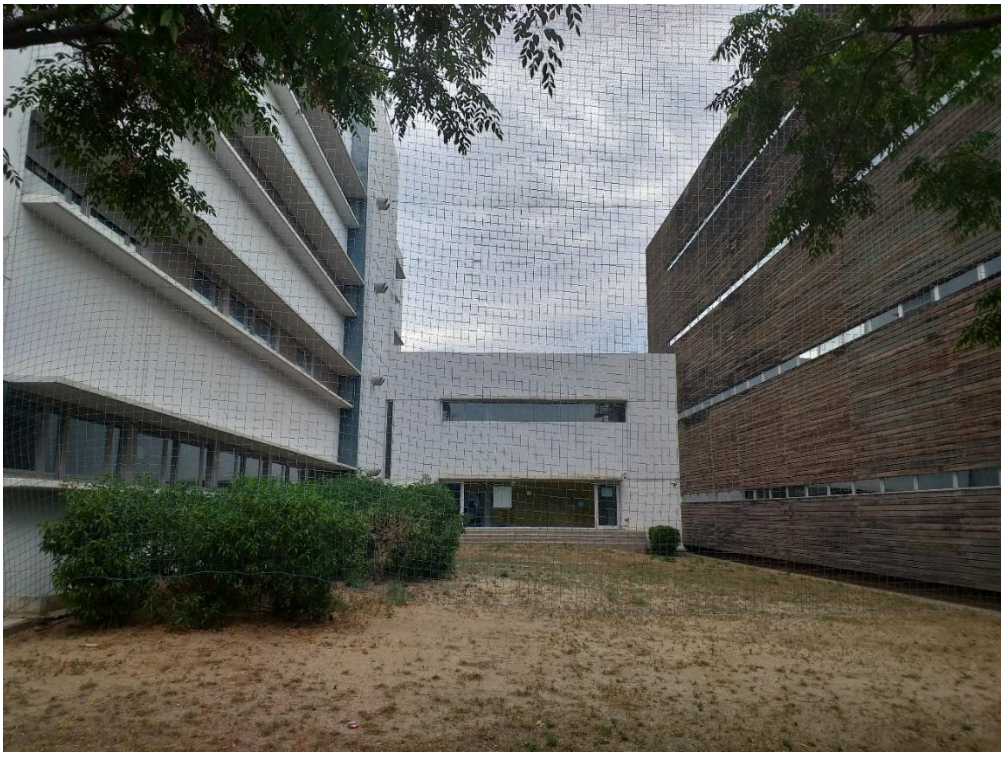


Figure 1: The controlled space in the Campus for flight tests

Activity #9

Log analysis

Aim

During the flight, the autopilot records a large amount of data that can be analysed later. In this activity you will learn how to do that.

Background

There are two types of logs. Telemetry logs (also known as “tlogs”) are recorded by the ground station when you connect the drone to your computer via a telemetry radio. Data is recorded by the ground station during the flight. Dataflash logs are stored on the autopilot during the flight and can be download into the computer after a flight.

You can learn more about both types of log in these web pages:

Tlogs: https://ardupilot.org/copter/docs/common-mission-planner-telemetry-logs.html#common-mission-planner-telemetry-logs

Dataflash logs: https://ardupilot.org/copter/docs/common-downloading-and-analyzing-data-logs-in-mission-planner.html

Both types of logs can be processed using Mission Planner, obtaining a large variety of tables and graphs that make easier the analysis of the drone behaviour. You can learn more on this in this video:

https://youtu.be/xhBd9KSAfzg

To do

After your flight tests, download the dataflash logs into your computer and generate the graphs and tables that show more clearly that your drone behaved properly, according to the flight plan in activity 8. Write a short report with this analysis and include it in your blog.

Activity #10

Phase 1 review

In this activity you will have the opportunity to review the work done during phase 1 of the transversal project.

During phase 1, you may have had difficulty understanding some of the concepts. Or perhaps, being tasks that have been carried out in a team, there is an activity in which you have not participated enough and you prefer to do it again to understand it well.

Of course, it is not possible to disassemble the drone platform and start from scratch. But most of the activities can be repeated without the need to disassemble the platform. In particular, these are the activities can be repeated:

3	Radio control	Configuration of the ratio for remote control of the different devices of the drone
5	The autopilot	Configuration and first calibrations of the autopilot.
7	More calibrations	Final set up, calibration of ESC and motor test.
8	Flight tests	First flight test of the platform
9	Log analysis	Recovering and analysis of telemetry logs coming from the first flight tests.

The activity is voluntary because you may not need to repeat any of the tasks. However, even then, you will be welcome because you may be able to help other colleagues to understand the areas where they are having the most difficulty.

Activity #11

Static flight plans

Introduction

During the transversal project, we are going to consider three ways to perform a drone mission, illustrated in figure 1.

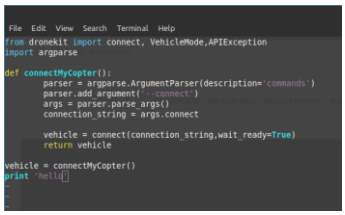
Manual flight: The pilot in command sends the commands to the drone from the radio station. This is what you did in activity 8.

Static flight plan: The flight plan is designed at the ground station (for instance, using Mission Planner) and loaded into the autopilot, which is capable of executing it autonomously.

Dynamic flight plan: The autopilot receives the flight plan instructions from a program (for instance Python + Dronekit library) running on the on-board computer (for instance a Raspberry Py). The program can make decisions regarding the flight plan, at runtime. For example, in case of detection of a particular object through the on board camera, the program can order a particular scan pattern of the area to take detailed images.

(a)

(b)



(c)

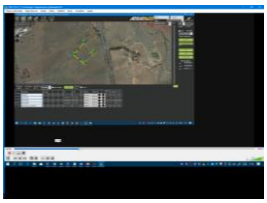
Figure 1: Three ways to fly the drone: (a) manual flight, (b) static flight plan and (c) dynamic flight plan

Mission planner has a simulator, often referred to as the Software In The Loop (SITL), that simulates the autopilot behaviour. This simulator is ideal for training and testing. In particular, we will use the simulator to learn how to create and test our flight plans (both static and dynamic) without the need to fly the drone.

In particular, in this activity you will learn to use the Mission Planner simulator to prepare and test static flight plans.

Preparing a simple mission

What the video to learn how to prepare a very simple plan.



https://youtu.be/YZBvGOybAho

Elaboration

Now that you know how to create and simulate a very simple plan is time to explore the possibilities of Mission Planner. As shown in figure 2, there are many options to prepare a mission and to graph the logs. Explore these options (for instance, how do you change the speed?). Together with your team mates, prepare two or three examples of not so simple flight plans using some of these options and include this material in the next version of your blog.

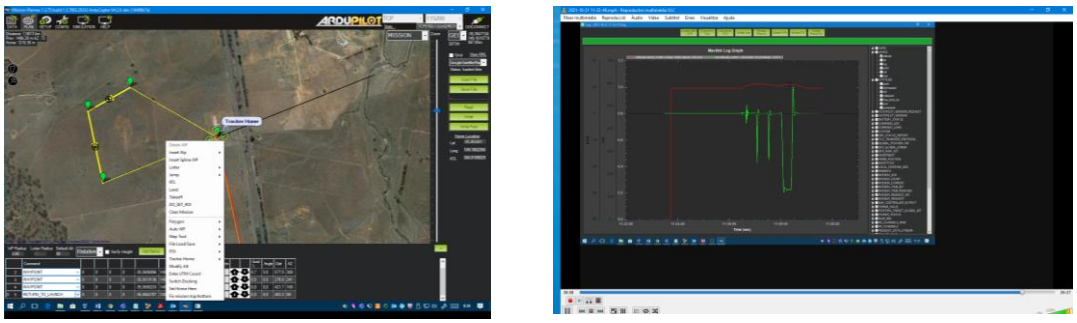


Figure 2: Many options to prepare a static flight plan and to graph the logs

Activity #12

Dynamic flight plans

Introduction

Dynamic flight plan mean that during the mission an on board computer can change the flight plan if certain event is detected through on board devices (camera, sensors, etc.).

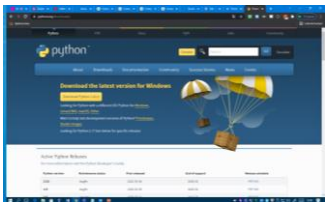
Our drone platform will have a Raspberry Pi. This will be our on board computer. The programs for our computer will be written in Python (using the PyCharm development tool) and these programs will use the DroneKit library to interact with the autopilot (see figure 1).

Figure 1: The technologies and tools that you wlll need to learn to implement dynamic flight plans

In future activities you will install the Raspberry Pi in the platform and connect it to the autopilot and the ground station. By the moment, you will learn to use Phyton + DroneKit to interact dynamically with Mission Planner in simulation mode.

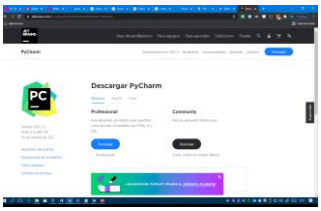
Install Python and PyCharm

This is the first step to do. Here you have the link to download Python.



https://www.python.org/downloads/

PyCharm is our favourite tool for developing Python applications. You can download it here.



https://www.jetbrains.com/es-es/pycharm/download/#section=windows

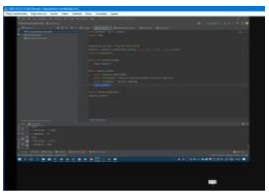
A very simple program

Look at the program shown in figure 2. This is not an example of how to generate a dynamic flight plan, but will be useful to start. Assume that the autopilot is executing a static flight plan (like those you learned to prepare and simulate in Activity 11). The program in figure 2 connect with the autopilot and gets some telemetry data (Altitude and Heading) every 2 seconds while the drone is flying.

```
from dronekit import connect
import time

connection_string = "tcp:127.0.0.1:5763"
vehicle = connect (connection_string, wait_ready = True, baud = 115200)
print ("Conexion realizada")

while not vehicle.armed:

while vehicle.armed:
    print (vehicle.mode.name)
    print ("Altitud", vehicle.location.global_relative_frame.alt)
    print ("Heading", vehicle.heading)
    time.sleep(2)

vehicle.close()
```

Figure 2: A simple example to extract some telemetry data dynamically using DroneKit

Obviously, we must first install the DroneKit in the PyCharm environment so that we can then import it in our programs. Look at this video for a demo of this first example.



https://www.youtube.com/watch?v=JKUTOZ-tBP0

Guiding the drone dynamically

Look know to the code in figure 3. In this case, our program instructs the autopilot to fly to a series of waypoints in sequence. First, note that we have a function to arm and takeoff until reaching the target altitude (this function will be useful for many of your projects). Then the program reads the commands already loaded into the autopilot (in other words, reads a static flight plan). Then extract those commands corresponding to waypoints. Then the program instructs the autopilot to arm and takeoff and ask to go to every waypoint in sequence. Note that we use a very simple function to detect when the drone has arrived to the waypoint (when the current position is equal to the previous position then the drone is stopped and we conclude that it has reached the waypoint). Finally, the program instructs the autopilot to return to lunch.

```
from dronekit import connect, VehicleMode, LocationGlobalRelative
import dronekit
import time
from datetime import datetime
def arm_and_takeoff(aTargetAltitude):
    """
    Arms vehicle and fly to aTargetAltitude.
    """

    print("Basic pre-arm checks")
    # Don't try to arm until autopilot is ready
    while not vehicle.is_armable:
        print(" Waiting for vehicle to initialise...")
        time.sleep(1)

    print("Arming motors")
    # Copter should arm in GUIDED mode
    vehicle.mode = VehicleMode("GUIDED")
    vehicle.armed = True

    # Confirm vehicle armed before attempting to take off
    while not vehicle.armed:
        print(" Waiting for arming...")
        time.sleep(1)

    print("Taking off!")
    vehicle.simple_takeoff(aTargetAltitude)  # Take off to target altitude

    # Wait until the vehicle reaches a safe height before processing the goto
    #  (otherwise the command after Vehicle.simple_takeoff will execute
    #   immediately).
    while True:
        print(" Altitude: ", vehicle.location.global_relative_frame.alt)
        # Break and return from function just below target altitude.
        if vehicle.location.global_relative_frame.alt>=aTargetAltitude * 0.95:
            print("Reached target altitude")
            break
        time.sleep(1)


def compare_location ( previous, current):
    if previous.lat == current.lat and previous.lon == current.lon:
        return True
    else:
        return False


connection_string = "tcp:127.0.0.1:5763"
vehicle = connect (connection_string, wait_ready = True, baud = 115200)
print ("Connected")

print ('Reading static flight plan commands')
cmds = vehicle.commands
cmds.download()
cmds.wait_ready()
print ('Commands')
for cmd in cmds:
    print (cmd)

raw_input("Press Enter to continue...")
```

```
arm_and_takeoff(10)
print ('Selecting waypoints')
waypoints = [x for x in cmds if x.command == 16]
print (waypoints)
vehicle.airspeed = 3

for wp in waypoints:
    print("Going ...")
    print (wp)
    point = LocationGlobalRelative(wp.x, wp.y, wp.z)
    vehicle.simple_goto(point)
    time.sleep(1)
    arrived = False
    previousPosition = vehicle.location.global_frame
    time.sleep(1)
    while not arrived:
         arrived=compare_location(previousPosition,
              vehicle.location.global_frame)
        previousPosition = vehicle.location.global_frame
        time.sleep(1)
    print ("arrived")

print("Returning to Launch")
vehicle.mode = VehicleMode("RTL")
time.sleep(30)
vehicle.armed = False
```

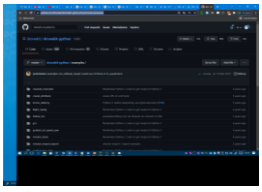
Figure 3: An example showing how to guide the drone through a series of waypoints

See the next video for a demonstration of this example.

https://www.youtube.com/watch?v=YoW7KvPWv-E

Elaboration

Investigate other possibilities of dronekit. For instance, look at the DroneKit site in GitHub:



https://github.com/dronekit/dronekit-python/tree/master/examples

You will find there many examples. Look at the code of these examples for inspiration and practice with some of them. Meet your team mates and try to prepare two or three interesting examples of your own creation and include them in the next version of your blog.

Activity #13

Interacting with the real autopilot

Introduction

In Activity 12 you learned to prepare a Python program that uses the DroneKit library to interact with the autopilot, but in simulation mode.

In this activity, we will go one step further. Now your program will interact directly with the real autopilot, to get some simple data. As in the previous activity, the program will continue to run on your computer to which you will connect the drone platform via USB. Therefore, the program will only get data from autopilot while the drone is stopped on your table. You can already foresee that in future activities you will learn to install your programs on the Raspberry Pi so that they can interact with the autopilot during the flight.

Do not forget to take notes, images, videos for your blog

Some background

As shown in figure 1, the autopilot has two parts: the orange cube (which is really the autopilot) and the board to distribute the connections. This board allows external devices to be connected to the autopilot's serial ports (in addition to the power supply). The most important connections are described in table 1.

Figure 1: The cube and the distribution board

USB	SERIAL0
Telem 1 (telemetry radio)	SERIAL1
Telem 2	SERIAL2
GPS	SERIAL3
ADB-S	SERIAL5

Table 1: Connections to the different serial ports of the cube.

Through SERIAL1 and SERIAL2, the autopilot exchanges information packets with connected devices (for example, with Mission Planner via telemetry radio). These information packets use the MAVLink protocol, and contain all kinds of information related to the autopilot (location, battery status, heading, etc.). You can find more information on this here:

https://ardupilot.org/copter/docs/common-telemetry-port-setup.html#common-telemetry-port-setup

https://docs.cubepilot.org/user-guides/autopilot/the-cube-module-overview#peripheral-ports

To carry out this activity we are going to connect the computer to Telem2 (SERIAL2) using the cable in the figure 2. It will be through this connection that our Python program will interact with the autopilot using DroneKit Python library.

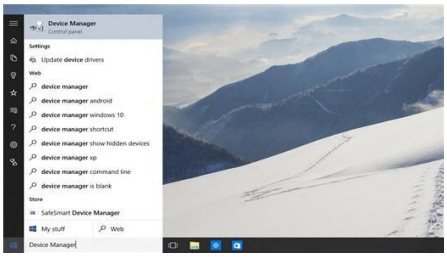
Figure 2: The cable you will need to connect your computer to the autopilot in this activity

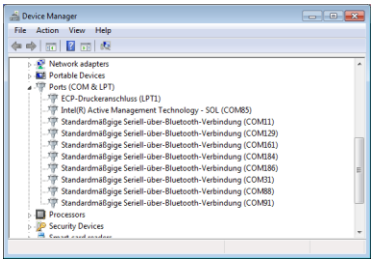
Step for the activity

The steps described in table 2 will guide you in this activity. The collection of images in figure 3 will help you. Look at the clarification notes for some of the steps.

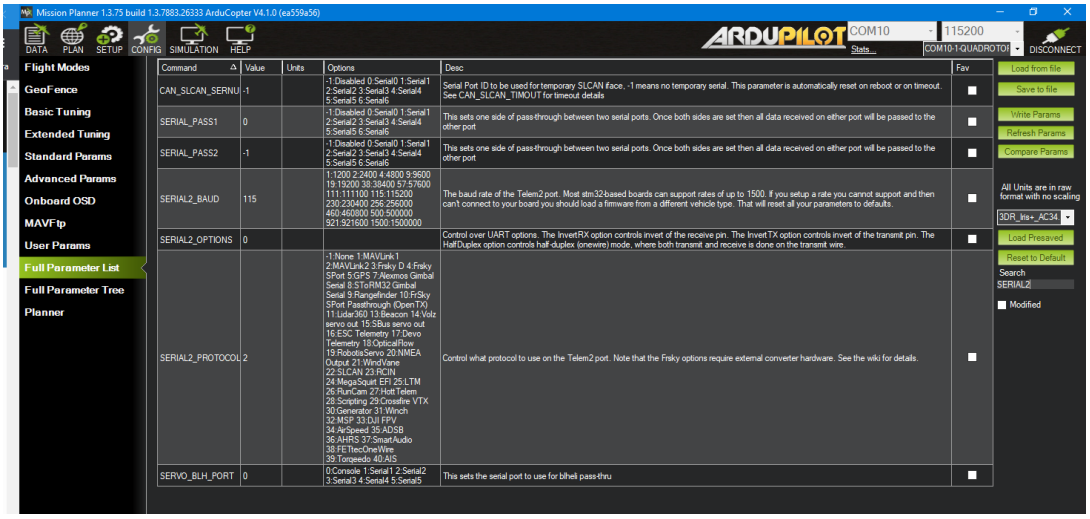
1	Connect your computer with the autopilot via USB (as usual) and also to Telem2 using the cable in figure 2.	
2	Identify in your computer the port where Telem2 has been connected	Figure 3.1
3	In Mission Planner configure SERIAL2_BAUD to 115. This will be the speed for the connection.	Figure 3.2
4	Reboot the autopilot: DATA -> Actions -> Preflight_Reboot	
5	Prepare a simple program in Python like the one shown in the figure. Note the change in the connection string.	Figure 3.3
6	Run your program. See below notes for two possible run errors	

Table 2: Steps for the activity





3.1



3.2

```
from dronekit import connect
import time

connection_string = "com22" # This is the port identified in step 2

vehicle = connect (connection_string, wait_ready = True, baud =
115200)
print ("Connected")
print(" Autopilot Firmware version: %s" % vehicle.version)
print(" Global Location: %s" % vehicle.location.global_frame)
print(" Velocity: %s" % vehicle.velocity)
print(" Battery: %s" % vehicle.battery)
print(" Heading: %s" % vehicle.heading)
print(" Is Armable?: %s" % vehicle.is_armable)
vehicle.close()
```

3.3

Figure 3: Images to support different steps of this activity

Note on possible error 1

Probably in the first try to run the program you will have a run time shown in figure 4. The system misses a library to manage serial port communication. You must include library pyserial, as shown in figure 5.

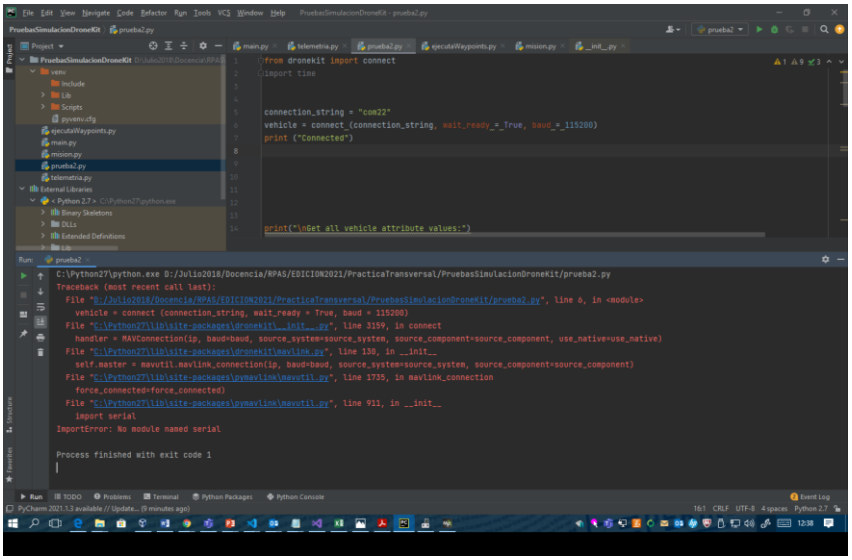


Figure 4: Missing pyserial library

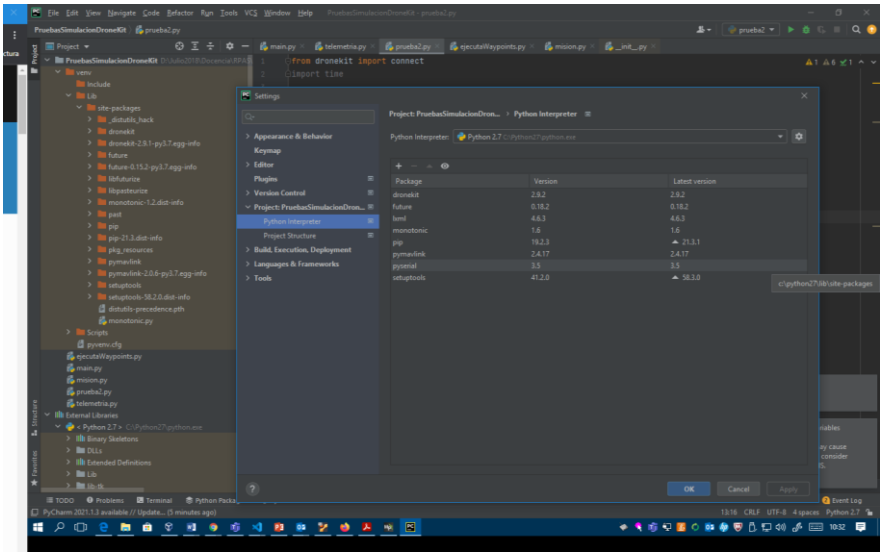


Figure 5: including pyserial library

Note on possible error 2

Figure 6 shows a possible second run time error. In this case, error indicates that DroneKit is receiving some information that it does not know how to interpret. Indeed, the DroneKit library (which has not been updated recently) is not able to interpret the information it receives (in MAVLink packets) from the ADB-S antenna, which is connected to the SERIAL5 of the autopilot. To solve this problem, it is necessary to disable SERIAL5 so that the autopilot does not include this information in the MAVLink packets.

The operation is similar to the one shown in the figure 3.2, but now we must configure SERIAL5 to indicate that SERIAL5_PROTOCOL is -1 (and then reboot again the autopilot with DATA -> Actions -> Preflight_Reboot).

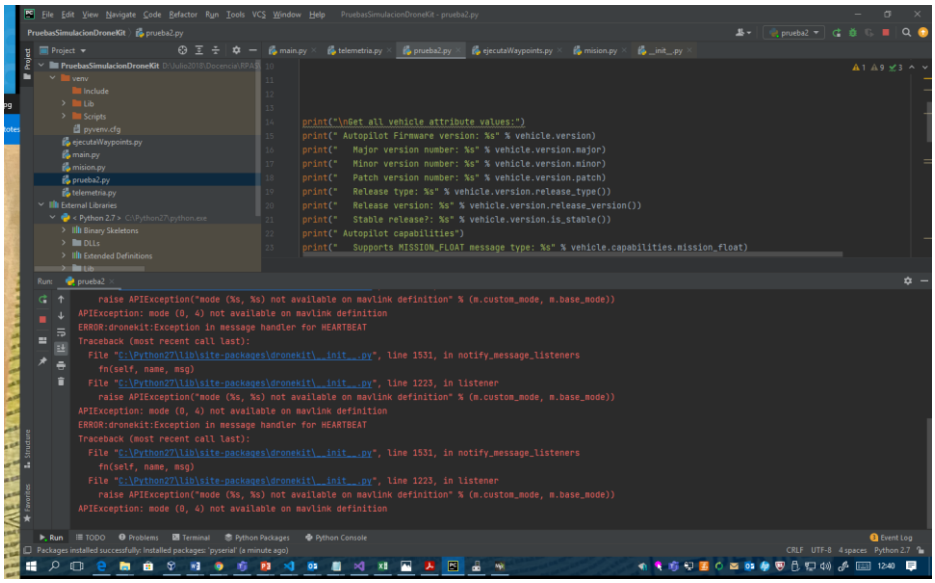


Figure 6: Current version of DroneKit cannot interpret information comming from ADB-S (SERIAL5)

Open question

You may have wondered why we need a new connection between the autopilot and your computer through TELEM1(SERIAL2) if there are already two connections: one through the USB connector (SERIAL0) and the other through TELEM1 (SERIAL1), corresponding to the telemetry radio.

See if you can get the autopilot data with the same Python program, but using those existing connections.

Activity #14

Raspberry Pi basic configuration

Introduction

Our drone platform includes an on-board companion computer, which is a RaspBerry Pi (RPi) and a camera. These devices are included in the RPi kit that you will find in the box (see figure 1). The RPi will run python programs that will interact with the platform's devices. For example, during the flight, our program can capture the images from the camera, process them using, for example, the OpenCV library, and send commands to the autopilot to change the flight plan if a particular object has been detected in the image. Naturally, the program will be developed (and perhaps tested) on your computer, which must be able to connect with the RPi to send it the program and to download files generated during the mission.

Figure 1: The RPi kit

Before integrating the RPi (and the camera) into the drone platform, it is necessary to configure it so that it can be connected with our computer to install there the libraries required to run our python programs. In this activity, you will learn how to perform these configuration tasks.

Communications

As shown in figure 2, the RPi can act simultaneously as a wireless access point and as a Wi-Fi client. Our computer will be able to connect to the RPi through the wireless access point. In this way, we will be able to login in the RPi and also to transfer files. On the other hand, the RPi can connect to a router to access Internet. This connection will be necessary, for example, in the RPi

configuration process, when we have to install the libraries that our programs will need. The internet connection may also be necessary during the flight, for example, to send to the cloud the images that are being captured with the camera, telemetry or other sensor data.

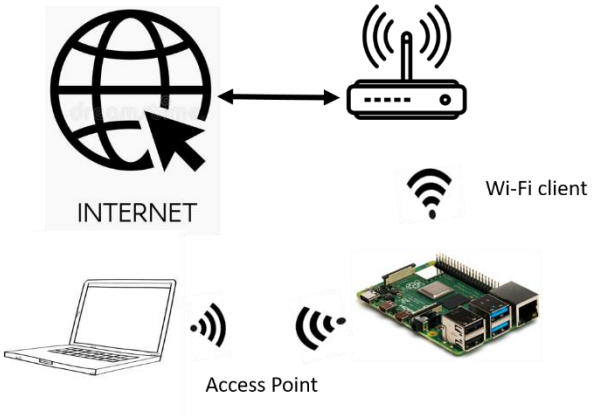
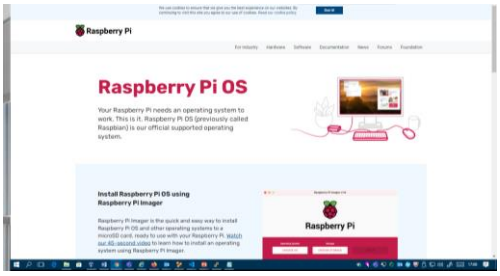


Figure 2: The RPi provide an access point and can operate as Wi-Fi client

Installing the operating system and initial configuration

The first step in the RPi setup process is to install the Linux operating system (Ubuntu 64-bit distribution). The operating system will be loaded onto the microSD card from our computer. To take this step, we will use the Raspbery Pi Imager, that can be downloaded here:



https://www.raspberrypi.org/software/

When running the RPi Imager we will be able to select the OS version and install it onto the microSD, that we must have inserted in our computer (see figure 3). Note that you should install the **Ubuntu-Server 20 (LTS) 64-bit image**.

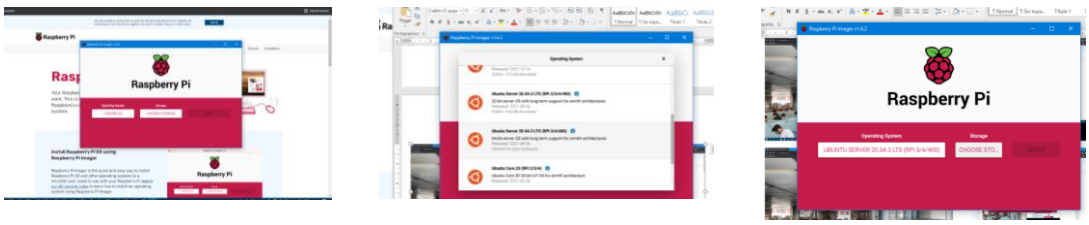


Figure 3: Installing the Ubuntu image into the microSD

Finally, we must insert the microSD into the RPi, as shown in figure 4.

Figure 4: Inserting the microSD into the RPi

The next step is to log in into the RPi to prepare the files necessary to configure the connections (access point and Wi-Fi client). To do this, you need to connect the RPi to three different devices (see figure 5):

- To a keyboard, through one of the RPi USB ports
- To a monitor with HDMI output to the microHDMI input
- To a router that provides internet access, through the Ethernet cable.

Naturally, the RPi must also be powered. Your teachers will facilitate the access to the different devices you need.

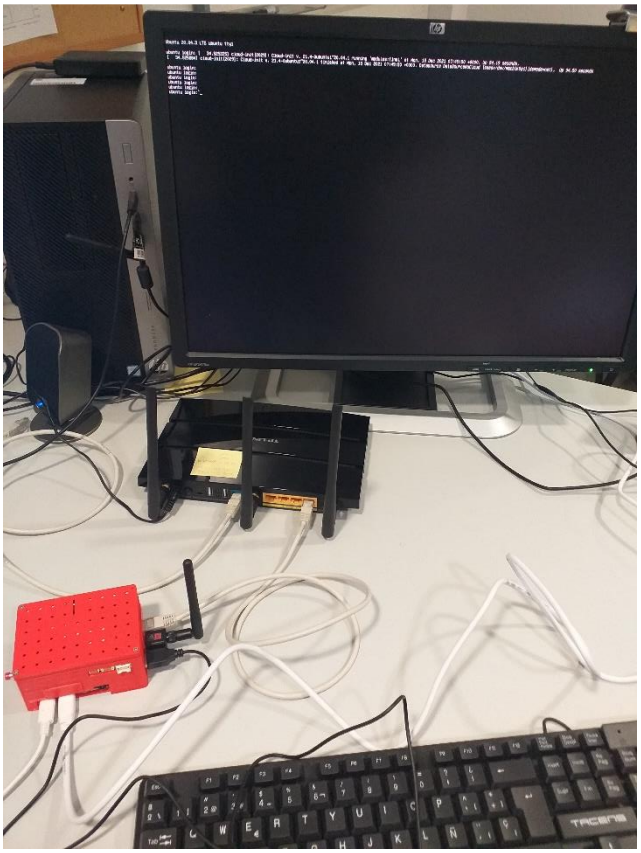


Figure 5: Connections requited for the first log in into the RPi

The credentials for the initial log in into the operating system are as follows:

```
Username: ubuntu
Password: ubuntu
```

As soon as you log in for the first time, you will be prompted to change the password.

Once logged in, you need to run the following commands, some of them requiring internet access (sudo allows you to run the commands with root privileges):

```
sudo apt update
sudo apt install network-manager
sudo bash -c "echo 'network: {config: disabled}' > /etc/cloud/cloud.cfg.d/99-disable-
network-config.cfg"
```

Then, you must edit the network configuration file:

```
sudo pico /etc/netplan/50-cloud-init.yaml
```

This file must have the following content, as shown in figure 6. In this file some parameters are established for the connection between the RPi and our computer via Ethernet cable (which, for now, will not be necessary).

```
network:
    renderer: NetworkManager
    ethernets:
        eth0:
            dhcp4: true
            optional: true
            addresses:
                - 10.0.0.1/24
    wifis:
        wlan1:
            addresses:
                - 10.10.10.1/24
            optional: true
            access-points:
                "AAAAA":
                    password: "BBBB"
                    mode: ap
        wlan0:
            dhcp4: true
            access-points:
                "CCCCC":
                    password: "DDDDDDD"

    version: 2
```

Figure 6: Contents of the file required to configure the RPi internet connections

On the other hand, the access point (`wlan1`) is configured. You must decide the name that this access point will have ("AAAAA") and the password ("BBBB") to be used by any device that wants to connect to the RPi using that access point. Take into account that the password ("BBBB") must be an alphanumeric code with at least 8 characters.

Finally, the connection with a router that provides Internet access to the RPi (`wlan0`) is configured. We must specify the credentials ("CCCCC" and "DDDDD") for the router to be used. Your teachers will give you credentials for a router when you need them.

Take into account that you must follow the exact format of the yaml file and its indentation, as shown in figure 6. Do not use tabs for indentation, each new level is prepended by four spaces.

The following commands are required to activate the configuration:

```
sudo netplan generate
sudo netplan apply
sudo reboot
```

From this moment on, any device with a Wi-Fi connection will locate the access point of the RPi and will be able to connect by Wi-Fi and log in via SSH (for log in) or via FTP (for file transfer) at

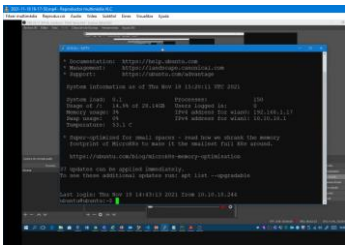
'**ubuntu@10.10.10.1'**. To do that, you will need an SSH and FTP client installed in your computer. PUTTY is a good option.

In addition, if the RPi has been connected to a router, we will see that the connected device has access to the internet through the RPi.

In case the connection with the RPi via SSH fails, you may need to run these commands:

```
sudo apt-get purge openssh-server
sudo apt-get update
sudo apt-get install openssh-server
```

Watch this video for a demo on how to log in into the RPi, how to use some basic Linux commands that will be useful and how to transfer files via FTP.



https://youtu.be/AaKHinsZ0LY

Installing Python

You will need a Python interpreter to run your programs in the RPi. These are the commands you need to install it:

```
sudo apt update
sudo apt upgrade
sudo apt install python3 –y
sudo apt install libgl1-mesa-glx
sudo apt install python3-pip python3-dev
```

Now, it is time for you to create and run your first "Hello world" in Python in your RPi.

```
python3 hello.py
```

Activity #15

Controlling the button and LEDs

Introduction

The module (red box) containing the RPi also includes various peripherals. Among others, it has a button and 5 RGB LED lights (see figure 1). These peripherals can be controlled from a python program and will be useful in our missions. For example, the LED lights can indicate if the RPi has an internet connection or inform about the status of the battery. The button can be used to instruct our program to start performing some pre-flight operations.

In this activity we are going to learn to connect the LED lights and the button to the RPi and to prepare programs that interact with these peripherals.

Figure 1: LEDs and button in the RPi module

Pinout

The RPi 4 has 40 pins (GPIO) available to interact with peripherals. Figure 2 shows the position of the pins on the board and the pin configuration.

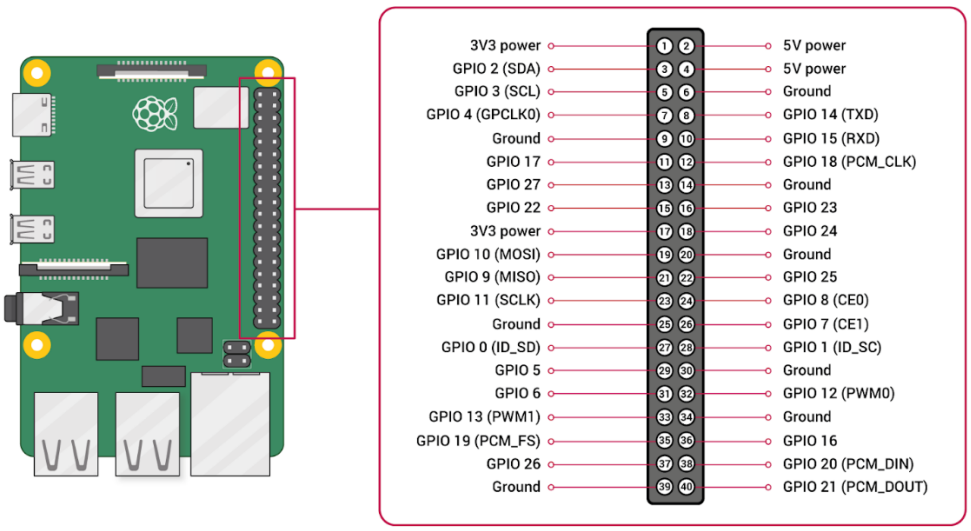


Figure 2: Position of the pins on the board and pin configuration

Some of the pins have a specific purpose (for example pins that supply 5 volts, or pins for serial communication). But several are general purpose and we can use them to control the LEDs and the button.

As figure 3a shows, the LEDs are controlled with 4 signals: ground (GND), control signal (DIN), 5 vols (5VDC) and again ground. On the other hand the button is controlled with two signals: ground and control. Therefore, in order for our programs to interact with the LEDs and the button, it is necessary to connect the pins of these peripherals with the appropriate pins of the RPi. For example, the pin of the LEDs that requires 5 volts can be connected to pin 2 of the RPi and the control pin can be connected to any of the general purpose pins (for example pin 12, which corresponds to GPIO18). Figure 3b shows just those two connections. The remaining connections are made in the same way (in the case of the button, it does not matter which pin the control and ground are connected to, because the button simply closes the circuit).

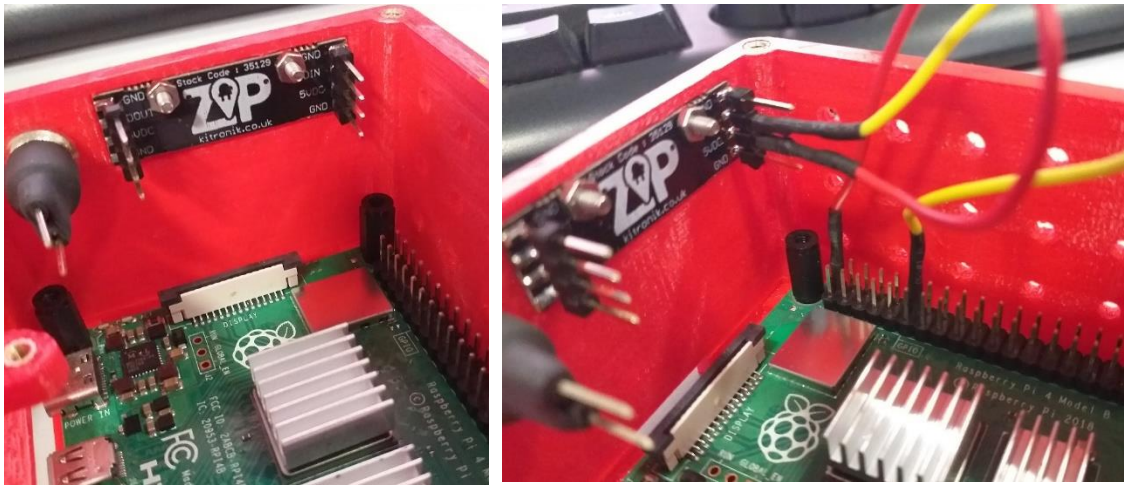


Figure 3: (a) Pins to control the LEDs and the button. (b) Two of the connections (5V and control) required to operate the LEDs.

Once the necessary connections have been made, the library that will allow us to control the peripherals from our python programs (the gpiozero library) must be installed in the RPi.

```
sudo apt update
sudo apt install python3-gpiozero
```

Now we can create and run programs that interact with the LEDs and the button. Figure 4 shows two simple examples that illustrate how to do this. The program in figure 4a writes something in the console each time the button is pressed, assuming that the control signal for the button has been connected to pin 03 (GPIO2). In the example shown in figure 4b, when the button is pressed, three of the LEDs light up sequence every two seconds, each showing a different color. The control signal for the LEDs has been connected to pin 12 (GPIO18).

It is important to note that these programs must be run with root privileges:

```
sudo python3 examples.py
```

Do not be surprised if you need to install new libraries in order to run example 4b.

```
sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
```

```
sudo python3 -m pip install --force-reinstall adafruit-blinka
```

```
from gpiozero import Button
from signal import pause

def say_hello():
    print ("Hello")

button = Button (2)
print ("Press button")
button.when_pressed = say_hello
pause()
```

(a)

```
from gpiozero import Button
from signal import pause
import board
import neopixel
import time

def lights():
    print ("Button pressed")
    #red
    pixels[0] =(255,0,0)
    time.sleep(2)
    pixels[0] = (0,0,0)
    #green
    pixels[1] = (0,255,0)
    time.sleep (2)
    pixels[1] = (0,0,0)
    #blue
    pixels[2] = (0, 0, 255)
    time.sleep (2)
    pixels[2] = (0,0,0)


pixels = neopixel.NeoPixel (board.D18,5)

button = Button (2)
print ("Press button")
button.when_pressed = lights
pause()
```

(b)

Figure 4: Two simple examples demonstrating the use of gpiozero to control the button and the LEDs

You can explore all the possibilities of gpiozero library here:

https://gpiozero.readthedocs.io/en/stable/index.html

Try to develop and interesting and original example and include it into the blog of the transversal project.

Activity #16

Installing and using the camera and the OpenCV library

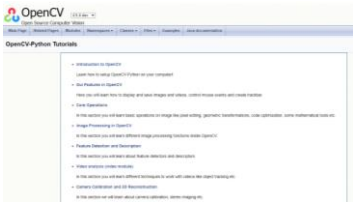
Introduction

The RPi kit contains a camera that will be installed on the drone platform and connected to the RPi. In this way, images can be taken during the flight and processed on board. Our programs will then be able to make flight plan decisions based on the content of the images captured by the camera.

For all this to be possible, it is necessary to learn to process images, using the OpenCV library, learn to connect the camera to the RPi and finally learn to write programs that capture images and videos. That is the content of this activity.

Installing OpenCV

OpenCV stands for Open Computer Vision. The library includes functions for motion detection, object recognition or 3D reconstruction from images. Today is the most popular library for machine vision. You can find more information here:



https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

To install OpenCV in the RPi you need to run the following commands . Be patient. This will take a few minutes.

```
sudo apt update
sudo apt upgrade –y
sudo apt install libgl1-mesa-glx
sudo apt install python3-pip python3-dev
sudo pip3 install -U opencv-contrib-python
```

A simple program

Figure 1 shows a simple code to test that the OpenCV installation is correct. The program simply loads an image and saves it in another file but in grayscale.

```
# import the cv2 library
import cv2

# The function cv2.imread() is used to read an image.
img = cv2.imread('example.jpg')

# The image is converted to gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Save the image
cv2.imwrite('examplegray.jpg',gray)
```

Figure 1: A simple program to test OpenCV

Run this program in your RPi. You will have to put to the RPi via FTP the initial image and get the resulting image.

The OpenCV library has very powerful functionalities. For example, the program shown in figure 2 is capable of detecting people's faces in a picture and frames each of the faces with a rectangle. To do this, the program uses a classifier already trained to recognize faces. Check that the program works correctly (you will have to put in the RPi an image with human faces).

```
import cv2

face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

img = cv2.imread('faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)

print ("number of faces ", len(faces))

for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

cv2.imwrite('facesdetected.jpg',img)
```

Figure 2: A program to identify human faces in a picture

Installing the camera

The camera should be connected to the RPi as shown in figure 3.

Figure 3: Connecting the camera to the RPi

Then, you must add the following line to the file **/boot/firmware/config.txt**:

```
start_x=1
```

And install some new libraries:

```
sudo apt install libraspberrypi-bin libraspberrypi0 libraspberrypi-dev
sudo pip3 install picamera
```

From that moment on, the OpenCV functions that capture images taken by the camera can be used. The program in figure 4 is a simple example. It takes an image from the camera when the user presses the button on the RPi and saves the image in a file.

```
from gpiozero import Button
from signal import pause
import cv2
cap = cv2.VideoCapture(0)
def held():
    print ("Button pressed")
    ret, frame = cap.read()
    if ret:
      cv2.imwrite('image.jpg', frame)

button = Button (17)
print ("Press the button 2 seconds")
button.when_held = held
pause()
```

Figure 4: A program that takes and saves a pictures then the button is pressed

Invent some nice application using LEDs, buttons, camera and image processing with OpenCV and show the results in your blog.

Sending images to the ground station

The RPi is now capable of taking images and storing them locally for further processing when the drone comes back home and these images are downloaded into the ground station (your computer). However, an interesting alternative is to send the images to the ground station in streaming, during the flight. To do that, your computer should be connected to the RPi access point and run a program that receives the images that are being send by the RPi. Figure 5 shows the Python scripts that must be run in your computer and in the RPi.

```
# import required libraries
from vidgear.gears import NetGear
import cv2

# define tweak flags
options = {"flag": 0, "copy": False,
"track": False}

# Define Netgear Client at given IP
address and define parameters

client = NetGear(
    address="10.10.10.XXX",
    port="5454",
    protocol="tcp",
    pattern=0,
    receive_mode=True,
    logging=True,
    **options
)
```

```
# import required libraries
from vidgear.gears import NetGear
import cv2

# Open suitable video stream
stream = cv2.VideoCapture(0)

# define tweak flags
options = {"flag": 0, "copy": False,
"track": False}

# Define Netgear server at given IP
address and define parameters

server = NetGear(
    address="10.10.10.XXX",
    port="5454",
    protocol="tcp",
    pattern=0,
    logging=True,
    **options
)
```

```
# loop over
while True:

    # receive frames from network
    frame = client.recv()

    # check for received frame

    if frame is None:
        break

    # Show output window
    cv2.imshow("Output Frame", frame)

    # check for 'q' key if pressed
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break

# close output window
cv2.destroyAllWindows()

# safely close client
client.close()
```

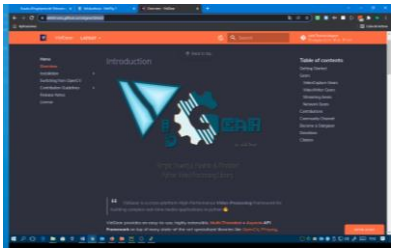
(a)

```
# loop over until KeyBoard Interrupted
while True:

    try:
        # read frames from stream
        (grabbed, frame) = stream.read()

        # check for frame if not grabbed
        if not grabbed:
            break

        # send frame
        server.send(frame)

    except KeyboardInterrupt:
        break

# safely close video stream
stream.release()

# safely close server
server.close()
```

(b)

Figure 5: Scripts to be run in your computer (a) and in the RPi (b).

The RPi runs a loop where images are taken from the camera and immediately send to the computer. The computer runs a loop that waits for the images coming from the RPi and shows them in a window. Both scripts use VidGear, that is a library that facilitates the development of complex real-time media applications in python. You can learn more about this library in the link below.



https://abhitronix.github.io/vidgear/latest/

Install the library in the RPi:

```
pip install -U vidgear[core]
```

and include it also in your PiCharm project.

The key point in the scripts shown in figure 5 is the IP address that must be used for the communication. This required some explanation.

Remember that any device that wants to connect to your RPi should use the access point that was configured in the RPi. This access point has a default gateway (“10.10.10.1”), and uses the Dynamic Host Configuration Protocol (DHCP). That means that any device connected to the access point will be assigned a different IP address in the range from “10.10.10.2” to

“10.10.10.254”. This is the IP that must be used for communications between this device and the RPi.

To identify the IP address that was assigned to your computer when it connected the access point you must run the command:

`ipconfig (Windows)`

`ifconfig (Linux)`

You will get an answer similar to that shown in figure 6 (in Windows). In this case, the IP address is “10.10.10.131”. This is the IP address that must be used in both scripts shown in figure 5.

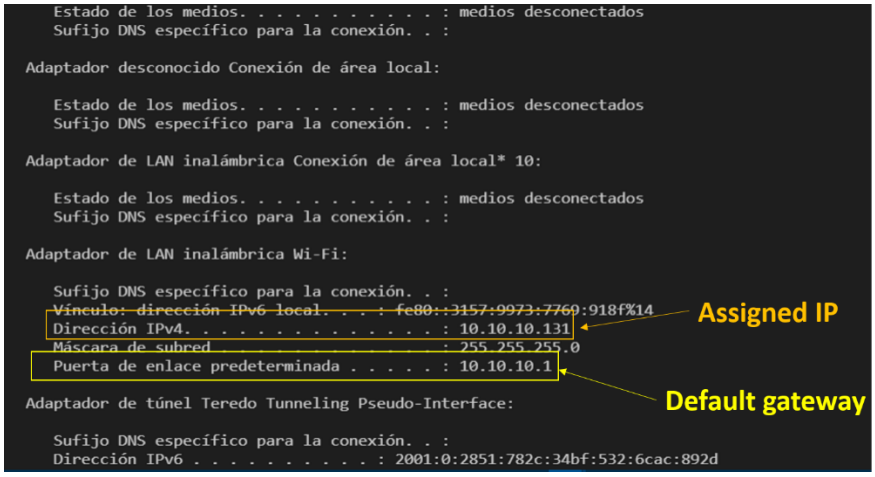


Figure 6: Result of “ipconfig” (Windows) to identify the assigned IP address

Run these scripts and you will see in your computer, in real time, the images taken by the camera in the RPi.

Activity #17

Connecting the RPi with the autopilot

Introduction

In activity #13 you learned to interact with the autopilot from a Python program that was running on your computer. The example program connected to the autopilot and got some information like battery status, etc.

Now is the time to learn how to run that program directly on the RPi. To do this, it will be necessary to learn how to connect the RPi to the autopilot. That is what you are going to learn in this activity.

Do not forget to collect material (images, videos, inventions and reflections) to document your work on the blog.

The UART protocol and UART connectors in the RPi and autopilot

The RPi and the autopilot will be connected through a serial communication link that follows the UART (Universal Asynchronous Receiver/Transmitter) protocol. This protocol defines a set of rules for the serial data exchange (bit by bit) between two devices. The protocol is very simple. It uses only two communication wires between the connected devices and a ground connection on both (see figure 1).

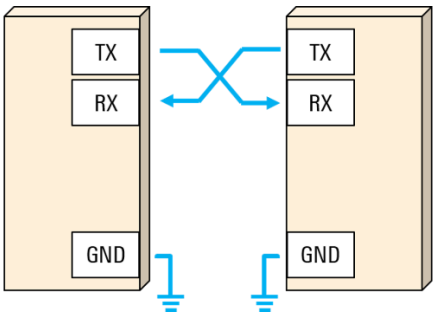
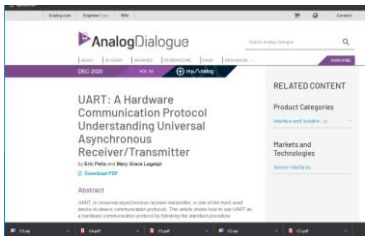


Figure 1: UART protocol only needs two signals (TX and RX) and GND

You can find more information about how this protocol works in the link below.



https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

Both the autopilot and the RPi have connections for UART communications. In fact, in activity #13 we already used serial communication between the computer and the autopilot. Remember that using the cable shown in the figure 2 you connected your computer (USB port) with the TELEM2 connection of the autopilot. That connection, through which your computer interacted with the autopilot, uses the UART protocol.

Figure 2: The cable used to connect your computer with the autopilot in activity #13. This connection uses the UART protocol

Figure 3 shows the pin configuration of the TELEM1 and TELEM2 connections. There are 6 pins. The pins come into play in UART communications are pin 2 (TX), pin 3 (RX) and pin 6 (GND). We can therefore deduce that when we connect the cable shown in the figure 2 to TELEM2, the wire closest to the red wire in that cable is the RX wire of the computer (which connects to the TX of TELEM2) and the next wire is the TX of the computer (which connects to the RX of TELEM2). This information is relevant for the first test that we will do in this activity: communicating your computer with the RPi. But first, let's see how the UART ports of the RPi are.

TELEM 1 & 2

Pin #	Name	Dir	Wire Color	Description
1	VCC_5V	out	red / gray	Supply to GPS from AP
2	MCU_TX	out	yellow / black	3.3V-5.0V TTL level, TX of AP
3	MCU_RX	in	green / black	3.3V-5.0V TTL level, RX of AP
4	MCU_CTS (TX)	out	gray / black	3.3V-5.0V TTL level or TX of AP
5	MCU_RTS (RX)	in	gray / black	3.3V-5.0V TTL level or RX of AP
6	GND	-	black	GND connection

Figure 3: Pin configuration of TELEM 1 & 2 in the autopilot

Tal y como muestra la figura, la RPi tiene dos pines dedicados a la comunicación serie UART: GPIO14 y GPIO15 (en realidad el modelo RPi 4 tiene otros pines que pueden usarse para comunicaciones series de manera que pueden configurarse hasta 4 UARTs diferentes).

As figure 4 shows, the RPi has two pins dedicated to serial UART communication: GPIO14 and GPIO15 (actually the RPi 4 model has other pins that can be used for serial communications so that up to 4 different UARTs can be configured).

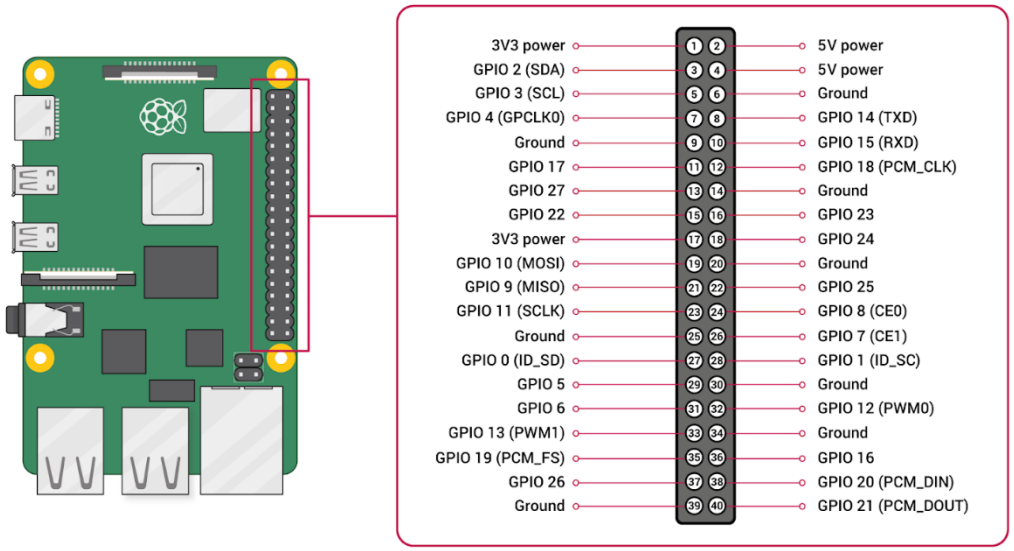


Figure 4: RPi 4 pinout configuration. GPIO14 and GPIO15 are used for UART communication.

On the other hand, the red box that contains the RPi has a connector prepared for the UART communication with the outside. This connector is shown figure 5 (which also shows the connector separated from the box for better understanding).

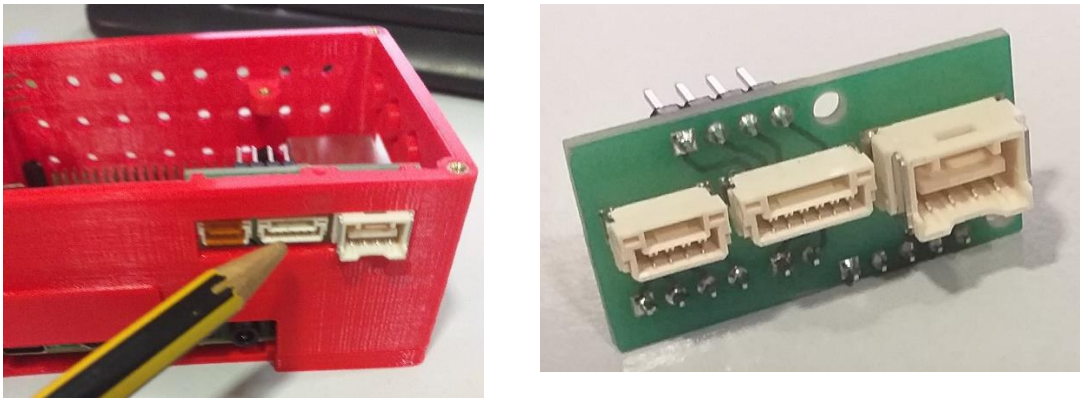


Figure 5: Connector in the red box prepared for UART communication

Figure 5 shows that the connector offers 4 pins corresponding to + 5V (left side end in the figure), GND (right side end) and the two pins for the TX and RX wires. The pins corresponding to TX, RX and GND should be connected to the corresponding pins of the RPi (the pin with + 5V can be ignored). The connections required depends on the situation, as shown in the following.

Comunicating the computer and the RPi

Lets connect now your computer with the RPi through a UART connection. You will not need that during the transversal project, but this is a nice exercise that will also help you that the UART in the RPi is working properly.

We need the cable shown the figure 2. Remember that the wire closest to red one in that cable corresponds to the RX of the computer. Therefore, the second pin, from the left, of the connector shown in the figure 5 should be connected to GPIO14 (green wire in figure 6) and the third pin should be connected to GPIO15 (yellow wire in figure 6). The GND pin can be connected to any of the pins labeled Ground on the RPi (black wire in figure 6).

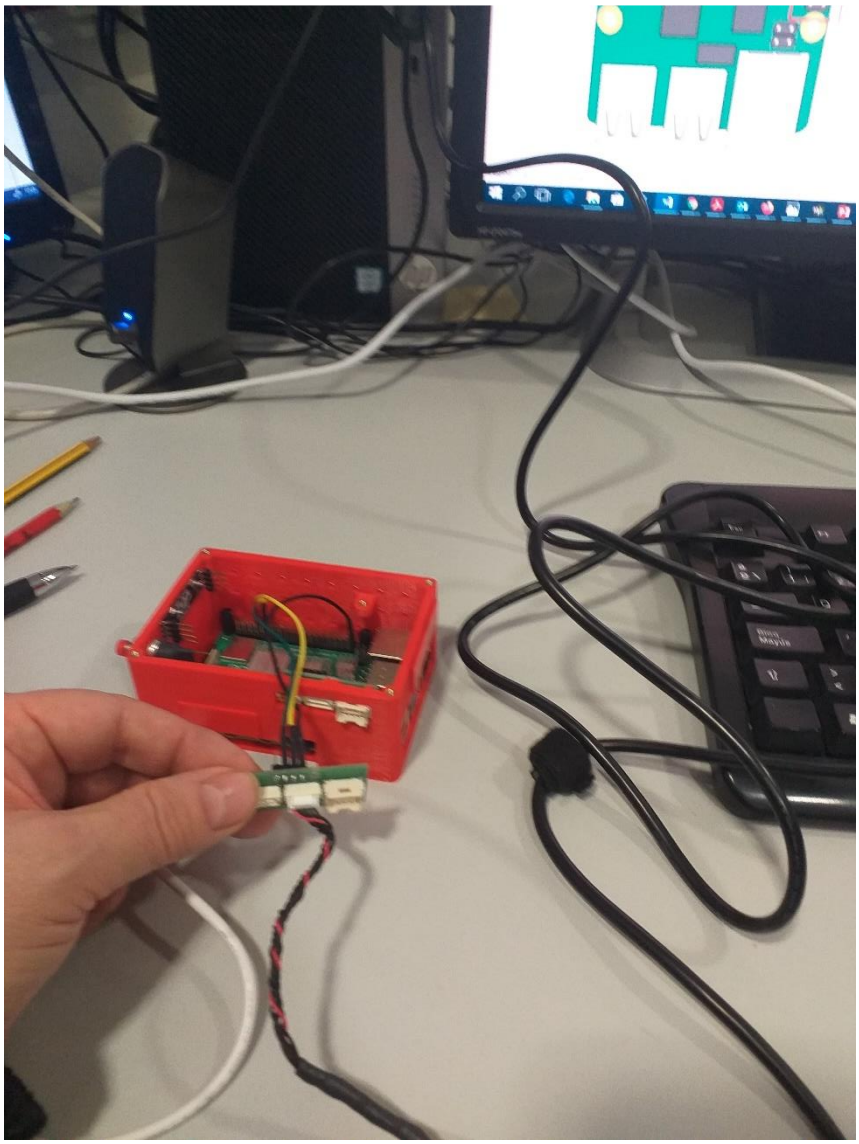


Figure 6: Connections required to communicate via UART your computer with the RPi

Once these connections have been made, the connection via UART between the computer (through one of the USB ports) and the RPi is correctly established. Now we are going to carry out the software configurations, which are somewhat more laborious.

Log in in the RPi and do the following:

1. Back up the original config.txt and cmdline.txt files

```
sudo cp -pr /boot/firmware/cmdline.txt /boot/firmware/cmdline.txt-orig
sudo cp -pr /boot/firmware/config.txt /boot/firmware/config.txt-orig
```

2. Edit `/boot/firmware/config.txt` to comment out the `enable_uart=1` like below,

```
#enable_uart=1
cmdline=cmdline.txt
```

3. Remove the console setting `console=serial0,115200` from `/boot/firmware/cmdline.txt`

4. Disable the Serial Service which used the miniUART

```
sudo systemctl stop serial-getty@ttyS0.service
sudo systemctl disable serial-getty@ttyS0.service
```

```
sudo systemctl mask serial-getty@ttyS0.service
```

5. Add the user which will use the miniUART to `tty` and `dialout` group

```
sudo adduser ${USER} tty
sudo adduser ${USER} dialout
```

6. Finally, reboot the RPi.

After these operations you have the device `/dev/ttyS0` ready for UART communication in the RPi.

Now we need a program to exchange data with your computer through the UART device. Install `minicom` in the RPi:

```
sudo apt-get update
sudo apt-get install minicom
```

Now, you can run minicom:

```
sudo minicom –s
```

In the configuration menu select serial port setup and then configure the serial port as shown in figure 7.

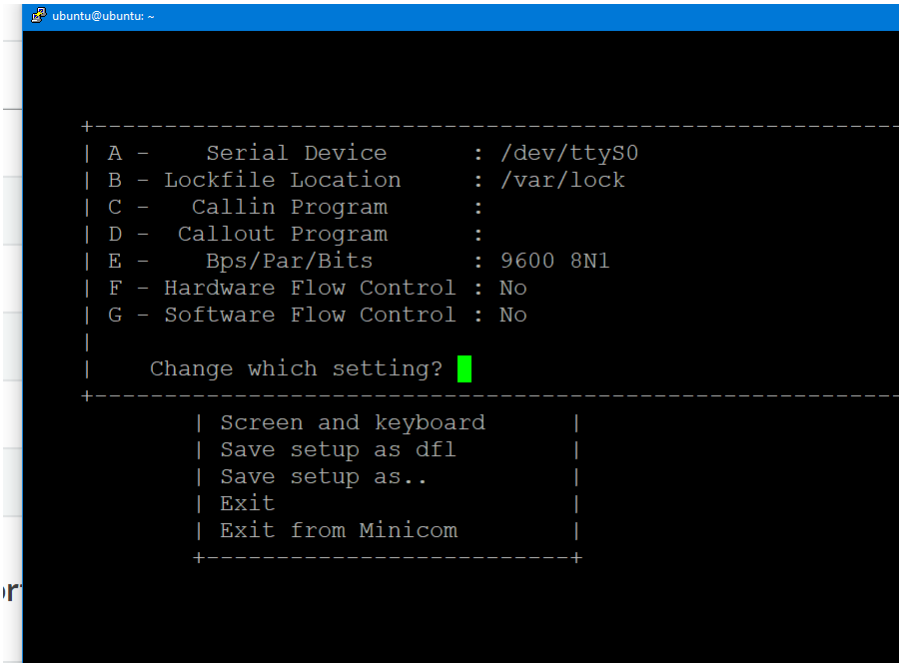


Figure 7: Configuration required in `minicom` for UART communication with your computer

Finaly select Exit and you will be ready to type any text to be send to your computer.

You also need an adecuate program in your computer to communicate with the RPi. You can use PuTTY for this, as shown in figure 8 (assuming that the cable is connected to the USB port COM23):

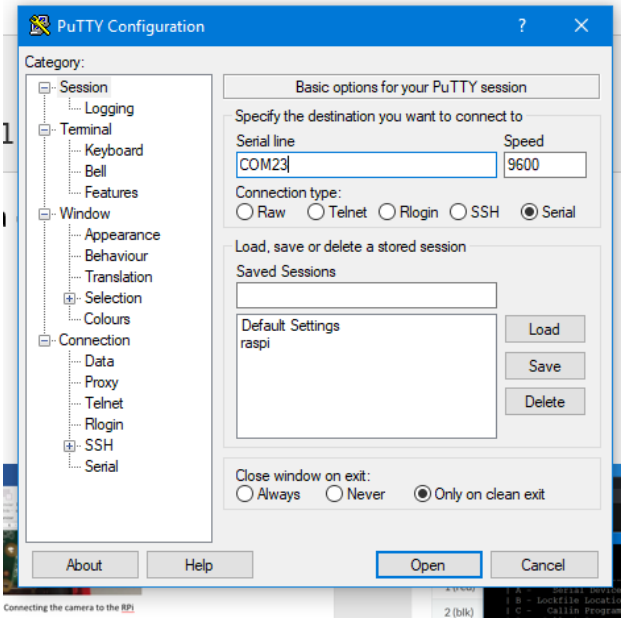


Figure 8: PuTTY configuration to connect via UART with the RPi

When opening you will be able to exchange text between your computer and the RPi though the UART connection.

See this short video showing all these operations.

https://youtu.be/zqeCHRouxy4

Comunicating the autopilot and the RPi

Now is time to connect the RPi with the autopilot. To that purpose, we need the cable shown in the figure 8, connecting TELEM2 with the UART connector in the RPi.

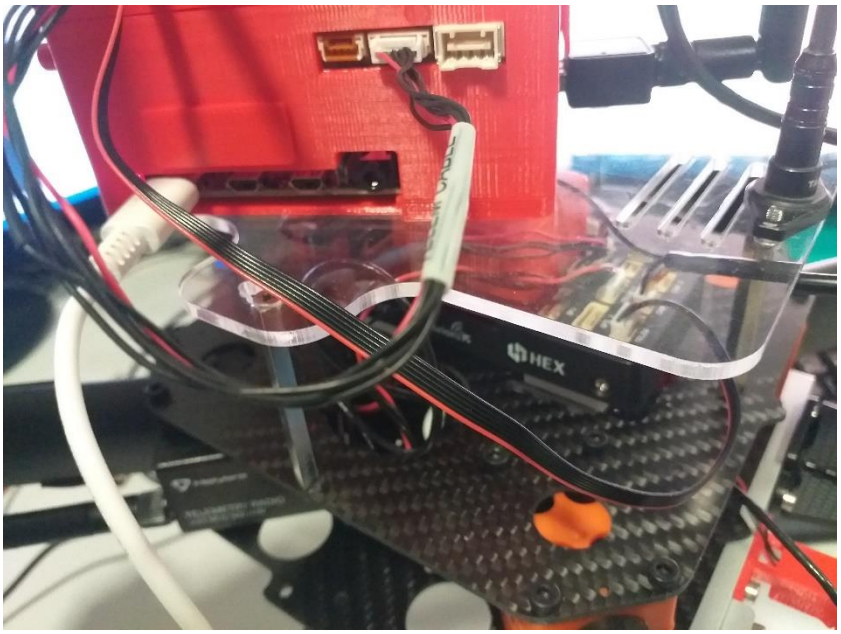


Figure 8: Connecting the RPi to TELEM2

But attention: remember that the wire close to the red one coming from TELEM2 is TX from the autopilot and we are connecting it to the green wire shown in figure 6, which goes to GPIO 14, which is also TX in the RPi side. So we must change the connections in the red box. The green cable must to GPIO 15 and the yellow to GPIO14.

Now, everything is ready to run a program in the RPi that sends commands to the autopilot. We can use the same program used in activity #13, shown in figure 9.

```
from dronekit import connect
import time

connection_string = "/dev/ttyS0"


vehicle = connect (connection_string, wait_ready = True, baud =
115200)
print ("Connected")
print(" Autopilot Firmware version: %s" % vehicle.version)
print(" Global Location: %s" % vehicle.location.global_frame)
print(" Velocity: %s" % vehicle.velocity)
print(" Battery: %s" % vehicle.battery)
print(" Heading: %s" % vehicle.heading)
print(" Is Armable?: %s" % vehicle.is_armable)
vehicle.close()
```

Figure 9: Python script to be run in the RPi to get some information from the autopilot

Note that we have changed the connection string so that the RPi uses the adequate device. On the other hand we fixed a higher baud rate. Finally, note also that in order to run this program in the RPi we must install dronekit:

```
pip install dronekit
```

On the side of the autopilot, you must be sure that the baud rate for TELEM2 is also 115200 (use Mission Planner to check/change this).

Now you can run the Python script and obtain some information from the autopilot, exactly as you did in activity #13, when the script was run in your computer.

Autorun scripts when starting up the RPi

During the transversal project it will be useful to be able to run in the RPi some Python scripts automatically, even before log in. For instance, imagine that as soon as you reboot the RPi you want to see the leds indicating with different color sequences that:

- The RPi has internet connection
- The platform battery is ok
- The drone is armable
- Etc.

To do that you can use the crontab file, which allows the user to specify a list of tasks to be executed at specific times. You must prepare the Python script (`myscript.py`) to be run automatically when the RPi is starting up. Then:

1. Edit your crontab list by typing:

```
sudo crontab -e
```

2. Select nano if you are prompted to select an editor. A file opens.

3. Add a line at the end of the file that reads like this (assuming that your script is in your root directory):

```
@reboot python3 /home/ubuntu/myscript.py
```

The line has to begin with @reboot which tells the system to run every time you boot the RPi.

4. Save and exit. In nano, you do that by hitting CTRL + X, answering Y and hitting Enter when prompted.

Explore this mechanism. Invent a useful auto-run script and test it. Show the results in your blog.