Spherical Fully Covered UAV with Autonomous Indoor Localization

Agustin Ramos(✉), Pedro Jesus Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero

GRVC Robotics Laboratory, University of Seville, Seville, Spain
aramosll@us.es

Abstract. This paper presents a UAV (Unmanned Aerial Vehicle) with intrinsic safety which can interact with people and obstacles while flying in an indoor environment in an autonomous way. A system description including mechanical features, the design of the external protective case, electrical connections and the communication using the Robot Operating System (ROS) between the different devices is presented. Then, the dynamic model of the aerial system taking into account the protective case, the local positioning algorithm (Hector SLAM) and the control models implemented are also described. Different experimental results, which include simulation in Gazebo and real flights are shown to verify the positioning system developed. Two additional experiments have also been tested to validate two emergency safety systems in case of a failure in the position estimation is detected.

Keywords: UAS applications · Onboard localization · Intrinsic safety

1 Introduction

In the last years, the range of applications of UAVs has grown exponentially [1]. Nowadays, UAVs are not only used in observation applications such as mapping, exploration, surveillance or localization, but also in applications in which the aerial platform becomes an aerial robot that is able to physically interact with the environment [2].

The applications of these vehicles to deliver parcels, transport cargo or others which involve working with people are daily growing. For instance, [3] presents a scenario in which a UAV interacts with people to improve the productivity and efficiency of a company and [4] shows deep learning techniques for UAV interaction and collision avoidance. However, in general, most of aerial robots have not been designed to cooperatively work with people.

Several designs for UAVs that fly in proximity or interact with people have been proposed in the literature, and most of them include a protective case to increase safety during operation. For example, [5] compares the behavior against collisions of UAVs with spherical covers, either fixed to the frame or gimballed. A teleoperated spherical UAV commercialized by Flyability [6] follows this last design and has been used for industrial inspection. [7] presents a UAV with an origami-inspired protective case which allows interaction with humans and to reduce its size. [8] shows a hybrid

M. F. Silva et al. (Eds.): ROBOT 2019, AISC 1092, pp. 355–367, 2020.
https://doi.org/10.1007/978-3-030-35990-4_29

quadrotor with a cage that allows to fly and roll. To sum up, most of these systems are teleoperated and only mount a small inspection camera to reduce size as much as possible.

This paper proposes an aerial robot that is fully surrounded by a spherical cover so that it can safely interact and co-work with people, and it also has enough payload to implement a local positioning system onboard to fly indoors using a laser ranger.

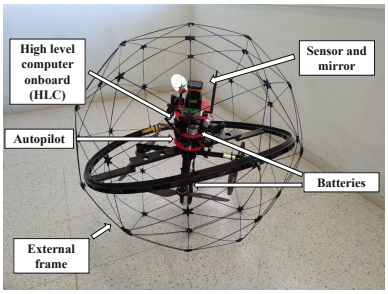

Fig. 1. The aerial platform and the main devices.

Thus, the main contribution of this paper is focused on the use of a system with intrinsic safety, which is suitable to co-work with people in the same environment (see Fig. 1). The proposed system consists of an aerial vehicle with a protective case which acts as a passive system to absorb small impacts. The aerial robot also implements a positioning system to fly autonomously in an indoor environment without depending on external measurements.

The organization of this paper is structured as follows: Sect. 2 presents the hardware and software architecture of the aerial vehicle. Section 3 explains the mathematical model of the system, the algorithm developed to position it and the control model implemented. Section 4 shows the tests performed to evaluate the correct localization. Finally, Sect. 5 includes the conclusions and future works and applications of the solution proposed.

2 System Description

In this section, the mechanical description of the aerial system, the avionics as well as the architecture of communications between the different devices are presented.

In Fig. 2 it can be observed the components of the aerial frame: the autopilot, the high-level computer and the sensor, which communicate with the PC Ground Station. Furthermore, the frame includes the batteries and the motors of the UAV.

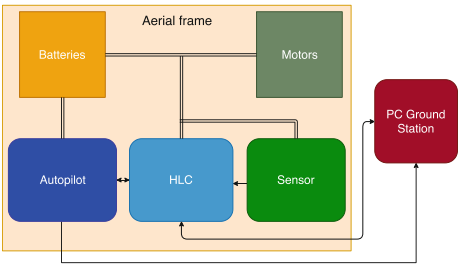

Fig. 2. System architecture. The double link lines represent the electrical connections. The directional and bidirectional connectors represent the communication between devices.

2.1 Aerial System

The aerial vehicle has been designed using an external carbon fiber structure with 3 rotational degrees of freedom, as the frame of a gyroscope. This allows rotating the external side of the platform in case of touching gently with obstacles or living beings, absorbing the impact and transforming it into rotational kinetic energy. Inside the external case, a cross layout quadrotor is in charge of maintaining the attitude of the vehicle and the payload.

Fig. 3. CATIA model for the simulation experiments.

For the simulation tests, a simplified CATIA model has been designed with the same size as the real aerial platform, which is shown in Fig. 3. This model has been imported in Gazebo, the open-source 3D robotics simulator [9].

A. Case Design

The spherical case consists of thin carbon fiber rods with a thickness of 2 mm, being the diameter of the sphere of 870 mm and the weight of 1.7 kg. The mechanical specifications of the aerial vehicle have been determined to have a payload of at least 200 g in addition of the laser sensor.

The rods of the external case form a truncated icosahedron structure. This polyhedron is composed of 12 regular pentagonal faces and 20 regular hexagonal faces. It is

an Archimedean solid which can be constructed from an icosahedron with the 12 vertices cut off one third of each edge.

B. Aerial Platform

As was above mentioned, the frame selected was a custom quadrotor with cross configuration using the T-motor MN4006-23 380 KV and T-motor propellers 14x4.8L as power plant.

The power supply system for the autopilot and the propulsive system is a 6 cells LiPo battery with a capacity of 7000 mAh. On the other hand, for the positioning sensor and the high level computer onboard (HLC), another independent circuit is connected to other 3s LiPo battery with 2700 mAh of capacity. The electrical schemes are represented in Fig. 4. The purpose of the Battery Eliminator Circuit (BEC) modules is to transform the voltage of the batteries (22,2 V or 11,1 V) to 5 V (the supply voltage of the autopilot and the HLC).

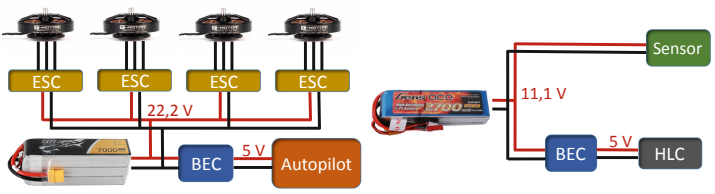

Fig. 4. On the left, the electrical scheme of the autopilot and the motors. On the right, the electrical scheme of the sensor and the HLC.

2.2 Avionics

The avionics architecture is represented in Fig. 5 in which it is also detailed the different communication protocols between the different devices.

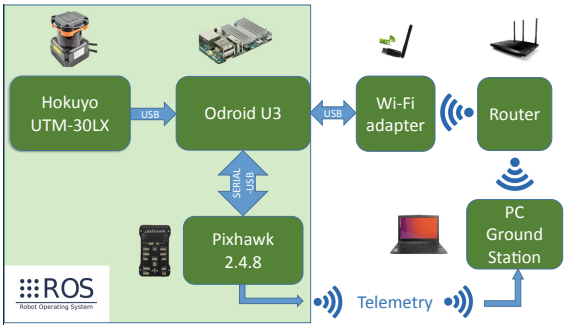

Fig. 5. Scheme of the communication architecture between devices.

A. Hardware description

The Pixhawk 2.4.8 is the autopilot board of the UAV [10]. This has embedded the inertial measurement unit, the magnetometer and runs the flight stack which is a customized version of the PX4 code [11]. The autopilot is physically connected to the Electronic Speed Controller (ESCs) of each of the four rotors and to the HLC through the TELEM2 port using the serial protocol. Furthermore, the port TELEM1 of the Pixhawk is used to connect the air side radio-modem and send information directly to the PC Ground Station using MavLink [12]. The Pixhawk has also connected other sensors and devices (GPS & Compass, receiver, buzzer and switch) as well as the BEC module (the voltage regulator) to turn on the autopilot and power the ESCs.

On the other hand, the HLC is an Odroid U3 [13] which is the 'core' of the avionics. This is a single-board computer and it communicates using its 3 USB ports with the Hokuyo UTM-30LX [14], the Pixhawk and the PC Ground Station, respectively. Thus, the Hokuyo is a scanning laser rangefinder sensor which sends the scan data to the Odroid, as it is detailed in Sect. 3.2, the Pixhawk communicates with it as it is described above and the third port is connected to a Wi-Fi adapter to communicate the Odroid with the PC Ground Station via SSH. This SSH protocol communication is used to log in remotely to the Odroid from the PC Ground Station and in that way to be able to launch several commands in this device. The PC Ground Station receives and monitors the state of the vehicle through the ground-side radio-modem.

To implement the communication between the Hokuyo sensor, the Pixhawk autopilot, and the HLC Odroid U3 has been required to use ROS, as explained in more detail in the next subsection.

B. Software description

Due to the aim of this UAV is to fly indoors autonomously, a local positioning system based on the Hokuyo UTM-30LX laser has been implemented to obtain a position estimation of the vehicle suitable to be included in the position control loop. This sensor provides scan data information sweeping an area of 270° and with a maximum range of 30 m.

The software layer has been implemented under a ROS Kinetic [15] framework running in an Ubuntu 16.04 OS, which are open-source. The main advantage of using ROS is that this is a publisher/subscriber system that easily interconnects different nodes which can be implemented in different programming languages. The software of the autopilot is PX4, which is running in the Pixhawk board [11, 16]. The communication between ROS and PX4 is accomplished through serial communication using the MavLink protocol. In addition, it has been used the open-source Mavros node running as an interpreter between ROS and MavLink [17]. Finally, this research uses a UAV abstraction layer (UAL) [18, 19] that allows a high level custom communication between the UAV and the users (the Ground Control System).

3 Modelling, Localization and Control

3.1 Modelling

The dynamic model of a classical multirotor is usually presented as:

$$M(\xi)\ddot{\xi} + C\left(\xi, \dot{\xi}\right) + G(\xi) = F + F_{ext} \tag{1}$$

Where M is the generalized inertia matrix, C is the Coriolis and centrifugal terms, G represents the gravity component, F is the generalized vector force developed by the rotors and F_{ext} are the external and unknown forces.

Usually, if an aerial platform interacts with the environment the vector of F_{ext} will be composed by the three forces and the three momentums as follows: $F_{ext} = \left[F_{ext_x} F_{ext_y} F_{ext_z} \tau_{ext_x} \tau_{ext_y} \tau_{ext_z}\right]$. However, the main advantages of the presented design is that the spherical case has an articulated link with the multirotor core and the propulsive system, so assuming that the bearing friction is null, the vector of external forces is: $F_{ext} = \left[F_{ext_x} F_{ext_y} F_{ext_z} 0\,0\,0\right]$. Therefore, the spherical case not only provide us an external protection to interact with the multirotor, but also acts in the dynamic model reducing the disturbance on the multirotor when it interacts with the environment. This is due to the kinematic energy that can be assumed by the external case.

3.2 Localization

The position and orientation of the UAV is obtained using the Hector SLAM algorithm [20, 21]. The reason for using this 2D algorithm is because it works with low computational capacity compared to other SLAM algorithms. With it, it has been possible to get the horizontal position estimation (x, y) and the orientation of the vehicle (ψ) while obtaining a map of the flight scenario.

The procedure for obtaining the required information is as follows (see Fig. 7 to observe the list of topics):

First, a ROS package called hokuyo_node [22] is used. This is a driver for Hokuyo laser range-finders which gives several ROS topics, services and sensor parameters. The package shows in the ROS topic "/scan" scans data provided by the sensor. The scan topic is a LaserScan message [23] that contains, among other things, the range data in meters which are stored in a float vector called ranges.

Second, another ROS package named hector_slam serves to get the mapping and the positioning of the vehicle. Using this set of packages, a script is launched to call the Hector SLAM algorithm implemented, which transforms the previous scan data in the topic /slam_out_pose. This is a PoseStamped message [24] that contains the position x, y in meters and the orientation yaw (ψ) in quaternion form of the aerial vehicle.

Third, a script to relay the data of /slam_out_pose to a topic named/uav_1/mavros/vision_pose/pose has been implemented to add the z estimation and two safety systems explained below. Moreover, in order to add the estimation of the z position to the last topic, a system with a mirror has been developed beside the Hokuyo sensor. The mirror is placed opposite the sensor with a rotation of 45º with respect to the xy plane, as it can

see in Fig. 6. Thereby, the samples of the laser scan which reflect on the mirror go vertically to the roof of the room.

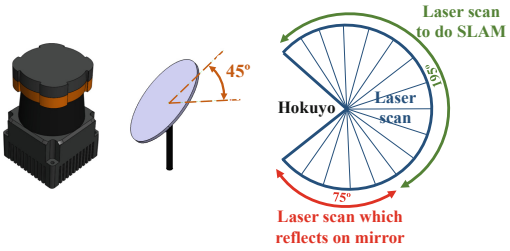

Fig. 6. On the left, model of the Hokuyo sensor and the mirror rotated 45°. On the right, distribution of the samples of the laser sensor.

Due to the Hector SLAM package uses all the scan data provided by the 1080 samples or steps which sweep the 270º detection angle of the sensor [14], it has been necessary to modify part of the code of the Hector SLAM package to avoid to do SLAM in the side where the mirror is located. In this way, part of the mapping script belonging to the Hector SLAM package has been changed, doing that the algorithm only effects between the step number 300 and 1080 (195º), because the mirror is placed between the steps 1 and 300 (75º) of the sensor (Fig. 6).

This way to estimate the z position is measuring the distance between the UAV and the roof. Therefore, to estimate the height of the robot is necessary to change the roof reference to the ground, being *h(t)* the height in the instant *t, range(t = 0)* the distance measured in the initial instant and *range(t)* the distance measured in the instant *t*.

$$h(t) = range(t = 0) - range(t) \tag{2}$$

To improve the z estimation, it is analyzed a ray beam of several steps or samples which reflect on the mirror, calculating the average of the measured distances and removing the measures lower than 20 cm to be considered the structure of the sphere itself. However, the laser scan to do SLAM does not need to remove any sample considered wrong because of the structure, given that the higher amount of samples to process the algorithm and the very thin carbon fiber rods allow to create the map of the scenario and estimate the position without interferences.

This method to obtain the z estimation has been developed to fly at constant height, i.e., in environments with roofs where there is not a large variation in height.

Once x, y, z and yaw estimation is obtained and published in the topic /uav_1/mavros/vision_pose/pose, this information is interpreted and fusion with the rest of the internal sensors in the Pixhawk by the PX4 EKF to get the position and the orientation of the aerial vehicle. This data is required to fly autonomously or in position hold mode, as will be shown in the control Sect. 3.3.

To send information between both topics, two PX4 parameters must be changed to enable the external position and orientation estimation. These are the EKF2_AID_MASK which select the source of the fusion data in the Extended Kalman Filter and the parameter EKF2_HGT_MODE to enable vision fusion an external altitude estimator (z axis) [25].

Figure 7 shows the scheme about the sending data between topics.

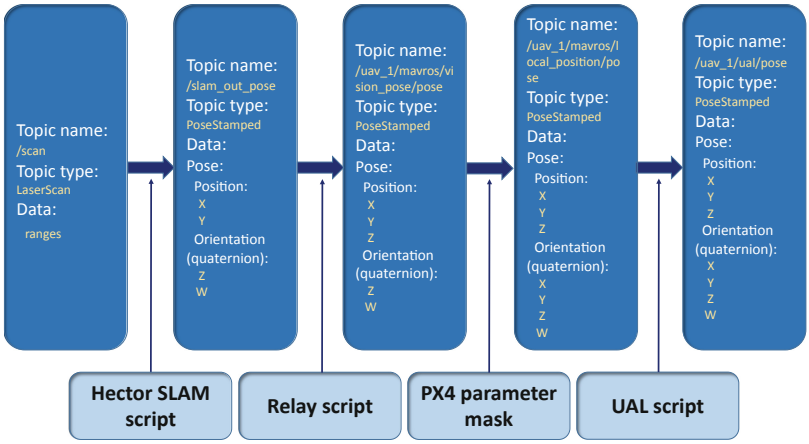

Fig. 7. Scheme of the ROS topics involved in the localization process.

The authors have also created two safety systems in case an emergency caused the vehicle to leave the autonomous flight. The first, if there is a loss of roof reference, the UAV will land automatically.

The second safety system is activated in case the robot reaches a high linear velocity in the axis x or y. That is due to the Odroid U3 can process the localization and the mapping of the algorithm if the UAV flies at not very high velocities. Because of that, the system switches to altitude hold mode when the aerial vehicle is in an autonomous flight and the velocity in the xy plane is high. In Sect. 4 some graphics which show the results of these safety systems are explained.

3.3 Control

The control algorithm of the multirotor enveloped in the spherical case is the standard control scheme with a cascaded PID linear controller as the one presented in Fig. 8.

Although the advantage of the external case could be exploited with a dedicated controller, this paper it was mainly focused on providing the autonomous capabilities of flying in an indoor scenario, leaving the development of applied control algorithms which improve the performance during the operation taking into account the presence of the external case to a future work.

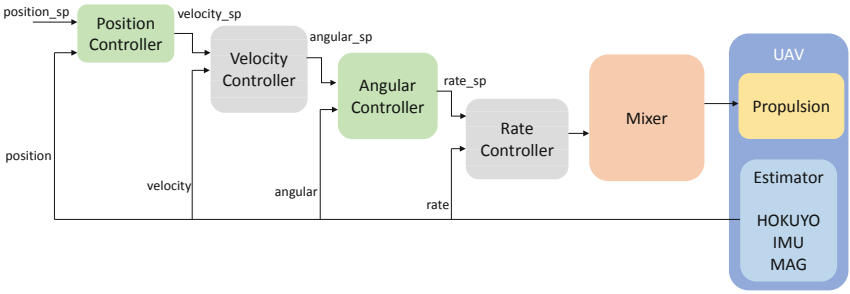

Fig. 8. Control scheme of the UAV.

4 Simulation and Experimental Results

Several simulated and experimental tests have been carried out to evaluate the quality of the localization and the mapping methods in an indoor scenario and also the safety systems mentioned in the previous section. The results are shown along this section as well as some simulation videos which can be found in [26] in which it is possible to see the performance of the algorithm during an autonomous mission. Finally, a real flight video has also been included to be able to observe how to obtain a map in a real environment [26].

In Fig. 9 it can be observed the x and y positions of the UAV compared with the set points commanded respectively. The reference of the position controller is shown in red and the state of the aircraft in blue. During this test, it has been commanded two different waypoints at 2 and 4 m for the x axis, and -2 and -4.5 m for the y axis. As it can see in blue, the position control of the vehicle reaches appropriately the target and it is well tuned in steady state. Although it seems that the controller is very low, this is the effect of limiting the linear velocities of the vehicle which was a mandatory action to grant the stability of the position estimation as it was explained in Sect. 3.2.

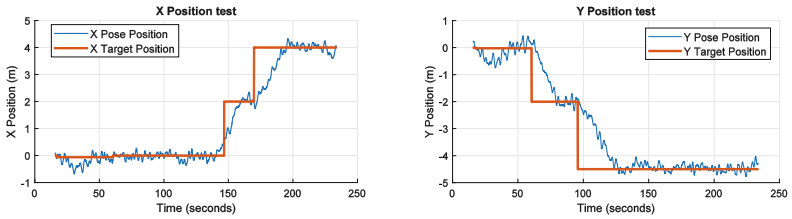

Fig. 9. x and y position of the aerial vehicle. WP tracking.

One of the solutions proposed to accelerate the convergence time of the position controller maintaining the safety conditions, consisted on establishing a limit in the maximum distance between the different waypoints removing the limits on the velocity

controller. By this way, in this simulation the pose gets to reach each target faster than with the first experiment. Figure 10 shows the test.

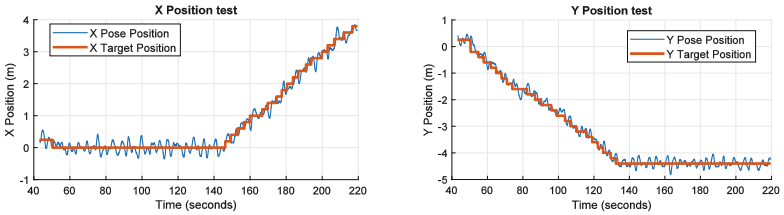

Fig. 10. x and y position of the aerial vehicle. Path tracking.

On the other hand, the results of the estimation and the controller in the z coordinate and yaw angle are shown in Fig. 11. These results clearly show that the tracking of the desired altitude is good enough and validate the solution proposed to estimate of the height and the orientation of the aircraft presented in Sect. 3.2.

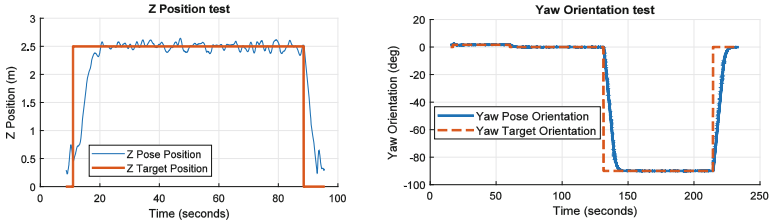

Fig. 11. z position and Yaw orientation of the aerial vehicle.

The yaw angle starts at zero degrees due to the algorithm fixes its own local reference frame when it is initialized. The controller and the estimation results have also been evaluated through the results obtained in Fig. 11. In this experiment, the maximum angular speed in yaw was established in 10 deg/s not only to increase the safety conditions during the autonomous operation avoiding a possible saturation in the motor mixer, but also to improve the mapping results and avoid localization mistakes.

In this test, an autonomous mission has been programmed for this UAV doing first a takeoff, later a translation in x axis 2 and 4 m, a rotation of 90° in yaw angle after that a translation in y axis −2 and −4.5 m and finally a landing. To do this, UAL ROS Services have been used [18, 19].

Regarding the safety systems mentioned in Sect. 3.2, different tests have been carried out (see Figs. 12 and 13). These tests can be also found in [26]. First, in the z emergency test, it can see several flight modes which have been represented by different colors in Fig. 12. The modes 1 and 5 (blue) are LANDED_ARMED, this means that the vehicle is not flying but ready to do it due to the motors are turns on and armed. Mode 2 (green) TAKING_OFF means that the UAV is flying but in the process to

reach the takeoff height commanded. Mode 3 (yellow) FLYING_AUTO is when the aerial vehicle is flying in an autonomous mission sending waypoints different to a takeoff and a land order. Finally, mode 4 (red) is LANDING, the process by which the UAV stops flying. This result shows that while the positioning system detects a roof to reference the z position of the vehicle it allows to perform the take-off maneuver and to fly autonomously. However, as it can be observed, at the moment in which the roof disappears, the z emergency system is activated and the vehicle lands automatically.

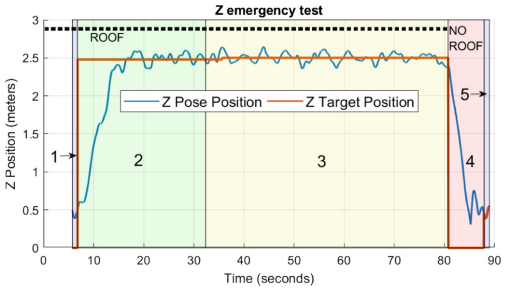

Fig. 12. z emergency test of the aerial vehicle.

The second safety system is the velocity test. It is about changing the flight mode in altitude hold in case of the xy linear velocity exceeds a limit. For this experiment, a waypoint in the x direction has been commanded limiting the xy velocity in autonomous mode in 1.0 m/s. As can be seen in Fig. 13, the flight modes have also been represented by colors being the mode 1 (blue) LANDED_ARMED, the mode 2 (green) TAKING_OFF, the mode 3 (yellow) FLYING_AUTO and the mode 4 (cyan) FLYING_MANUAL. In this last mode the UAV is manually controlled by a safety pilot, except in the height of the vehicle. Therefore, the UAV begins in landed state. Then, the vehicle takes off and after that starts to move in the x direction. In case the xy velocity exceeds the threshold of 0.3 m/s five times checking with a frequency of 10 Hz, the UAV switches to altitude hold mode if the vehicle was flying auto.

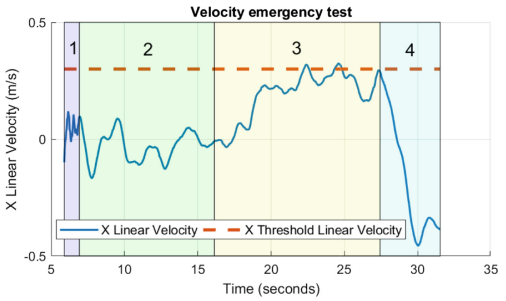

Fig. 13. Velocity emergency test of the aerial vehicle.

5 Conclusions

This paper has presented the design of an aerial vehicle in which a local positioning system has been developed to fly autonomously indoors with intrinsic safety. The mechanical parts of the external protective case of the quadrotor, the electrical configuration and the communication of the electronic devices have been explained. A mathematical model of the UAV, as well as, the integration of the localization system in the platform and a control model have been exposed.

The localization and mapping systems have been tested in simulation and in several indoor environments with walls, doors, containers and other obstacles which allow creating a map with the algorithm developed. Last, as a future work there would be still to test it in industrial areas, like factories, warehouses or similar to check a correct positioning, mapping and flight in these environments.

Acknowledgment. This work has been supported by the national project ARM-EXTEND (DPI2017-89790-R) funded by the Spanish RD plan and HYFLIERS H2020-ICT-2017-1-779411 projects.

References

1. Valavanis, K.P., Vachtsevanos, G.J.: Handbook of Unmanned Aerial Vehicles. Springer, Dordrecht (2015)
2. Sanchez-Cuevas, P.J., Heredia, G., Ollero, A.: Multirotor UAS for bridge inspection by contact using the ceiling effect. In: International Conference on Unmanned Aircraft Systems (ICUAS), Miami, pp. 767–774. IEEE (2017)
3. Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., Siegwart, R.: A UAV system for inspection of industrial facilities. In: IEEE Aerospace Conference, Montana. IEEE (2013)
4. Gandhi, D., Pinto, L., Gupta, A.: Learning to fly by crashing. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, pp. 3948–3955. IEEE (2017)
5. Briod, A., Kornatowski, P., Zufferey, J., Floreano, D.: A collision-resilient flying robot. J. Field Robot. **31**, 496–509 (2014)
6. Flyability webpage. https://www.flyability.com/. Accessed 3 Oct 2019
7. Kornatowski, P.M., Mintchev, S., Floreano, D.: An origami-inspired cargo drone. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, pp. 6855–6862. IEEE (2017)
8. Kalantari, A., Spenko, M.: Design and experimental validation of HyTAQ, a hybrid terrestrial and aerial quadrotor. In: IEEE International Conference on Robotics and Automation, Karlsruhe, pp. 4445–4450. IEEE (2013)
9. Gazebo webpage. http://gazebosim.org/. Accessed 3 Oct 2019
10. Pixhawk documentation page. https://docs.px4.io/en/flight_controller/pixhawk.html. Accessed 3 Oct 2019
11. Meier, L., Honegger, D., Pollefeys, M.: PX4: a node-based multithreaded open source robotics framework for deeply embedded platforms. In: IEEE International Conference on Robotics and Automation (ICRA), Seattle, pp. 6235–6240. IEEE (2015)

12. Atoev, S., Kwon, K.R., Lee, S.H., Moon, K.S.: Data analysis of the MAVLink communication protocol. In: International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, pp. 1–3. IEEE (2017)
13. Odroid U3 documentation page. https://www.hardkernel.com/shop/odroid-u3/. Accessed 3 Oct 2019
14. Hokuyo documentation page. https://www.hokuyo-aut.jp/search/single.php?serial=169. Accessed 3 Oct 2019
15. ROS Kinetic page. http://wiki.ros.org/kinetic. Accessed 3 Oct 2019
16. PX4 documentation webpage. https://px4.io/documentation/. Accessed 3 Oct 2019
17. ROS Mavros webpage. http://wiki.ros.org/mavros. Accessed 3 Oct 2019
18. Real, F., Torres-González, A., Ramón-Soria, P., Capitán, J., Ollero, A.: UAL: an abstraction layer for unmanned aerial vehicles. In: 2nd International Symposium on Aerial Robotics (ISAR), Philadelphia. Springer, Cham (2018)
19. UAL documentation page. https://github.com/grvcTeam/grvc-ual/wiki. Accessed 3 Oct 2019
20. Hector SLAM documentation. http://wiki.ros.org/hector_slam. Accessed 3 Oct 2019
21. Kohlbrecher, S., Von Stryk, O., Meyer, J., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, pp. 155–160. IEEE (2011)
22. Hokuyo node documentation. http://wiki.ros.org/hokuyo_node. Accessed 3 Oct 2019
23. LaserScan message documentation. http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html. Accessed 3 Oct 2019
24. PoseStamped message documentation. http://docs.ros.org/melodic/api/geometry_msgs/html/msg/PoseStamped.html. Accessed 3 Oct 2019
25. PX4 parameter reference guide. https://dev.px4.io/en/advanced/parameter_reference.html. Accessed 3 Oct 2019
26. Link to the video of the experiments. https://www.dropbox.com/sh/4evf4xgn4hslycz/AADmZHtEL8xDlwrCMvzvflCia?dl=0