VALENCIA COLLEGE

Department of Electrical and Computer Engineering Technology (ECET)
Division of Engineering, Computer Programming, and Technology (ECPT)

Pandemic Project
Orlando Section

Sanitizing Drone,
The Flying Cleaner

Performed By
Tamer Taha and Anthony Amaya

Supervised by
Dr. Ali Notash

April 7th, 2021

1. Introduction

In this report we will go over the most recent updates of the Sanitizing Drone Pandemic Project as of the 7th of April 2021.

In the following sections we will be discussing the contributions and the work that has

been done to complete the objectives we have set. Due to the pandemic we decided to build two prototypes of the drone which both team members agreed on. However, each of team members cooperated to build all the components through video meetings. The Sanitizing Drone became a shared responsibility.

2. Hardware Design and Component Testing

In the last month we focused heavily on getting into a complete design of the project to get it ready to be tested for flights and spraying operations. This design came together with well planed steps as it follows:

1) The F4 flight controller was mounted on the middle layer of the frame after soldering all the required wiring to it.
2) This middle layer that provides a printed circuit board to distribute power to all the electrical components was soldered to a supply line that will be connected later with the S5 LiPo Battery.
3) The receiver was also mounted on that middle layer and connected with servo cables. One using IBUS communication that will be bounded with the FLYSKY transmitter. The other servo is using PPM communication which is connected with the controlled buck converter that is controlling the pump.
4) The buck controller was connected to the receiver as mentioned in step 3. As well as connecting the DC-DC line to the pump which is converted 18.5V to 12V which is the operating voltage of the pump.
5) As the middle layer having all the components mounted to it. The top layer was mounted on top of the middle layer.
6) The top layer as was shown last week had the 4 ESCs and 4 Motors mounted on it. Each set of Motor and ESC is mounted on each arm of the quadcopter.

7) Lastly, the battery was mounted in the middle of that top layer, to be later connected to the PCB on the middle layer.
8) The last part is going to be mounting the pump and the tank to the bottom layer.

After completing steps from 1-7 the following figure below shows the drone in an 90% complete form.

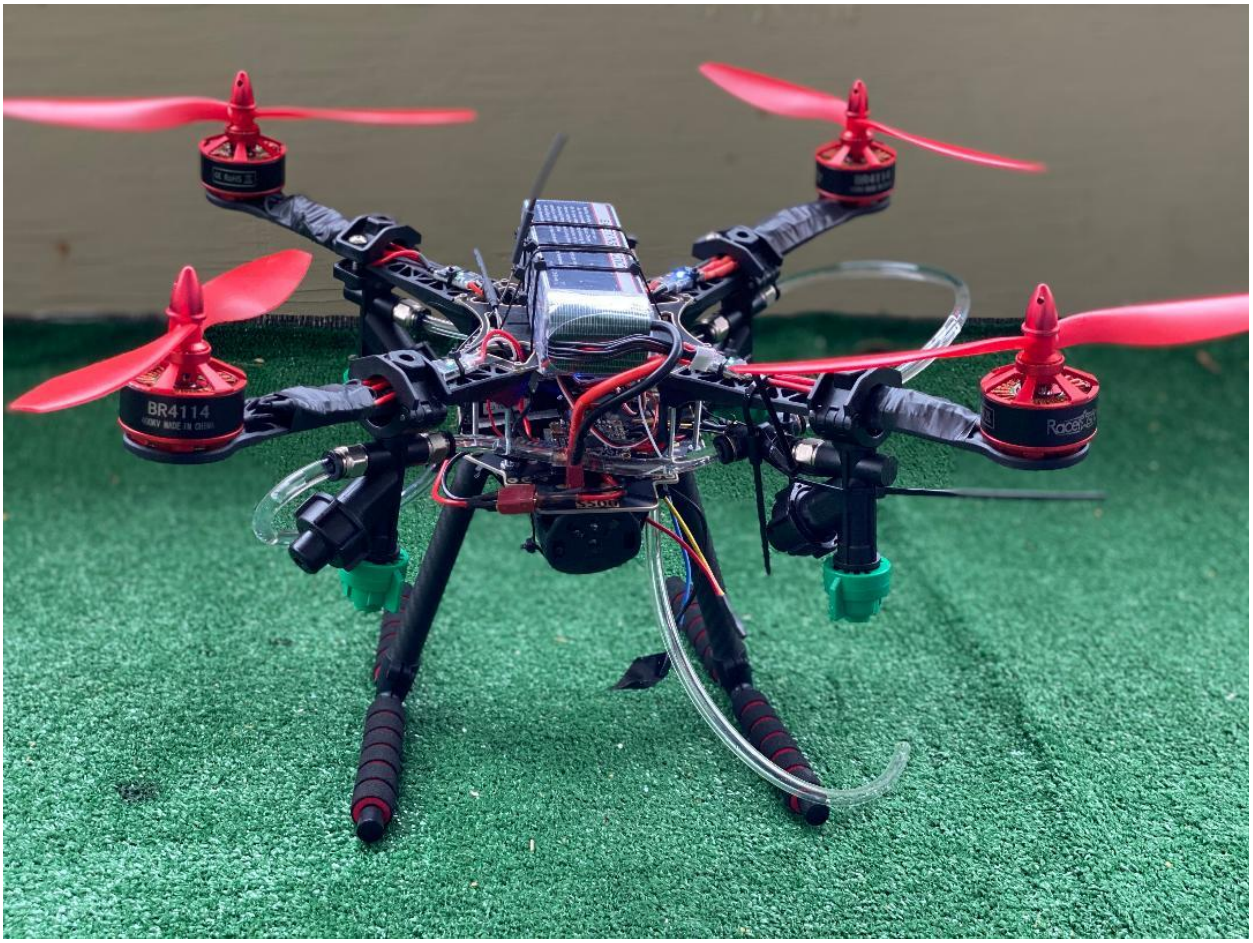
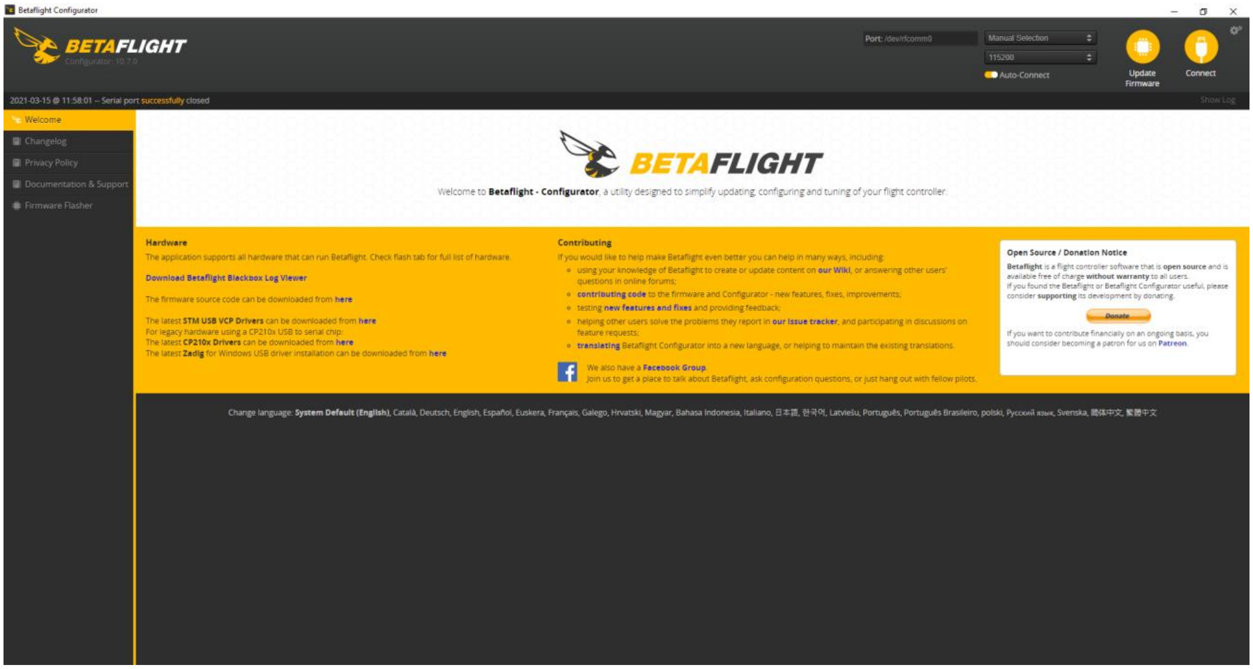


Figure 1: Drone 90% completed

3. Software programming and component testing

In this step we worked on finalizing the drone and transmitter modules by connecting the three layers of the drone, binding the transmitter and receiver. We started programming the flight controller and configuring its settings by connecting it to a PC then binding it with the Beta Flight firmware:



2021-03-15 @ 11:58:01 -- Serial port successfully closed

Welcome to **Betaflight - Configurator**, a utility designed to simplify updating, configuring and tuning of your flight controller.

Hardware

The application supports all hardware that can run Betaflight. Check flash tab for full list of hardware.

Download Betaflight Blackbox Log Viewer

The firmware source code can be downloaded from **here**

The latest **STM USB VCP Drivers** can be downloaded from **here**
For legacy hardware using a CP210x USB to serial chip:
The latest **CP210x Drivers** can be downloaded from **here**
The latest **Zadig** for Windows USB driver installation can be downloaded from **here**

Contributing

If you would like to help make Betaflight even better you can help in many ways, including:

- using your knowledge of Betaflight to create or update content on **our Wiki**, or answering other users' questions in online forums;
- **contributing code** to the firmware and Configurator - new features, fixes, improvements;
- testing **new features and fixes** and providing feedback;
- helping other users solve the problems they report in **our issue tracker**, and participating in discussions on feature requests;
- **translating** Betaflight Configurator into a new language, or helping to maintain the existing translations.

We also have a **Facebook Group**.
Join us to get a place to talk about Betaflight, ask configuration questions, or just hang out with fellow pilots.

Open Source / Donation Notice

Betaflight is a flight controller software that is **open source** and is available free of charge **without warranty** to all users.
If you found the Betaflight or Betaflight Configurator useful, please consider **supporting** its development by donating.

If you want to contribute financially on an ongoing basis, you should consider becoming a patron for us on **Patreon**.

Change language: **System Default (English)**, Català, Deutsch, English, Español, Euskera, Français, Galego, Hrvatski, Magyar, Bahasa Indonesia, Italiano, 日本語, 한국어, Latviešu, Português, Português Brasileiro, polski, Русский язык, Svenska, 简体中文, 繁體中文

Figure 2: Beta Flight

After connecting the drone to the firmware, we noticed that the USB driver method is not recognized by the computer. We researched that problem, and three programs are required to communicate with the flight controller by a Microsoft-based operating system and these programs are STM USB VCP Drivers, CP210x Drivers, and Zadic.

We downloaded these programs and configured different settings to get a stable connection between the operating system and the drone.

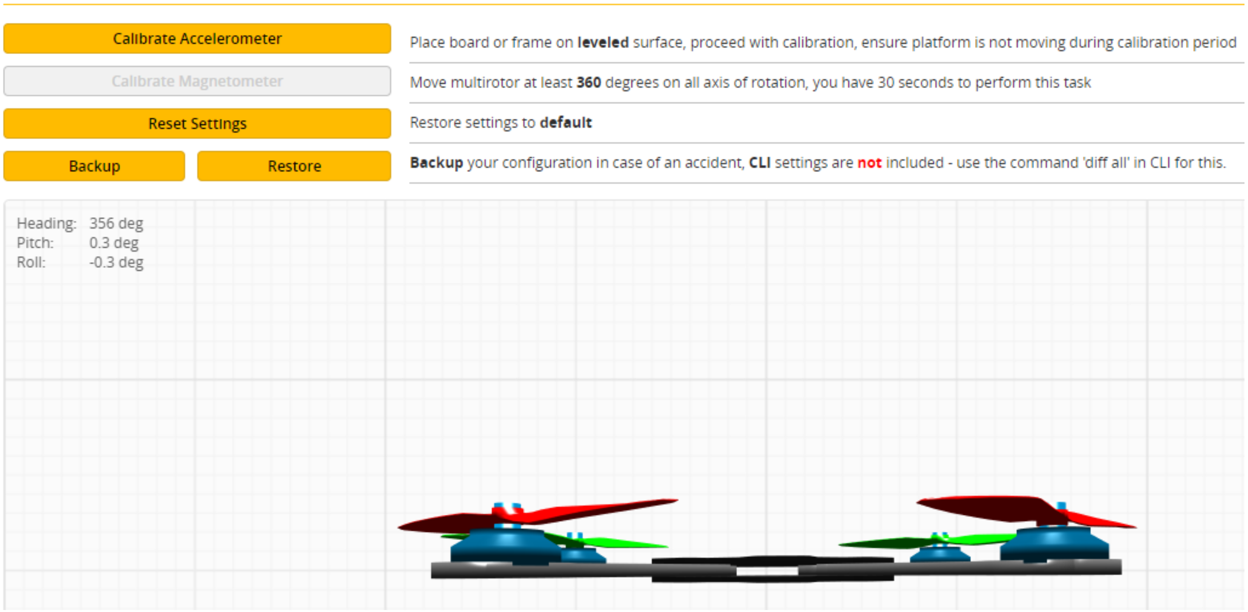
After connecting properly, we started working on configuring the settings and updating the chip's firmware as it follows:

1) Update firmware:

To update the chip's firmware, we need to connect the FC to the computer while holding the reboot button located on the chip to connect it in the reboot mode. After getting into the reboot mode, we need to update the chip to the most recent compatible firmware. The compatible version of the chip we are using can be found by typing the command "-version" in the CLI command window. For the F4 FC, that version is BETAFLIGHT4. This firmware can be located online or programmed locally.

2) Calibrating the accelerometer:

This step was performed by resting the drone on a leveled surface, then hitting the calibrate accelerometer button as shown below:



Place board or frame on **leveled** surface, proceed with calibration, ensure platform is not moving during calibration period

Move multirotor at least **360** degrees on all axis of rotation, you have 30 seconds to perform this task

Restore settings to **default**

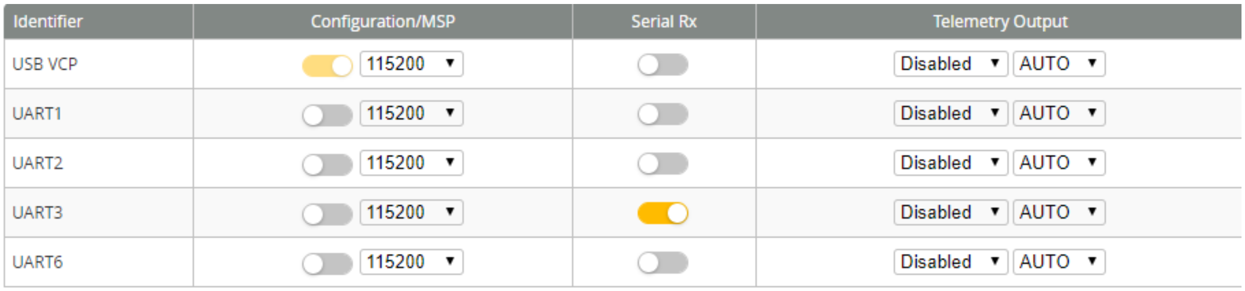
Backup your configuration in case of an accident, **CLI** settings are **not** included - use the command 'diff all' in CLI for this.

Heading: 356 deg
Pitch: 0.3 deg
Roll: -0.3 deg

Figure 3: Calibrating the accelerometer

3) Setting ports:

In this step, we need to allow only one UART port to be activated to allow communication between the drone and the receiver. Each flight controller has a different UART port that needs to be activated and that can be located by reading the manual for your flight controller. In our case, the F4 FC uses the UART3 port as a communicator between its firmware and the receiver. You also must make sure that ONLY one UART is activated as the figure below shows:

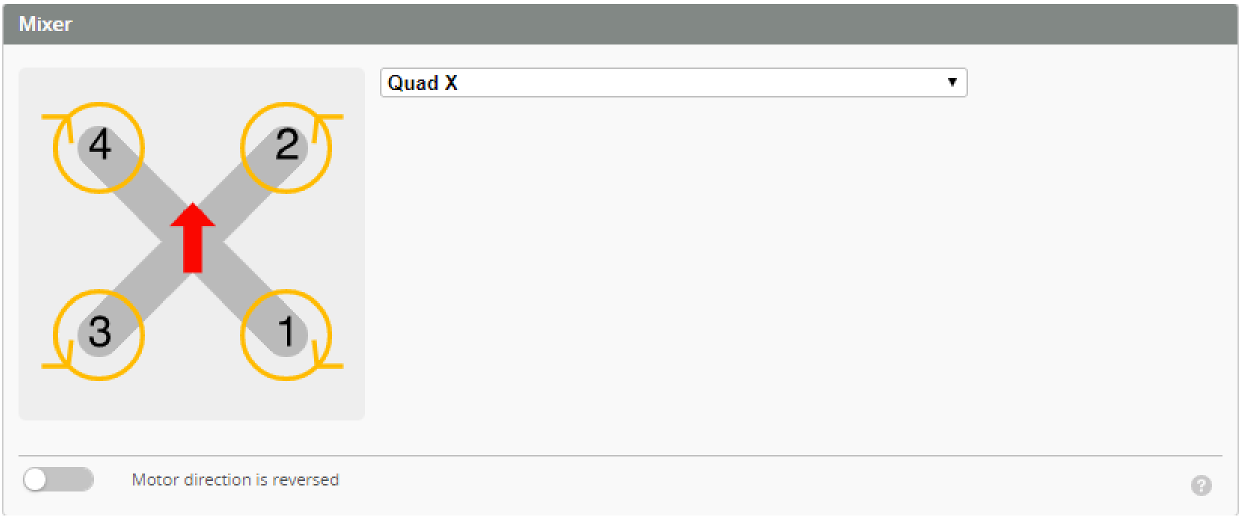


Identifier	Configuration/MSP	Serial Rx	Telemetry Output
USB VCP	● 115200 ▼	○	Disabled ▼ AUTO ▼
UART1	○ 115200 ▼	○	Disabled ▼ AUTO ▼
UART2	○ 115200 ▼	○	Disabled ▼ AUTO ▼
UART3	○ 115200 ▼	●	Disabled ▼ AUTO ▼
UART6	○ 115200 ▼	○	Disabled ▼ AUTO ▼

Figure 3: Port Selection

4) F4 Configuration:

In this step, we went over different configurations to set up the flight controller properly. Firstly, the quadcopter type, our drone is a quadcopter type:



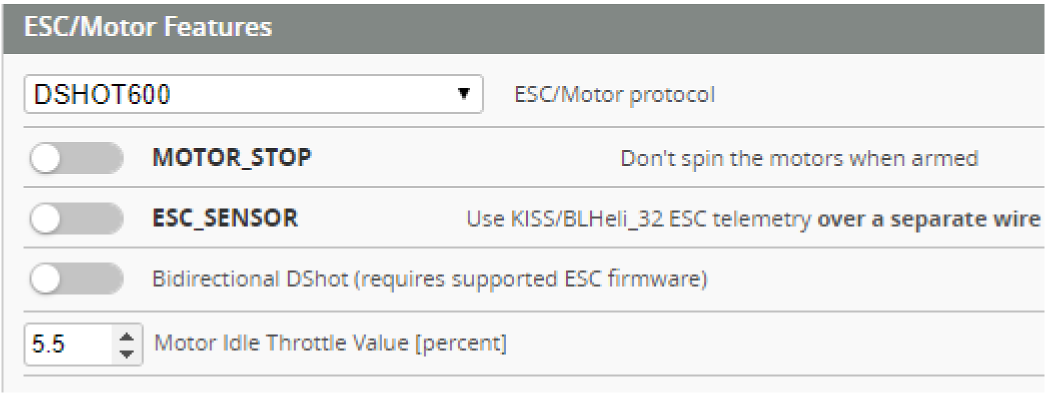
Mixer

Quad X

Motor direction is reversed

Figure 4: Copter Type

The second option is ESC/Motor Features, this option is to set up the communication method between the electronic speed controllers & the motors. The BLHeli_32Bit ESC we are using supports Dshot communication up to 1200. However, our firmware allows up to Dshot 600 and that is what we decided to go for:



ESC/Motor Features

DSHOT600 ESC/Motor protocol

MOTOR_STOP Don't spin the motors when armed

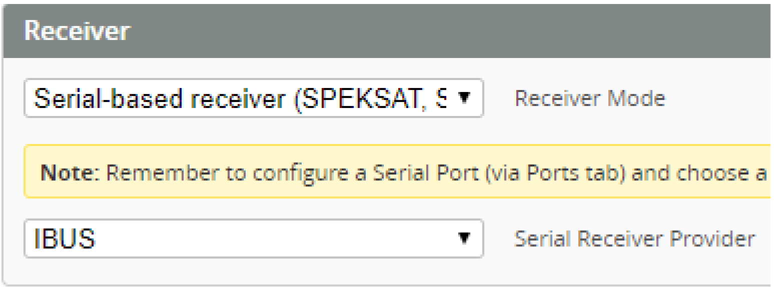
ESC_SENSOR Use KISS/BLHeli_32 ESC telemetry **over a separate wire**

Bidirectional DShot (requires supported ESC firmware)

5.5 Motor Idle Throttle Value [percent]

Figure 5: ESC communication method

Lastly, is the receiver mode select and the serial provider. In our case, we are using a FLYSKY transmitter bounded with an FS-IA6B receiver, this communication method uses IBUS communication and a serial-based receiver:



Receiver

Serial-based receiver (SPEKSAT, S ▾ Receiver Mode

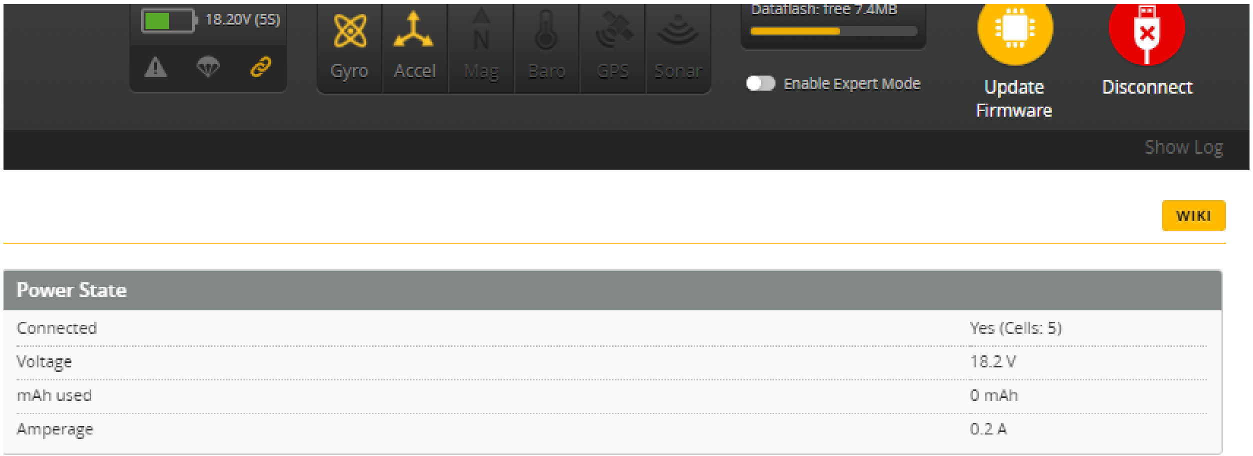
Note: Remember to configure a Serial Port (via Ports tab) and choose a

IBUS ▾ Serial Receiver Provider

Figure 6: Transmitter communication method

5) Setting up battery count:

For the ADC on the flight controller to read the battery voltage properly, we need to adjust the cell count of the battery we are using in the command CLI window by typing "set auto_profile_cell_count = 5" then type "save". In our case we used 5 as the cell count because we are using a 5s battery then if we connect the battery to the drone, we will notice that the firmware will start reading the battery voltage, current draw, and mAh:

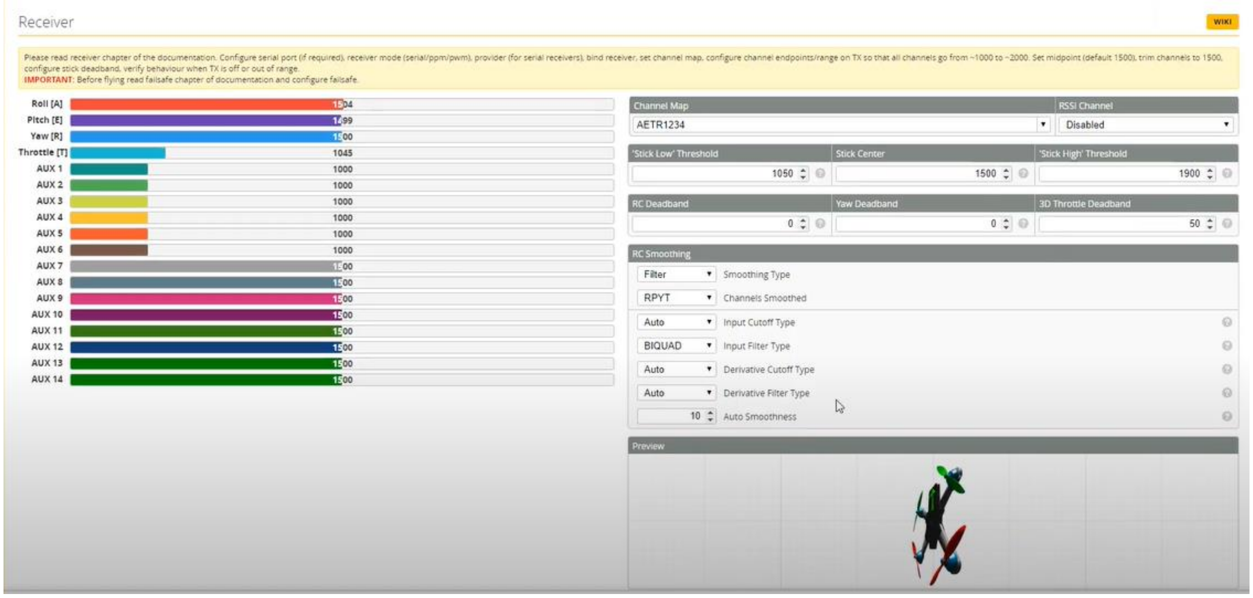


Power State	
Connected	Yes (Cells: 5)
Voltage	18.2 V
mAh used	0 mAh
Amperage	0.2 A

Figure 7: Battery cell count

6) Setting up receiver and channels:

In this part, after assigning channels in the handheld transmitter, we are going to check if each channel is assigned correctly in the simulation firmware. The following video shows how we performed that test:



Receiver

Please read receiver chapter of the documentation. Configure serial port (if required), receiver mode (serial/ppm/pwm), provider (for serial receivers), bind receiver, set channel map, configure channel endpoints/range on TX so that all channels go from ~1000 to ~2000. Set midpoint (default 1500), trim channels to 1500, configure stick deadband, verify behaviour when TX is off or out of range.
IMPORTANT: Before flying read failsafe chapter of documentation and configure failsafe.

Roll [A]	1504
Pitch [E]	1499
Yaw [R]	1500
Throttle [T]	1045
AUX 1	1000
AUX 2	1000
AUX 3	1000
AUX 4	1000
AUX 5	1000
AUX 6	1000
AUX 7	1500
AUX 8	1500
AUX 9	1500
AUX 10	1500
AUX 11	1500
AUX 12	1500
AUX 13	1500
AUX 14	1500

Channel Map	RSSI Channel
AETR1234	Disabled

'Stick Low' Threshold	Stick Center	'Stick High' Threshold
1050	1500	1900

RC Deadband	Yaw Deadband	3D Throttle Deadband
0	0	50

RC Smoothing

Filter	Smoothing Type
RPYT	Channels Smoothed
Auto	Input Cutoff Type
BIQUAD	Input Filter Type
Auto	Derivative Cutoff Type
Auto	Derivative Filter Type
10	Auto Smoothness

Figure 8: Transmitter channels

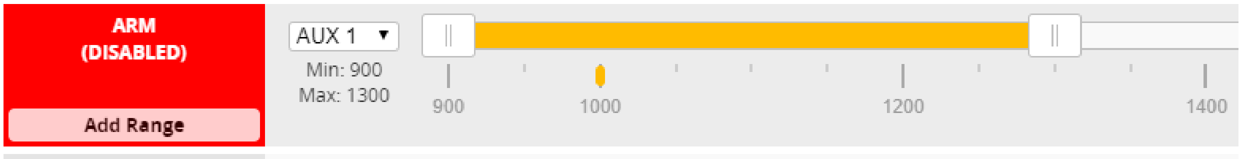
7) Setting up motors:

In this part, we are going to perform a test on each motor to check if it's receiving a signal using the Beta Flight firmware, as well as the rotation direction of each motor. The following video shows how we performed that test:

https://www.youtube.com/watch?v=kK3z24oupBY

8) Setting up arming mode:

Arming the aircraft before the flight has two purposes: to prevent the motor from turning when the pilot is not ready to fly (a safety feature) prevent takeoff before the autopilot is fully configured and ready to fly. This mode was set in the modes menu as shown below:



ARM (DISABLED)

Add Range

AUX 1
Min: 900
Max: 1300

Figure 9: Arming frequency

After setting all the configurations above the drone is set to be tested by arming it with the transmitter then start performing flying tests.

3. Flying operations testing.

In this step after setting all the electrical components of the drone and programming the flight controller, the drone was put to the test and check its flying behavior and adjust as needed. The results were not perfect in the beginning, because the drone was drifting to different directions when we arm it and then throttle up to gain altitude. After various testing of the both the hardware and the software, we concluded that the drone is shifting to different directions due to weight imbalance. The battery was not mounted in a final set position because we still need to add the pump and the tank to the drone. The following videos show these flying operations testing:

1) https://www.youtube.com/watch?v=MBg5eC7ug5I
2) https://www.youtube.com/watch?v=mCopbICKMqY&t=1s
3) https://www.youtube.com/watch?v=gSY8m-Cl694

As we notice in these videos, the drone was only tested to throttle up. However, it was going up and drifting into different directions due to weight imbalance. We are planning to work on that in the coming month of April. After we mount the pump and tank to the bottom layer.

4. Tank design & Assembly

As we discussed earlier, designing the tank according to the space available below the bottom layer of the drone. This tank was designed using solid works firmware and is going to be 3D printed by the open lab faculty of Valencia college. However, we faced a problem in the design. The open lab team that is helping us print this tank referred to a problem in our design which is when this tank will be printed, they will not be able to break down the supports inside the empty space of the tank because of the lack of accessibility to that space. We solved that problem by adjusting our design by replacing the top surface of the tank with a drawer where it could be slide in and out as figure 10 below shows:

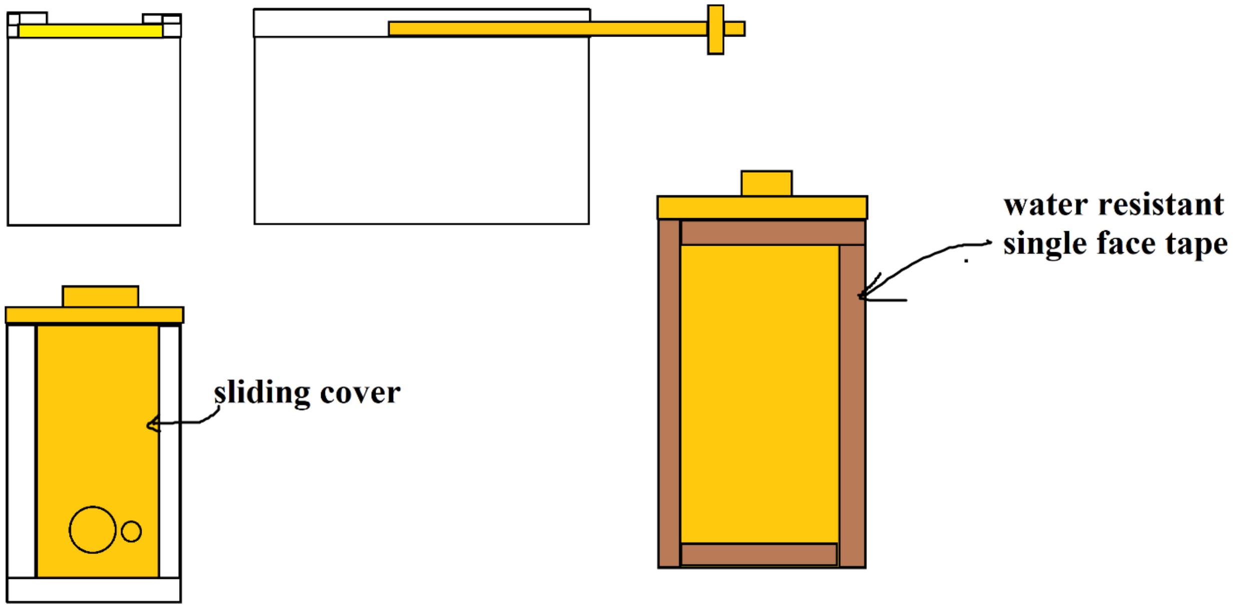


Figure 10: New Tank design idea

This design will provide a maintenance friendly tank with accessibility to break down the supports in the empty space.

The new tank was printed successfully and will be mounted during the following month.

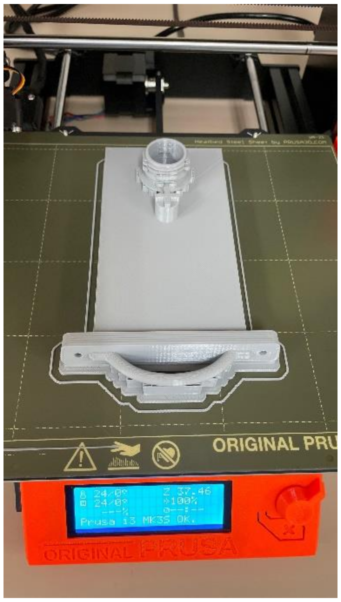


Figure 11: Tank drawer

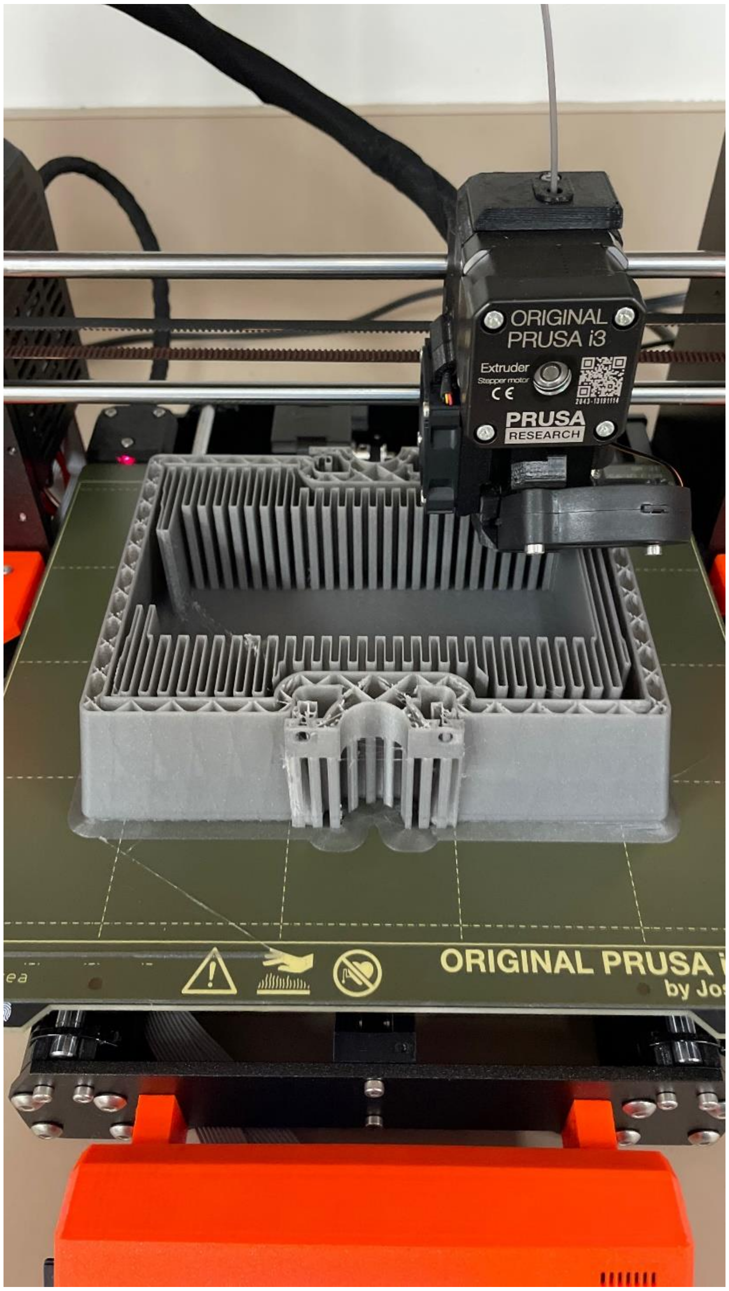


Figure 12: Tank printing process