NOVA.DroneArena: design and control of a low-cost drone testbed

Hugo Cabrita[1] and Bruno Guerreiro[1,2]
[1]DEEC and CTS/UNINOVA, NOVA School of Science and Technology, 2829-516 Caparica, Portugal.
[2]Institute For Systems and Robotics, LARSyS, 1049-001, Lisbon, Portugal.
E-mails: hugo.cabrita@yahoo.com and bj.guerreiro@fct.unl.pt

***Abstract*—In order to improve the development of unmanned aerial vehicles it is mandatory to make, not only virtual simulations using proper software, but also real experimental trials. Furthermore, to enable rapid prototyping, a controlled environment with all the necessary hardware and software must be thoroughly prepared. The creation of a testbed will solve this need and that is precisely the objective of this article, allowing future drone developments in NOVA School of Science and Technology. In order to be able to follow the trajectory of the drone and ensure that its position is known with a considerable accuracy and low latency, the testbed will rely on Marvelmind Indoor Navigation System. Regarding this system and so that it was possible to have an idea of how accurate it can be, a series of experiments were done with the objective of achieving the best configuration out of it. Furthermore, it is shown that it can be integrated with an autopilot module, in this case PX4 Mini was used. Consequently, so that the communication was as fast and reliable as possible, a Wi-Fi module was used, namely ESP2866. The testbed is then validated by conducting a simple experimental trial inside its perimeter.**

***Index Terms*—drone, testbed, Marvelmind, PX4, control**

I. Introduction

The evolution of technology in the world we live nowadays has been exponentially growing. With that growth there's always the need of something more, something better, something able to solve more problems and in other situations to improve the way things are done, both in terms of time and efficiency. Rotary-wing unmanned aerial vehicles (RUAVs), commonly called drones, appeared with the capacity of solving a lot of problems and improving the way things are done. The list of cases in which this recently developed technology acts is vast. Examples of this technology include architectural 3D rendering, something that before was a time-consuming and difficult task, nowadays with the usage of drones it became an extremely simplified task, or the plethora of military applications, such as supporting forest surveillance in the critical fire period and aerial photography/footages. Another dramatically increasing use of these devices are for transport and delivery purposes, for example to assist vaccine supply chains in low and middle income countries that face enormous challenges [1], and with the evolution its applications will go even further.

* This work was partially funded by FCT project REPLACE (PTDC/EEI-AUT/32107/2017) which includes Lisboa 2020 and PIDDAC funds, project CAPTURE (PTDC/EEI-AUT/1732/2020), and also projects CTS (UIDB/00066/2020) and LARSYS (UIDB/50009/2020).

Although their applications are enormous, they require a lot of researching and development, and obviously, often face some difficulties, such as obtaining an accurate position of the drone in real time, or even perform a trajectory planning, obstacle avoidance and coordination between multiple drones.

The need of implementing a controller that is able to satisfy our needs and is able to perform consistently is a fundamental part of the design of an UAV. It currently is one of the aspects that has been the scope of a significant amount of research over the last decade and certainly will continue to be a focus. Additionally, in order to test the developed controller it is recommended to perform virtual simulations, ensuring its reliability. Furthermore, there is a need to make real experimental trials with drones, therefore it is required to have a reliable testbed which offers the needed conditions to accomplish it, contemplating the necessary software and hardware.

As explained in [2], testbeds have a big importance when it comes to multi-robot research with UAVs as they are usually lightweight and easily influenced by the dynamics of the environment, being able to change their state in a matter of milliseconds. This testbed consists in a sort of arena that has all the required sensors to be able to perform tests with UAVs in the best conditions possible, even allowing trials with multiple UAVs. This structure will provide the conditions to experiment controllers that are already designed in autopilot modules or even custom controllers that haven't been tested before.

In this matter, we can divide the controllers into two big categories: linear controllers and non-linear controllers. As the focus is not to give an example of each, this article will only provide a set of possible controllers that can be replicated and tested in the testbed.

As for linear controllers, some examples that already proved to be possible to achieve decent results are PID (Proportional-integral-derivative) [2], LQR (Linear-quadratic regulator) [3] and MPC (Model predictive model) [4], [5] controllers. In terms of non-linear controllers, SMC (Sliding mode controller) [6], [7], feedback linearization [8] and backstepping [9] have proved to be competent controllers to be chosen.

With regards to the positioning system responsible of capturing the location of the drone(s) there are a lot of solutions that deserve some attention. Some of these are ultra-wide band (UWB) techniques, 2D lasers attached to a rotation servo, visual tags (e.g. ArUco markers) and motion capture systems

(e.g. OptiTrack Motion Capture, VIVE Lighthouse, Pozyx and Marvelmind Indoor Navigation System).

One problem with having autonomous flying aerial vehicles is to know their exact position, for example if you would to fly a drone outdoors, the GPS system can be used even though it has an intrinsic uncertainty, as the conditions in outdoor trials usually allow to have uncertainties around a couple of meters, without compromising safety. Nonetheless, this can be a problem if there is an autonomous drone near obstacles.

Using this GPS module indoors would be completely unbearable as the conditions don't allow to have such position uncertainty, while the GPS signal reception quality is so degraded that the module might not even be able to compute a position solution. In order to avoid this problem, the testbed should come handy allowing to perform experimental trials of drone(s) in a safe environment with an irrelevant imprecision in their position. In order to achieve this goal, the choice of what positioning system to use is an important matter as it can be done by many methods.

The solution encountered after considering various options was a MoCap system, namely Marvelmind, this motion capture system will be responsible by following the drone's trajectory and capture its position. Furthermore, the position will be transmitted to a ROS topic and sent to the PixHawk 4 Mini autopilot. With this solution, and using a proper estimation algorithm that will fuse the data from the sensors belonging to the IMU with the external position coming from Marvelmind, the quadrotor will be able to know its local position and all its states.

The paper is organized as follows. Section II provides the details about the drone testbed, including how the design of the testbed was done, an analysis of the MoCap system used and its integration with a PX4 autopilot and the calibration of the used coordinate systems. Section III shows that the interaction between all the elements composing the testbed and the results obtained in an experimental trial using a Wi-Fi module. Section IV states a brief conclusion about the presented work.

II. Drone testbed

The list of required material, the necessary preparations, the hardware needed, and the software development needed in order to achieve a fully working drone testbed is a rather long, complicated and time-consuming process.

Starting from the beginning, to be able to design the drone testbed there's a physical need to have a place to properly mount it, as it's a rather big arena, the Faculty of Science and Technology provided the necessary installations, inside a room of the Department of Electrical and Computer Engineering. The room is big enough to create a decent testbed with dimensions approximately of 2m x 3.5m x 3m (width x length x height).

The testbed is delimited with a net which function is to protect the drones from crashing into unwanted places and most importantly works as a protection for the people who are controlling the quadrotors and whoever might be in the surroundings. Fig. 1 shows a sketch of how the testbed is implemented. Another point worth mentioning is that safety is rule number one when maneuvering with aerial vehicles, specially indoors. As a safety measure, two ropes were installed and both of them should be attached to the drone's frame, one on the bottom and the other on the upper part.

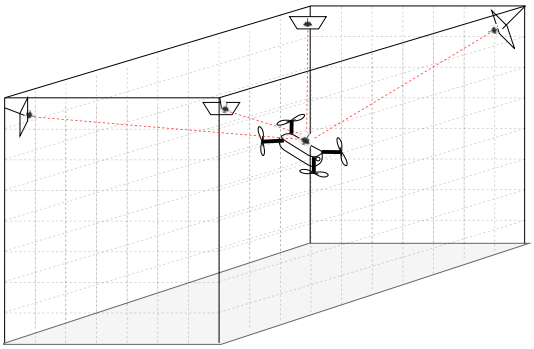

Fig. 1: Testbed drawing

The chosen positioning system to be used was the Marvelmind Indoor Navigation System. This system is composed by six elements: five ultrasonic beacons and a modem. In each of the four upper corners of this net surrounding the arena there will be a stationary beacon, mounted on a rotatable support that is roughly pointing to the center of the arena. These stationary beacons will be responsible for giving the drone's position as accurately as possible. The drone being tested will be carrying a moving beacon, which in practice is the same as the other stationary beacons, the only difference is that this one will be configured to be a moving/mobile beacon (commonly refereed as the hedgehog). All these five beacons will then communicate with the modem which will be connected by USB to the computer running Linux.

Marvelmind provides a simple software that goes by the name of "Dashboard" which facilitates the configuration of these beacons. It allows to configure a large list of settings such as the update rate (Hz) in which the beacons operate, choose an address for each individual beacon, choose whether a beacon is stationary/mobile and it also offers the possibility to monitor in real-time each of the beacons. Upon setting the system up and configuring all the parameters, the hedgehog's position should be mapped. Figures 2 shows the result of the final testbed.

A. Analysis of Marvelmind indoor navigation system

A vast sequence of tests were conducted in order to have an idea of how accurate and how fast the data retrieved by this system is.

In first place, an exhaustive test will be done in order to verify its real accuracy, keeping in mind that the accuracy announced by the manufacturer is roughly 2cm. Furthermore, the main goal of these tests are to understand the impacts some of the parameters have, in order to achieve the most optimal results from this system in terms of position's accuracy, while at the same time maintaining the latency as low as possible.

Fig. 2: Testbed

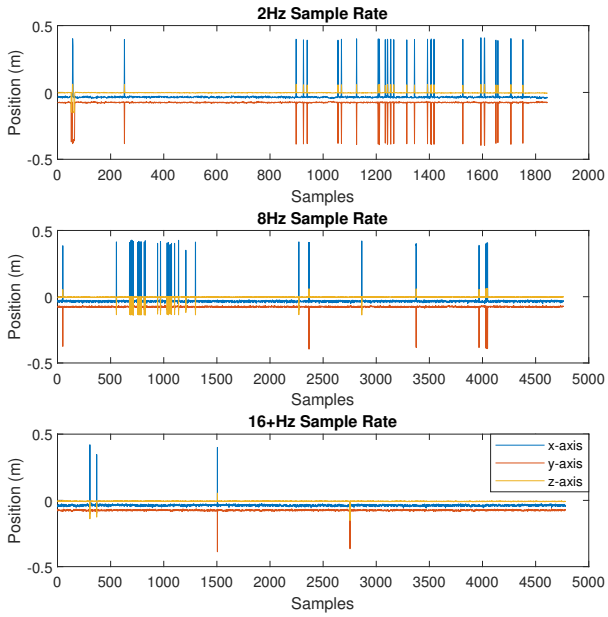

Fig. 3: Output with different update rates (Raw data)

The way this test will be conducted is by placing one of the beacons configured as a mobile beacon (hedgehog) in a certain position (x_0, y_0, z_0) and collect the position given by the system for a certain amount of time, collecting the data which contains the registered position during all the time span, and then properly analyze in order to retain the necessary conclusions from it. These tests will be done with different update rates (Hz), yet maintaining the same environmental conditions throughout the realization of each individual tests while Marvelmind's filtering was turned off.

At this point ROS received the measured values from Marvelmind, and fortunately it provides the necessary documentation with some code examples of possible ways on how to achieve the desired goal. These code examples are basically ROS nodes, each with a specific objective [10]. It was based on these examples that a ROS node was created in order to read the values from Marvelmind and then publish in a ROS topic. With this node, a series of ROS bags were recorded for each test. Upon that, the data can be read through MATLAB and subsequently plotted and analyzed.

The resulting output from these experiments, while varying the update rate of the system from 2 Hz, 8Hz and 16+Hz (maximum update rate) can be seen in Fig. 3, all of these tests have roughly around ten or more minutes of duration.

Taking a quick glance at the results shown in Fig. 3, it's rather obvious that the results are not as good as expected. Unwanted positions are retrieved quite often as there are *spikes* in the graphs, meaning that the position that Marvelmind outputs is quite wrong in those points, which could mislead the controller and cause undesired behaviour. In terms of the different update rates, it's obvious that the results obtained with the higher update rate (16+Hz) provides less wrong position coordinates.

After that, a rolling average algorithm was tested and, furthermore, the filtering method provided by Marvelmind's software itself was enabled. All the resulting outputs were compared and ultimately the configuration that presented a higher accuracy was by enabling Marvelmind's filter and using the highest update rate. It's worth noting that to enable Marvelmind's filter it is needed to choose an input window, the chosen one was 5. We can see these results in Fig. 4.

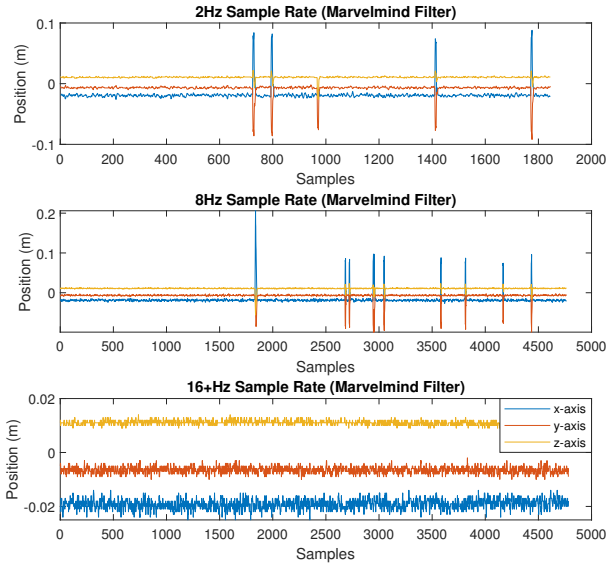

Fig. 4: Output with different update rates (Marvelmind filter)

A quick analysis to the results obtained using this method showed that the mean deviation of x, y and z axis were,

respectively, 0.14cm, 0.10cm and 0.06cm.

Furthermore, so that it was possible to perform experimental trials in the testbed, the PX4 autopilot was gathered. In order to get the best out of the Extended Kalman Filter (EKF2) used for the state estimation of the quadrotor throughout autonomous flight missions inside the developed testbed, it is completely important that the coordinate system used in the testbed and in the autopilot is the same, so that the position received in the PX4 autopilot is the one expected.

B. Testbed and autopilot coordinate systems calibration

Before the calibration, it was clearly noticeable that the coordinate system was not the same, because while maintaining the direction of the movement and varying the yaw angle of the autopilot module, different results were obtained. These results clearly allowed to understand that the coordinate system had to be adjusted so that we can avoid incoherences in the position between both systems. Incompatible coordinate systems would certainly cause unwanted behaviour and increase the chances of crashing the drone.

The autopilot is expecting to receive the coordinates in NED system (X North, Y East, Z Down) [11]. The steps that were done to proceed with the coordination alignment were:

- Establish a position (in the middle of the testbed) which coordinates would be the origin of the referential with $x = 0, y = 0, z = 0$ and mark the position on the ground so it's a visually known position
- Place the drone in this exact position with the hedgehog beacon attached to it
- Align the x-axis from Marvelmind with the magnetic north pole as accurately as possible making the needed rotations on the z-axis, which in our case was 35 degrees.
- Make the needed rotation around x-axis so that the z-axis is facing down - obtaining the NED coordinate system. In our case, as the z-axis was facing up, the rotation was 180 degrees.
- Move Marvelmind's submap so that the established origin position would be the desired one and adjust the height so that the current position is the ground ($z = 0$).
- Move the quadrotor around each axis and visualize the movements in QGC whilst at the same time move its yaw angle to make sure everything is calibrated.

During these calibrations it was possible to see that when trying to align the quadrotor with the north magnetic (where yaw angle is equal to zero degrees) the results were diverging more than desired hence the need to re-calibrate the integrated compass, which can be done easily resourcing QGC. Upon this calibration the results obtained were quite accurate having small divergences, always lower than 5 degrees, which is pretty much an acceptable result.

After all these important adjustments done in "*Dashboard*" we achieve a coherent coordinate system that allows a safe autonomous flight inside the testbed.

C. Integration of Marvelmind system with PX4

Note that before you proceed with any experimental trial of a controller you should use a virtual simulator (e.g. Gazebo) beforehand, ensuring the reliability of the controller.

In terms of the integration between Marvelmind's system and the autopilot PX4, it can be done by multiple methods. Be aware that in order to allow PX4 to an accept an external position and ignore the signal from the GPS module (which does not need to be installed), it is needed to change some parameters in the ground control station (e.g. QGroundControl).

One of the methods is by electrically connecting Marvelmind's hedgehog to PX4 by an adapted cable, where Marvelmind will transmit the position via NMEA 0183 protocol. Although, this method only works if Super-beacons from Maverlmind are being used.

The two following methods are done using ROS as a bridge to transmit the coordinates from Marvelmind to the autopilot, while using Marvelmind's own communication protocol.

The first one is by using a telemetry radio which can be connected to the computer via USB, and the receiver is connected directly in the autopilot.

Furthermore, the solution that is best suitable for a testbed using an external positioning system is by using a Wi-Fi module. This way it is possible to enhance the update rate of the transmission of MAVLink messages. The used and proposed Wi-Fi module is the ESP8266 which operates at 2.4GHz.

At this point, a ROS node was created. This node was responsible by gathering the hedgehog's position and publish into a ROS topic. And then, another ROS node was created in order to subscribe this topic and publish it into */mavros/vision_pose/pose*, which is the topic that PX4 provides to accept position from an external source.

The state estimator used by the autopilot was the EKF2 which shows to be efficient in these scenarios as this algorithm works on a delayed *fusion time horizon*, meaning it can fuse data from sensors with different time delays.

By connecting the autopilot to QGroundControl while at the same time transmitting the hedgehog's position, it was possible to visualize if the data we were sending and the data that is being estimated in LOCAL_POSITION_NED is roughly the same. To visualize this comparison, it is needed to set the parameter MAV_ODOM_LP = 1 so that PX4 will stream back the received external pose as MAVLink ODOMETRY messages.

In a first approach, we observed how well the EKF2 state estimator reacted to the external position by manually moving the hedgehog. Taking the x-axis as an example we can observe Fig. 5 which shows that the estimation done by EKF2 is not great, yet the reason for these undesired results are clearly due to the delay of the received positions from Marvelmind. We can clearly see that the estimation is taking into account the results coming from the IMU and then it should adjust its location based on the coordinates sent from the external source, but as the autopilot doesn't know that the coordinates are delayed it thinks that the drone is still moving, hence the

resulting overshoots when some harder movements are done. A couple of other tests were made, and it was noticeable that the more aggressive the movement would be, the worse the estimation would be.

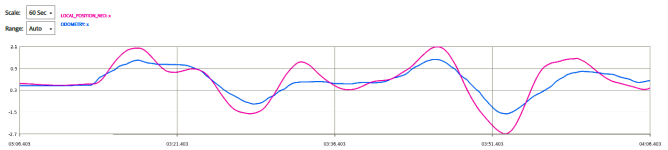

Fig. 5: EKF2 results on x-axis (Delay not configured)

In order to calibrate the delay of the system, multiple tests were made, coming to the conclusion that the delay between the the position being captured by Marvelmind and reaching PX4 is roughly 400ms, therefore the parameter EKF2_EV_DELAY was adjusted accordingly. The results obtained afterwards are shown in Fig. 6

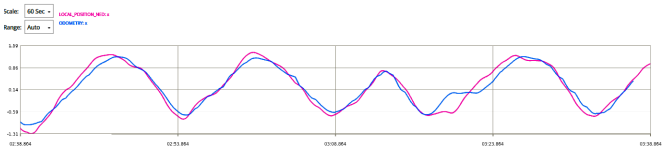

Fig. 6: EKF2 results on x-axis (After delay calibration)

At this point, the results in terms of state estimation and testbed validation in order to allow safe flights to be made inside its perimeter are achieved. The only missing step is to proceed with real experimental trials.

III. Experimental trials using a WiFi module

In first place, so that the Wi-Fi module was able to communicate through MAVLink messages, there was the need to flash the pre build binaries. The FT232 adapter, which can be seen in Fig. 7, provides the ability to make the communication between USB and UART, allowing to flash the mentioned file to the Wi-Fi module upon establishing the electrical connection between these two devices, which can be seen in the same figure. It's important to make sure that the FTDI module is set to 3.3V (instead of 5V) in order to avoid damaging ESP2866.

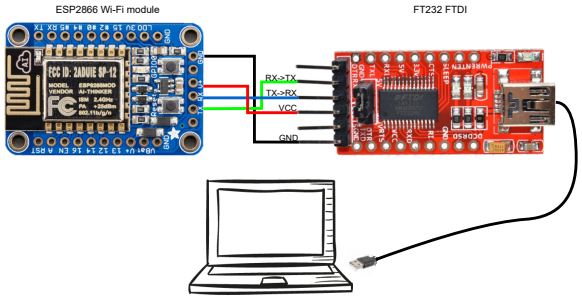

Fig. 7: Electrical connection between ESP2866 and FT232 to flash pre built binaries

Upon connecting the FTDI module via USB to the computer using Linux operating system and making sure that *esptool* software is installed, it remains to set ESP2866 into *flashmode*, which can be done by simply pressing both *Reset* and *GPIO-0* buttons at the same time and then releasing the *Reset* button in first place. A red LED will start blinking with low intensity ensuring that the *flashmode* is enabled.

After flashing the firmware, which is a process that takes less than a minute, ESP2866 was configured as an access point. This allows the computer to connect directly to its Wi-Fi network, which by default, has both the SSID and password set to *pixracer*. To check the status and configure some parameters it is possible to access this network's IP.

Additionally, a cable was made to directly connect ESP2866 into TELEM1 port of PX4 autopilot, by soldering the TELEM1 connector to servo connectors. The autopilot was connected to the computer to finish the configuration, as well as readjust the baud rate of TELEM1 port in QGC to match the one configured in the Wi-Fi module (921600).

Finally, the conditions to perform experimental trials were achieved, hence some preparations had to be done. The blades were attached to the aircraft, the battery was plugged and made sure it was charged enough to complete the whole flight. The protection ropes were attached to the aircraft, the safety measures were double-checked and the flight was done. QGC automatically connected to the Wi-Fi module and started transmitting MAVLink messages.

The interaction between Marvelmind and the autopilot using this ESP8266 microchip can be seen in Fig. 8.

Fig. 8: Interaction scheme between Marvelmind and ESP2866

Furthermore, it's important to ensure that the transmission of data between QGC and the autopilot is smooth and that the parameter MAV_0_MODE is set to External vision, allowing to maximize the number of messages that are transmitted.

The objective of the experimental trial was easy, arm the rotors, takeoff to an altitude of 1 meter and then land again. To perform this trial, the takeoff velocity was lowered.

The results of this trial were recorded to a *rosbag* and plotted in MATLAB so that it is easier to analyze the data. This data can be seen in Fig. 9 where it is possible to see some small oscillation in X and Y axis (roughly 20cm) which might be a result of the mentioned facts. However, the results shown are very solid, the trial was completed with success and the altitude established as reference was correctly achieved, there

was not unwanted movements and there was not the need of any external force to stop the vehicle's movement.

This experimental trial fortifies and validates the work developed in terms of the testbed, integration with the external MoCap system, and even the ESP8266 Wi-Fi module introduced which enhances the performance of the autonomous flights using an external position system. This flight can be seen in the Youtube video: https://youtu.be/lrrOZvvSejY.

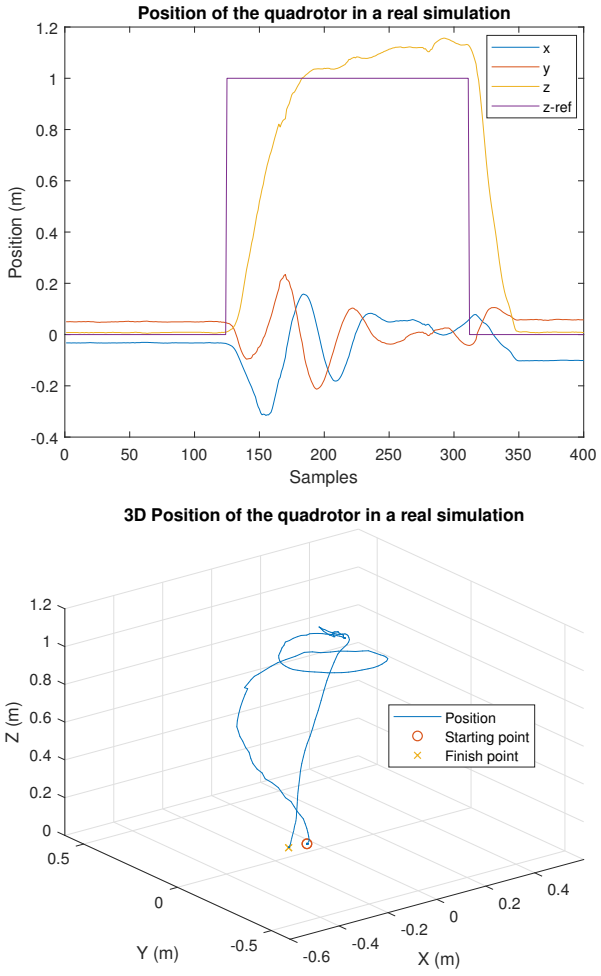

Fig. 9: Experimental trial using ESP8266

IV. Conclusion

The main objective of having a safe environment that provides the infrastructure and is capable of enhancing the development of drone controllers was well accomplished.

The safety measures showed to be appropriate considering the environment in cause. It allows to safely hold the aerial vehicle, preventing it from crashing while keeping it inside the perimeter of the testbed.

Regarding Marvelmind's system, it showed to be a low-cost positioning system that is capable of justifying its price when it comes to tracking trajectories of aerial vehicles. It showed to have a great balance between the accuracy and latency of the computed position.

As for the integration between Marvelmind and PX4's autopilot, after testing three methods, we showed that the best option to consider when it comes to indoor experimental trials using an external positioning system is by using a Wi-Fi module, namely the ESP2866. It truly enhances the transmission rate of MAVLink messages, providing the conditions that are needed to use PX4 within the established recommendations.

In [12] it is possible to find the master thesis in which this article was based and some aspects are far more detailed.The ROS nodes used throughout this article can be accessed in [13]. In this repository. Additionally, an example of a rotary-wing drone controller can also be found.

Acknowledgment

The authors would like to express their gratitude to the Control and Decision scientific area of the Department of Electrical and Computer Eng. at NOVA School of Science and Technology, in particular, to Prof. Fernando Coito and Prof. Luís Palma, for providing the necessary resources and cooperation environment to build this testbed.

References

[1] L. A. Haidari, S. T. Brown, M. Ferguson, E. Bancroft, M. Spiker, A. Wilcox, R. Ambikapathi, V. Sampath, D. L. Connor, and B. Y. Lee, "The economic and operational value of using drones to transport vaccines," *Vaccine*, vol. 34, no. 34, pp. 4062–4067, 2016.

[2] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[3] A. Dharmawan, T. K. Priyambodo, *et al.*, "Model of linear quadratic regulator (lqr) control method in hovering state of quadrotor," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3, pp. 135–143, 2017.

[4] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances," *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[5] A. Hernandez, H. Murcia, C. Copot, and R. De Keyser, "Model predictive path-following control of an ar. drone quadrotor," in *XVI Latin American Control Conference The International Federation of Automatic Control, Cancun, Mexico*, 2014.

[6] C. Edwards and S. Spurgeon, *Sliding mode control: theory and applications*. Crc Press, 1998.

[7] S. Vaidyanathan and C.-H. Lien, *Applications of sliding mode control in science and engineering*, vol. 709. Springer, 2017.

[8] A. Das, K. Subbarao, and F. Lewis, "Dynamic inversion with zero-dynamics stabilisation for quadrotor control," *IET control theory & applications*, vol. 3, no. 3, pp. 303–314, 2009.

[9] D. Cabecinhas, R. Cunha, and C. Silvestre, "A nonlinear quadrotor trajectory tracking controller with disturbance rejection," *Control Engineering Practice*, vol. 26, pp. 1–10, 2014.

[10] "Ros marvelmind package." URL: "https://bitbucket.org/marvelmind_robotics/ros_marvelmind_package/src/master/", Accessed: 04.10.2020.

[11] "Using vision or motion capture systems for position estimation." URL: "https://dev.px4.io/master/en/ros/external_position_estimation.html", Accessed: 25.10.2020.

[12] H. Cabrita, "Design and control of a rotary wing drone testbed," Master's thesis, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, 2021.

[13] H. Cabrita, "Design and control of a rotary wing drone testbed." https://github.com/brunojnguerreiro/nova-drone-arena.git, 2021.