

# UPWIND Project POCI-01-0145-FEDER-031447

## Activities Report

Sérgio Vinha

April 7 - October 6, 2020

## 1 Introduction

The UPWIND project aims to Airborne Wind Energy Systems (AWES), that generate energy by high altitude wind. The idea of generating power with tethered wings flying at high speed in a cross wind direction has been first proposed by Miles Loyd in 1980 [1]. Since then, a great variety of proposals has been reported. One of them [2] is to use a glider with autonomous control of the tethered flight in order to achieve an optimal production of energy.

## 2 Developed Work

### 2.1 Software

The main focus of this work is to test a trajectory controller in a glider. For that, we use the path-following guidance method reported in [3], developed by previous UPWIND members and implement in *Ardupilot*. *Ardupilot* has three different flight stacks for each drone, plane or rover type autonomous vehicles.

The navigation control of any plane (or glider in our case) in *Arduplane* [4] is done by using the L1 distance reference to the desired point in the trajectory (figure 1) and perform the demanded acceleration using the formula:

$$a_s = 2 \frac{V^2}{L_1} \sin(\eta)$$

However, in the guidance method of UPWIND (figure 2), we define the closest point in the path and then a reference target point is defined as the point distancing L from the closest point in the path. In the end, the L1 resulting from this approach is given by:

$$L_1 = \sqrt{d^2 + L^2}$$

After this modification in the *Arduplane* firmware (V4.1.0dev) and simulating the navigation of the plane we got the results present in figure 3. We can

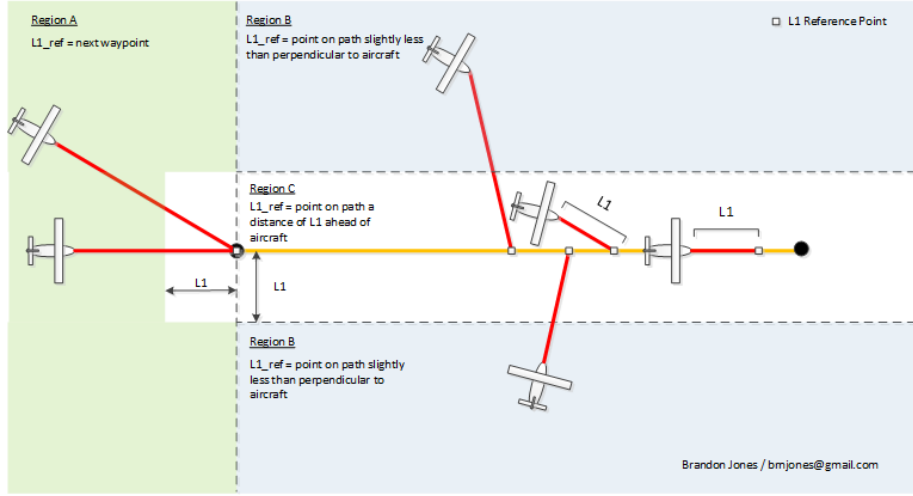


Figure 1: L1 navigation theory used in *Arduplane*

conclude that the new guidance method is capable of following the desired path given the waypoints. In the simulation present in figure 3 there are various  $L$  distances. The simulation performance is for the software in the loop general plane. A new parameter was then added to the parameters list of *Arduplane*, so we can test and change when performing real flights. A stable  $L$  can then be chosen accordingly to the flights of our glider, since our plane will not perform equal to the general plane parameters and we have to tune them.

In the flight mode loiter, the plane does a circular trajectory around the point where the mode was selected. Flying with a tether in this mode would lead to an accumulation of twist on the cable. In order to avoid this, the aircraft can fly along a curve consisting of two (circular) segments of opposite

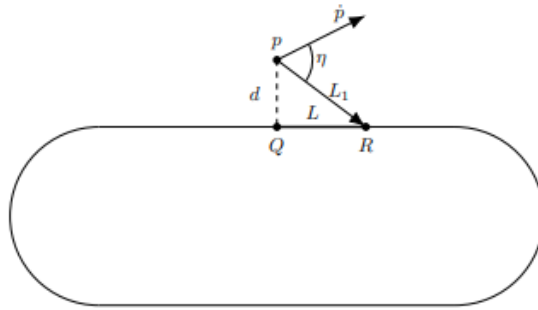


Figure 2: L navigation used in UPWIND guidance method



orientation, resembling the figure eight pattern. So a new mode called *upwind* was created. This mode follows the method applied by AWEsome project [2] with more improvements. When we command the plane to enter this mode, the aircraft starts to navigate along a figure eight pattern with center on home position (figure 4).

Since we are an organization from Porto - Portugal, it makes sense that we do the simulations tests in our airport (figure 5). Of course these are simulations and we would not fly a small plane like EasyStar at Oporto airport.

At any moment, we would need to fly our real plane in real conditions. In the first approach, we simply removed the L1 control and added our L control in order to test the theory. However, this requires to compile, initiate the software and restart the missions. So, a new parameter had to be added to *Arduplane* flight stack. The new parameter is "L-use" that is 0 when we want to use L1 control or 1 when we want to test our new control L. This allows a change in real time that can work along another parameter - "L-dist" - in order to tune the plane controller to the best performance possible. This reduces the probability of crashing the plane in real simulations. When the plane is not performing the desired path, we simply change the control definition using the parameters in ground station (*Mission Planner* or *QGround Control*) and it returns to normal control (L1) that was already been developed by *Ardupilot* team and have a lot of tests.

In order to get the simulations as close as possible to real conditions, we used JSBSim [5]. JSBSim is a sophisticated flight simulator that is used as the core flight dynamics system for several well known flight simulation systems. This required a model of the plane. In our first approach, we adopted the model of the EasyStar I [6], that is similar to the EasyStar II. However, this first version of the EasyStar did not have ailerons and adding that feature to our model was not an easy task. So, after that we decided to use the model of the plane Strato Surfer [7] that also have the same dimensions of the EasyStar II. We had to make some adjustments in order to work with JSBSim. We change the specifications of the new model plane, accordingly to the manual and the result is present in table 1.

| Plane            | EasyStar II | Strato Surfer |
|------------------|-------------|---------------|
| <b>Wing area</b> | 28 $dm^2$   | 29,364 $dm^2$ |
| <b>Wing span</b> | 1366 $mm$   | 1588 $mm$     |
| <b>Chord</b>     | 0,17087 $m$ | 0,17939 $m$   |
| <b>Power</b>     | 200 $W$     | 350 $W$       |

Table 1: Planes specifications

As we can see on the table 1, the specifications of the two models are almost the same. There are not any clues of the center of gravity of the real plane, so we kept the same center of gravity from the older plane (0.078, 0.00, -0.035). From the first tests of the model we had to make some adjustments to configuration

values for our airframe. Those include "RLL2SRV\_P" and "PTCH2SRV\_P" [8]. We adopted the value 2 to be the proportional gain from roll demands to ailerons and also the value 2 to be the proportional gain from pitch demands to elevator. Higher values allow more servo response but can cause oscillations. In our simulations the model had lower oscillations in straight lines when using a lower value than the default one. This was true when using either L or L1 navigation protocol. In the real flight these values will need to be tuned again.

Considering the main application of all the work developed that is to fly in high altitudes where are high winds, we needed to add the wind to our simulations. So we changed the environment conditions with two simple commands. "param set SIM\_WIND\_DIR xx" and "param set SIM\_WIND\_SPD xx". The first one changes the direction of the wind and the second one the speed of the wind. In the first approach we defined the wind to be 45 degrees from north direction and a speed of  $10\text{km/h}$ . Also, a new test "branch" was made in order to test the controllers performance in straight lines (waypoint navigation) and also in curve lines (flying 8-figure). The two navigation controllers were tested with  $L=50$ . The results are shown in figure 6.

All the simulations done until this point were done without any force reproduction of the tether in the plane. This needed to be also tested before adding a rope to the plane in the real world, or we could destroy the plane due to excess force in the plane. There is still work to do on this way, so we could properly fly the plane in the best conditions.

## 2.2 Hardware

Since we already had a glider (Easy Star II), it was essential to acquire some essential parts compatible with this glider in order to test in real flights. We had not any telemetry radio and the kit only supported manual flights. We had to buy a new receiver so our radio control could be compatible with the *Pixhawk 4* we owned and would be installed in the glider. This receiver uses SRXL - Serial Receiver Link protocol. This is an open data transfer protocol which allows to transport control data from a RC receiver to another device like an autopilot by only using one single line. We can change the flight modes and see the telemetry info with the telemetry radio, however we want to manually control the aircraft in certain occasions like takeoffs and lands. A table with all the components used in this build are present in the table 2 and a image with all electronics in figure 8a.

The glider, as mentioned earlier, is the Multiplex Easystar II (figure 7a). The kit we had is ready to fly, however some parts needed to be assembled and glued (figure 7). For instance, the mounts for the servos on the wings and also the servo rods required a proper glue specially for styrofoam.

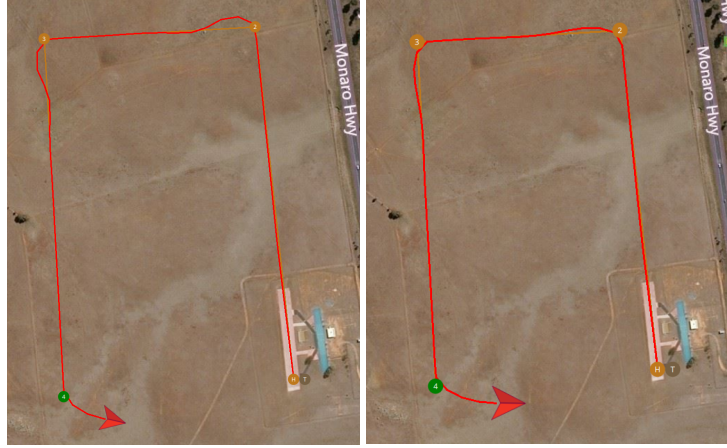
In the first contact with the autopilot board, we loaded the latest version of the firmware. This first approach was done to check if every hardware part we use is working properly. Later we will use our own version, like we described in the last section of this report. After that, we configured the system accordingly to described in *Ardupilot* documentation. Firstly, we calibrated the radio con-

| <b>Part</b>        | <b>Link</b>                  |
|--------------------|------------------------------|
| <b>Plane</b>       | Multiplex Easystar II RTF    |
| <b>Battery</b>     | Roxxy 3S 20C EVO 2200        |
| <b>Transmitter</b> | Multiplex Smart SX           |
| <b>Receiver</b>    | Multiplex RX-4/9 FLEXX       |
| <b>Autopilot</b>   | Holybro Pixhawk 4            |
| <b>GPS</b>         | Holybro GPS Module           |
| <b>Telemetry</b>   | Holybro Telemetry Kit 433MHz |

Table 2: Plane parts

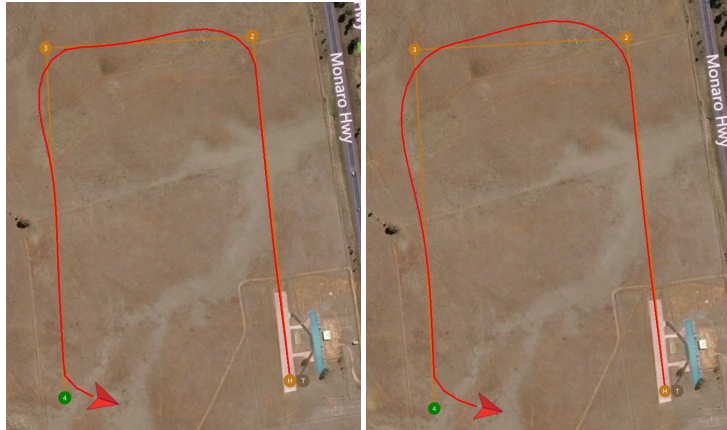
trol, checking if we needed to reverse the signal in the servos. The channel 1,2,3 and 4 of the radio controller were assigned to Ailerons, Elevator, Rudder and Throttle respectively. Then, we calibrated the accelerometer and the compass of the board. Finally, it was time for the autopilot system assembly (figure 8b).

The battery that comes with the RTF kit has a MPX connector. However, the power module uses a XT60 universal connector. Special connectors were needed to connect the battery to the power module (XT60 - MPX and MPX - XT60). We wired all the necessary hardware together and mounted it on the plane (figure 9). The GPS/compass module is placed in the nose of the aircraft as far as possible from the motor to reduce interference. Since the battery to power pixhawk, the servos and the motor is heavy, we studied the center of mass so we could find the best position to place the battery. The assembly of the plane was done and it was ready to perform the first real flights (figure 10).



(a)  $L=20$

(b)  $L=30$



(c)  $L=50$

(d)  $L=80$

Figure 3: Results on waypoint navigation with different  $L$  distances

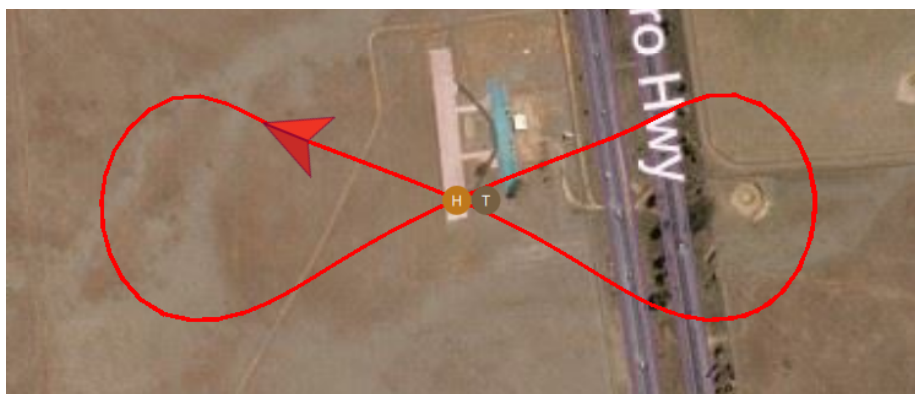


Figure 4: UPWIND mode



Figure 5: Plane ready to takeoff at Oporto airport



(a) L1 simulation



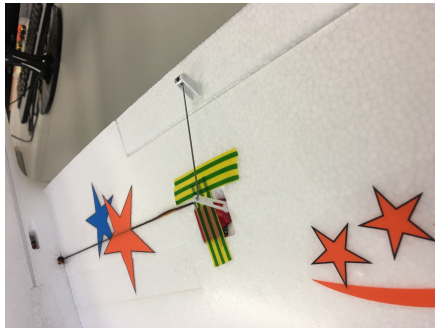
(b) L=50 simulation

Figure 6: Results on waypoint navigation with different navigation controllers





(a) Multiplex Easystar II RTF kit



(b) Servo mounts on an aileron

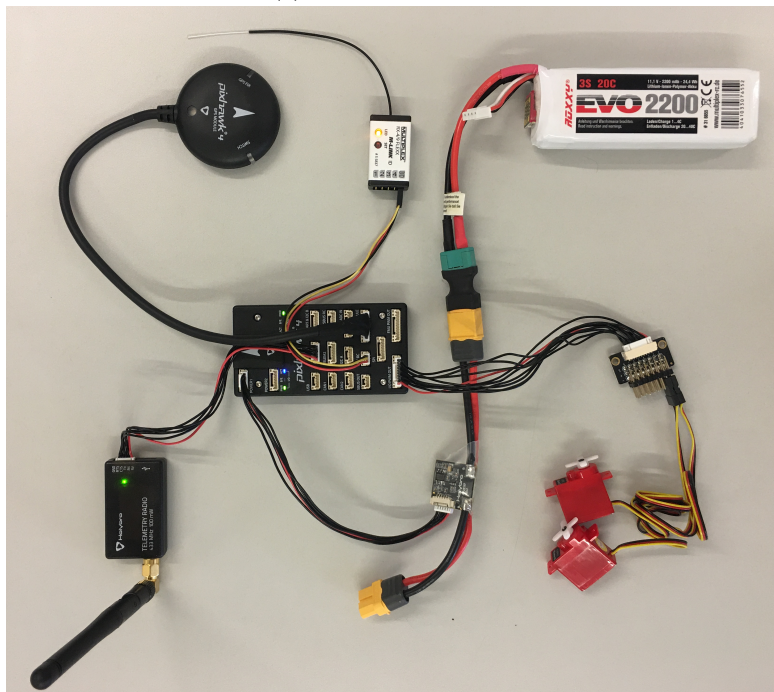


(c) Servo rod on the rudder

Figure 7: Parts Assembly



(a) Required Hardware



(b) Hardware Connected

Figure 8: Pixhawk 4 and other Modules



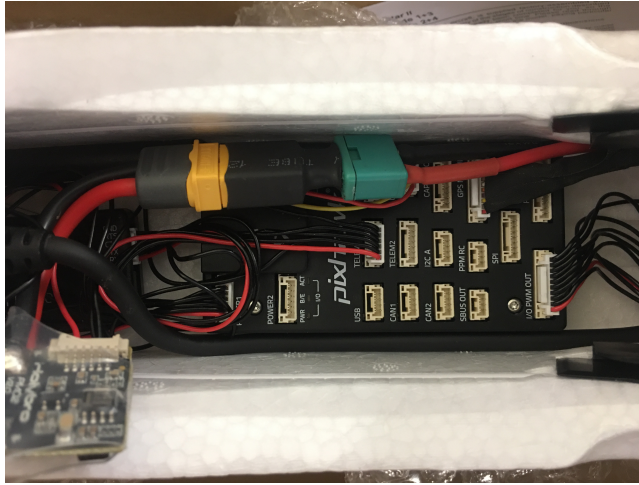


Figure 9: Hardware Position



Figure 10: Plane completed for first flight

## References

- [1] Miles L. Loyd. *Crosswind kite power*. *Journal of Energy*. 1980.
- [2] Philip Bechtle et al. *AWEsome: An open-source test platform for airborne wind energy systems*. 2017.
- [3] G. B. Silva, L. T. Paiva, and F. A. C. C. Fontes. “A Path-following Guidance Method for Airborne Wind Energy Systems with Large Domain of Attraction”. In: *2019 American Control Conference (ACC)*. July 2019.
- [4] *Arduplane*. <http://ardupilot.org/plane/index.html>.
- [5] *JSBSim manual*. <http://jsbsim.sourceforge.net/JSBSimReferenceManual.pdf>.
- [6] *EasyStar I model*. [https://github.com/ArduPilot/apm\\_planner/blob/master/files/flightgear/Aircraft/EasyStar/EasyStar-set.xml](https://github.com/ArduPilot/apm_planner/blob/master/files/flightgear/Aircraft/EasyStar/EasyStar-set.xml).
- [7] *Strato Surfer model*. <https://github.com/UASLab/OpenFlightSim/tree/master/Simulation/aircraft/StratoSurfer>.
- [8] *Configuration Values*. <https://ardupilot.org/plane/docs/configuration-files-for-common-airframes.html>.