



MEASUREMENT SYSTEM FOR THE CALCULATION OF THE RELATIVE POSITION BETWEEN A SET OF UAVS

CARLOS ANDRÉS GÁLVEZ CALDERÓN

UNIVERSIDAD DEL VALLE

FACULTY OF ENGINEERING

ELECTRICAL AND ELECTRONICS ENGINEERING SCHOOL

ACADEMIC PROGRAM OF ELECTRONICS ENGINEERING

SANTIAGO DE CALI, 2019



MEASUREMENT SYSTEM FOR THE CALCULATION OF THE RELATIVE POSITION BETWEEN A SET OF UAVS

CARLOS ANDRÉS GÁLVEZ CALDERÓN

ING. EVAL BLADIMIR BACCA CORTÉS, PH.D.

DR.-ING. ESTEBAN EMILIO ROSERO GARCÍA

UNIVERSIDAD DEL VALLE

FACULTY OF ENGINEERING

ELECTRICAL AND ELECTRONICS ENGINEERING SCHOOL

ACADEMIC PROGRAM OF ELECTRONICS ENGINEERING

SANTIAGO DE CALI, 2019

ACKNOWLEDGEMENTS

La realización de este trabajo no hubiera sido posible sin el apoyo de mis directores. Quiero agradecer de forma especial al profesor Bladimir Bacca a quien admiro muchísimo. Desde que fue mi profesor en 7mo semestre me ha motivado a crecer todos los días en el ámbito personal y académico. Le doy gracias a todos los profesores de la escuela de ingeniería electrónica de la Universidad del Valle que semestre tras semestre forjaron la persona que soy ahora. Agradezco al profesor Elkin de Jesús Salcedo quien creyó en mí y me brindó todas las herramientas para desarrollar los conocimientos adquiridos durante la carrera.

Agradezco a mi padrino, Joel Rojas, quien me ha acompañado y apoyado desde el inicio. Su ejemplo ha sido para mí una motivación en todo momento. A mi mamá gracias por todo el esfuerzo y amor que ha puesto para permitirme cumplir mis metas. A mi hermana gracias por el apoyo incondicional que me ha brindado. A mi papá gracias por siempre estar ahí. A mi novia, mi morenita, mil gracias por motivarme cada día en el desarrollo y culminación de la carrera.

Por último, quiero dar gracias a todos y cada uno de mis familiares y amigos que de una u otra forma me apoyaron en este proyecto.

RESUMEN

Con los avances tecnológicos de la actualidad, sistemas como las aeronaves no tripuladas (UAV – Unmanned Aerial Vehicle) son de gran importancia para el progreso. Las aplicaciones que se pueden dar a un UAV son muy diversas, pueden ejecutar labores tan simples como ser un juguete hasta labores tan complejas como ejecutar misiones militares. Su utilización de forma individual resulta muy beneficiosa, pero sus capacidades pueden ser aumentadas haciendo uso de mas de un UAV. Para hacer uso de mas de un UAV es muy importante conocer la posición relativa de cada uno por dos razones. La primera razón es que se puedan ejecutar las labores con precisión. La segunda razón es que se deben evitar colisiones. Las aplicaciones que se pueden dar a un grupo de UAVs son: vigilancia, exploración, rescate, transporte, control de incendios, militares, entre muchas otras.

En este proyecto se llevó a cabo el diseño y la implementación de un sistema para el cálculo de la posición relativa entre un grupo de UAVs. El sistema fue denominado GALS – Group Autonomous Location System. Un sensor LiDAR 2D (Light Detection And Ranging) fue agregado a la estructura de los UAVs a utilizar. La posición relativa es calculada partiendo del análisis de imágenes de rango obtenidas con el sensor LiDAR. El análisis de las imágenes permite calcular la distancia y la dirección a las cuales se encuentran los UAV cercanos. Al mismo tiempo se obtienen los datos correspondientes a los obstáculos en el rango de operación del sensor. Los puntos que no corresponden a UAVs son considerados como posibles obstáculos. Los datos generados por GALS son reportados al software de control de vuelo del UAV.

El usuario tiene la posibilidad de visualizar la información generada por GALS. La visualización se realiza a través de la interfaz gráfica de la estación base. Desde esta interfaz, el usuario tiene la posibilidad de configurar los diferentes parámetros del sistema como son: Configuración del sensor LiDAR; Modificación de límites de visualización; Calibración del sensor laser; Comenzar y detener la operación de GALS.

El sensor LiDAR utilizado es pequeño y liviano, características importantes para equiparlo en un UAV. Su rango máximo es de 40 metros, lo que permite una amplia visualización el entorno. La precisión de los datos de rango generados permite a GALS detectar y ubicar UAVs en un rango entre 1 y 4 metros. Se realizaron pruebas al sistema ubicando UAVs en 12 posiciones del rango de detección. El error medio obtenido en distancia fue menor a 8cm y en angulo menor a 5 grados. El éxito en detección llego al 93%.

La información generada por GALS es almacenada en forma de archivos de texto. La información contenida en los archivos de texto permite el análisis posterior al vuelo de los datos generados. También es posible el análisis posterior al vuelo del comportamiento del sistema ante las solicitudes del usuario.

Finalmente, se obtuvo un grupo de 4 UAVs en la capacidad de detectar y localizar otros UAVs y obstáculos cercanos. Su ubicación relativa con respecto a su grupo y los objetos detectados son ahora parte de la información disponible en el sistema de control de vuelo.

Keywords: UAV, Posición Relativa, Drones Colaborativos, Drones Autónomos, Detección, LiDAR.

ABSTRACT

With the current technological advances, systems such as Unmanned Aerial Vehicle (UAV) are of great importance for science's progress. There are a variety of applications that can be given to an UAV. UAVs are capable of perform tasks as simple as being a toy. But they are also capable of perform tasks as complex as executing military missions. The use of a single UAV is useful, but its capabilities can be increased by using more than one UAV. The flight control for a group of UAVs need the knowledge of the relative position between UAVs. The first reason is that the group's tasks can be executed with precision. The second reason is that collisions must be avoided. The applications that can be given to a group of UAVs are: surveillance, exploration, rescue, transport, fire control, military, among many others.

In this project, the design and implementation of a measurement system for calculation of the relative position among a group of UAVs was carried out. The system was called GALS - Group Autonomous Location System. A 2D LiDAR sensor (Light Detection And Ranging) was added to the structure of the UAVs. The relative position is calculated from the analysis of range images obtained with the LiDAR sensor. The analysis of the images allows to calculate the distance and the direction at which the nearby UAVs are located. The obstacle's data in the operating range of the sensor are also obtained. The points that do not belong to UAVs are considered as possible obstacles. The data generated by GALS is reported to the flight control software of the UAV.

The user has the possibility to visualize the information generated by GALS. The visualization is done through the base station's graphics interface. From the interface, the user has the possibility to configure the different system parameters such as: LiDAR sensor's configuration; Modification of visualization limits; Calibration of the laser sensor; Start and stop the operation of GALS.

The LiDAR sensor is small and light, important features to use it in an UAV. Its maximum range is 40 meters, which allows a wide visualization of the environment. The accuracy of the range data allows GALS to detect and locate UAVs in a range between 1 and 4 meters. The system was tested by placing UAVs in 12 positions in the detection range. The average error obtained in distance was less than 8cm and less than 5 degrees in angle. The success in detection reached 93%.

The information generated by GALS is stored as text files. The information in the text files allows the post-flight analysis of the data generated by GALS. Post-flight analysis of the behavior of the system is also possible.

Finally, a group of 4 UAVs was obtained. Each UAV has the capacity to detect nearby UAVs and obstacles. Their relative location and the detected objects are available in the flight control system now.

Keywords: UAV, Relative Position, Collaborative Drones, Autonomous, Detection, Machine Vision, LiDAR.

CONTENT TABLE

Acknowledgements	3
Resumen	4
Abstract	5
Content table	6
List of Figures	8
List of tables	10
1. Introduction	11
1.1. Problem Statement	12
1.2. Objectives	12
1.2.1. Objective of Study	12
1.2.2. Specific objectives	12
1.3. Proposed solution	13
1.4. Document structure	14
2. State of the Art and Theoretical Framework	15
2.1. State of the art	15
2.2. Theoretical Framework	19
2.2.1. Unmanned Aerial Vehicle	19
2.2.2. UAV Classification	20
2.2.3. LiDAR	20
2.2.4. GPS/GNSS	21
2.2.5. RTK	22
2.2.6. RUP	22
2.3. final Observations	24
3. Design and implementation of GALS	25
3.1. Design of GALS	25
3.1.1. Functional and non-functional Requirements	25
3.1.2. Conceptual Diagram	26
3.1.3. Class diagram of GALS	27
3.1.4. Flowchart of GALS	31
3.2. Implementation of GALS	33

3.2.1. The sensor: Laser Range Finder	33
3.2.2. The Identification of UAV problem using a laser range finder	36
3.2.3. Identification Algorithm.....	38
3.2.4. Calculation of the relative position	41
3.2.5. Communication Module with the Flight Control App	45
3.3. Storage module (android).....	48
3.4. Base station	49
3.5. Final observations.....	53
4. Validation and tests	55
4.1. FUNCTIONAL tests	55
4.1.1. Automatic Detection of UAV using a Laser Range Finder	55
4.2. Field Tests	73
4.3. conclusions	75
5. Conclusion and future works.....	76
5.1. conclusions	76
5.2. Future Works	77
6. Annex	78
7. Bibliography.....	79

LIST OF FIGURES

Figure 1.1. Current UAVs applications. From top to bottom and from left to right: a) Recue [2]; b) Aerial photography [3]; c) Fire control [4]; d) Security [5]; e) Military [6].	11
Figure 1.2. One hundred UAVs organized in the sky [8].	11
Figure 1.3. Communication lines between sensor and processing devices on each UAV [17][18][19][20].	13
Figure 2.1. Basic explanation of GNSS RTK operation [38].	22
Figure 2.2 Chart for description of RUP development process [41].	23
Figure 3.1. Conceptual diagram of GALS.	27
Figure 3.2. Class diagram for the Arduino.	27
Figure 3.3. Class diagram for the Smartphone.	29
Figure 3.4. Class diagram for the Base Station	30
Figure 3.5 Flowchart of the Arduino's program execution.	32
Figure 3.6 Flowchart of the Smartphone's program execution.	33
Figure 3.7. Sweep V1 360, LiDAR from Scanse.	34
Figure 3.8. Structure for the calibration of the LiDAR sensor.	34
Figure 3.9. Comparison of sensor readings against ideal behavior.	35
Figure 3.10. Comparison of sensor readings against ideal behavior (Zoom).	36
Figure 3.11. Conceptual diagram for the identification of UAVs process.	36
Figure 3.12. 2D image captured with the Sweep LiDAR 360. a) The sensor is located in position (0,0). b) Enlarged view of a captured cylinder.	37
Figure 3.13. Cylinder used to perform identification tests with LiDAR.	37
Figure 3.14. Test data for the identification process.	38
Figure 3.15. Resulting points after the range delimitation step.	38
Figure 3.16.. a) Angle of separation between two consecutive points. b) Distance of separation between two consecutive points.	39
Figure 3.17. Result after the false edge elimination process.	39
Figure 3.18. Remaining objects in the test data after the diameter identification.	40
Figure 3.19. Arc length measurement of a cylindrical object.	40
Figure 3.20. a) Left: A cylinder observed from two different positions. b) Right: A rectangle observed from two different positions.	41
Figure 3.21. Results of the identification process of cylinders in 2D data.	42
Figure 3.22. UAV prototype, part of the "Recue and exploration with multi-agents" project.	42
Figure 3.23. UAV size. a) Side view. b) Top view.	43
Figure 3.24 a) GALS's cylinder size. b) Final structure of the UAV.	43
Figure 3.25. UAV's motor operation.	44
Figure 3.26. a) Left: UAV motor, MT2216 [46]. b) Right: Arduino DUE, development platform [47].	44
Figure 3.27. GALS Communication's block diagram.	45
Figure 3.28. Data frame for the communication module.	45
Figure 3.29. Block diagram of the Storage module operation.	48
Figure 3.30. Block diagram of the Storage module operation.	49
Figure 3.31. Screenshot of the Base Station application.	49
Figure 3.32. Communication protocol between the Ground Station and the Smartphone.	50
Figure 3.33. Screenshot of the GALS tab in the Ground Station program.	50
Figure 3.34. Picture of the UAV with GALS and the Base Station.	52

Figure 3.35. Screenshot of the execution of the Base Station application.	52
Figure 3.36. a) Piece of text file saved by the base Station. b) Plot of the saved data.	53
Figure 4.1. Physical configuration for the Flight Controller + GALS GUI.	56
Figure 4.2. Test positions used for the static test.	73
Figure 4.3. Picture of the test location for static test.....	74
Figure 4.4. Plot of the results for the static test.....	74
Figure 6.1. Annex directory structure.	78

LIST OF TABLES

Table 2.1. Compilation of Scientific papers related to this Project.....	16
Table 2.2. Classification of the UAVs depending on their physical configuration.....	20
Table 3.1 Functional requirements for GALS.....	26
Table 3.2 Non-Functional requirements for GALS.....	26
Table 3.3 Average range readings for calibration	35
Table 3.4 Indicators usage in communication module.....	46
Table 3.5 Arduino-Smartphone API.....	47
Table 3.6 Smartphone-Ground Station Protocol messages added for GALS	51
Table 4.1 GALS Functional Requirements	56
Table 4.2 Functional Test: Arduino-Smartphone Communication.....	57
Table 4.3 Functional Test: LiDAR Data Acquisition.....	58
Table 4.4 Functional Test: UAVs Identification.....	60
Table 4.5 Functional Test: Data Filtering.....	62
Table 4.6 Functional Test: LiDAR Configuration	64
Table 4.7 Functional Test: GALS Data Transmission.....	66
Table 4.8 Functional Test: Integration with Flight Control Software	68

Chapter 1

INTRODUCTION

1.1. Problem Statement

1.2.1. Objective of Study

1.2.2. Specific objectives

1.3. Proposed solution

1.4. Document structure

Nowadays, the use of Unmanned Aircraft Vehicles (UAV) has become increasingly common. In its beginnings, it was a military artifact and now it has become an almost daily object. The spread of this technology has made available to science a powerful tool for obtaining data. Currently, UAVs perform tasks in diverse areas, like searching for missing persons, aerial photography, fire control, security and military applications, among others [1]. An example of each of those applications can be seen in the figure 1.1.



Figure 1.1. Current UAVs applications. From top to bottom and from left to right: a) Recue [2]; b) Aerial photography [3]; c) Fire control [4]; d) Security [5]; e) Military [6].

The massive production of this type of technology (UAV) in all of its varieties, has allowed its development not only as an individual device, but also as a group formed by several UAVs. It has even been controlled the formation of hundreds of UAVs [7], as can be seen in figure 1.2.



Figure 1.2. One hundred UAVs organized in the sky [8].

This project is part of a bigger research project, which objective is the development of a group of UAVs that will perform tasks of data collection for precision agriculture. In this context, one of the challenges of using multiple UAVs is that they can collide with each other. In order to maintain a formation, the UAV must know its own position in relation to the environment or other UAVs. With the knowledge of the own position, the UAV will even be able to avoid objects.

The present project aims to give continuity to the research with UAVs, giving them sufficient perception to visualize their environment and to recognize other UAVs that may be around them.

1.1. PROBLEM STATEMENT

A drone, known as unmanned aerial vehicle (UAV), is a flying vehicle that has no crew, its flight can be controlled by a remote control or through preloaded flight plans [9]. Initially they were conceived for military use, but recently it has become popular among the civil population. The uses that UAVs currently receive range from photographic shoots at social events to scientific research. UAVs cost have achieved prices so low that it is common to see them as entertainment or toys.

They can be found in various physical configurations, initially they had the characteristics of an airplane and currently it is common to find them in the form of a quadcopter. A quadcopter is a central body supported by rotors located in ascending vertical direction. Quadcopter type provides a better control of the vehicle in the air but has a much lower displacement speed. Due to the low cost of this type of aircraft, nowadays is economically profitable to have multiple aircraft in flight. A group of drones represents new challenges like: how to perform displacements without drones suffering a collision?. This problem is directly related to the knowledge of the relative position of each drone with respect to the others. For the calculation of the relative position of an UAV different sensors have been used, on-board and off-board. Examples of that sensors are: cameras [10],[11],[12], GPS [7], microphones [13], laser beams [14],[15], IMUs [7] and WiFi devices [16].

Each sensor presents a solution to the problem posed but each one has its particular limitations. The cameras provide information about surroundings, but it is limited by the required lighting to properly work. The GPS gives a global reach but the accuracy is low, and it does not work indoors. The microphone achieves good precision and good range of operation, but it only measures the direction of a source of sound, not the distance to that source. The laser presents a complete solution, but it is only profitable for single point or 2D data, and it is difficult to detect another drone using 2D information. IMUs provide a lot of information to work with, but they are very sensitive to noise.

The present project aims to propose a solution to the calculation of the relative position of an UAV using drones of quadcopter type. The engineering question is formulated: How to determine the relative position of a drone with respect to other drones?

1.2. OBJECTIVES

1.2.1. Objective of Study

- To develop a 2D measurement system for the calculation of the relative position between a set of UAVs.

1.2.2. Specific objectives

1. To perform a state of the art for the calculation of the relative position between a set of UAVs.

2. To design a 2D measurement system for the calculation of the relative position between a set of UAVs.
3. To implement the 2D measurement system for the calculation of the relative position between a set of UAVs.
4. To validate the operation of the developed system.

1.3. PROPOSED SOLUTION

In figure 1.3 the developed solution is shown. The operation starts with the acquisition of sensor data. The processing of the data is performed in the Arduino. Finally, the data obtained is provided as a tool to the flight control system that runs on the Android.

In order to solve the engineering question, the design and implementation of the GALS system (UAV-Group Autonomous Location System) was carried out. The GALS system provides a drone with the ability to perceive its surroundings, detecting and locating nearby UAVs. The GALS system is oriented to using in multi-agent UAVs groups. The system is divided into 3 stages. In the first stage the acquisition of the range data from the LiDAR sensor is done. In the second stage the analysis of the obtained data is performed in order to detect and localize nearby UAVs. In the third stage the obtained data are transmitted to the flight control system.

Each UAV is equipped with a LiDAR sensor, an Arduino development platform, an Android device and a known cylindrical radio cover. The LiDAR sensor provides 2D range images with an operating distance of 1 to 40 meters. The Arduino allows the processing of the range data, also establishes communications with the sensor and with the Android. The Android device runs the flight control system, which is the final receiver of the data obtained. The communication between the LiDAR sensor and the Arduino is done using an embedded protocol, which runs over a serial protocol. The communication between the Arduino and the Android is done using an embedded protocol, which runs over a USB protocol.



Figure 1.3. Communication lines between sensor and processing devices on each UAV [17][18][19][20].

1.4. DOCUMENT STRUCTURE

The document's structure is built as a book, in which the chapters shows the development and the solution given to the each of the specific objectives.

In the second chapter, entitled: "State of the art and theoretical framework" is presented at the beginning the review of national and international background of this project. The related projects are compared in a table where its main characteristics are shown. Then, the projects are analyzed individually and separated in groups of similar characteristics. Then a general analysis of the background and a conclusion is made. Finally, the theoretical framework is shown, where concepts that are necessary for the understanding of the following chapters are described.

In the third chapter, entitled: "Design of GALS" The results obtained in the design phase of the project are described. This chapter includes the documents related to the RUP method. The communication protocols used between modules are also specified.

The fourth chapter, entitled: "Implementation of GALS" describes in detail the hardware and software used to accomplish the specifications stated in the design phase. The operation of the detection software and its implementation is shown. It is also shown the operation of each module and the way in which they communicate.

In the fifth chapter, entitled: "Validation and test" tests are performed on the implemented prototype. It starts with integration tests to check the general perform of the system. Then field tests are carried out in which the operation of the system is validated in uncontrolled environments.

Finally, in the sixth chapter, entitled: "Conclusions and future work" it shows the results obtained after the implementation and validation phases for the first prototype. It is also shown the limitations of the system and the projects that could arise in the future.

Chapter 2

STATE OF THE ART AND THEORETICAL FRAMEWORK

- 2.1. State of the art
- 2.2. Theoretical Framework
 - 2.2.1. Unmanned Aerial Vehicle
 - 2.2.2. UAV Classification
 - 2.2.3. LiDAR
 - 2.2.4. GPS/GNSS
 - 2.2.5. RTK
 - 2.2.6. RUP
- 2.3. final Observations

This chapter begins with the presentation of the State-Of-The-Art for the calculation of the relative position in mobile robotics. National and international projects are analyzed individually. For their comparison, the following characteristics are taken into account: Reference point, Special stamp usage, Range, Accuracy, Returned data, Number of vehicles tested and Approximate sensor cost. Based on its characteristics the projects are joint into categories. The categories are: visible light-based, laser-based, sound-based, RF-based, Dead reckoning and Data fusion.

The theoretical framework aims to describe some concepts that will be useful for the reader with this document. The theoretical framework is composed of: UAV, LiDAR, GPS, GPS RTK, GNSS and RUP.

At the end of this chapter final remarks are given. The State-Of-The-Art is used as starting point to choose the system for implementation.

2.1. STATE OF THE ART

The problem of finding the relative position of a drone with respect to its equals, or its environment has been approached multiple times during the last few years. On-board and off-board sensors have been used. A comparison of works whose approach is directly related to what is intended in this project is shown in Table 2.1.

In this table, the field "Category" shows the classification of the project; the field "Position relative to" shows the reference point used to calculate the relative position; the field "Special stamp" shows, if used, the marks used for reference; the field "Range" shows the operating range of the prototype; the field "Accuracy" shows the maximum error in the measurement; the field "Returned data" shows the information that the positioning system returns; the field "Number of vehicles tested" shows the maximum number of UAVs tested simultaneously; and, the field "Approximate cost" shows the approximate cost of the materials for the implementation of each positioning system.

The classification of the projects has been done according to the variable sensed by the receiver, this is the case of the categories visible light-based, laser-based, sound-based and RF-based. In addition, the category "Dead reckoning" is presented, where an inertial measurement unit is used (IMU). The current position is calculated starting from an earlier position and the data that the IMU provides. Finally, there is the category "Data fusion" which has projects that combine two or more of the already mentioned categories.

Table 2.1. Compilation of Scientific papers related to this Project.

Reference	Category (Sensor)	Position relative to	Special stamp	Range [m]	Accuracy [m]	Returned data	Number of vehicles tested	Approx. sensor cost [USD]
[9]	Visible Light (Camera)	QR codes	QR Code	1.8	0.02	Relative position	1	70
[10]	Visible Light (Camera)	Other drone	Passive marker	3	0.001	Distance and direction	2	70
[11]	Visible Light (Camera Off-board)	Cameras	Passive marker	5	-	Relative position	2	1680
[14]	Laser (IR Laser)	Other drone	None	14	0.04	Distance	2	160
[15]	Laser (LiDAR)	Environment	None	30	0.05	Relative position	1	4810
[13]	Sound (Microphone)	Other drone	Buzzer	60	15°	Direction	6	10
[21]	RF (GPS)	Environment	None	Global	3	Global position	1	60
[16]	RF(WiFi)	WiFi APs	None	7	0.002	Relative position	1	150
[27]	RF (UHF RFID)	RF emitters	RF Tag	7	0.13	Relative position	1	2000
[22]	RF(Bluetooth)	BLE emitter	BLE beacon	10	2	Relative position	1	200
[24]	Dead reckoning (IMU)	Start point	None	100	0.7	Relative position	2	450
[23]	Data fusion (IMU and RF)	Start point and RF emitters	None	10	0.35	Relative position	0	600
[12]	Data fusion (GPS and camera)	Environment and other drones	None	3m-Camera Global-GPS	-	Distance , direction and global position	2	110
[7]	Data fusion (GPS and IMU)	Environment and start point	None	300	0.003	Global position	100	12300
[26]	Data fusion (IMU, WiFi, Barometer, GPS, Magnetometer and Light sensor)	Known environment	None	70	5.76	Global and relative position	5	450
[28]	Data fusion (IMU, Magnetometer, GPS and Camera)	Environment and other drones	None	100	11	Global and relative position	3	400

On the visible light-based category, the camera is the main sensor. The camera allows the identification of surrounding objects and thus you can obtain your own position. This sensor has been used in the projects [9], [10] and [12]. The range of operation at its best reach 3 meters, which is a low range for the purpose of this project. Another drawback is that it needs a good illumination to work properly. A camera can be bought for less than 110 USD but estimate the self-position with a single camera is very complex. A great accuracy can be reached with this sensor, up to $\pm 1\text{mm}$.

The use of cameras has also been tested by locating them off-board. A clear example of this is shown in the document [11]. 24 cameras are placed in the ground to capture the relative position of a drone. It has a better range of operation with up to 5 meters. The weak point of this configuration is that it only works in the space located within the polygon determined by the lenses of the cameras. The number of required cameras is also a limitation because of the cost, which for 24 cameras is approximately 1680 USD.

The visible light category, shows an economic sensor that can achieve good accuracy. It has limitations on the operating range, the illumination required and the complexity of its implementation. it was determined that the best use of a camera is on board because of the coverage area.

The Laser-based category presents a sensor capable of measuring the distance to an object (LiDAR). A simplified version of its use can be seen on [14]. Here an infrared laser sensor is used to stay at a fixed distance of an object. It has advantages regarding the operating range (14m) with an accuracy of $\pm 4\text{cm}$. The sensor used has the size of a coin and it doesn't require an illuminated environment to work properly. The sensor can be bought for only 160 USD. The weak point of this method is that it performs the distance measurement in a single point.

A more sophisticated use of the laser is shown in [15]. In this project a Hokuyo 30LX is used, it has 270° and 30m of scanning range and an accuracy of $\pm 50\text{mm}$. The cost of the sensor is 4810 USD which is not cheap. The drawback of the two sensors presented is the difficulty of determining a UAV with 2D horizontal or single point data.

The laser category shows a sensor of excellent performance in range and accuracy. There are two characteristics that can limit the implementation of these sensors in UAVs. The first one is that those sensors are not cheap. The second one is that the weight of this type of sensor increases considerably for improved properties.

On the Sound-based category is the lowest price project [13]. Its total price is 10 USD because it only uses a buzzer and three microphones. The sound of the buzzer is used to find the direction in which the surrounding drones are. Each drone has a specific frequency buzzer so it is possible to differentiate one drone from another through filters. It has an operating range of 60 meters and a precision of 15° . It has 2 drawbacks, the first one is that it can be affected by ambient noise. The second one is that it does not measure the distance with respect to other drones, just the direction where they are.

The sound category shows an economic sensor with good accuracy and range. The limitation with the delivery of the distance with respect to other UAVs discards it for use in this project. Even so, combining its use with a sensor that delivers the missing magnitude, would be feasible of implementation.

On the RF-based category, the GPS is the main sensor. The GPS uses a global coverage coordinate system, which is very useful for pre-loaded flight applications, as can be seen in [21]. The project [21] shows a GPS application where a plan for a trip can be loaded with a starting point and a final point. This project also provides the possibility to plan the trip with waypoints, through which the aircraft must pass to reach its goal. The drawback

of this sensor is that it does not have a good accuracy, normally it provides data with an error of ± 3 meters. A strong point of the GPS is that its price is low, it can be purchased for only 60 USD.

Other RF-based method is shown in [16]. The project [16] makes use of a smartphone with Android OS and three WiFi access points. The location of each AP is static and known. The distance between the cell phone and one access point is calculated from the signal strength. The use of triangulation with the calculated distances makes it possible to obtain the position of the smartphone. This method presents important advantages, the first one is that the accuracy that it achieves is high (2mm) and the second one is that its cost is low. It presents a limitation and it is that as the required coverage area increases, the number of required APs increases exponentially.

The project [16] is just one among many other projects that make use of smartphones [22], [24], [26]. In [22] a centralized system is presented using a server. In the study area, Bluetooth Beacons are positioned, each one has a defined position. Getting the RSS (Received Signal Strength) and with the help of the server, the smartphone is able to obtain its own position. It has advantages because of the low price. Difficulties are found in the accuracy due to the non-linearity of the RSS with respect to the distance to the emitter.

The project [27] shows the use of UHF RFID, extending its identification function to a location function. As has been shown in [16] and [22] the RSS is also used here for pose estimation. In this case the processing is done on an external device to the tracked object. The tracked object only has to carry a small UHF RFID tag. It has advantages because it releases the tracked object of processing loads and electric loads because the label is passive. It has a disadvantage with the cost of transmitters and antennas because it is high.

The RF category shows the use of different sensors, such as GPS and WiFi. The GPS provides good range and price, but the accuracy of 3m may be insufficient for this project. The WiFi stands out in price and accuracy, but the range can be a limiting characteristic for its use.

In the Dead reckoning-based category, the IMU is the main sensor. In [24] the dead reckoning method is used with the IMU of a smartphone. It makes use of the initial position, the IMU data and a map preloaded, for the calculation of the actual position. The accelerometer of the IMU is used to calculate the acceleration vector. Additionally, the IMU's magnetometer is used to calculate the orientation. With the acceleration and the orientation, the change of position is calculated. This project has advantages in accuracy because of the use of the particle-filter [25]. Its implementation was made in 2D movements so it is important for the present project.

The dead reckoning category shows a sensor that stands out from all points of comparison. The limitations to its use are the noise because of the motors and the fact that its error is cumulative over time.

One way to face the limitations of using one sensor is to increase the information received. More information can be obtained by adding one or more sensors to the system. The projects that use two or more sensors will be analyzed into the Data fusion-based category.

The GPS by itself provides worldwide coverage, but a special configuration of this sensor combined with IMU data is presented in [7]. It uses an RTK GPS which needs a base station and a synchronization time (30 minutes) to work properly. An accuracy of up to ± 3 mm and an operating range of 300 meters is reached. It is achieved the coordination of 100 drones simultaneously. The drawback of the RTK GPS is that it has the highest price, approximately 12300 USD.

The project [23] shows the combination of dead reckoning and RF-based categories. The project is the best example of data fusion because it takes the best of each method to achieve a synergy that ends in a better accuracy of the data. The inertial method (IMU) has long-term errors due to the integration operations that have

to be performed. The RF (RSS) method presents low accuracy in the calculated position. [23] has advantages because it corrects IMU long-term errors with the RF measurements.

The project [26] shows the use of all sensors of a smartphone. It uses the IMU, WiFi module, barometer, GPS, gyroscope and light sensor. In this case a previously took dataset is used to compare it with the readings in real time and thus obtain the current position. The project advantages and disadvantages are conditioned by the size and accuracy of the data set. With a dataset taken with a single smartphone an error of 5.76m was obtained in a rectangular area with 100m of side.

In the project [28] the range of the GPS is combined with the camera's indoor use capability. They also take advantage of having a group of vehicles with GPS-challenging environments. A leader-follower topology is used, where the leader has sensors designed for challenging environments. Followers get the leader's position through the camera. The followers have low-cost sensors and correct their trajectory with the acquired leader's position. The project has economic advantages because the followers are vehicles from low cost. It presents a weak point because of its centralized system. If the leader fails in his objective, the followers will also fail.

The data fusion category shows that the combination of sensors results in better system performance. The use of GPS is an excellent option because it adds global coverage to the positioning of the vehicle. The use of the GPS must be combined with another sensor that provide indoor capabilities and better accuracy in short distances.

2.2. THEORETICAL FRAMEWORK

The theoretical framework's subsections describe some concepts that will be useful for understanding the components and references on this project. It is described the term UAV, its origin, its evolution over time and some forms of its classification. It is shown the applicability of LiDAR that was already shown in the "Laser-based" category of the State of the art. To achieve the LiDAR understanding it is described its operation and the type of data that it returns. There is also shown the applicability of GPS and RTK GPS that was also observed in the State of the art, its operation is also described, and its generalized form called GNSS. It is also shown the RUP methodology, its objectives, its components and the expected results of its use, in order to describe the guidelines that will delimit the development of this project.

2.2.1. Unmanned Aerial Vehicle

UAV, defined as unmanned aerial vehicle, is an aircraft that does not have human crew and can be autonomous or remotely operated [29]. The UAV was conceived for military use and it has had commercial applications: "This technology was initially conceived as an aeronautical robot, able to fly autonomously and fulfill a mission with defined objectives" (Pinilla, W. 2015) [30].

In military aviation, the UAVs was initially used for aerial recognition and later as electronic blocking systems. Historically, they were used in the War of the Persian Gulf (1990) and in the War of Bosnia (1992). Afterwards UCAV (Unmanned Combat Aerial Vehicle) were used, equipped with air weaponry for air combat and bombing. The UCAV were also used in the wars of Afghanistan (2001) and Iraq (2003) [30].

UAV is not only the aerial vehicle, but also the sensors, as well as the control systems, and the equipment that collects information and the communication systems. The main characteristics of an UAV are: autonomy, payload, dimensions, deployment capacity and type of mission [31].

2.2.2. UAV Classification

In the Table 2.2 is shown the classification of the UAVs depending on their physical configuration. In the second column is shown an example of every type and in the columns three and four the Pros and Cons are shown [32]:

Table 2.2. Classification of the UAVs depending on their physical configuration.

Type of UAV	Example	Pros	Cons
Multi-Rotor	 Quadcopter	Accessibility Ease of use VTOL* and hover flight Can operate in a confined area	Short flight times Small payload capacity
Fixed-Wing	 Plane	Long endurance Large area coverage Fast flight speed	Launch and recovery needs a lot of space no VTOL*/hover Harder to fly, more training needed Expensive
Single-Rotor	 Helicopter	VTOL* and hover flight Long endurance (with gas power) Heavier payload capability	More dangerous Harder to fly, more training needed Expensive

***VTOL: Vertical Take-Off and Landing. [33]**

The UAVs depending on their mission are usually classified into 6 types [34]:

- Target: Designed to simulate aircraft or enemy attacks in the systems of defense on ground.
- Recognition: Designed to collect and send military information.
- Combat (UCAV): They are used to fight and to accomplish dangerous missions.
- Logistics: Designed to carry payload.
- Investigation and Development: Designed to test systems under development.
- Commercial and Civil: Designed for civil purpose.

2.2.3. LiDAR

LIDAR, which stands for Light Detection and Ranging, is a remote sensing method that uses light in the form of a pulsed laser to measure ranges (variable distances) to the target. These light pulses generate precise, three-dimensional information about the shape of the target and its surface characteristics. LIDAR systems allow

scientists and mapping professionals to examine both natural and manmade surfaces with accuracy, precision, and flexibility [35].

The principle behind LiDAR is starts with a small light shoot at a surface, and then measure the time it takes to return to its source. Light travels very fast - about 300,000 kilometers per second. Only with the advancements in modern computing technology LiDAR become possible.

The actual calculation for measuring how far a returning light photon has travelled to and from an object is as follows:

The LiDAR instrument fires rapid pulses of laser light at a surface, some at up to 150,000 pulses per second. A sensor on the instrument measures the amount of time it takes for each pulse to bounce back. Light moves at a constant and known speed so the LiDAR instrument can calculate the distance between itself and the target with high accuracy. By repeating this in quick succession the instrument builds up a complex 'map' of the surface it is measuring [36].

LIDAR data is known as point cloud. Point clouds are large collections of points in 3D. Point clouds are typically used to measure physical world surfaces. They have applications in robot navigation and perception, depth estimation, stereo vision, visual registration, and in advanced driver assistance systems (ADAS) [37].

2.2.4. GPS/GNSS

GNSS stands for Global Navigation Satellite System, and it is the standard generic term for satellite navigation systems that provide autonomous geo-spatial positioning with global coverage. This term includes e.g. the GPS, GLONASS, Galileo, Beidou and other regional systems. GNSS is a term used worldwide The advantage to having access to multiple satellites is accuracy, redundancy and availability at all times. Though satellite systems don't often fail, if one fails GNSS receivers can pick up signals from other systems. Also if line of sight is obstructed, having access to multiple satellites is also a benefit. Common GNSS Systems are GPS, GLONASS, Galileo, Beidou and other regional systems [38].

The United States' Global Positioning System (GPS) consists of up to 32 medium Earth orbit satellites in six different orbital planes, with the exact number of satellites varying as older satellites are retired and replaced. Operational since 1978 and globally available since 1994, GPS is currently the world's most utilized satellite navigation system [38].

"Real time" independently of other meanings is the obtaining information at the moment of occurrence of the phenomenon or process, i.e. at the same instant. In the case of GNSS measurements it means obtaining the desired quality at the same time of the measurement. The instrument must be capable of receiving information additional to that transmitted by the GNSS satellites. According to the above, there are many options to perform GNSS measurements in Real Time, the most important are:

- Augmentation systems, like WAAS or EGNOS.
- Radio systems (VHF and UHF bands), like RTK.
- Subscription systems, like Omnistar, CNAV.
- Internet and cellphone network, like Ntrip.

2.2.5. RTK

RTK is a type of GNSS that is useful for applications that require high accuracies. RTK is a technique that provides positions that are orders of magnitude more precise than, for example, simple GPS system. RTK techniques are complicated. The basic concept is to reduce and remove errors common to a base station and rover pair, as illustrated in Figure 2.1. At a very basic conceptual level, there is a very precise system in the base station that computes the position with low error and then it sends accuracy information to the rover (simpler system) to decrease its errors [39].

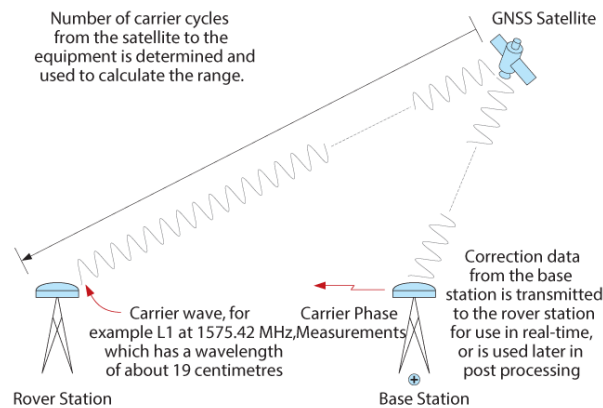


Figure 42 Real-Time Kinematic

Figure 2.1. Basic explanation of GNSS RTK operation [38].

Some of the applications for GNSS RTK systems are listed below:

- Geodetic Control
- Mining
- Navigation
- Bathymetry
- GIS data collection and mapping
- Civil works

2.2.6. RUP

RUP, defined as Rational Unified Process, is a methodology for software development that also can be extended to hardware development. RUP is a product developed by Rational Software, a company of IBM [40].

The main documents in RUP are the real use cases. Real Use Cases are a technique of capture of requirements that forces to think in terms of importance for the user, not only in terms of functions. A Use Case is defined as a piece of system functionality that provides the user with added value. Use Cases represent the functional requirements of the system. In RUP, Use Cases are not just a tool for specifying system requirements. They also guide their design, implementation and testing. Use Cases are an integrating element and a guide to work [40].

The architecture of a system is the organization or structure of its most relevant parts. The architecture allows a common vision among all involved (developers and users), and a clear perspective of the complete system, necessary to control the process. In the case of RUP, in addition to using the Use Cases to guide the process, special attention is given to the early establishment of a good architecture. RUP is not strongly impacted by subsequent changes during construction and maintenance [40].

The correct balance between the Use Cases and the architecture is very similar to the balance of form and function in product development. For achieve the balance, the strategy proposed in RUP is to have an iterative and incremental process, where the work is divided into smaller parts or mini projects. The balance can be achieved during each mini project, and throughout the development process. Each mini project can be viewed as an iteration with which you get an increase that produces a growth in the product [40].

The process to be performed is illustrated in RUP with a two-dimensional chart, which can be seen in figure 2.2.

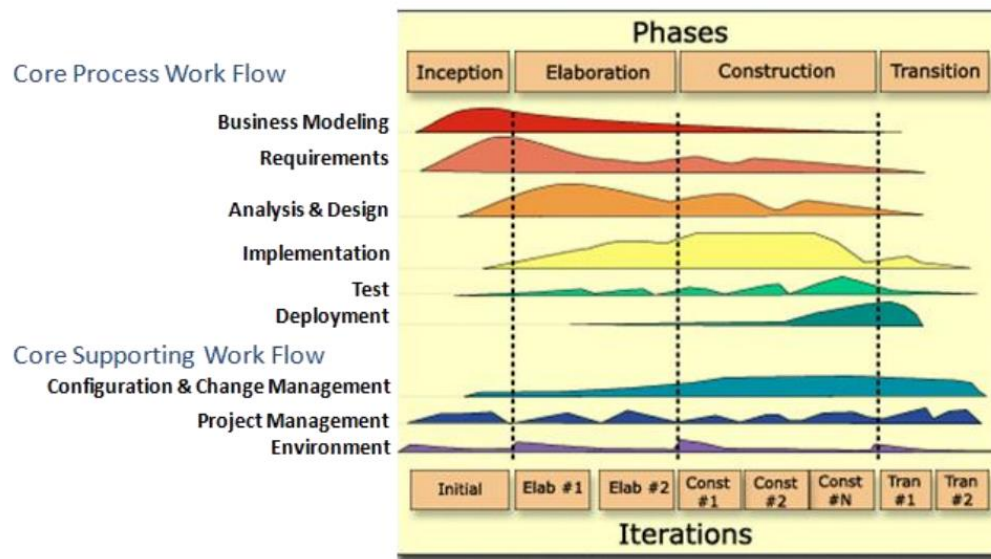


Figure 2.2 Chart for description of RUP development process [41].

As can be seen in Figure 2.2 the diagram flows both on the horizontal axis and on the vertical axis, each axe is described individually below [41].

Horizontal axis: Represents time, it is considered the axis of the dynamic aspects of the process. Indicates the life cycle characteristics of the process expressed in terms of phases, iterations and goals. RUP consists of four phases: Start, Elaboration, Construction and Transition. As mentioned above each phase is subdivided into iterations at the same time.

Vertical axis: Represents the static aspects of the process. Describes the process in terms of process components, disciplines, workflows, activities, and roles.

2.3. FINAL OBSERVATIONS

The state-of-the-art report shows that each sensor offers advantages and disadvantages in the different evaluation criteria. Therefore, it is necessary to choose the sensor that provides more advantages, according to the requirements of a specific project.

This project requires great accuracy to locate the nearby UAVs and a wide range to provide enough coverage area for the set of UAVs. The following sensors did not fulfill the first requirement: Camera, WiFi, RFID and Bluetooth. Another important point to take into account in the sensor selection is that the cost-benefit ratio should be adequate. The sensor GPS RTK is rejected because it is very expensive. The microphone is rejected because it does not give the distance measurement. Expecting to use the system under different environment conditions the GPS sensor is rejected. The IMU sensor is rejected because it is required for each UAV to be capable of identify the other UAVs position without communications. Considering the requirements of this project, it is decided to use a laser sensor.

The laser sensors have three big categories depending on the dimensions of the data retrieved: 1D, 2D and 3D. Because of the complexity of the identification of a UAV the use of 1D laser is discarded. Due to the amount of payload that a UAV can carry, the 3D laser is also discarded. Additionally, the use of 3D laser implies a high economic investment.

Because of the exposed reasons, it is decided to use a 2D laser. Among the different models available on the market it is highlighted the LiDAR Sweep360 sensor from the manufacturer Scanse. It is light (120 g), low power (2.25 W) and has a simple serial communication. The size of the sensor is also an advantage because with 6x6x6 cm of volume it does not require a big UAV. The price of the Sweep 360 in the market is around \$340.00 USD so its price is also an advantage.

The two main characteristics of the sensor are the range (40 m) and the accuracy (maximum error of 2 mm). These characteristics besides providing information about the UAVs nearby, provides important information regarding the environment that surrounds it.

As described in the theoretical framework, RUP is a software development methodology extensible to hardware development. In this project, the RUP methodology was used for the entire design process. The design process is shown in the next chapter, chapter 3.

Chapter 3

DESIGN AND IMPLEMENTATION OF GALS

3.1. Design of GALS
3.1.1. Functional and non-functional Requirements
3.1.2. Conceptual Diagram
3.1.3. Class diagram of GALS
3.1.4. Flowchart of GALS
3.2. Implementation of GALS
3.2.1. The sensor: Laser Range Finder
3.2.2. The Identification of UAV problem using a laser range finder
3.2.3. Identification Algorithm
3.2.4. Calculation of the relative position
3.2.5. Communication Module with the Flight Control App
3.3. Storage module (android)
3.4. Base station
3.5. Final observations

This chapter shows the results of the design and implementation stage of GALS. Section 3.1 shows the developed RUP documentation. The documents developed are: Functional requirements; Conceptual diagram; Class diagram; Flowchart. The RUP documentation is used as a guideline for the implementation of GALS. The implementation is shown in section 3.2. The sensor used is described, the calibration process performed is also shown. Then, the detection process is described. At the end of the implementation section, the communication protocols used between the 3 main blocks of GALS are described. At the end of this chapter, final observations are presented. The results and conclusions regarding the design and implementation of GALS are described.

This chapter shows the relevant results obtained with RUP. More detailed documents are shown in the folder **Annex1_RUPDocs**. That folder contains the documents shown in this chapter along with the real use cases and the sequence diagrams.

3.1. Design of GALS

This section shows the results of the design stage of GALS. The design is based on the RUP methodology. The RUP methodology was described in chapter 2. The following sub-sections show, respectively: the requirements for the prototype (3.1.1). The general conceptual diagram that shows every module and the interaction between them (3.1.2). The class diagrams showing the software structure of each module (3.1.3). The flowcharts that show the timeline of events and the communication protocols between modules (3.1.4).

3.1.1. Functional and non-functional Requirements

This section shows the requirements that will be used as a starting point for the implementation of GALS. The functional requirements describe what GALS must do. The non-functional requirements do not influence the operation of GALS but they are necessary for its implementation.

The functional requirements for GALS are shown in table 3.1. In the column "Ref #", each requirement is listed. In the "Functions" column, each requirement is described. The "Category" column indicates whether the requirement is optional (O) or essential (E).

Table 3.1 Functional requirements for GALS.

Ref #	Functions	Category
1.1	Allow communication between Smartphone and Arduino	E
1.2	Allow acquisition of data from the laser sensor	E
1.3	Perform an UAV identification in the data received from the laser sensor	E
1.4	Perform a filtering of the data received from the laser sensor	E
1.5	Allow software configuration of the laser sensor	E
1.6	Allow transmission of the relative position of other drones and obstacles from the Arduino to the Smartphone	E
1.7	Integrate the results with the Flight Control Software that currently runs on the devices	E

The non-functional requirements for GALS are shown in table 3.2. In the column "Ref #", each requirement is listed. In the "Description" column, each requirement is specified. The "Category" column indicates whether the requirement is optional (O) or essential (E).

Table 3.2 Non-Functional requirements for GALS.

Ref #	Description	Category
1.0	Software	E
1.1	Android Studio version 2.3.3	
1.2	Arduino IDE version 1.8.4	
1.3	Sweep360 Library for Arduino	
1.4	Sweep360 firmware version 1.4	
1.5	Android OS version 4.4	
2.0	Sensor	E
2.1	Minimum accuracy: 1/2 UAV's radio	
2.2	Minimum range: 10 m	

3.1.2. Conceptual Diagram

This section shows the conceptual diagram designed for the implementation of GALS. The conceptual diagram is shown in figure 3.1. The three main blocks are: the LiDAR sensor, the Arduino device and the Android device. The sensor generates the environmental data. Then, the Arduino device performs the acquisition and processing of the data. Also, the Arduino also communicates with the Android device to exchange information and execute commands. The Android device executes the flight control system, which receives the information from the Arduino. The flight control system can also make changes to the system through commands sent to the Arduino.



Figure 3.1. Conceptual diagram of GALS.

3.1.3. Class diagram of GALS

GALS is a system that makes use of 3 subsystems. Figure 3.1 shows 2 of these subsystems: the Arduino and the Smartphone. The third subsystem is the base station, from which the user can interact with both the UAV and GALS. The base station is described in more detail in section 3.4. In section 3.1.3.1, the class diagram of the Arduino software is shown. In section 3.1.3.2, the class diagram of the Smartphone software is shown. In section 3.1.3.3, the class diagram of the software of the base station is shown. The interaction between the different existing classes is also shown in each section. The methods that were modified or created to allow the operation of GALS are shown in color red.

3.1.3.1. Class diagram for Arduino software

In the original UAV's design an Arduino MEGA was used. The main function of the Arduino MEGA was to receive the power values for the motors through the USB connection. After that, the power values were converted into PWM signals to be delivered to the motors. With the implementation of GALS, the Arduino MEGA was replaced with an Arduino DUE. The function of delivering the power signal to the motors was maintained but there are several changes in this software. Figure 3.2 shows the methods implemented to satisfy the new requirements described in Table 3.1.

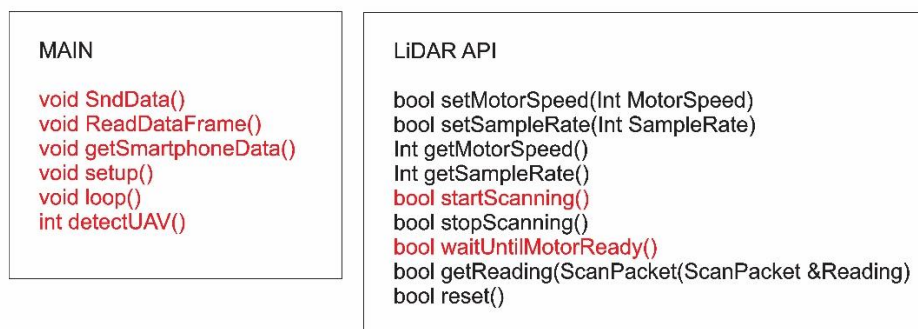


Figure 3.2. Class diagram for the Arduino.

Figure 3.2 shows the methods used from the LiDAR API. There are two methods that were modified. The modification was made because the original methods used blocking algorithms. The calls to the LiDAR API are made from the loop method of the Arduino's main class.

The methods used in the Arduino software are described below.

A. Main

1. **void SndData ()**: There are global arrays created to send and receive data over USB. In the Main thread the arrays are loaded with data. When the SndData() method is called, it collects the data, organizes it according to the communication protocol and sends it to the smartphone.
2. **void ReadDataFrame()**: The messages received from the Smartphone are loaded into global arrays. Once a message is received, the ReadDataFrame method is called. This method decodes the message. Then, for each type of message, the state of a flag (local variable) is changed. Each flag allows the execution of a specific action.
3. **void getSmartphoneData()**: This method is executed periodically every 1ms. Its function is to poll the Smartphone asking for messages to be transmitted.
4. **void Setup()**: This method is mandatory for Arduino software. This method is executed once before any other method. Within this method the system variables and communication interfaces are initialized.
5. **void loop()**: This method is also mandatory for Arduino software. This method is the main thread and runs cyclically during the entire execution time of the program. In this method a state machine is used. The data coming from the LiDAR is received. Then the data is processed. Finally, the data is sent to the Smartphone. Other tasks are executed in this method when the messages arrive from Smartphone.
6. **int detectUAV()**: This method is used for the detection of UAVs in the range data. The detection result is stored in global arrays. The number of detected UAVs is returned.

3.1.3.2. Class diagram for Smartphone software

Unlike the Arduino's software in which the code was written almost from scratch, the Smartphone was based on an existing flight control software. Figure 3.3 shows the class diagram of the software that runs on the Smartphone. The classes whose purpose is the flight control were omitted.

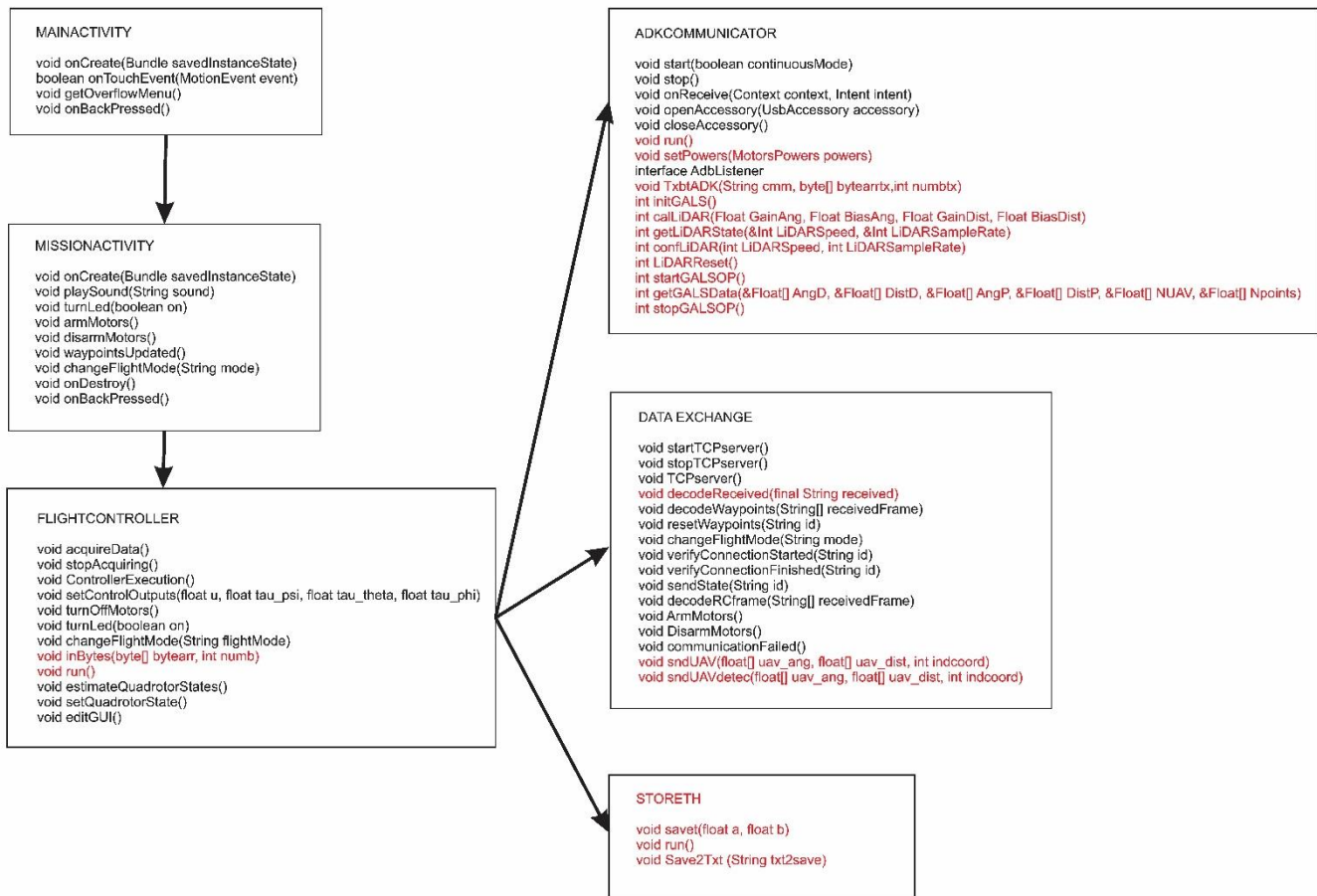


Figure 3.3. Class diagram for the Smartphone.

The methods used in the Smartphone software are described below.

A. MainActivity

This class is responsible for displaying the initial screen. Then it automatically calls the MissionActivity class.

B. MissionActivity

This class presents the main interface of the flight control software. It initializes the system to start the operation. Finally, it calls the FlightController class.

C. FlightController

1. ***void inBytes(byte[] bytearray, int numb):*** This method is used for the reception and decoding of the messages coming from the Arduino. It uses the AdkCommunicator class as interface for USB communication.

D. AdkCommunicator

1. ***void run():*** This method is used to receive messages through the USB port. The original method was modified to fit the requirements of GALS.
2. ***void setPowers(MotorsPowers powers):*** This method is in charge of send the power values for the motors. The original method was modified to fit with the requirements of GALS.
3. ***void TxbtADK(String cmm, byte[] bytearraytx, int numbt):*** This method is used to send data frames to the Arduino.

4. ***int initGALS()***: It is explained in table 3.5.
5. ***int calLiDAR(Float GainAng, Float BiasAng, Float GainDist, Float BiasDist)***: It is explained in table 3.5.
6. ***int getLiDARState(&Int LiDARSpeed, &Int LiDARSampleRate)***: It is explained in table 3.5.
7. ***int confLiDAR(int LiDARSpeed, int LiDARSampleRate)***: It is explained in table 3.5.
8. ***int LiDARReset()***: It is explained in table 3.5.
9. ***int startGALSOP()***: It is explained in table 3.5.
10. ***int getGALSData(&Float[] AngD, &Float[] DistD, &Float[] AngP, &Float[] DistP, &Float[] NUAV, &Float[] Npoints)***: It is explained in table 3.5.
11. ***int stopGALSOP()***: It is explained in table 3.5.

3.1.3.3. Class diagram for Base Station software

The base station was modified to fit with GALS requirements. A new tab dedicated to GALS was added. Some of his classes were modified, it can be observed in figure 3.4.

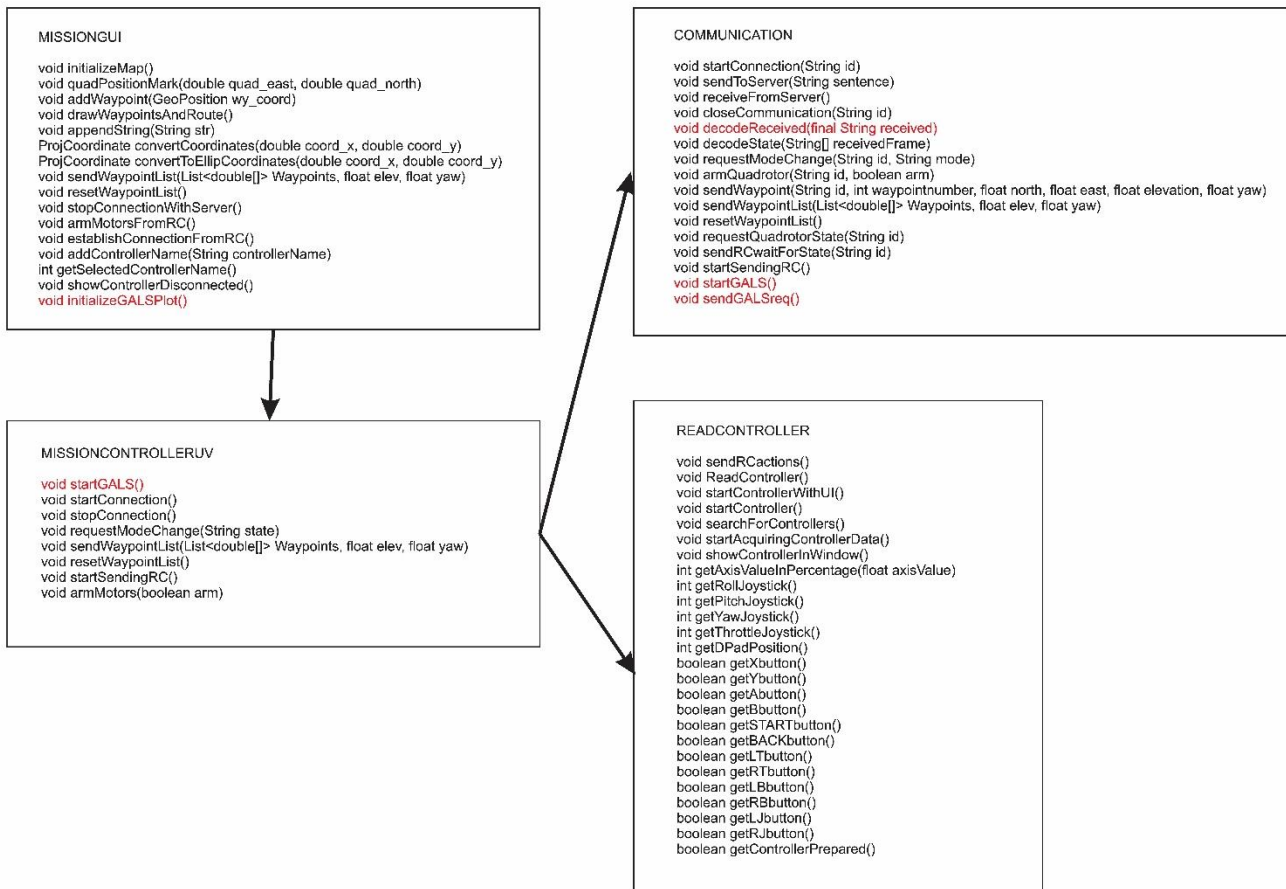


Figure 3.4. Class diagram for the Base Station

The methods used in the base station are described below:

A. MissionGUI

1. ***void InitializeGALSPlot()***: This method is used to generate the plot window in the GALS tab. Once the graph is initialized it is possible to plot from any other class of the application.

B. MissionControllerUV

1. ***void startGALS()***: This method is used to start the operation of GALS in the UAV. GALS execution starts on both the Smartphone and the Arduino.

C. Communication

1. ***void decodeReceived(final String received)***: This method is used to receive and decode messages from the Arduino. The method was modified to allow the reception of frames belonging to GALS.
2. ***void startGALS()***: Initiates the execution of periodic tasks that are necessary for the operation of GALS.
3. ***void sendGALSreq()***: This method is used to poll the Smartphone for frames belonging to GALS.

D. ReadController

This class is responsible for receiving and decoding the commands coming from the external USB control.

3.1.4. Flowchart of GALS

The flowchart of GALS shows the step by step of the program's execution in each module. Two modules are shown: The Arduino and the Smartphone. Each step shows the name of the method to be used.

The flowchart of the Arduino software is shown in Figure 3.5. The execution begins with a handshake with the Smartphone software. Then, the initialization of the LiDAR and the configuration checking are performed. The messages coming from the Smartphone software are then received. The messages are answered and the required configurations are made. The acquisition and processing of the range data is done, where the results are: the position of the detected UAVs, and a point cloud with the possible obstacles. At this point the execution can be stopped by command from the Smartphone software, or continue sending the information to the Smartphone and moving on to the next iteration.

Arduino

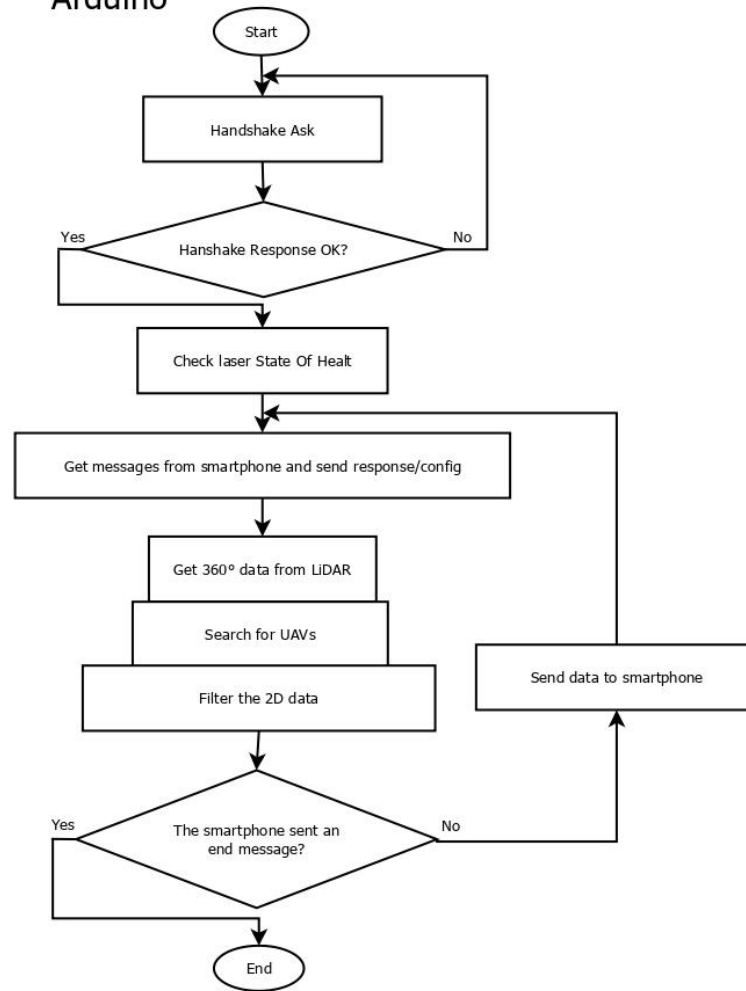


Figure 3.5 Flowchart of the Arduino's program execution.

The flowchart of the Smartphone software is shown in Figure 3.6. The execution begins with a handshake with the Arduino. Then, the Arduino software is asked for the current state of the sensor. The Smartphone software waits until the sensor is operative. After that, the configuration and the calibration variables are sent to the Arduino. Afterwards, information about UAVs and obstacles is received from the Arduino software if they are available. This information is sent to the flight control system. At this point, the operation of the system can be ended by command or continue to the next iteration.

Smartphone

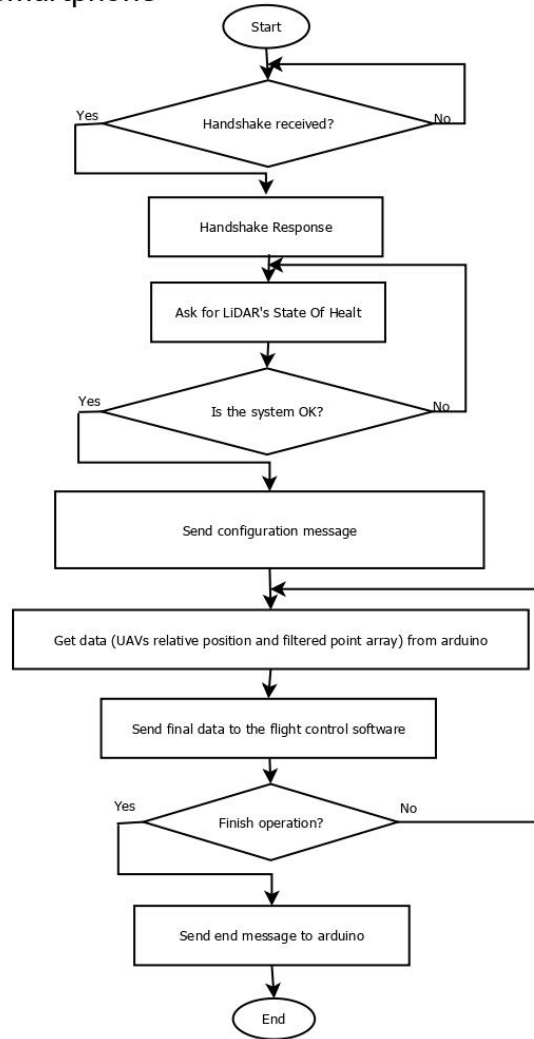


Figure 3.6 Flowchart of the Smartphone's program execution.

3.2. Implementation of GALS

In this section the implementation of GALS is shown. This section is divided into 3 sub-sections, each sub-section is related with one block of the system, figure 3.1. In sub-section 3.2.1, are described the selected sensor (First block, up to down.) and the calibration process. Sub-Section 3.2.2 shows the acquisition and processing of the data, both operations are executed by the Arduino device (Second block). Sub-Section 3.2.3 shows the communication structure used to transmit information between Android and Arduino devices.

3.2.1. The sensor: Laser Range Finder

The first block in the GALS design is the sensor (Figure 3.2). It is an essential block because the information generated is the raw material of GALS. Due to its importance, this section is dedicated to the presentation of the sensor (section 3.2.1.1) and the calibration process (section 3.2.1.2). The presentation of the sensor shows its most important characteristics and some of its specifications. The calibration process was carried out by contrast of tape measurements with the measurements delivered by the sensor.

3.2.1.1. Sweep V1 360

The Sweep V1 360 (Figure 3.7) is a LiDAR type sensor from the manufacturer Scans. The Sweep generates information in 2D in 360 degrees. This sensor has characteristics that fit with the requirements of this project. The sensor's most relevant characteristics are: accurate, wide range, low weight, low energy consumption and cheap. The sensor specifications are shown below:

Range: 40 meters
Sample rate: 1000samples/sec
Field of view: 360 degrees
Rotation frequency: 2-10 Hertz (Adj.)
UART TTL: 3.3V - 5V tolerance
Connection: USB
Source: 5V DC
Consumption: 400mA



Figure 3.7. Sweep V1 360, LiDAR from Scans.

3.2.1.2. Sensor calibration

In order to get readings accurately, it was carried out the calibration of the LiDAR sensors. The relationship between the data has linear behavior, so the least squares method was used to find the calibration coefficients [43]. To perform the calibration, measurements were taken at known distances. The relationship between measured distance and known distance were used to estimate measures of greater accuracy. The calibration structure can be seen in the figure 3.8. The components of the calibration structure were a mobile table (on which the sensor is located) and a wall. The wall was made of bricks and it had a white coating. The distance between the table and the wall were adjusted with the help of a tape measure. It was used 9 positions between 1 and 5 meters to take 190 distance readings in each position. With the collected data, the calibration was done following the method described below.

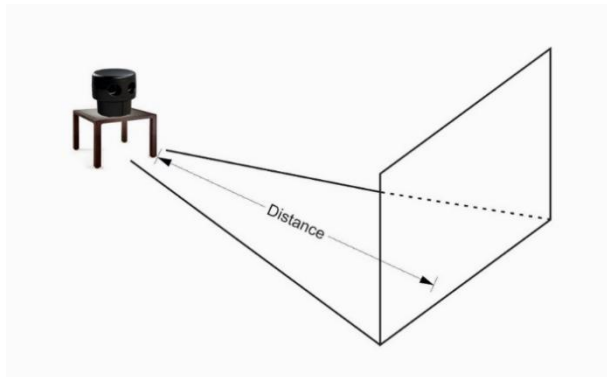


Figure 3.8. Structure for the calibration of the LiDAR sensor.

The relationship between the real distance and the estimated distance is described by equation 3.1. $\hat{y}$ represents the estimated distance of the real distance y . μ represents the average of the sensor data. k and b represent constants that minimize the mean square error that is described by equation 3.2. The calibration process is then reduced to:

1. To find the constant k with the operation described in (3.3).
2. To find the constant b with the operation described in (3.4).
3. To find the calibrated value of each reading, individually, using the equation (3.1).

$$\hat{y} = k \cdot \mu + b \quad (3.1)$$

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - k\mu_i - b)^2 \quad (3.2)$$

$$k = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)(\mu_i - \bar{\mu})}{\sum_{i=1}^n (\mu_i - \bar{\mu})^2} \quad (3.3)$$

$$b = \hat{y} - k\bar{\mu} \quad (3.4)$$

Table 3.3 shows the results obtained by taking measurements at known distances with each sensor. In the column "Real distance" the real distance at which the sensors were placed is shown. In the following columns, the heading "LiDAR S/N" shows the average of 190 readings for each position.

Table 3.3 Average range readings for calibration

Real distance [cm]	LiDAR 9488	LiDAR 9487	LiDAR 9489	LiDAR 9490
100	103.38	114.15	102.42	106.02
150	146.23	161.09	149.69	152.43
200	203.14	212.78	201.33	200.08
250	253.35	264.22	255.43	253.80
300	302.69	312.34	304.96	303.91
350	351.40	363.02	354.30	352.73
400	397.22	411.55	403.76	403.66
450	450.94	462.45	454.95	450.95
500	499.31	513.85	502.50	502.14

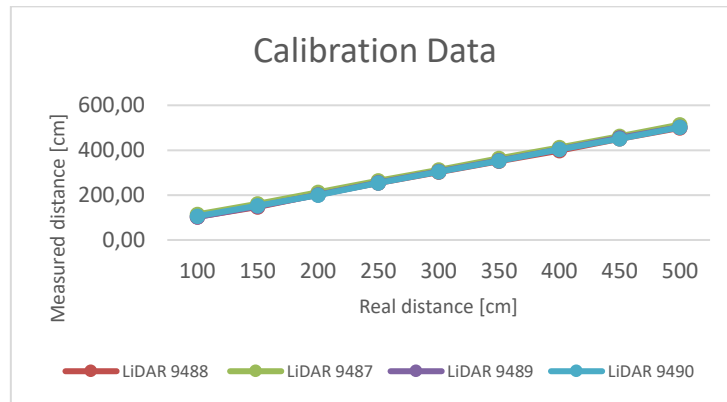


Figure 3.9. Comparison of sensor readings against ideal behavior.

In figure 3.9, it is shown a chart with the Table I's data. The purpose of this chart is to compare the behavior of each sensor against the ideal behavior. A closer view of the farthest measured point (450 to 500cm) is shown in Figure 3.10. The approach is made to observe in more detail the behavior differences between sensors.

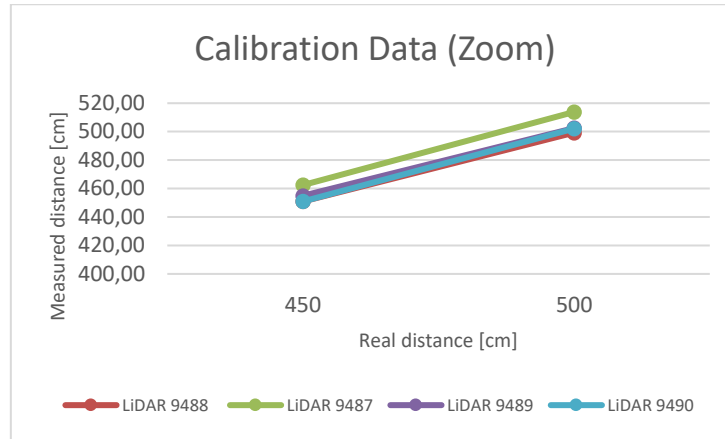


Figure 3.10. Comparison of sensor readings against ideal behavior (Zoom).

Due to the linear behavior, it is possible to find the characteristic line of each sensor. With the results of the linear regression it is possible to propose equations to obtain more accurate data using the data delivered by each sensor. The calibration process describes a function whose input is the value delivered by the sensor and its output is a value of greater accuracy. The calibration equation for each sensor are shown in equations 3.5 to 3.8.

$$f_{9488}(r) = 1.0017 \cdot r - 1.0857 \quad (3.5)$$

$$f_{9487}(r) = 1.003 \cdot r - 12.9919 \quad (3.6)$$

$$f_{9489}(r) = 0.9835 \cdot r + 0.7509 \quad (3.7)$$

$$f_{9490}(r) = 1.0047 \cdot r - 4.2246 \quad (3.8)$$

3.2.2. The Identification of UAV problem using a laser range finder

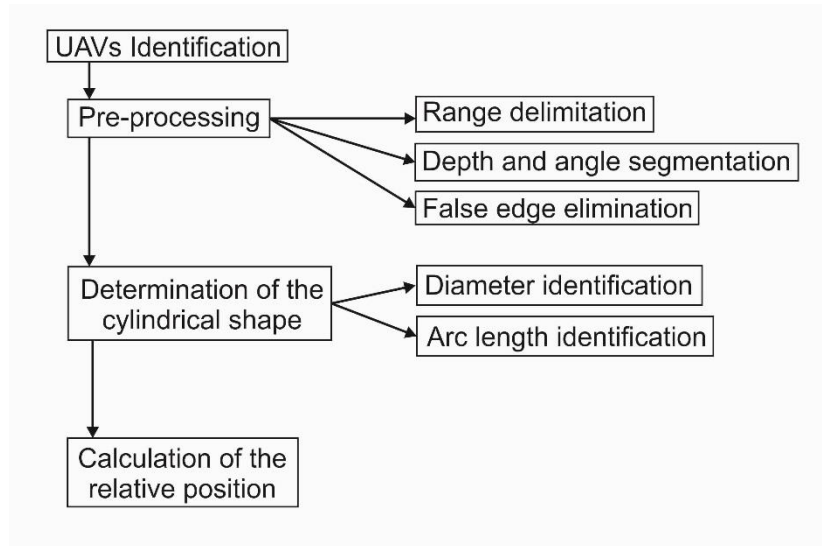


Figure 3.11. Conceptual diagram for the identification of UAVs process.

This subsection describes the detection process of UAVs from 2D data. As can be seen in figure 3.11 the identification of UAVs has three main parts:

1. The pre-processing of the 2D data.
2. The determination of the cylindrical shape.
3. Calculation of the relative position between the sensor and the cylinder.

The identification process is detailed in the sub-section 3.2.3. The relative position calculation is detailed in the sub-section 3.2.4.

The identification process of an UAV from a two-dimensional slice is difficult. In order to simplify identification tasks, a basic geometric shape is chosen: the cylinder. Then, a cylinder of known diameter is located at the top of each UAV to facilitate its identification. As can be seen in Figure 3.12 a cylindrical shape is easily identifiable in 2D data. In Figure 3.12, the sensor is located at the center of the coordinate frame. The place of the picture is a square-shaped room. The plot is generated from the laser sensor data. The 2D data visualization showed in Figure 3.12 were achieved using The MATLAB software 2013a version.

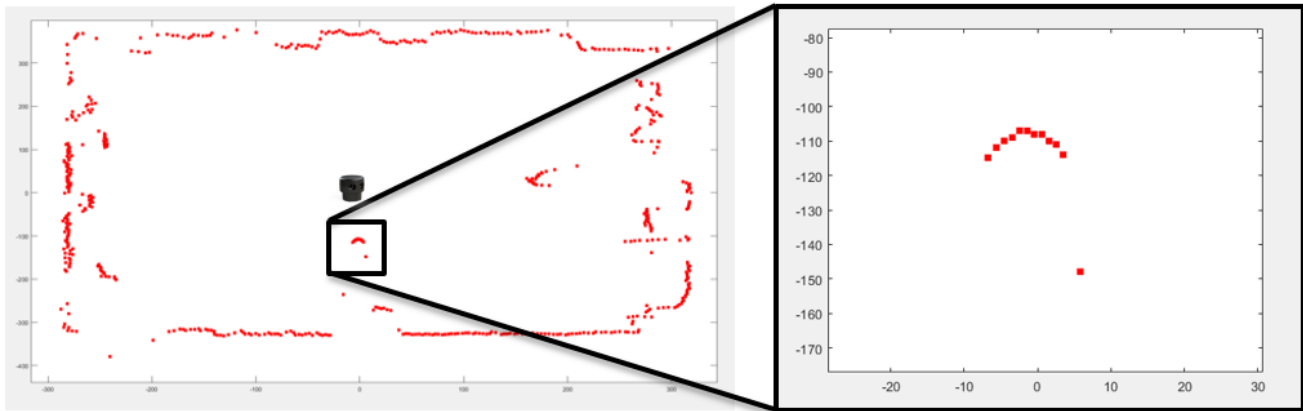


Figure 3.12. 2D image captured with the Sweep LiDAR 360. a) The sensor is located in position (0,0). b) Enlarged view of a captured cylinder.

The cylinder used in Figure 3.12 is shown in Figure 3.13. It is a cylinder with 11cm diameter located on a tripod. Different materials were tested, some of them were plastic, paper and wood. The cylinder used is made of wood, because this material generated the best results with respect to the variance in the laser readings.



Figure 3.13. Cylinder used to perform identification tests with LiDAR.

3.2.3. Identification Algorithm

The identification process can be achieved in 6 steps. The identification algorithm resolves the “Pre-processing” and the “Determination of the cylindrical shape” items showed in figure 3.10. The process is shown step by step using real data from the laser sensor. The test data used is shown in Figure 3.14:

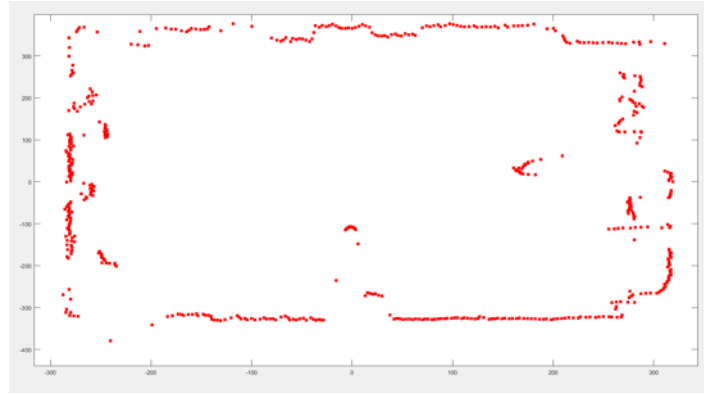


Figure 3.14. Test data for the identification process.

- a) **Range delimitation:** Initial filtering is made to the data. Only the points that are in the interest range are considered for identification. Even when the laser sensor has a maximum range of 40 meters, the objective of the GALS is to calculate the relative position between a set of UAVs at near distance. The range of interest ranges from 1 meter to 5 meters starting at the sensor position. As can be observed in Figure 3.15 the amount of points is significantly reduced. The range delimitation is done right after the data corresponding to the 360 degrees arrive. With the range delimitation the out-of-range points are also eliminated. The range of interest limits are user configurable.

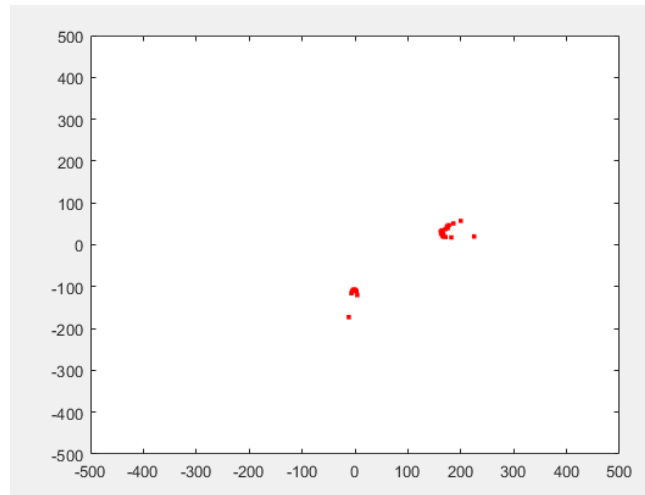


Figure 3.15. Resulting points after the range delimitation step.

- b) **Depth and angle segmentation:** The objective of the segmentation process is to differentiate between objects present in the capture. The limits of an object are determined by thresholds in the measurements of angle and depth between one point and the next one. This can be observed in Figure 3.16. The calculated thresholds are 7cm in distance and 2.52° in angle.

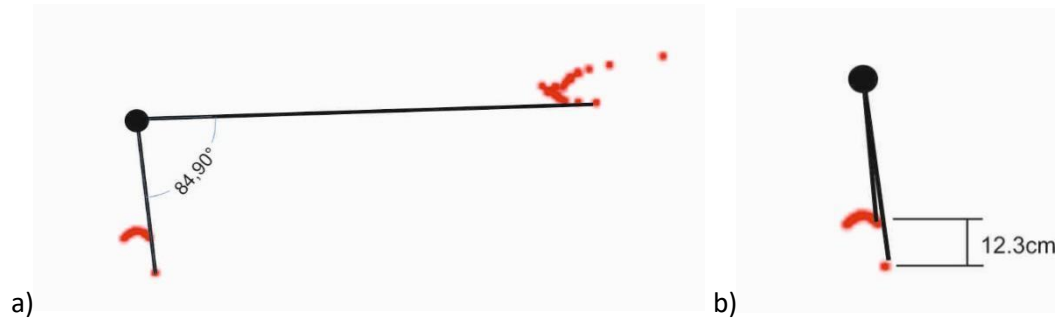


Figure 3.16.. a) Angle of separation between two consecutive points. b) Distance of separation between two consecutive points.

- c) **False edge effect elimination:** The false edge effect is a perception error that occurs at the edge of the objects. That effect appears because sometimes at the edge of the objects an abrupt change of depth is present. The result of the false edge effect is a false-reading, normally a point between two objects that does not match with any object in the reality.

After the segmentation process, two well-defined objects can be seen in figure 3.16. In figure 3.16b, it is shown a false edge effect point. That kind of points are easily identifiable because they remain as a single point object. The identified objects are stored in memory as vectors. Taking advantage of the vector form to represent data, the vector length is used as selection criteria. The false edge effect is just an intermediate point between two objects, its length is at most 2. Thus, vectors whose length is 2 or less, are eliminated. In this case a vector of length 1 is located at 12.3cm of the nearest segment.

Then, after the false edge effect elimination, only two objects remain in the process as shown in Figure 3.17.

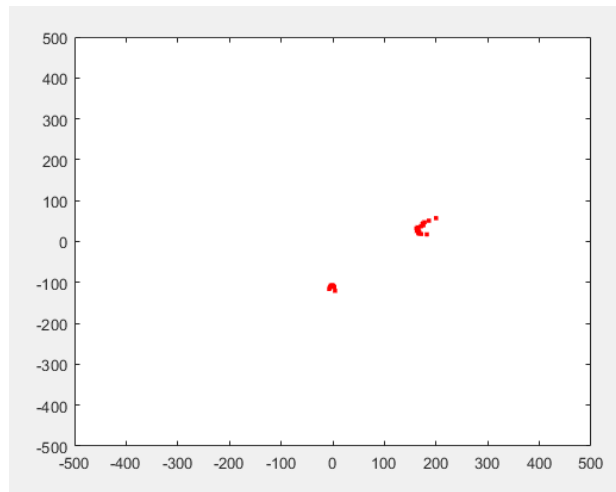


Figure 3.17. Result after the false edge elimination process.

- d) **Diameter identification:** The assumption that the objects found have a cylindrical shape is made. Their "diameter" is calculated. The objects that have a "diameter" different from the expected ($\approx 11\text{cm}$) are discarded. The calculated uncertainty of the diameter measurement is $\pm 3\text{cm}$. Due to the uncertainty in the measurement process, every object with diameter bigger than 7cm ($11\text{cm} - 3\text{cm}$) and lower than 13cm ($11\text{cm} + 3\text{cm}$) remain in the identification process. To calculate the diameter, the Euclidean distance between the starting point and the ending point of the object is used. The calculation is done using Cartesian coordinates. The distance is calculated using equation 3.9, where the coordinates of the

starting point are $(x1, y1)$ and for the ending point are $(x2, y2)$. For the two objects in the test data, the diameter measurement was 10.29cm for the object in the lower part and 29.68cm for the object in the right part.

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \quad (3.9)$$

Now it is clear that only one object remains in the test data. The results can be observed in figure 3.18.

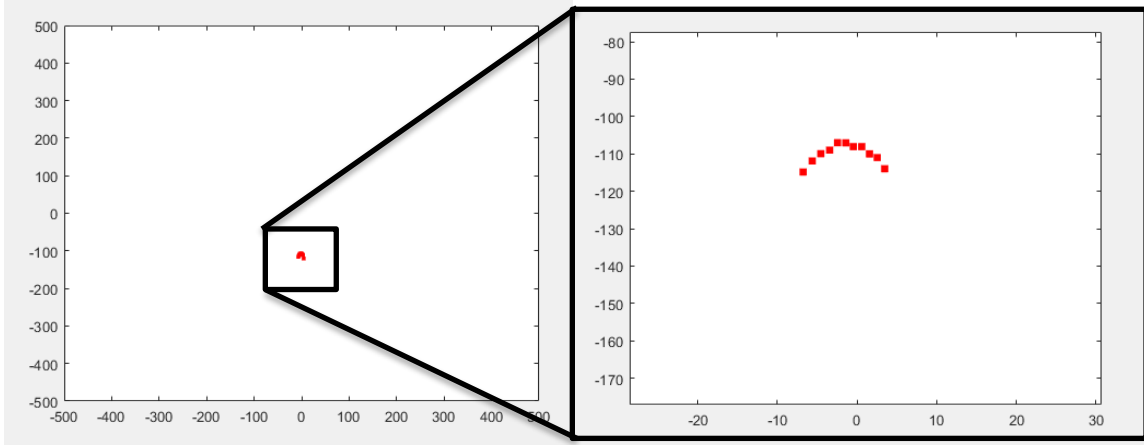


Figure 3.18. Remaining objects in the test data after the diameter identification.

- e) **Arc length identification:** A shape identification is made in the remaining objects. The arc length of the cylindrical shapes is calculated. The calculation is done using the angle created between the starting point and the ending point of the object with respect to the sensor. Using the angle and the distance at which the object is located (polar coordinates of the points at the edges of the object were used) the arc length is calculated. For the calculation of the arc length, the equation (3.10) was used [44]. Where (A_i, R_i) are the polar coordinates (Angle, Distance) of the starting point and (A_f, R_f) are the coordinates of the ending point. The procedure can be observed graphically in Figure 3.19.

$$ArcLength = (A_i - A_f) * \left(\frac{R_i + R_f}{2}\right) \quad (3.10)$$

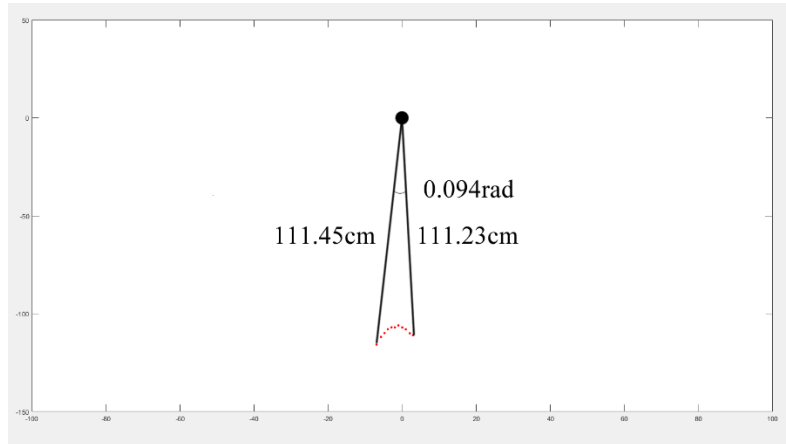


Figure 3.19. Arc length measurement of a cylindrical object.

As can be seen in figure 3.19 the arc length measurement was done from the sensor perspective, so its value must be slightly bigger than the diameter measurement. The distance at which the initial point

(R_i) and the end point (R_f) of the object are located should be the same. Therefore, if the subtraction ($R_i - R_f$) is performed, the result should be zero. In the reality, this value is close to zero, and therefore it is possible to use it as selection criteria. Regardless of the angle in which a cylinder is observed, it will have its limits at the same distance from the observer. This feature is used to allow the elimination of non-cylindrical objects. An example of these cases is shown in figure 3.20 a) and b). In order to consider cylinders in the entire range of interest (1 to 5 meters) the arc length is used. Then, those objects where the operation ($R_i - R_f$) results in less than one third of the arc length ($\frac{1}{3} \cdot ArcLength$) are conserved, and those that do not comply with the condition are eliminated.

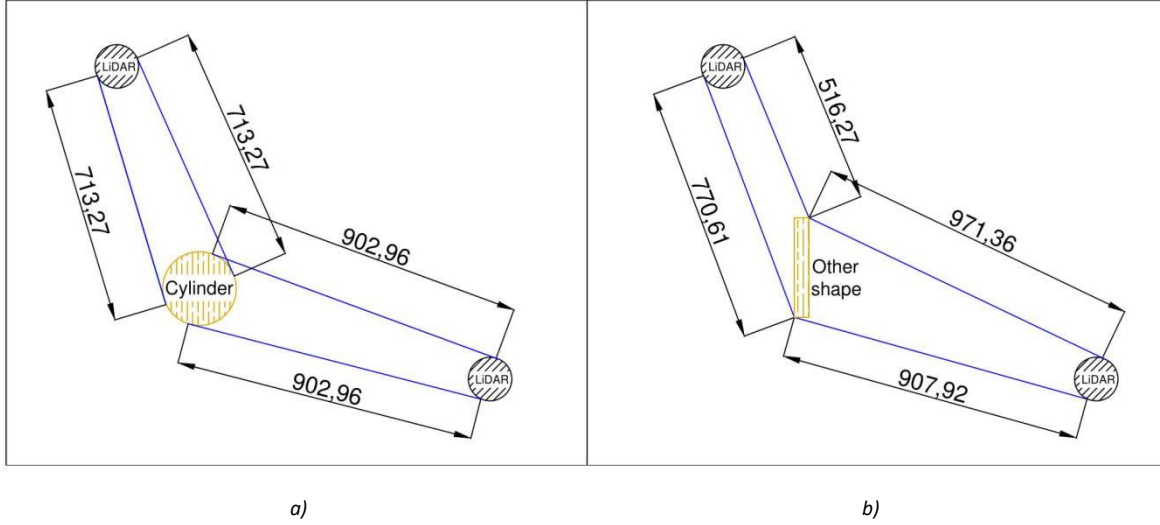


Figure 3.20. a) Left: A cylinder observed from two different positions. b) Right: A rectangle observed from two different positions.

The case shown in figure 3.19 has a cylinder, which the calculated arc length was 10.46cm. The operation ($R_i - R_f$) results in 0.22 cm. Then it is clear that $0.22 < 3.48$ and therefore the object was identified as a cylinder of approximately 11cm diameter. The value 3.48 is the result of the operation ($\frac{1}{3} \cdot 10.46$).

At the end of the process, the points of the cylindrical objects of known diameter were stored in memory. At this point, every object remaining in memory was considered as an UAV. The “Determination of the cylindrical shape” process has ended, the remaining process is the calculation of the relative position between the laser sensor and the cylinders (UAVs). The “calculation of the relative position” is described in the following sub-section (3.2.4).

3.2.4. Calculation of the relative position

Using the results of the identification process, the relative position calculation is performed. The process begins with the calculation of the central point of the identified cylinders. To obtain the polar coordinates of the cylinder’s center point, there are used the equations (3.11) for the distance and (3.12) for the angle. To calculate the distance, it is used the distance between the sensor and the midpoint of the object, and then it is added a half of the diameter. Every object (cylinder) is stored in memory as a vector; the midpoint is then the vector’s component whose index is the half of the vector length. For the calculation of the angle, the average between the angles corresponding to the edge points of the object is done; the start point angle and the end point angle are the vector’s component whose index are zero and the vector length. The polar coordinates of the center

point correspond to the relative position of the cylinder. The result obtained in the initial tests is shown in figure 3.21.

$$CenterDistance = MidPointDistance + \frac{Diameter}{2} \quad (3.11)$$

$$CenterAngle = \frac{StartPointAngle + EndPointAngle}{2} \quad (3.12)$$

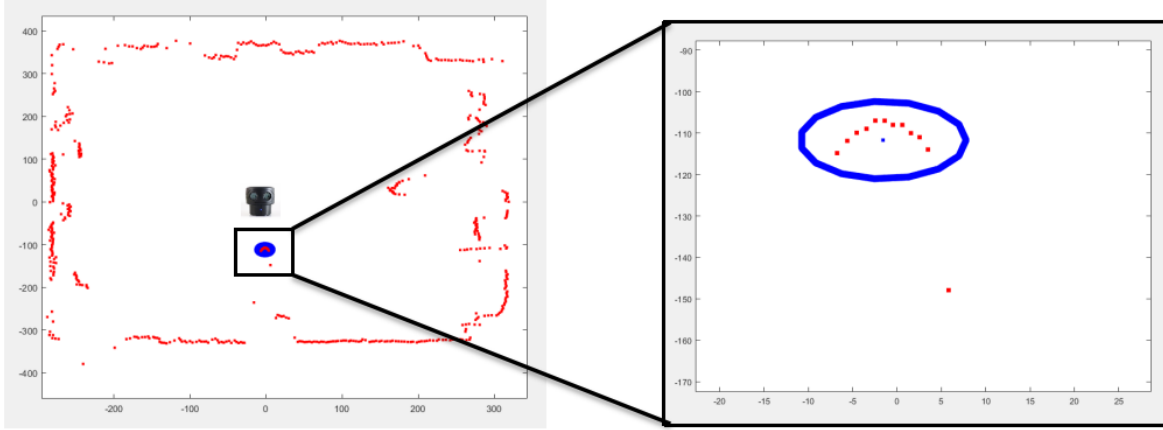


Figure 3.21. Results of the identification process of cylinders in 2D data.

The result of the identification process of cylinders in 2D data is the location and diameter of each cylinder. Below is the result of the identification process in the test data. The distance and diameter are in cm. The azimuth is in radians.

- Sensor-Cylinder distance: 111.68
- Azimuth: 4.698
- Cylinder's diameter: 10.29

3.2.4.1. UAVs Physical Structure



Figure 3.22. UAV prototype, part of the "Recue and exploration with multi-agents" project.

The GALS system was implemented in four UAVs. The UAVs used were the main result of the project "Rescue and exploration with UAVs"[45]. The UAVs were already in operation with remote controlled flight at the time

of the implementation of the GALS system. The UAVs used were of quadcopter type. In figure 3.22 it is shown one of the prototypes. In figure 3.23 a) and b) it can be seen the size of the UAV and its parts.

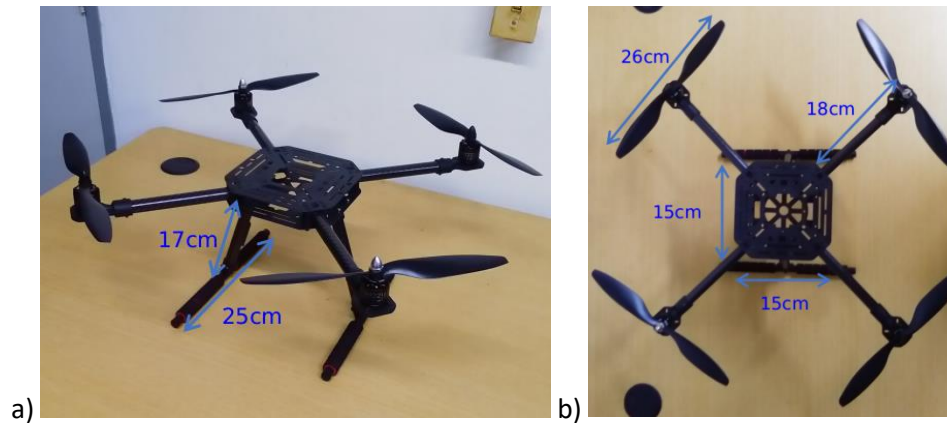


Figure 3.23. UAV size. a) Side view. b) Top view.

At the top of the UAV's structure it was added a cylinder. The cylinder is made of wood and has a diameter of 11cm as can be seen in figure 3.24 a). The cylinder is a fundamental part for the operation of GALS. The LiDAR sensor is located in the inner part of the wooden cylinder. In the middle part of the cylinder there is a methacrylate ring that serves as a window to the LiDAR sensor. In the figure 3.24 b) the final structure of the UAV is showed.

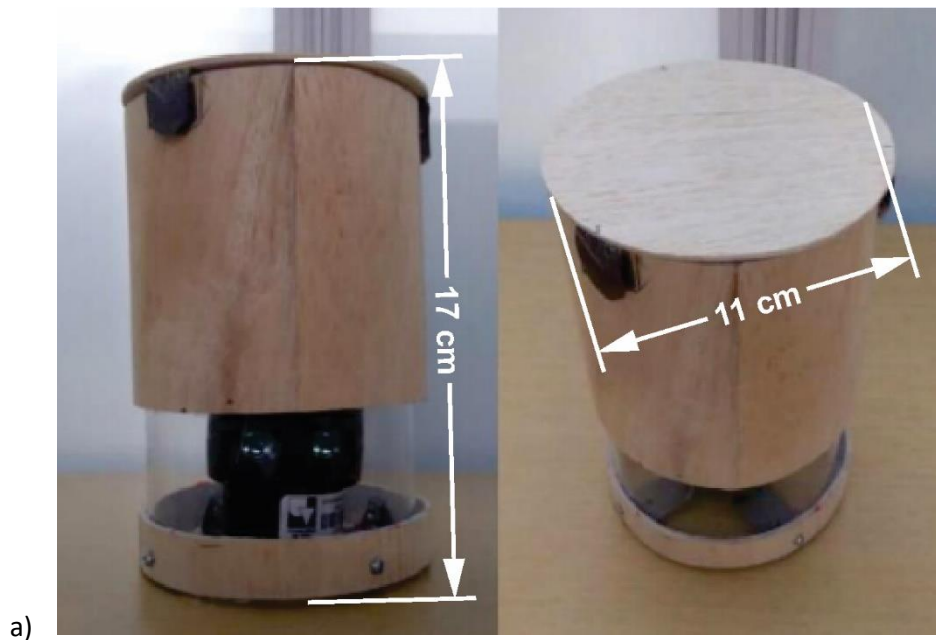


Figure 3.24 a) GAL's cylinder size. b) Final structure of the UAV.

The flight control system was implemented using the sensors of a LG Nexus 5X Smartphone. The Smartphone is located in the physical structure of the UAV. The Smartphone send the control signal to an Arduino DUE development platform over USB. In the USB connection, the Arduino DUE acts as the HOST and the Smartphone acts as the DEVICE. The Arduino DUE is responsible for transforming the received control signal into four PWM signals. The duty cycle of the PWM signals defines the amount of energy delivered to each motor. Figure 3.25 shows a general scheme of operation of the UAV flight control system.

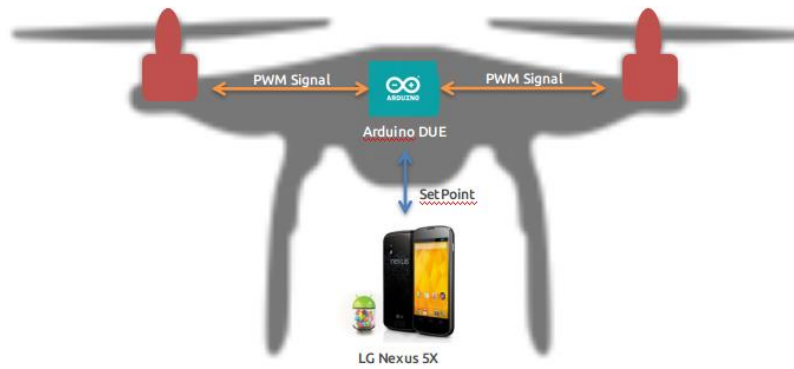


Figure 3.25. UAV's motor operation.

The engines used in the UAVs was the MT2216 of the manufacturer EMAX. Each motor provides a load capacity of 960gr. Figure 3.26 a) show the MT2216 engine.

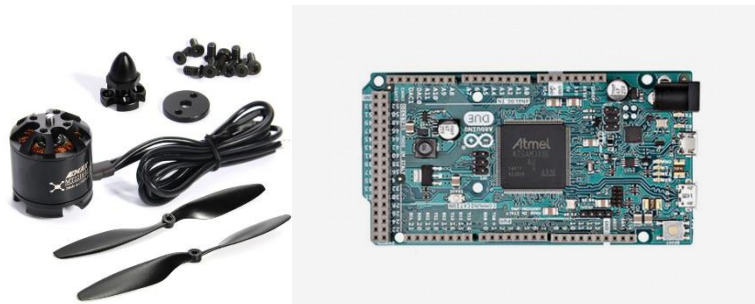


Figure 3.26. a) Left: UAV motor, MT2216 [46]. b) Right: Arduino DUE, development platform [47].

The LG Nexus 5X device was chosen because it meets the following requirements:

1. Multi-core, High frequency CPU.
2. WiFi connection capability.
3. GPS enabled.
4. USB Open Accessory enabled.
5. Sensors: accelerometer, gyro, compass and barometer.

The Arduino DUE was not part of the original design. At the beginning an Arduino Mega ADK was used. The Arduino Mega was chosen because it complied with the requirements of the original design:

1. Accessory Development Kit enabled.
2. 4 PWM output pins.
3. 12V voltage source support.

The GALS design added two additional requirements, listed below:

4. Volatile memory bigger than 9KB.

5. 5V output pin with up to 650mA support.

The fourth requirement is related to the storage and processing of the LiDAR sensor's data. The fifth requirement is related to the power required by the LiDAR sensor. Due to the additional requirements, it was necessary to change the Arduino platform from an Arduino Mega ADK to an Arduino DUE. Figure 3.26 b) shows the Arduino DUE development platform.

3.2.5. Communication Module with the Flight Control App

The communications module was designed to allow the flight controller to access the GALS functions. In Figure 3.27 it can be seen a block diagram of the GALS's modules, and the way in which the information flows between them. In case of the communication between the Smartphone and the Arduino, a communication protocol was designed, it uses an USB connection. In case of the communication between the Arduino and the LiDAR, it uses the protocol provided by the manufacturer of the LiDAR, it uses a standard Serial connection.

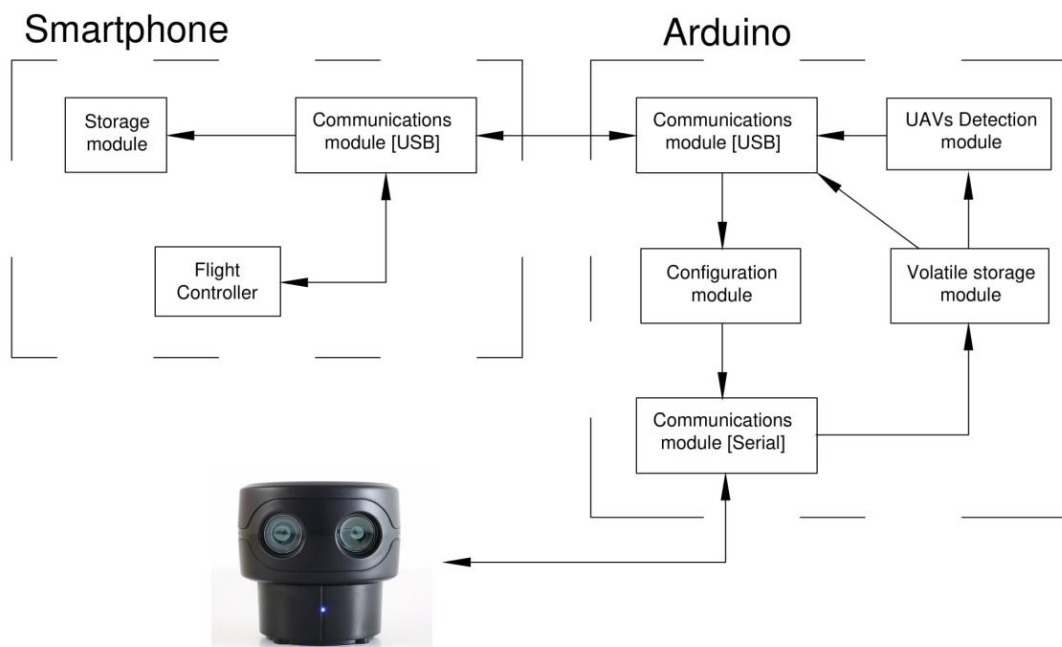


Figure 3.27. GALS Communication's block diagram.

The basic component of the communications module (USB) are the data frames to be exchanged. Figure 3.28 shows the structure of the data frame. The data frame has synchronization bytes at the beginning and at the end (Header / Footer). The two bytes that follow the header correspond to the indicator, which determines the purpose of the data. The data is contained between the indicator and the footer. 8 bytes of data are carried in each frame.

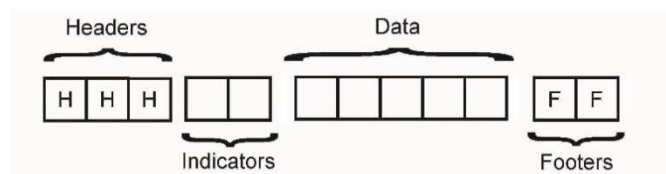


Figure 3.28. Data frame for the communication module.

The indicators available in the communications module are shown in table 3.4. The “action” column defines the operation to be performed. The “category” column defines the category of each indicator. The “from-to” column defines the sender and the receiver for each indicator. The “Request/Response” column defines whether the indicator is sent as a request or as a response. The “indicator” column shows the indicator for the data frames. The “data” column defines the data that is sent in the data frame for each indicator.

Table 3.4 Indicators usage in communication module

CATEGORY	ACTION	FROM-TO	Request - Response	INDICATOR	DATA
Agreements	connection establishment (Handshake)	Smartphone-Arduino Arduino-Smartphone	Req. and Res.	[O][K]	[X][X][X][X][X][X][X][X]
	End of operation	Smartphone-Arduino	Req.	[E][N]	[X][X][X][X][X][X][X][X]
	Done	Arduino-Smartphone Smartphone-Arduino	Res.	[D][N]	[X][X][X][X][X][X][X][X]
SOH	State of Health (SOH), Ask	Smartphone-Arduino	Req.	[S][S]	[X][X][X][X][X][X][X][X]
	State of Health (SOH), Response	Arduino-Smartphone	Res.		[N (Not OK) or (OK)][X][X][X][X][X][X]
Configuration	Start operation	Smartphone-Arduino	Req.	[C][C]	[A][X][X][X][X][X][X][X]
	Stop operation				[O][X][X][X][X][X][X][X]
	Get state				[X][X][X][X][X][X][X][X]
	Set speed				[S][1-10 Hz][X][X][X][X][X][X]
	Set sample rate				[R][1->500Hz; 2->750Hz; 3->1000Hz] [X][X][X][X][X][X]
	Reset				[T][X][X][X][X][X][X][X]
	State	Arduino-Smartphone	Res.	[C][D]	[N (Not OK) or Y (OK)] [Speed value][Sample rate value] [X][X][X][X][X][X]
	Calibrate	Smartphone-Arduino	Req.	[A][D]	GainDist BiasDist
				[A][A]	GainAngle BiasAngle
Data transfer	Get UAV and Obj. data	Smartphone-Arduino	Req.	[D][R]	[X][X][X][X][X][X][X][X]
	Send UAVs data	Arduino-Smartphone	Res.	[D][U]	Angle Distance
	Send point cloud		Res.	[D][P]	Angle Distance
	End of the message		Res.	[D][F]	[X][X][X][X][X][X][X][X]

The flight controller does not directly use the data frames of the communication module. The design of GALS was oriented to simplicity. The simplicity was reflected in the implementation with its API-type design. The Smartphone-Arduino API has two objectives. The first objective is to allow the flight controller system to configure the GALS and the LiDAR sensor. The second objective is to allow the flight controller system to get the GALS's and the LiDAR's sensor data. The AdkCommunicator class is used to send and receive messages with the Arduino. The FlightController class uses the AdkCommunicator and the Smartphone-Arduino API together to reach the two objectives.

Table 3.5 shows the Smartphone-Arduino API functions. Each function name is shown in a row. In "Input" the input variable type and its description are shown. In "Output" the output variable type and its description are shown. In "Description" each function labor is described.

Table 3.5 Arduino-Smartphone API

initCommAdk	
INPUT	-
OUTPUT	uint8_t – Returns 1 if the communication and the LiDAR are running. Returns 0 if the LiDAR is not ready or the communication fails.
DESCRIPTION	A handshake message is sent to the Arduino. An integer value is returned from the Arduino to the Smartphone telling the state of the GALS.
calLiDAR	
INPUT	Float – GainAng, is the gain that will be applied to every angle reading from the LiDAR sensor. Float – BiasAng, is the bias that will be applied to every angle reading from the LiDAR sensor. Float – GainDist, is the gain that will be applied to every distance reading from the LiDAR sensor. Float – BiasDist, is the bias that will be applied to every distance reading from the LiDAR sensor.
OUTPUT	uint8_t – Returns 1 if the configuration was correctly applied to the LiDAR. Returns 0 if the configuration fails.
DESCRIPTION	A calibration message is sent to the Arduino. The LiDAR's information is in polar coordinates (Angle, Distance). The calibration is performed as follows: $RealAngle = (ReadingAngle * GainAng) + BiasAng$. $RealDistance = (ReadingDistance * GainDist) + BiasDist$. A more detailed explanation is in the section 3.2.1.
getLiDARState	
INPUT	&int – LiDARSpeed, Pointer to variable used to storage the sensor rotation speed. &int – LiDARSampleRate, Pointer to variable used to storage the sensor sample rate.
OUTPUT	uint8_t – Returns 1 if the LiDAR returns a valid value. Returns 0 if the LiDAR returns a non-valid value.
DESCRIPTION	A state request message is sent to the Arduino. The Arduino gets the LiDAR information and returns it to the Smartphone. Two values are returned, the rotation speed and the sample rate. The rotation speed can be 0 to 10 [Hz]. The sample rate can be 500, 750 or 1000 [Hz].
confLiDAR	
INPUT	Int – LiDARSpeed, sensor rotation speed. Int – LiDARSampleRate, sensor sample rate.
OUTPUT	uint8_t – Returns 1 if the configuration was correctly applied to the LiDAR. Returns 0 if the configuration fails.
DESCRIPTION	A state configuration message is sent to the Arduino. The Arduino configures the LiDAR according to LiDARSpeed and LiDARSampleRate values. The rotation speed can be 0 to 10 [Hz]. The sample rate can be 500, 750 or 1000 [Hz].
LiDARReset	
INPUT	-
OUTPUT	uint8_t – Returns 1 if the reset was correctly applied to the LiDAR. Returns 0 if the reset fails.
DESCRIPTION	A reset message is sent to the Arduino. The Arduino reset the GALS system including the LiDAR sensor. An integer value is returned from the Arduino to the Smartphone telling the result of the operation.
startGALSOP	
INPUT	-
OUTPUT	uint8_t – Returns 1 if GALS started correctly. Returns 0 if the start operation fails.
DESCRIPTION	A start message is sent to the Arduino. The Arduino starts the operation of the GALS system. An integer value is returned from the Arduino to the Smartphone telling the result of the operation.
getGALSData	
INPUT	&float[1200] – AngD, Pointer to variable used to storage the polar coordinates angle of the detected UAVs. &uint16_t[1200] – DistD, Pointer to variable used to storage the polar coordinates distance of the detected UAVs. &float[1200] – AngP, Pointer to variable used to storage the polar coordinates angle of the environment point cloud. &uint16_t[1200] – DistP, Pointer to variable used to storage the polar coordinates distance of the environment point cloud. &int – NUAV, Pointer to variable used to storage the number of detected UAVs. &int – NPoints, Pointer to variable used to storage the number of points in the 360° reading.
OUTPUT	uint8_t – Returns 1 if the operation was successful. Returns 0 if the operation fails.
DESCRIPTION	A data request message is sent to the Arduino. At the beginning the GALS get a 360° reading from the LiDAR. Then an UAV localization is performed. After that the 360° reading is filtered to eliminate the points corresponding to the identified UAVs. The GALS then return four matrixes with the UAVs and the PointCloud location (in polar coordinates). The number UAVs detected and the number of points in the PointCloud are also returned in order to correctly storage the information in each matrix.
stopGALSOP	
INPUT	-
OUTPUT	uint8_t – Returns 1 if GALS stopped correctly. Returns 0 if the stop operation fails.
DESCRIPTION	A stop message is sent to the Arduino. The Arduino stops the operation of the GALS system. An integer value is returned from the Arduino to the Smartphone telling the result of the operation.

3.3. Storage module (android)

It was implemented a storage module that runs on the Smartphone. The purpose of this module is to store in non-volatile memory the information generated by GALS. The storage of data allows debugging in GALS, but also the analysis of data even after landing the UAVs. The stored data contain information of relative positions, but also contains information about the environment that was captured by the LiDAR. The environment information is very wide thanks to the LiDAR's range that is up to 40m.

Figure 3.29 shows a block diagram that illustrates the operation of the storage module. This block diagram is directly related to the one showed in figure 3.27. The data are received at high speed over the USB connection. The received data is stored temporarily in a buffer that operates on volatile memory. A thread runs parallel to GALS and it is responsible for storing the received information in non-volatile memory.

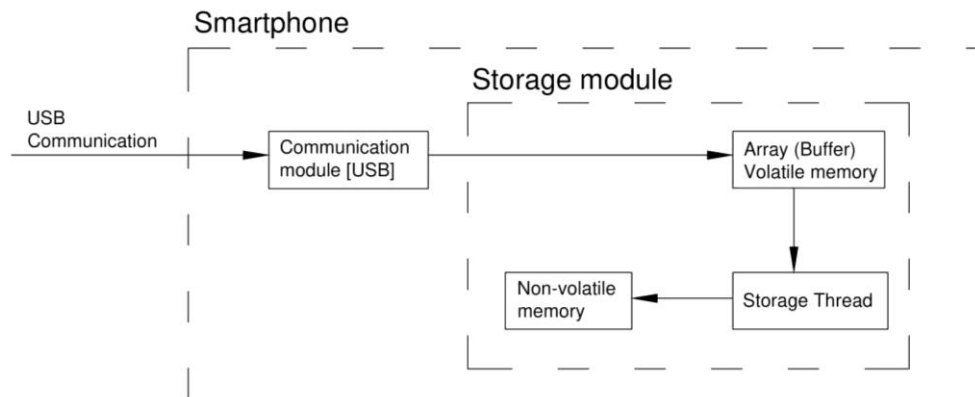


Figure 3.29. Block diagram of the Storage module operation.

The storage module was tested. Blocks of 1000 float values were sent to the Smartphone. The values are 500 coordinates from the LiDAR. When the storage tests were carried out without the buffer, it stores files with 700 float values approximately. The loss of data indicates that the transmission speed was 1.42 times higher than the storage speed. Because of that the storage with input buffer was developed. It was decided to use a buffer with an array of 500 float values. The value 500 is half the size of the block, so it is now possible to receive 1000 floating values without loss of data. In the final version of the code it was used the value of 1000 float values in the buffer to support the reception of 2000 float values. Blocks of size 2000 are received when the laser sampling frequency is set to 1000sps.

The type of the buffer used is FIFO (First input - First output) [48]. In this kind of buffer, the data is stored in volatile memory and wait "in row" to be stored in non-volatile memory. The buffer is a float array that is used as a ring as depicted in Figure 3.30. The index of the array returns to the initial position when the limit of the array is reached. The volatile data indicator increases fast, but not all the time. The non-volatile data indicator increases slowly but constantly.

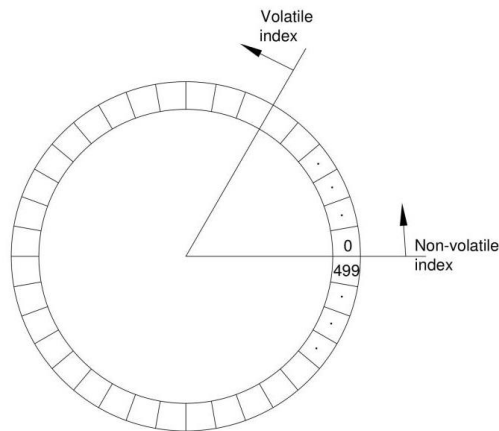


Figure 3.30. Block diagram of the Storage module operation.

Each cell of the ring represents one float number received by the Smartphone from the Arduino. A quantity of data equivalent to the size of the ring would be lost if the buffer size is not correctly chosen. The loss of data occurs when the volatile data indicator exceeds the position of the non-volatile data indicator by more than one "round".

3.4. Base station

The UAVs had an operational base station software at the time of starting this project. The base station software allows the user to interact with the UAV. The base station software was developed in Java and its executed in a computer. The user is able to indicate the directions of movement for the UAV. The user can set geographical locations for the UAV to flight over that points (Waypints). The UAV can accomplish the waypoints missions because the Smartphone has an embedded GPS sensor. Figure 3.31 shows the main screen of the base station application.

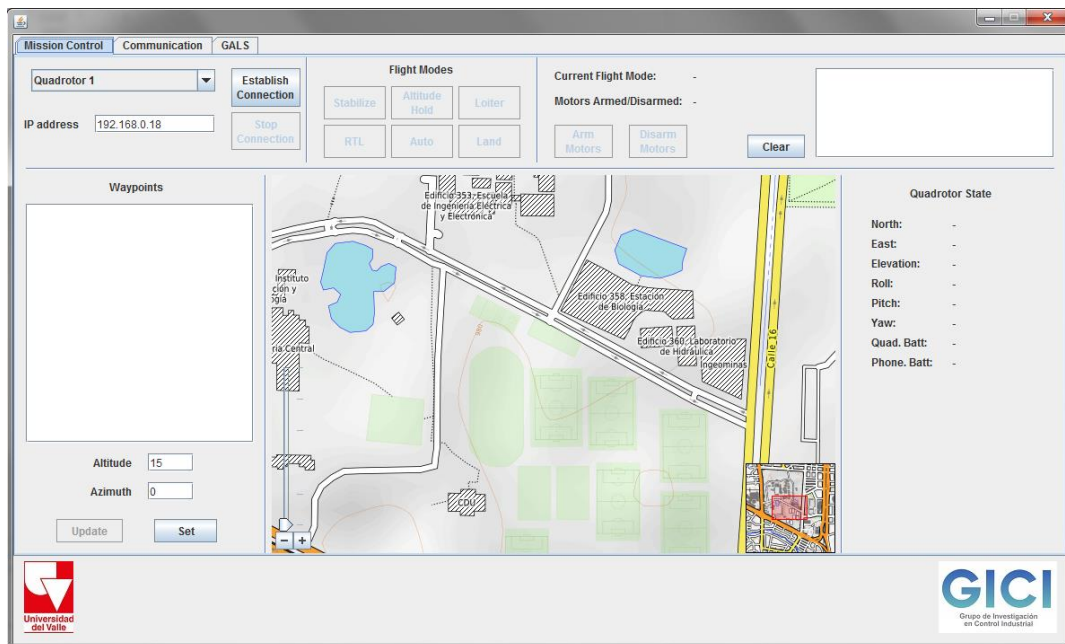


Figure 3.31. Screenshot of the Base Station application.

The Communications tab allows you to configure the connection between the Smartphone and the base station. The connection is established through WiFi. The Smartphone acts as an AccessPoint and the base station acts as a client. The WiFi technology is used with WebSockets so it is possible to transmit ASCII characters. The structure of the communication protocol used by the Smartphone and the base station is shown in figure 3.32.



Figure 3.32. Communication protocol between the Ground Station and the Smartphone.

The message is indeed a char array. Each frame has three components:

- **ID**: Identification of the message emitter
- **Function**: function selected from a list of pre-defined functions
- **Data**: piece of the message that carry the necessary information for the selected function.

The base station software had to be extended to receive the information generated by GALS. A tab dedicated to GALS was added. The GALS tab can be seen in Figure 3.33. From the new tab the user can configure the LiDAR parameters and also visualize the data generated by GALS.

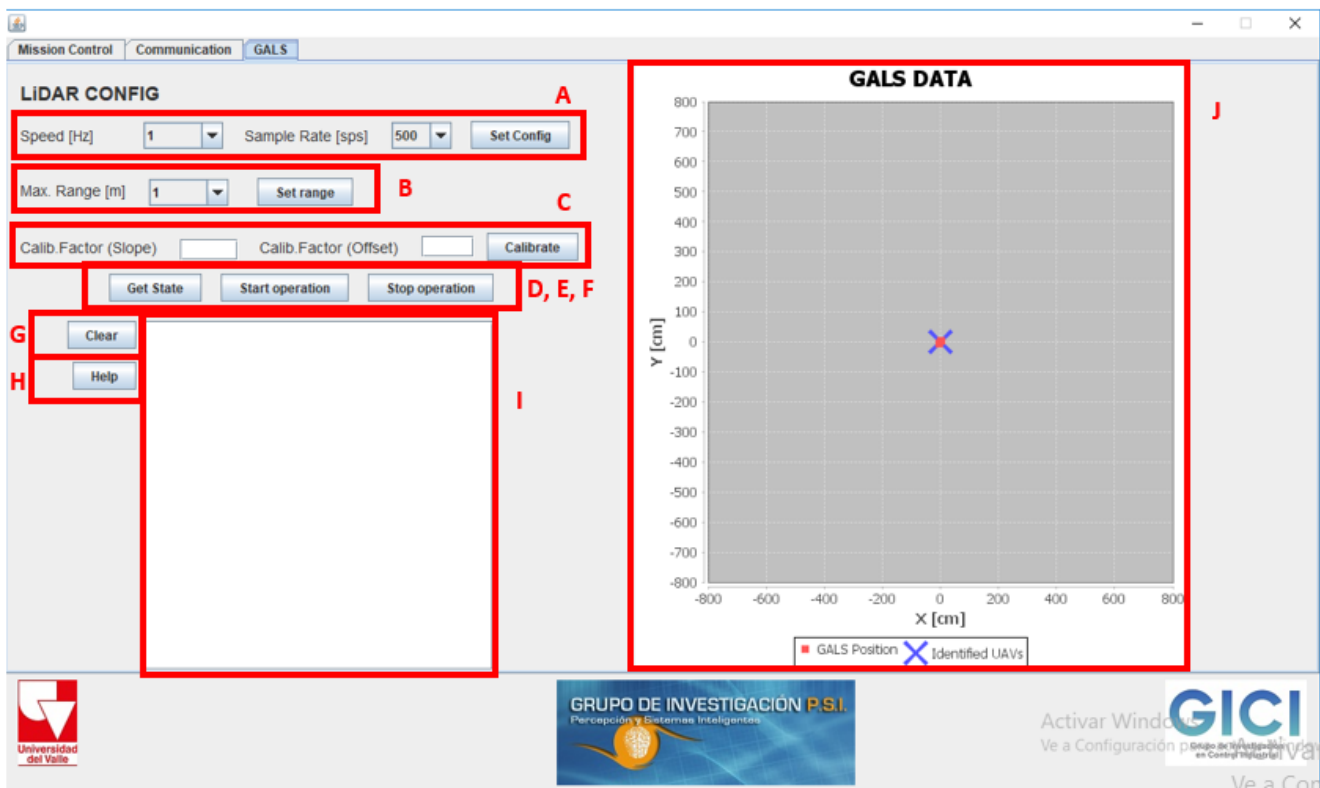


Figure 3.33. Screenshot of the GALS tab in the Ground Station program.

The elements part of the GALS tab are described below:

- a) **Set Configuration Button:** When pressed, a request is sent to GALS. The configuration set the operating parameters of the LiDAR. The parameters are the Speed and Sample Rate fields.
- b) **Set Range Button:** When pressed, sends a request to GALS. It sets the maximum operating distance of GALS. This button also changes the plotting limits. It takes the value of the Max Range field as input.
- c) **Calibrate Button:** Pressing this button sends a request to GALS to configure the LiDAR calibration parameters. It takes the Slope and Offset fields as input.
- d) **Get State Button:** Pressing this button sends a request to GALS. The request asks for the status of the LiDAR operation. The received response is displayed in the Logs window.
- e) **Start Operation Button:** Pressing this button sends a request to GALS. It requests the start of the operation. The start generates the reception of Range data. Once the operation is started it is not possible to make configurations in the LiDAR.
- f) **Stop Operation Button:** Pressing this button sends a request to GALS. It requests the stop of the operation. The stop generates the end of the Range data transmission. Once the operation is finished it is again possible to make configurations in the LiDAR.
- g) **Clear button:** Pressing this button the Logs window is cleared.
- h) **Help button:** Pressing this button generates a popup window. The window shows basic information about the use of the GUI.
- i) **Logs window:** This field shows GALS messages. The information displayed is range data, identified UAVs, configurations, confirmations received and faults / errors identified.
- j) **Plotting area:** In this field the information received from GALS is plotted. The range information is displayed as well as the location of the identified UAVs.

The data received from the Smartphone is stored in the base station for later analysis. The GALS's data is stored in text files. Table 3.6 shows the messages added to the Smartphone-Ground station protocol to support GALS.

Table 3.6 Smartphone-Ground Station Protocol messages added for GALS

ID	FUNCTION	DATA
UAV ID	LRF-ENV (LiDAR data)	Range, Angle, Range....
	LRF-POS (Identified UAVs)	Range, Angle, Range....
Ground Station ID	LOP (LiDAR Operation)	Start
		Stop
	LCONF (LiDAR Configuration)	Speed, Sample rate

In Figure 3.34, it is shown the UAV equipped with GALS. It is also showed a laptop that executes the base station software.



Figure 3.34. Picture of the UAV with GALS and the Base Station.

Figure 3.35 shows a screenshot of the base station software; here, it shows the data coming from GALS being received.

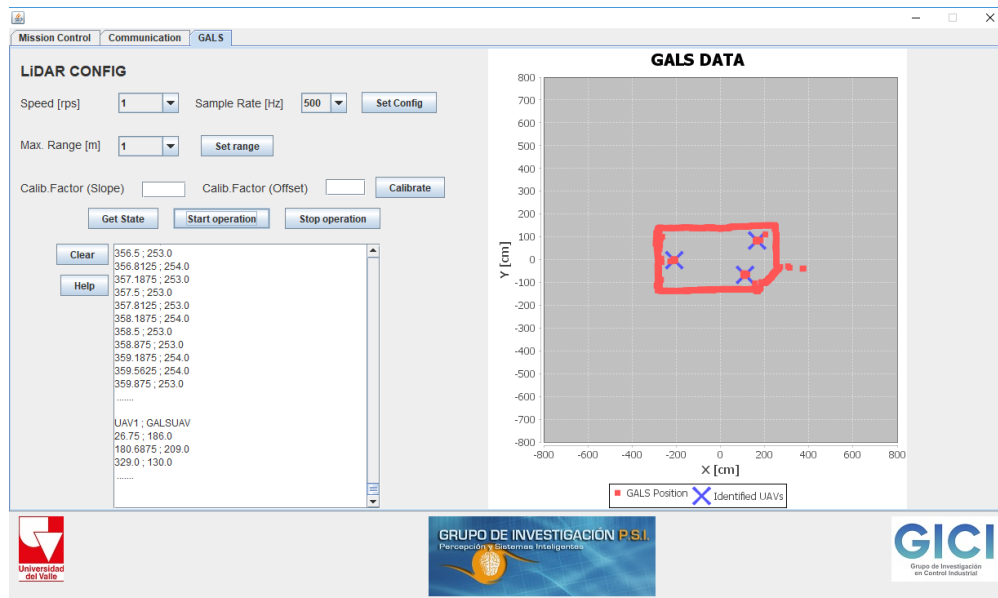


Figure 3.35. Screenshot of the execution of the Base Station application.

Base station's storage module was configured to store GALS's data as (Angle, Distance). The data corresponding to identified UAVs were also stored. Figure 3.36.a shows a text file generated by the base station. Figure 3.36.b shows a plot of the base station's data.

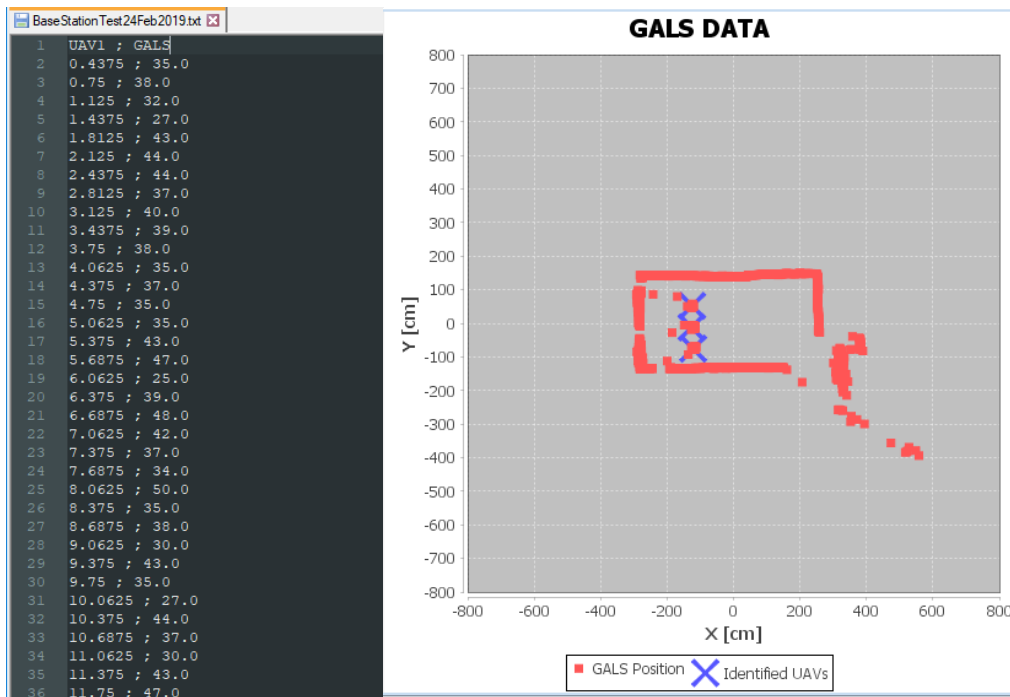


Figure 3.36. a) Piece of text file saved by the base Station. b) Plot of the saved data.

3.5. Final observations

The section 3.1 was completely dedicated to the design phase of GALS. The functional requirements showed in table 3.1 leaded the development of the first prototype. The main functional requirements are: Allow acquisition of data from the laser sensor; Perform an UAV identification in the data received from the laser sensor; Integrate the results with the Flight Control Software that currently runs on the devices. With the non-functional requirements, the preparation of the workspace was carried out. The conceptual diagram introduced the idea of the modular operation of GALS. The class diagrams were a fundamental piece in the development of the software. Finally, the flow diagrams were the basis for the operation and interaction of the modules, as well as their synchronization. With all the design guidelines available, the implementation of GALS began.

The section 3.2 was dedicated to the implementation phase of GALS. For the time of the implementation, there were excellent performance sensors, small enough to be equipped in an UAV. With the determination and calibration of the sensor, the implementation of GALS started. It was developed a method for the identification of nearby UAVs. That method achieved the expected results in laboratory tests. It is capable of identify the cylindrical shape of the objects in the reading. Additionally, it is capable of measure the diameter of the cylinders. The identification method showed the capability of identify the relative position of the UAV's cylinder in the range of 1 to 5 meters. Finally, a robust communication module with the flight controller was implemented. The communications module accomplishes the requirements in the usage and configuration of GALS.

The section 3.3 showed the way in which the data is stored in non-volatile memory. The storage module accomplishes the requirements: Storage the data received form GALS in non-volatile memory. The module allows the continuous and orderly storage of the data received by GALS. A file with the storage results can be found in the directory **Annex2_GALSStorageResults**.

The section 3.4 showed the base station operation. The GUI allows the user to use the GALS capabilities. The GUI also allows to configure GALS.

After the implementation phase, the validation and testing of the prototype continues. The qualitative and quantitative evaluation of the prototype is shown in chapter 4.

Chapter 4

VALIDATION AND TESTS

4.1. Functional tests

4.1.1. Automatic Detection of UAV using a Laser Range Finder

4.2. Field Tests

4.3. conclusions

This chapter shows the results of the tests and validations performed to GALS. The integration tests are shown in section 4.1. In the integration tests GALS was tested together with the flight controller. The data provided by GALS was analyzed in real-time with the information collected by the UAV's ground station, but also with the information provided by the Storage module. This chapter also shows the field tests performed on GALS. GALS was tested with the UAV as a single device. Initially, the device GALS+UAV was tested at fixed positions (Static test). After the static test, GALS was tested with a moving UAV (Tracking test). The conclusions of this chapter show the performance parameters of GALS, the achievements and the limitations that were found. Each test was designed to accomplish with the functional requirements included in the RUP.

The code and files part of GALS are stored in the folder **Annex3_Code&Files**. That folder contains the source files used to program each device that make part of GALS. This chapter make reference to the GALS's functional requirements and its fulfillment. Real-use case forms shows the tests results in detail. The Real-use case forms are stored in the folder **Annex1_RUPDocs**. At the end of this chapter Field Test results are analyzed. The data generated by GALS during the Field Test is stored in the folder **Annex3_GALSFTData**.

4.1. FUNCTIONAL TESTS

This section shows the tests performed to each GALS's module. In sub-section 4.1.1, the operation of the identification module was tested using the text files generated by the GALS's storage module. This test was performed executing GALS together with the flight controller. In sub-section 4.1.2 the report and notification module were tested. There was analyzed the text files generated by the notification and report module in the ground station.

4.1.1. Automatic Detection of UAV using a Laser Range Finder

The functional test began after the laboratory test ended. In the functional tests, GALS was executed together with the flight controller. The Smartphone storage module was used to verify that the execution flowed normally, and both systems (GALS and flight controller) were fulfilling the corresponding tasks. The Smartphone storage module was described in section 3.3. The storage module was configured to store the data corresponding to the LiDAR readings (Angle, Distance). The data corresponding to the identified UAVs were also stored. Figure 4.1 shows the physical configuration used in the test.



Figure 4.1. Physical configuration for the Flight Controller + GALS GUI.

The functional tests were carried out following the RUP documentation. 7 functional tests are reported, each test is related to the functional requirements of the system. Table 4.1 shows the functional requirements designed with RUP. The tables 4.2 to 4.8 have the following structure:

1. Requirements: Name of the test's functional requirement.
2. Test type: Type of test performed.
3. Required hardware: Hardware required to perform the test.
4. Required software: Software required to perform the test.
5. Objective (1): Main purpose of the requirement.
6. Objective (2): Main purpose of the test.
7. Test data: Data necessary to perform the test.
8. Procedure: Step-by-step of the test.
9. Expected results: Expected results for the test.
10. Results obtained: Results obtained after the test.
11. Notes: Author's notes about the test.

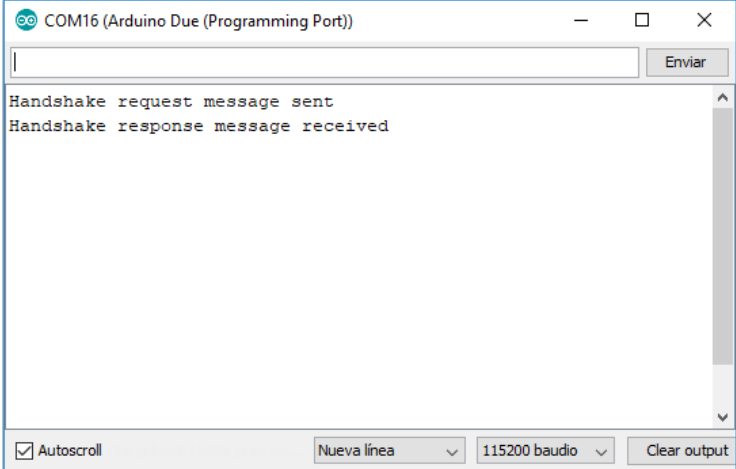
Table 4.1 GALS Functional Requirements

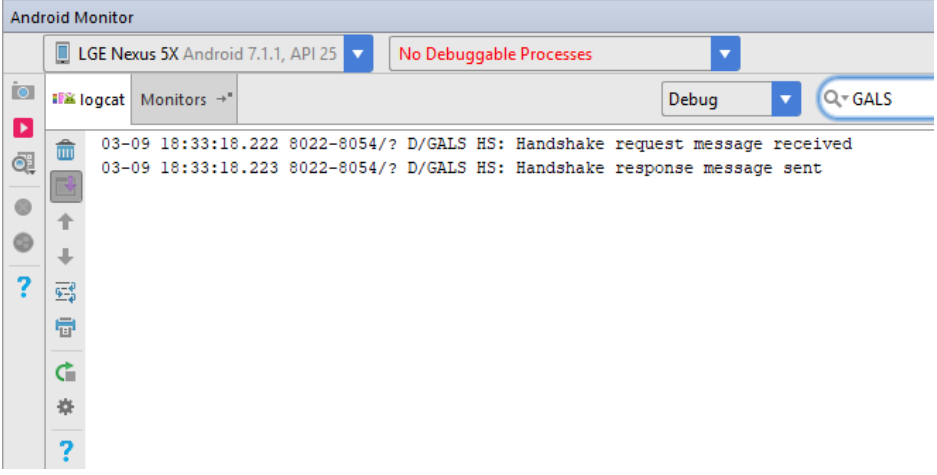
Ref #	Functions
1.1	Allow communication between Smartphone and Arduino
1.2	Allow acquisition of data from the laser sensor
1.3	Perform an UAV identification in the data received from the laser sensor
1.4	Perform a filtering of the data received from the laser sensor
1.5	Allow software configuration of the laser sensor
1.6	Allow transmission of the relative position of other drones and obstacles from the Arduino to the Smartphone
1.7	Integrate the results with the Flight Control Software that currently runs on the devices

4.1.1.1. Arduino-Smartphone Communication

This test verifies the operation of the communication link between the Arduino and the Smartphone. The entire test and its results are described in table 4.2.

Table 4.2 Functional Test: Arduino-Smartphone Communication

FUNCTIONAL TEST: ARDUINO-SMARTPHONE COMMUNICATION	
Requirements	Allow communication between Arduino and Smartphone
Test type	functional
Required hardware	Arduino DUE, Smartphone LG Nexus 4
Required software	Arduino IDE v1.8
Objective	To check the communication established between Arduino and Smartphone
HANDSHAKE BETWEEN ARDUINO AND SMARTPHONE	
Objective	To check the sending and receiving capabilities of both Arduino and Smartphone
Test data	None
Procedure	1. Turn on the devices (Arduino and Smartphone) 2. Connect the Arduino and the Smartphone through USB cable 3. Wait until automatic handshake messages are traded 4. Check results in Arduino's serial monitor and Smartphone's logs
Expected results	1. Evidence of handshake messages traded in Arduino's serial monitor 2. Evidence of handshake messages traded in Smartphone's logs
Results obtained	a) 

	 b)
Notes	None

In Table 4.2, screenshots of the procedure performed and the results obtained are shown. In the result a) The serial monitor of the Arduino is shown. This screenshot shows the request for a handshake message sent to the Smartphone. It is also observed that the message is answered correctly. In the result b) The logs of the Smartphone are shown. This screenshot shows the reception of the handshake message received from the Arduino. It is also observed that the message is answered correctly. The results show the fulfillment of the requirement for GALS.

4.1.1.2. LiDAR data acquisition

This test verifies the operation of the communication link between the Arduino and the LiDAR. The entire test and its results are described in table 4.3.

Table 4.3 Functional Test: LiDAR Data Acquisition

FUNCTIONAL TEST: LiDAR DATA ACQUISITION	
Requirements	Allow data acquisition from the laser sensor
Test type	functional
Required hardware	Scanse Sweep V1 LiDAR, Arduino DUE
Required software	Arduino IDE v1.8, Sweep Library v1.4
Objective	To check the exchange of data between LiDAR and Arduino
LiDAR DATA ACQUISITION AND SERIAL PRINTING	
Objective	To check that the 360-degree range data is correctly received in the Arduino
Test data	None
Procedure	1. Connect the Arduino and the LiDAR through Serial connection 2. Turn on both devices (Arduino and LiDAR)

	3. Send a “Start transmission” message from the Arduino to the LiDAR 4. Collect 360-degree range data in the Arduino 5. Serial print the data received in the Arduino
Expected results	1. Get 360-degree range data in the Arduino’s Serial monitor
Results obtained	COM16 (Arduino Due (Programming Port)) Enviar Distance: 364.00 cm 1061 Angle: 356.81 Deg Distance: 365.00 cm 1062 Angle: 357.12 Deg Distance: 364.00 cm 1063 Angle: 357.50 Deg Distance: 365.00 cm 1064 Angle: 357.81 Deg Distance: 362.00 cm 1065 Angle: 358.12 Deg Distance: 363.00 cm 1066 Angle: 358.50 Deg Distance: 364.00 cm 1067 Angle: 358.81 Deg Distance: 364.00 cm 1068 Angle: 359.12 Deg Distance: 363.00 cm a) ☒ Autoscroll Nueva línea 115200 baudio Clear output
Notes	None

In Table 4.3, screenshots of the procedure performed and the results obtained are shown. In the result a) the serial monitor of the Arduino is shown. In the screenshot the last 8 of the 1069 coordinates received are shown. The LiDAR was set to 1000sps configuration. The results show the fulfillment of the requirement for GALS.

4.1.1.3. UAVs identification

This test verifies the UAV identification process that runs on the Arduino. The entire test and its results are described in table 4.4.

Table 4.4 Functional Test: UAVs Identification

FUNCTIONAL TEST: UAVS IDENTIFICATION	
Requirements	Allow UAV identification in the data received from the laser sensor
Test type	functional
Required hardware	Scanse Sweep V1 LiDAR, Arduino DUE
Required software	Arduino IDE v1.8, Sweep Library v1.4
Objective	To check the correct identification of UAVs in 2D range data
UAVS IDENTIFICATION IN 2D RANGE DATA	
Objective	To perform an UAV identification in a 2D range data
Test data	None
Procedure	1. Connect the Arduino and the LiDAR through Serial connection 2. Turn on both devices (Arduino and LiDAR) 3. Send a “Start transmission” message from the Arduino to the LiDAR 4. Collect 360-degree range data in the Arduino 5. Execute the detection algorithm in the received data 6. Serial print the results
Expected results	1. Get the detected UAV information in the Arduino’s serial monitor
Results obtained	

```
COM16 (Arduino Due (Programming Port))

1065
Angle: 358.88 Deg
Distance: 272.00 cm

1066
Angle: 359.25 Deg
Distance: 274.00 cm

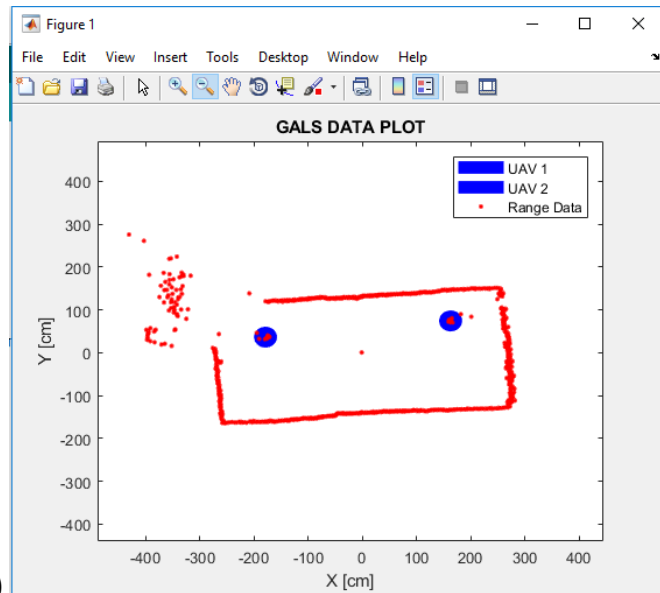
1067
Angle: 359.56 Deg
Distance: 269.00 cm

1068
Angle: 359.94 Deg
Distance: 271.00 cm

Cylinder No.:
1
Distance[cm]:
182.00
Angle[Deg]:
24.44
Diameter[cm]:
10.01
Cylinder No.:
2
Distance[cm]:
180.00
Angle[Deg]:
168.50
Diameter[cm]:
9.91
Detected UAVs:2
```

☒ Autoscroll Nueva línea 115200 baudio Clear output

a)



b)

Notes

None

In Table 4.4, screenshots of the procedure performed and the results obtained are shown. GALS was executed in a field of 5m x 3m. Two cylinders were located 180 cm away from the LiDAR. In the result a) the serial monitor of the Arduino is shown. GALS identified the cylinders and calculate the following location for the cylinders. The first cylinder was at 182 cm and at an angle of 24 degrees. For the second UAV the calculated location was at 180 cm and at an angle of 182 degrees. In the result b) it is shown a plot of the result. Result b) was obtained using the MATLAB software. The results show the fulfillment of the requirement for GALS.

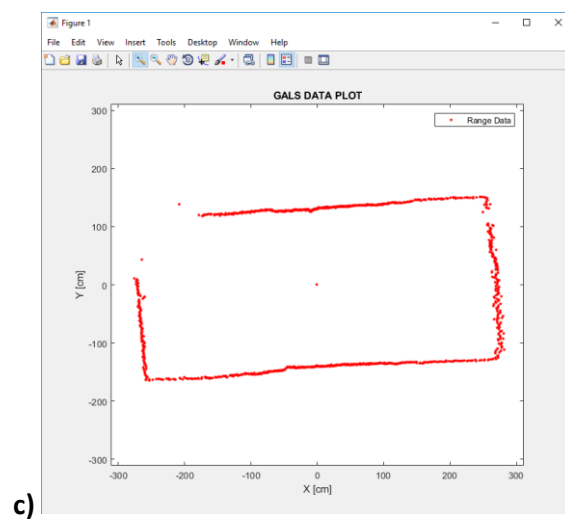
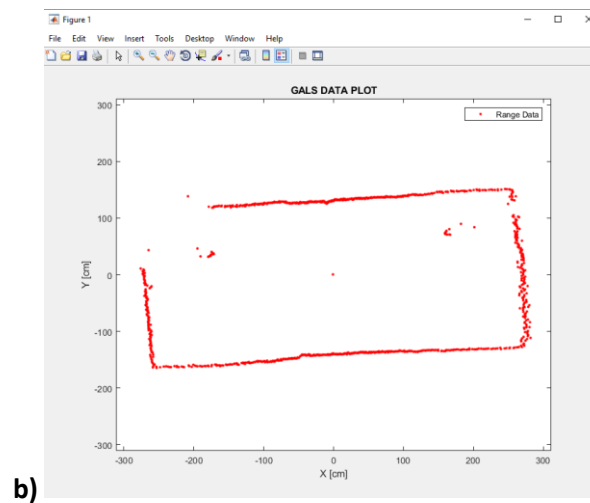
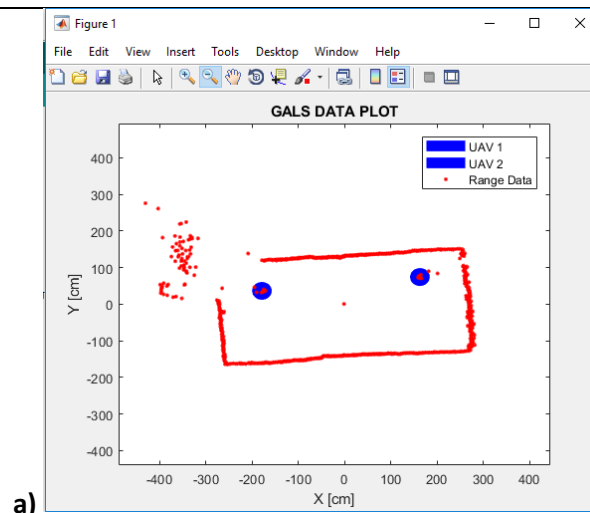
4.1.1.4. Data filtering

This test verifies that the data filtering is executing properly. The entire test and its results are described in table 4.5.

Table 4.5 Functional Test: Data Filtering

FUNCTIONAL TEST: DATA FILTERING	
Requirements	Allow the filtering of the range data. The points corresponding to UAVs are eliminated. The remaining points are assumed as obstacles.
Test type	functional
Required hardware	Scanse Sweep V1 LiDAR, Arduino DUE
Required software	Arduino IDE v1.8, Sweep Library v1.4
Objective	To check the correct filtering of 2D range data.
DATA FILTERING	
Objective	To perform data filtering in 360-degree range data
Test data	None
Procedure	1. Connect the Arduino and the LiDAR through Serial connection 2. Turn on both devices (Arduino and LiDAR) 3. Send a "Start transmission" message from the Arduino to the LiDAR 4. Collect 360-degree range data in the Arduino 5. Execute the detection algorithm in the received data 6. Serial print the results of the 360-degree range data and the detection process 7. Filter the 360-degree range data 8. Serial print the filtered data
Expected results	1. Get 360-degree range data including the points corresponding to UAV 2. Get 360-degree range data without the points corresponding to UAV

Results obtained



Notes

None

In Table 4.5, screenshots of the procedure performed and the results obtained are shown. In the result a) the detection result is shown. The filtering process was done in the data shown in table 4.4. In the result b) the data before filtering is shown. In the result c) it is shown a blank space at the point where the UAVs were located, the data was filtered correctly. The results show the fulfillment of the requirement for GALS.

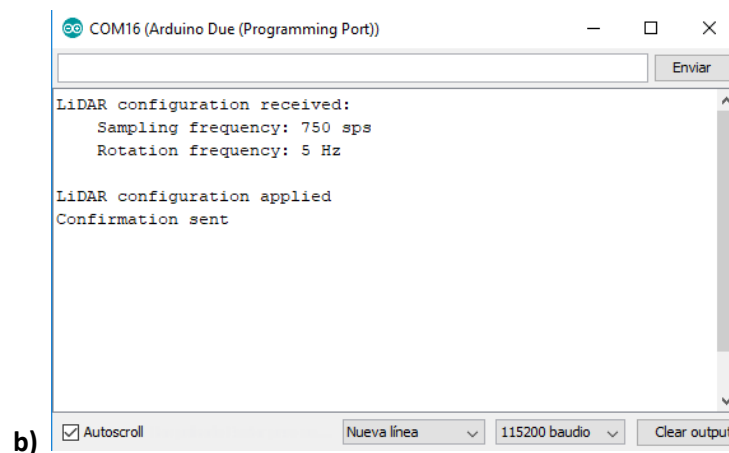
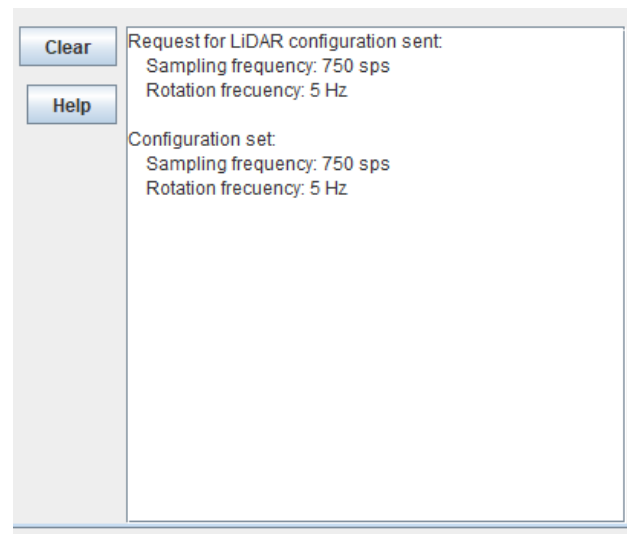
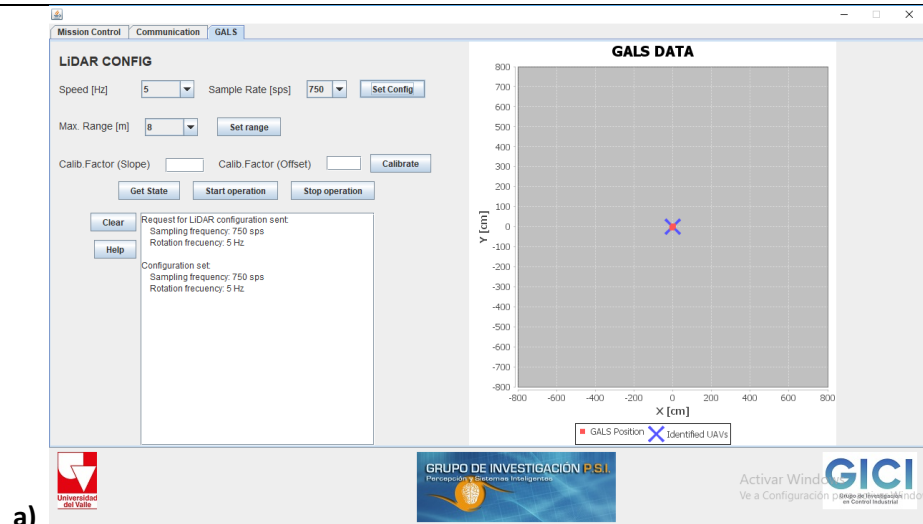
4.1.1.5. LiDAR configuration

The LiDAR has configurable parameters: The sampling frequency, the rotation frequency and the data acquisition status. This test verifies that GALS can change the configuration of the sensor correctly. The entire test and its results are described in table 4.6.

Table 4.6 Functional Test: LiDAR Configuration

FUNCTIONAL TEST: LiDAR CONFIGURATION	
Requirements	Allow software configuration of the laser sensor
Test type	functional
Required hardware	Scanse Sweep V1 LiDAR, Arduino DUE, Smartphone LG Nexus 4, Ground station computer
Required software	GALS code for Arduino DUE, Flight controller Android application with GALS add (QuadcopterControlUV), Java JRE v8.0, Flight controller Java application with GALS add (GCSUV)
Objective	To check the configuration capabilities of GALS over de LiDAR
LiDAR CONFIGURATION FROM GROUND STATION	
Objective	To perform a LiDAR configuration from the ground station
Test data	None
Procedure	1. Connect the Arduino and the LiDAR through Serial connection 2. Turn on Arduino and LiDAR 3. Turn on The Smartphone hotspot option 4. Connect the Smartphone and the Arduino through USB connection 5. Wait until the Flight controller Android application with GALS add (QuadcopterControlUV) automatically starts 6. Connect the ground station pc to the Smartphone hotspot over WiFi 7. Start the Flight controller Java application with GALS add (GCSUV) 8. Send configuration message from the ground station
Expected results	1. Get configuration confirmation in ground station's logs window 2. Observe the received configuration in the LiDAR operation

Results obtained



Notes

None

In Table 4.6, screenshots of the procedure performed and the results obtained are shown. In the result a) a screenshot of the base station software is shown. In the logs window it is shown the configuration sent and the confirmation from GALS. In the result b) the serial monitor of the Arduino is shown. It is shown the reception and application of the received configuration. The results show the fulfillment of the requirement for GALS.

4.1.1.6. GALS data transmission

An UAV with GALS has three types of communication. The communication types are: Serial Communication between the Arduino and the LiDAR; USB communication between the Arduino and the Smartphone; WiFi communication between the Smartphone and the base station. Each link is vital for the proper operation of GALS. The data exchanged through these links are: Digital PWM values; Range data; identified UAVs and Request/Response messages. This test verifies the operation of the communication links in GALS. The entire test and its results are described in table 4.7.

Table 4.7 Functional Test: GALS Data Transmission

FUNCTIONAL TEST: GALS DATA TRANSMISSION	
Requirements	Allow acquisition of the relative position of other UAVs and obstacles from the Arduino
Test type	functional
Required hardware	Arduino DUE, Smartphone LG Nexus 4, Ground station computer
Required software	GALS code for Arduino DUE, Flight controller Android application with GALS add (QuadcopterControlUV), Java JRE v8.0, Flight controller Java application with GALS add (GCSUV)
Objective	To check the transmission of data from the Arduino going through the Smartphone to the base station
GALS DATA: FROM ARDUINO TO BASE STATION	
Objective	Send the relative position of other UAVs and obstacles from the Arduino going through the Smartphone to the base station
Test data	UAVs coordinates, 360-degrees range data
Procedure	1. Turn on Arduino 2. Turn on The Smartphone hotspot option 3. Connect the Smartphone and the Arduino through USB connection 4. Wait until the Flight controller Android application with GALS add (QuadcopterControlUV) automatically starts 5. Connect the ground station pc to the Smartphone hotspot over WiFi 6. Start the Flight controller Java application with GALS add (GCSUV) 7. Wait for UAVs and range data to be transferred
Expected results	1. Observe the received data in the base station logs window 2. Observe the transmitted data in the Smartphone's memory as txt file

Results obtained

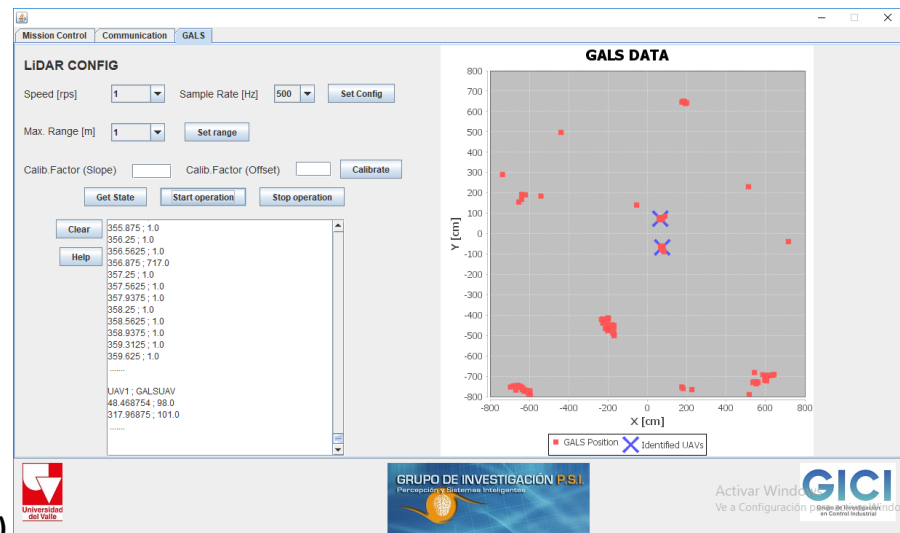
Test 45 & -45 1m.txt

```

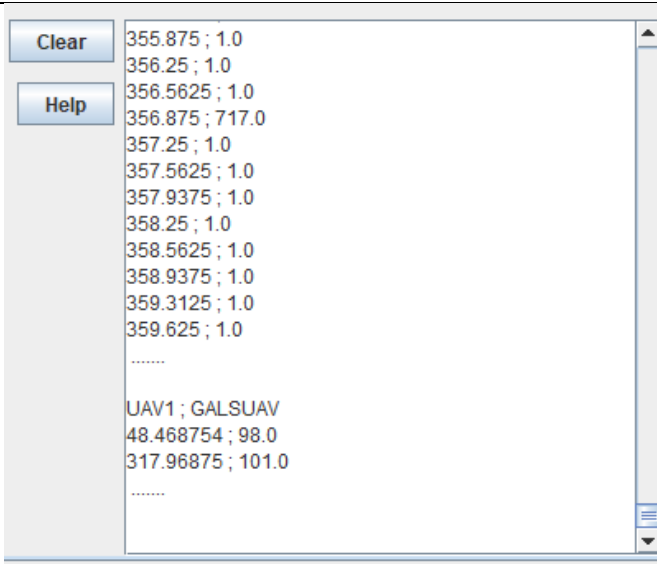
6432 352.1875;1.0
6433 352.5;1.0
6434 352.875;1.0
6435 353.1875;1.0
6436 353.5625;9448.0
6437 353.875;1.0
6438 354.1875;3399.0
6439 354.5625;3392.0
6440 354.875;3391.0
6441 355.1875;1.0
6442 355.5625;1.0
6443 355.875;1.0
6444 356.25;1.0
6445 356.5625;1.0
6446 356.875;717.0
6447 357.25;1.0
6448 357.5625;1.0
6449 357.9375;1.0
6450 358.25;1.0
6451 358.5625;1.0
6452 358.9375;1.0
6453 359.3125;1.0
6454 359.625;1.0
6455 ///////////////
6456 ///////////////
6457 ///////////////
6458 48.468754;98.0
6459 317.96875;101.0
6460 ///////////////
6461 ///////////////

```

a)



b)

	
Notes	None

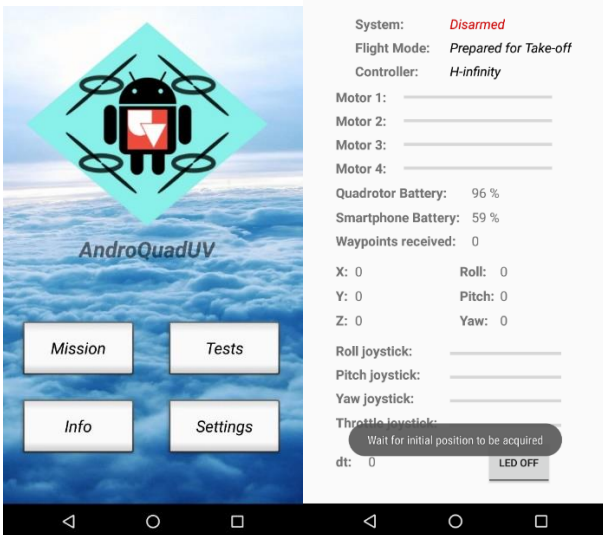
In Table 4.7, screenshots of the procedure performed and the results obtained are shown. In the result a) part of the text file stored in the Smartphone is shown. The text file has the information received from the Arduino. In the result b) It can be seen a screenshot of the base station application. In a red box it is shown the logs window, the data received from the Smartphone is displayed. It can be seen that the UAVs data were received as well as the 360-degree reading of the LiDAR. The results show the fulfillment of the requirement for GALS.

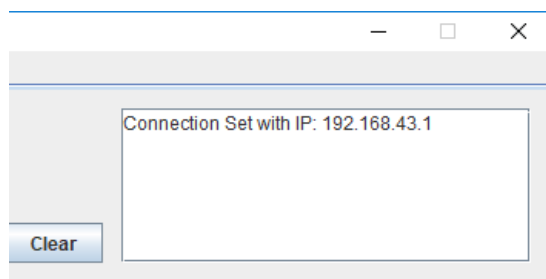
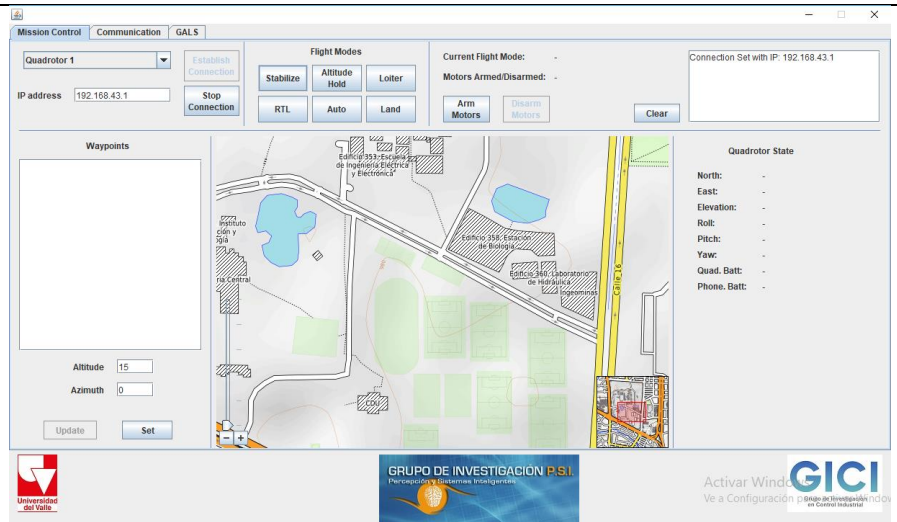
4.1.1.7. Integration with flight control software

Arduino, Smartphone and base station computer execute both GALS and flight control software. At the beginning of this project the UAVs only executed the flight control software. It is very important for GALS to does not obstruct the operation of the flight control software. This test verifies the operation of GALS and the flight control software executing together. The entire test and its results are described in table 4.8.

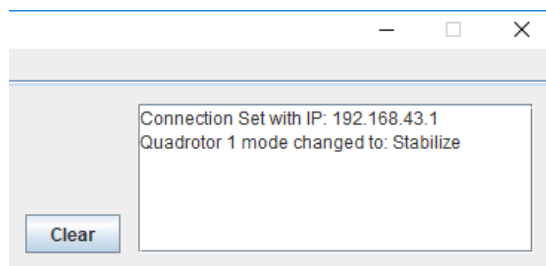
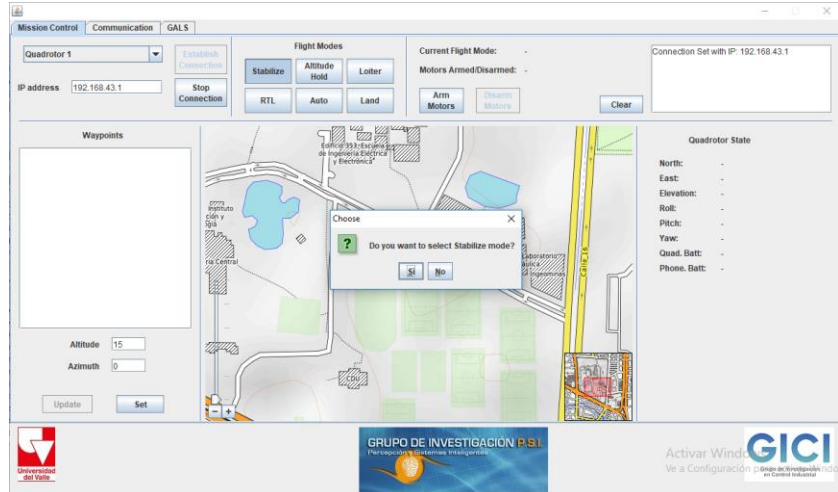
Table 4.8 Functional Test: Integration with Flight Control Software

FUNCTIONAL TEST: INTEGRATION WITH FLIGHT CONTROL SOFTWARE	
Requirements	Integrate the capabilities of GALS with the Flight Control Software that currently runs on the UAVs
Test type	Functional
Required hardware	Scanse Sweep V1 LiDAR, Arduino DUE, Smartphone LG Nexus 4, Ground station computer, External USB Control, Oscilloscope
Required software	GALS code for Arduino DUE, Flight controller Android application with GALS add (QuadcopterControlUV), Java JRE v8.0, Flight controller Java application with GALS add (GCSUV)

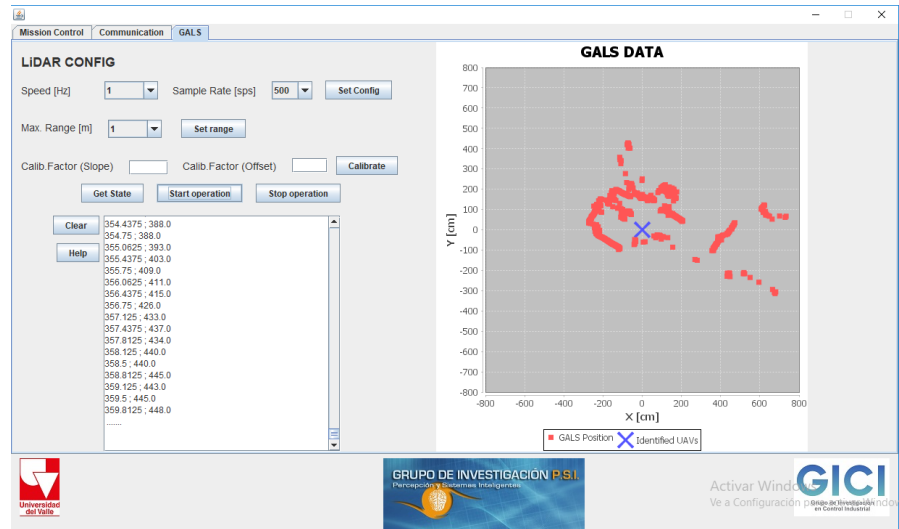
Objective	To check the execution of GALS and Flight Control software over the same hardware at the same time
GALS AND FLIGHT CONTROL SOFTWARE EXECUTION	
Objective	To check the behavior of both GALS and Flight Control Software when executing together
Test data	None
Procedure	1. Connect the Arduino and the LiDAR through Serial connection 2. Connect the Oscilloscope port to the PWM Output on the Arduino 3. Turn on Arduino and LiDAR 4. Turn on The Smartphone hotspot option 5. Connect the Smartphone and the Arduino through USB connection 6. Wait until the Flight controller Android application with GALS add (QuadcopterControlUV) automatically starts  7. Connect the ground station pc to the Smartphone hotspot over WiFi 8. Start the Flight controller Java application with GALS add (GCSUV) 9. Connect the external USB control 10. Start connection between the base station and the Smartphone



11. Set flight mode to Stabilize



12. Start GALS Operation



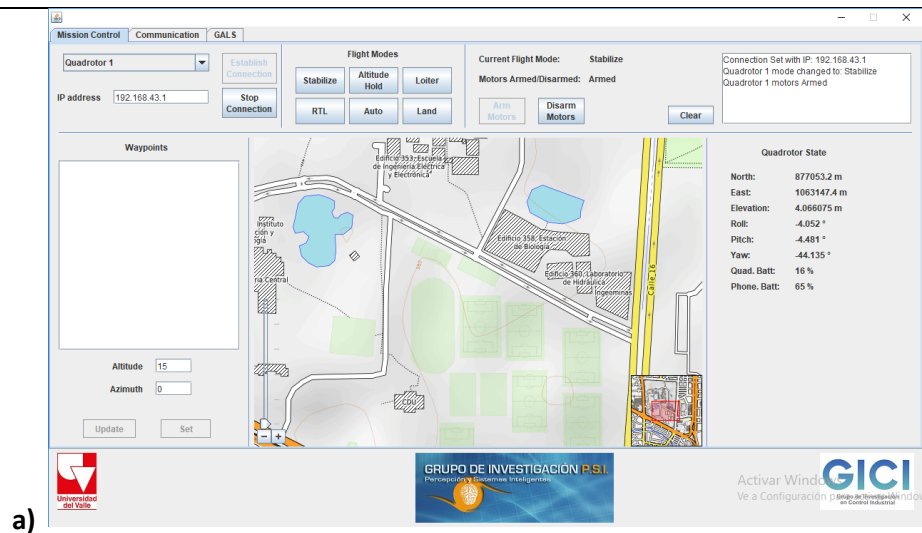
13. Put a set point to the motors



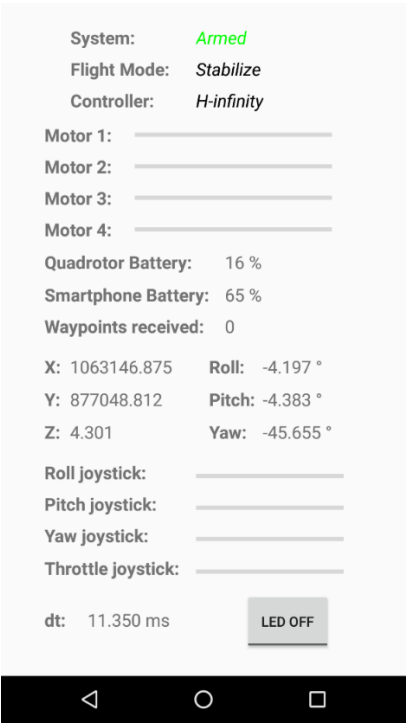
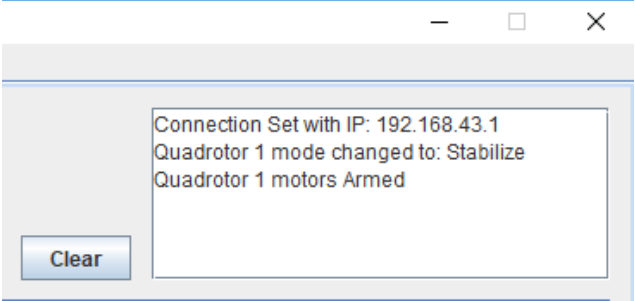
Expected results

1. Get the PWM output as set
2. Get GALS Data in the base station's logs

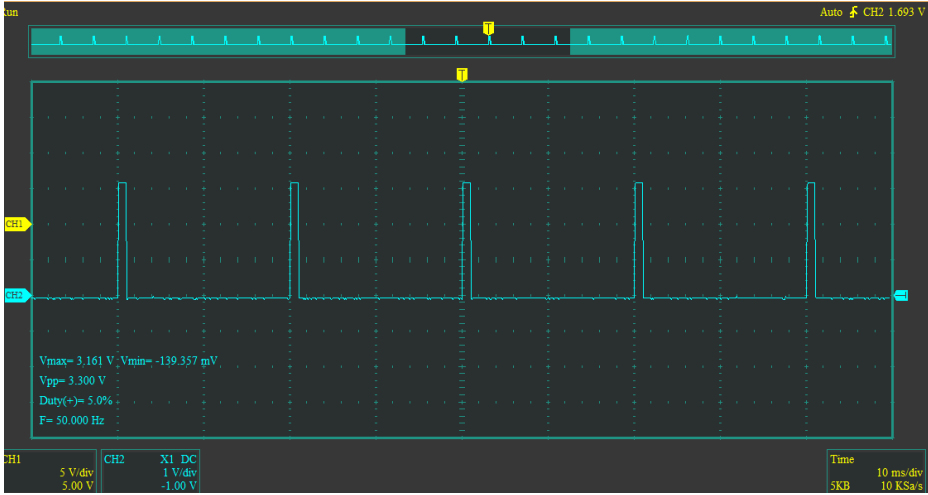
Results obtained



a)



b)



c)

Notes

None

In Table 4.8, screenshots of the procedure performed and the results obtained are shown. In the result a) the GUI of the flight control software is shown. In the result b) a screenshot of the Smartphone is shown when the motors were set to minimum set-point. In the result c) it is shown an oscilloscope screenshot of the PWM signal generated from the Arduino. The input of the oscilloscope was one of the motor controllers input signal. The results show the fulfillment of the requirement for GALS.

4.2. FIELD TESTS

In this section, it is tested the operation of GALS as part of the UAV. The GALS performance is shown at the end of this section. The UAVs were tested at ground height. The operation of each GALS's module was tested at the same time while they were working together with the flight controller.

The tests show the UAV with GALS operating together. GALS and the UAV operate at this point as a single device. For test purposes the UAV was tested at ground height. The main UAV was placed at the center of the tests field. Another UAV was placed at fixed positions to the initial UAV. The data generated by GALS was saved for later analysis. This test let to know the precision of the location results of GALS. The test configuration is shown in Figure 4.2. Each red circle is a test position. A picture of the location used for the test is shown in figure 4.3. The main UAV was placed at the center of the soccer field. The secondary UAV was placed at 1m and 180° of the main UAV.

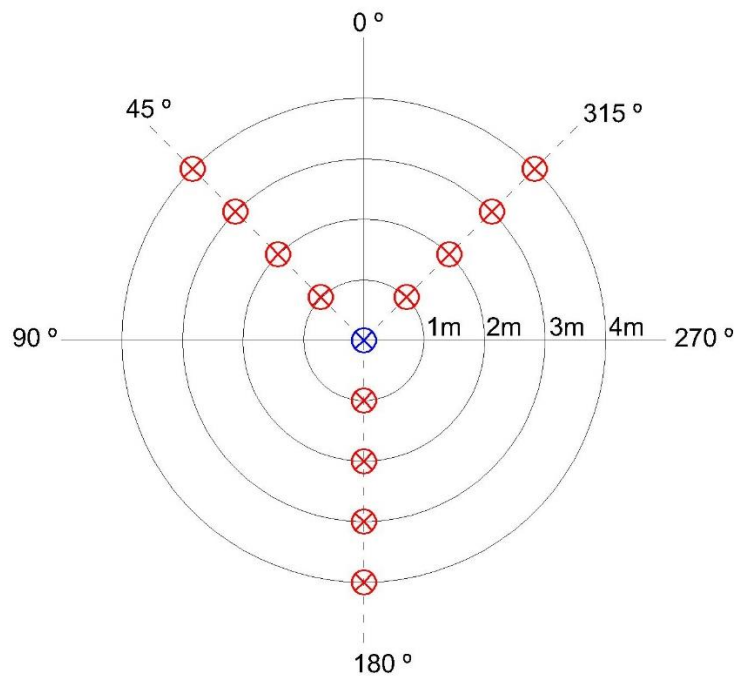


Figure 4.2. Test positions used for the static test.



Figure 4.3. Picture of the test location for static test.

The cylinder was placed at each position for 10 GALS's cycles. There are 12 positions for the test, so there are 120 detections. Figure 4.4 shows the data generated by GALS during the test. The plot was generated using the UAV position of every detection.

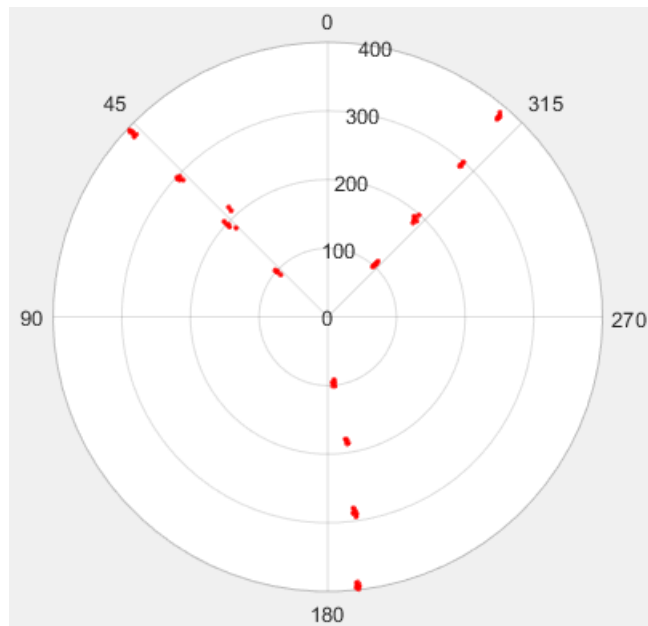


Figure 4.4. Plot of the results for the static test.

The performance of GALS was measured using the parameters described above. The value assigned to GALS for each parameter was calculated from the data stored by the Smartphone. The data stored by the base station was also used.

- **Sampling period:** Time required to get a 360° dataset and process it. The value obtained by GALS for this parameter was: 22.51 s. This value is the average between 50 readings.

- **Successful readings:** Percentage of successful UAV identifications with respect to the total UAV reported by GALS. In the tests, GALS reported 129 UAVs of which 120 corresponded to UAVs that were actually present. Therefore, the value obtained by GALS for this parameter was: 93.02%.
- **Erroneous readings:** Percentage of erroneous UAV identifications with respect to the total UAV reported by GALS. In the tests, GALS reported 129 UAVs of which 9 corresponded to UAVs that were not present. Therefore, the value obtained by GALS in this parameter was: 6.98%.
- **Distance-angle average error:** Accuracy in the calculation of the position of the identified UAVs. It was calculated the average of the error obtained in distance and angle. The results obtained by GALS for this parameter were Distance: 7.78 cm and Angle: 4.34 °.
- **Standard deviation in Distance-Angle:** This parameter evaluates the accuracy in the calculation of the position of the identified UAVs. The standard deviation was calculated in the calculation of distance and angle. The results obtained by GALS for this parameter were Distance: 2.86 and Angle: 1.13.

4.3. CONCLUSIONS

In section 4.1, the GALS integration tests were carried out. With the functional test tables that are part of RUP (4.2 to 4.8), the operation of the system was verified. Here, the following test were carried out: Arduino-Smartphone Communication, LiDAR Data Acquisition, UAVs Identification, Data Filtering, LiDAR Configuration, GALS Data Transmission and Integration with Flight Control Software. The functional tests show the fulfillment of all the functional requirements designed for GALS. There is also shown the step-by-step of the UAV+GALS operation.

In section 4.2, the GALS UAV identification and location results were tested. GALS operates for 120 rounds while another UAV was at fixed locations. The results allow the calculation of performance parameters of GALS. The following parameters were calculated: Sampling period, successful readings, erroneous readings, distance-angle average error and standard deviation in distance-angle.

The correct detection percentage is one of the GALS successes. In laboratory tests the value of successful detections reached 100%. The average distance-angle error, 7.78cm and 4.34 degrees respectively, is also a satisfactory result for GALS. Then, the uncertainty in the calculation of the position of the UAV is less than one UAV diameter. The standard deviation in Distance-Angle values show that the location process is accurate. The results obtained by GALS for this parameter were 2.86 and 1.13 for distance and angle respectively. The sampling period of GALS is 23 seconds. This value is higher than expected. 17% of this time is spent in data transmission through the communication links. The remaining 83% of time is spent in the LiDAR reset, which is a step that could not be avoided. The reset is used between a LiDAR reading and the next one. The contribution of time of the range data processing is negligible (20uS). The correct detection percentage is one of the GALS successes. In laboratory tests the value of successful detections reached 100%.

The next chapter include the conclusions of the project. The future works proposed are also shown.

Chapter 5

CONCLUSION AND FUTURE WORKS

5.1. conclusions

5.2. Future Works

5.1. CONCLUSIONS

The objective of this project was to design and implement a measurement system for calculating the relative position between a set of UAVs. This piece of software was called GALS. GALS was added to existing UAVs. The UAVs used has basic flight capabilities. The hardware and software of the UAVs was modified to accomplish with GALS requirements. A LiDAR sensor was added to the UAVs structure. The interface platform used to manage the motors was changed. Storage classes were added both in the UAV and on the base station. The UAVs GUI was modified to allow the use and configuration of GALS.

GALS is an alternative for UAVs to get environment information. The LiDAR sensor generates a 2D range plane of the environment. The range data is used to calculate the relative position of other UAVs. The GUI modifications allow the user to visualize the range data generated by GALS in real time.

Before the beginning of the design phase of GALS, a State-of-the-Art was made. The State-of-the-Art describe the latest developments related to the calculation of the relative position. The benefits generated each development, whether academic, commercial or military, were analyzed. The State-of-the-Art was the argument for the decisions made during the design phase. To do so, this work satisfies with the specific objective number 1 proposed for this project.

The implementation of GALS directly involves the LiDAR sensor and its data. The range data is used to describe de UAV surroundings. The range data are also used for the detection and calculation of the relative position of UAVs. The GUI of the UAV was modified. It was added a tab dedicated to GALS. The GALS tab has: 2D plotting field, logs window; Configuration panel for the LiDAR; Calibration panel for the LiDAR; Start/stop panel for GALS. Data acquisition and UAVs identification are performed continuously, 360 degrees at a time. The data generated by GALS allows the user to obtain environmental data directly from the perspective of the UAV.

The GALS design was made following the RUP methodology. The RUP guidelines were used to validate the operation of the system. The tests carried out show the fulfillment of each one of the RUP requirements. The functional tests checked the proper operation of GALS. It is also shown the step-by-step of the GALS operation.

The implemented prototypes operation was evaluated. The integration between GALS and the flight control software was successful. The detection process generates successful results in 93% of the cases. The average error in location was 7.78cm and 4.34 degrees for distance and angle respectively. The standard deviation in location was 2.83 and 1.13 for distance and angle respectively. The results obtained shows that the location process of GALS is accurate.

Finally, there are tools whose objective was to facilitate the development of GALS. The tools may be relevant for future developments that involve GALS. The first set of tools are the storage modules. GALS allows to save text files in the Smartphone and in the base station. The text files have the range and detection data generated

by GALS. The text files also have configuration and error messages generated by GALS. The text files allow the analysis of range data, detection results and operation data of GALS even when the mission is over. The second set of tools are the debugging tools of each GALS device. Debugging tools allow to get execution and error message. The Arduino generates debug messages through the serial port. The Arduino serial port can be accessed through the “USB programming” port. In the Smartphone debug messages are generated through LogCat. The LogCat can be accessed through the AndroidStudio development environment. In the base station debug messages are generated through the java console. Java console can be accessed through the NetBeans development environment.

5.2. FUTURE WORKS

This chapter shows the possible future works related to GALS. The future works are sort by its objective. The first group has the objective of improve the behavior of GALS. The second group has the objective of accomplish some goal making use of GALS. The possible future works are described below.

- **Improvements**

- **Continuous operation:** The detection process of GALS operates over 360-degrees data sets. The range points that describe a UAV are up to 22 (UAV at 1m). It is then possible to increase the efficiency of the system if the detection process executes over smaller data sets. The detection process could operate with a sliding detection window. An example of this operation principle is the radar used on ships.
- **Rotation/sample rate increase:** GALS allows to use the LiDAR data with any of its configurations. The user is capable of configure the LiDAR to operate at different rotation frequency and sampling rate. Otherwise, GALS detection process is design to operate using the highest density configuration (1000sps and 1Hz rotation frequency). GALS could be improved by extending its detection capabilities to faster configurations.
- **Security routines:** GALS is designed to operate under certain conditions. Unexpected scenarios may generate unexpected responses from GALS. The security of GALS could be improved by safety routines.

- **Future works**

- **Tracking:** Currently the detection process of GALS treats each detection as independent. In a future project, detections could be concatenated to track the displacement of an UAV.
- **Speed measurement:** It is possible to calculate the displacement of an object between detections. The period of time between detections is known. With these two parameters the relative speed of an object is calculable. If the object been tracked is static, it is then possible to calculate the speed of the GALS UAV.
- **Maps generation:** The generation of maps from range data is a procedure that has been studied in works like [49]. It is then possible to implement algorithms that allow the generation of environment maps for the UAV. The generation of maps is part of SLAM, a more complex procedure.
- **3D Reconstruction:** The generation of 3D models from range data is a procedure that has been studied in works like [50]. It is then possible to implement algorithms that allow generating 3D models. The UAV could be capable of 3D modeling objects and structures that surround it.

Chapter 6

ANNEX

There is relevant information for GALS that is not included directly in this document. This information is committed digitally. The annexes have been referred during the chapters. The directory structure for the annexes is shown in figure 6.1.

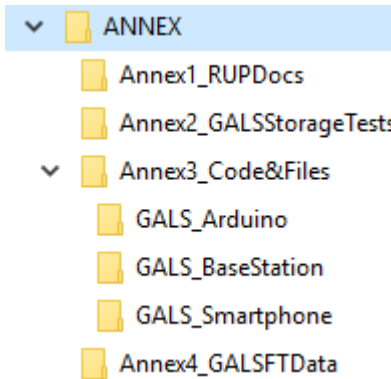


Figure 6.1. Annex directory structure.

Each annex directory is described below.

1. **Annex1_RUPDocs:** The directory has the RUP methodology files. There are included: conceptual diagram, functional / non-functional requirements, real-use cases, class diagrams and flow diagrams.
2. **Annex2_GALSStorageTests:** The directory has the results of the tests performed on the storage modules. Files from the Smartphone and the base station are included.
3. **Annex3_Code&Files:** The directory includes the programming file for Arduino "GALS_Arduino.ino". There are included the project files of the Smartphone "GALS_Smartphone". There are also included the project files of the base station "GALS_BaseStation".
4. **Annex4_GALSFTData:** The directory includes the text files generated by GALS during the field tests.

BIBLIOGRAPHY

- [1] intelligenia DYNAMICS. Aplicaciones y usos. Retrieved May 31, 2017, from http://www.iuavs.com/pages/aplicaciones_y_usos
- [2] Drone Helps Rescue Missing Hikers in Colorado - The Drive. Retrieved March 5, 2018, from <http://www.thedrive.com/aerial/11695/drone-helps-rescue-missing-hikers-in-colorado>
- [3] Aerial Photography Drone: Top Products Picks and Expert's Opinions. Retrieved March 5, 2018, from <http://mydronelab.com/best-pick/aerial-photography-drone.html>
- [4] Using Drones to Control Forest Fires - DRONELIFE. Retrieved March 5, 2018, from <https://dronelife.com/2015/06/11/using-drones-to-control-forest-fires/>
- [5] Drone startup Aptonomy introduces the self-flying security guard | TechCrunch. Retrieved March 5, 2018, from <https://techcrunch.com/2016/08/22/drone-startup-aptonomy-has-created-robotic-flying-security-guards/>
- [6] Chinese military drones rival the best of the U.S. - MarketWatch. Retrieved March 5, 2018, from <https://www.marketwatch.com/story/chinese-military-drones-rival-the-best-of-the-us-2016-12-01>
- [7] Hörtnner, H., Gardiner, M., Roland, H., Lindinger, C., & Berger, F. (2012). Spaxels, Pixels in Space. Retrieved from http://www.aec.at/futurelab/files/2012/07/spaxels_revision_final_20120615.pdf%5Cnpapers3://publication/uuid/F6E4A530-5E2C-444F-B6B2-60D2D0C099B1
- [8] Drone 100 – The World Record For Intel 2015 - SPAXELS®. (n.d.). Retrieved March 5, 2018, from <https://www.spaxels.at/show/drone-100-the-world-record-for-intel-2015/>
- [9] Internet of Things (IoT) news, blogs and analysis - IoTAgenda.com. Retrieved April 21, 2017, from <http://internetofthingsagenda.techtarget.com/>
- [10] Lugo, J. J., Masselli, A., & Zell, A. (2013). Following a quadrotor with another quadrotor using onboard vision. 2013 European Conference on Mobile Robots, ECMR 2013 - Conference Proceedings, 26–31. <https://doi.org/10.1109/ECMR.2013.6698815>
- [11] Rabbath, C. A. (2013). A finite-state machine for collaborative airlift with a formation of unmanned air vehicles. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 70(1–4), 233–253. <https://doi.org/10.1007/s10846-012-9692-7>
- [12] Okutake, T., Uchida, N., & Yamamoto, N. (2016). A Collaborative Safety Flight Control System for Multiple Drones. 2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 371–375.
- [13] Basiri, M., Schill, F., Lima, P., & Floreano, D. (2016). On-Board Relative Bearing Estimation for Teams of Drones Using Sound. *IEEE Robotics and Automation Letters*, 1(2), 820–827. <https://doi.org/10.1109/LRA.2016.2527833>
- [14] Zanobini, A. (2016). A drone anti-collision system: maintaining a fixed distance from a target during flight.
- [15] Li, R., Liu, J., Zhang, L., & Hang, Y. (2014). LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments. 2014 DGON Inertial Sensors and Systems, ISS 2014 - Proceedings.
- [16] Akbar, Y., & Riyanto, I. (2015). Local Area Positioning System (LAPS) for Indoor Navigation System.
- [17] LG Electronics. (2017). LG. Retrieved September 6, 2017, from <http://www.lg.com/uk/images/lg-mobile-phones/e960/gallery/medium01.jpg>
- [18] Arduino. (2017). Arduino. Retrieved September 6, 2017, from <https://www.arduino.cc/>
- [19] RobotShop inc. (2017). Robotshop. Retrieved September 6, 2017, from <http://www.robotshop.com/en/sweep-v1-360-laser-scanner.html>
- [20] Clipartpig.com. Clipart share. Retrieved September 14, 2017, from <https://clipartpig.com/cat/everything>
- [21] Silva, G., Rodriguez, M., Bacca, E. (2017). Sistema de Gestión de Vuelo de un Dron para la Captura Geo-referenciada de Imágenes. Universidad del Valle, 2017.
- [22] Wang, Y., Yang, Q., Zhang, G., & Zhang, P. (2016). Indoor Positioning System Using Euclidean Distance Correction Algorithm with Bluetooth Low Energy Beacon. 2016 International Conference on Internet of Things and Applications (IOTA), 243–247.
- [23] Chengming, L., Gaifang, X., Hai, Y., & Wei, L. (2016). Shearer positioning for mining fleet cooperative movement in, 503–506.
- [24] Racko, J., Brida, P., Perttula, A., Parviainen, J., & Collin, J. (2016). Pedestrian dead reckoning with particle filter for handheld smartphone. 2016 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2016, (October), 4–7.

- [25] Turner, L., & Sherlock, C. (2013). An introduction to particle systems, (May), 1–25. Retrieved from <http://www.lancaster.ac.uk/pg/turner/PartileFiltering.pdf>
- [26] Torres-Sospedra, J., Jiménez, A. R., Knauth, S., Moreira, A., Beer, Y. (2017). The smartphone-based offline indoor location competition at IPIN 2016: Analysis and future work. *Sensors (Switzerland)*, 17(3), 1–17.
- [27] Nick, T., Götze, J., John, W., & Stönnner, G. (2011). Localization of passive UHF RFID labels using an unscented Kalman filter with relative position information. *RFID SysTech 2011; 7th European Workshop on Smart Objects: Systems, Technologies and Applications*; 1–7.
- [28] Vetrella, A. R., & Fasano, G. (2016). Cooperative UAV navigation under nominal GPS coverage and in GPS-challenging environments. *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow, RTSI 2016*.
- [29] Clift, L. G., & Clark, A. F. (2015). Determining positions and distances using collaborative robots. *2015 7th Computer Science and Electronic Engineering Conference, CEEC 2015 - Conference Proceedings*, 189–194.
- [30] Medina, C., Cuellar, S., & Mojica, P. (2015). Drones. *Superintendencia de Industria Y Comercio Colombia*.
- [31] Sánchez, G., Valenzuela, M. M., & Cadavid, E. S. (2013). Vehículos aéreos no tripulados en Latinoamérica. España, 85. Retrieved from <http://www.infodefensa.com/servicios/publicaciones/empresa-informe-infodefensa-vehiculos-aereos-no-tripulados-en-latinoamerica.html>
- [32] Australian UAV. (n.d.). Types of Drones: Multi-Rotor vs Fixed-Wing vs Single Rotor- Australian UAV. Retrieved May 29, 2017, from <https://www.auav.com.au/articles/drone-types/>
- [33] VTOL Technologies Ltd. (n.d.). VTOL Technologies | RPAS | BVLOS. Retrieved May 29, 2017, from <http://www.vtol-technologies.com/>
- [34] EcuRed. (n.d.). Vehículo aéreo no tripulado. Retrieved May 29, 2017, from https://www.ecured.cu/Vehículo_aéreo_no_tripulado
- [35] US Department of Commerce, N. O. and A. A. What is LIDAR. Retrieved from <http://oceanservice.noaa.gov/facts/lidar.html>
- [36] Bluesky International Limited. (n.d.). LiDAR UK. Retrieved May 29, 2017, from <http://www.lidar-uk.com/how-lidar-works/>
- [37] MathWorks. 3-D Point Cloud Processing - MATLAB & Simulink. Retrieved May 29, 2017, from <https://www.mathworks.com/help/vision/3-d-point-cloud-processing.html>
- [38] Symmetry ELECTRONICS. GNSS and GPS. Retrieved May 29, 2017, from <http://www.semiconductorstore.com/blog/2015/What-is-the-Difference-Between-GNSS-and-GPS/1550>
- [39] Novatel. Real-Time Kinematic (RTK) | An Introduction to GNSS | NovAtel. Retrieved May 30, 2017, from <https://www.novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/real-time-kinematic-rtk/>
- [40] Rational Unified Process (RUP). Retrieved from <http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesRUP.pdf>
- [41] Bacca, E. Introduction to Software Development. Universidad Del Valle.
- [42] Spectra. (2016). Receptor GNSS SP60. Retrieved from <http://www.spectraprecision.com/media/custom/upload/File-1478170077.pdf>
- [43] Cang Ye, & Borenstein, J. (2002). Characterization of a 2D laser scanner for mobile robot obstacle negotiation. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 3(May), 2512–2518.
- [44] Wen Yu, Erik Zamora, Alberto Soria, "Ellipsoid SLAM: A Novel Set Membership Method for Simultaneous Localization and Mapping," *Autonomous Robots (Impact Factor: 1.75)*, 40(1):125-137, 2016.
- [45] "RESCATE Y EXPLORACIÓN CON UNA FORMACIÓN DE MULTI-AGENTES", C. I. 2915, Universidad del Valle.
- [46] MT2216 brushless motors. Retrieved from: http://rc-planeta.pl/1223-thickbox_default/emax-mt2216-cw-810kv-228w-zestaw-1045.jpg
- [47] Arduino DUE. Retrieved from: <https://store.arduino.cc/usa/arduino-due>
- [48] Stukjunger, B. P. (2013). Design tip Benefits in using FIFO buffer embedded in ST MEMS sensors, (December), 1–7.
- [49] Jan Latecki, L., Lakaemper, R., Sun, X., & Wolter, D. (n.d.). Building Polygonal Maps from Laser Range Data. Retrieved from <https://pdfs.semanticscholar.org>
- [50] M. G. Ocando, N. Certad, S. Alvarado and Á. Terrones, "Autonomous 2D SLAM and 3D mapping of an environment using a single 2D LIDAR and ROS," *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), Curitiba, 2017*, pp. 1-6.