

## Article

# Sensor Information Sharing Using a Producer-Consumer Algorithm on Small Vehicles

Rodrigo Vazquez-Lopez <sup>1</sup>, Juan Carlos Herrera-Lozada <sup>1</sup>, Jacobo Sandoval-Gutierrez <sup>2,\*</sup>,  
Philipp von Bülow <sup>2</sup> and Daniel Librado Martinez-Vazquez <sup>2</sup>

<sup>1</sup> Instituto Politécnico Nacional, Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC), Unidad Profesional Adolfo López Mateos, Ciudad de México 07700, Mexico; rvazquezl1800@alumno.ipn.mx (R.V.-L.); jlozada@ipn.mx (J.C.H.-L.)

<sup>2</sup> Departamento de Procesos Productivos, Universidad Autónoma Metropolitana Unidad Lerma, Estado de México 52005, Mexico; p.von@correo.ler.uam.mx (P.v.B.); d.martinez@correo.ler.uam.mx (D.L.M.-V.)

\* Correspondence: j.sandoval@correo.ler.uam.mx; Tel.: +52-728-282-7002 (ext. 3011)

**Abstract:** There are several tools, frameworks, and algorithms to solve information sharing from multiple tasks and robots. Some applications such as ROS, Kafka, and MAVLink cover most problems when using operating systems. However, they cannot be used for particular problems that demand optimization of resources. Therefore, the objective was to design a solution to fit the resources of small vehicles. The methodology consisted of defining the group of vehicles with low performance or are not compatible with high-level known applications; design a reduced, modular, and compatible architecture; design a producer-consumer algorithm that adjusts to the simultaneous localization and communication of multiple vehicles with UWB sensors; validate the operation with an interception task. The results showed the feasibility of performing architecture for embedded systems compatible with other applications managing information through the proposed algorithm allowed to complete the interception task between two vehicles. Another result was to determine the system's efficiency by scaling the memory size and comparing its performance. The work's contributions show the areas of opportunity to develop architectures focusing on the optimization of robot resources and complement existing ones.

**Keywords:** absolute position system; cooperative algorithm; intercepting vehicles; indoor positioning; robot framework; UWB sensors



**Citation:** Vazquez-Lopez, R.; Herrera-Lozada, J.C.; Sandoval-Gutierrez, J.; von Bülow, P.; Martinez-Vazquez, D.L. Sensors Information Sharing Using a Producer-Consumer Algorithm on Small Vehicles. *Sensors* **2021**, *21*, 3022. <https://doi.org/10.3390/s21093022>

Academic Editor: Simon Tomažič

Received: 23 February 2021

Accepted: 21 April 2021

Published: 25 April 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The use of an operating system has made it possible to standardize the technologies for the problem of cooperativity between several robots or vehicles. To this end, the devices, the sensors, the communication modules, the protocols, and the programming languages are designed to be compatible with each other. This conception of unified platforms such as ROS, Kafka, AEROSTACK, MAVLink among others, resolves several hardware problems and allows a more user-friendly environment for software developers [1–4]. However, high compatibility leads to a significant increase in resource usage. Another consequence is to lose the diversity of solutions that are the result of particular case investigations.

In this research, a case study shows the importance of maintaining research with custom designs than generic designs. A basic platform scenario requires resources at three fundamental layers. The perception layer (indoor systems) uses a GoPro-type internal camera [5,6] or a VICON type external camera sophisticated system [7,8]. The communication layer involves the use of transmitters, routers, and standardized protocols such as WiFi, Zigbee, or Bluetooth [9]. The third layer of applications involves the use of a compatible information management system, i.e., master and slave workstations using a computer architecture with an operating system such as Ubuntu or similar. The above implies a

high energy consumption in managing information compared to the energy required for actuators who achieve vehicles' displacement. However, it is essential to consider the advantages of these solutions. For example, end-users, developers, and interested people can program solutions in a more friendly way.

In another perspective, there are tailor-made solutions so specific, complex, and customized that they require specialized equipment and are not compatible with other systems. Therefore, the proposal is based on a methodology that combines customized solutions with generic ones to be implemented in low-performance vehicles or incompatible with operating systems. For this, a reduction or replacement of tools is proposed without losing compatibility with high-level technologies. Given the characteristics of this architecture, algorithms based on hardware or technology constraints should be implemented. Finally, the actual functionality should be validated through a known task and where the optimization of resources can be compared.

A significant difference between the design of a generic and a customized platform lies in taking advantage of the sensor's characteristics. In particular, UWB sensors allow sending messages and obtaining the position by triangulation. Consequently, known tools cannot be applied directly, and an intermediate algorithm has to be used. The libraries of programming languages based on operating systems are very advanced compared to embedded systems. It is important to remember that the migration of a tool or algorithm of custom technology to a general one is always possible. However, in the opposite direction, it is not always guaranteed. Therefore, it is essential not to miss the opportunity to investigate code optimization for embedded systems.

### 1.1. Contribution

The design and implementation of an alternative architecture for small vehicles is the main contribution of this work. Simultaneous localization and sharing information with a UWB technology reduced computing resources and consumption energy. A suitable producer–consumer algorithm gives a solution to manage the concurrent processes in reduced-scale hardware. Other contributions are listed as follow:

- Enable incompatible hardware with high-level architectures.
- Manage information packets between embedded systems.
- Achieve an interception task among vehicles avoiding the use of additional hardware or complex infrastructure.
- Make compatible an architecture that uses operating systems with one that does not.
- Adapt to the confinement conditions when performing experiments at home.
- The small vehicle platform allows test different autonomous navigation strategies, lowering the risk and cost of large-scale testing.

### 1.2. Organization

The article is organized as follows. Section 2 presents a brief review of the state-of-the-art in robotic systems architectures. The customized architecture for small vehicles and its underlying concepts are presented in Section 3. The experiments, results and discussions are developed during Section 4. Conclusions and future Work can finally be found in Section 5.

## 2. Background and Related Work

### 2.1. Robotic System Architectures

Software-oriented solutions in multi-vehicle tasks have consistently been standardized on architecture with at least three layers with particular purposes. These layers' functions are the quantification of physical variables, the communication between sensors and processing devices, and the applications that solve the tasks [10].

This architecture seeks to centralize the sensors' information in an integrated manner on the devices running the applications. Sophisticated communication systems are used to guarantee the exchange of information. The processing and storage devices use operating

systems to program the solutions to the tasks. The above clearly shows that the software-oriented perspective allows for more excellent compatibility. However, poor performance or incompatibility can occur with specific problems.

#### 2.1.1. Localization Sensors

The Inertial Navigation Systems (INS) are fed with data from their Inertial Measurement Unit (IMU), composed mainly of sensors such as gyroscopes and accelerometers. INS are cost-effective solutions that do not require additional infrastructure. The main disadvantage of these systems lies in the cumulative position error that grows during the operation of the vehicle [11]. However, the combination of the information from an INS with a global positioning system allows to improve the accuracy of the positioning estimates [12].

A variety of technologies applied to indoor global positioning systems have been sought. The results in [13] showed that accuracy depends on the implementation and methodology used. In particular, the technologies, which were applied in the development of autonomous vehicles, are global positioning systems by vision such as VICON [7,8], or Optitrack [14,15], Virtual Reality (VR) systems [16,17] and RF technology [18–20]. Meanwhile, the use of Ultra-Wide Band (UWB), part of the RF technologies, attracted interest during the last years [21], because with this technology it is also possible to obtain a good performance in terms of accuracy [22].

Typical tasks that autonomous vehicles must fulfill depend on the quality of the information about the vehicle's absolute and relative position within its workspaces, such as Simultaneous Localization and Mapping (SLAM), obstacle avoidance, trajectory tracking, and cooperation. The integration of absolute and relative positioning systems is, therefore, necessary. The most recent research efforts present various technologies to achieve a good precision in detecting the global and relative position and thus perform a satisfactory SLAM. On the other hand, UWB and vision systems have been combined to improve the estimation of the position in the workspace [23]. Other strategies bet on the fusion of UWB data with those coming from INS [24], and Optitrack [25]. Additionally, it has been tried to improve UWB systems' results on a platform of mobile robots using Gaussian processes [26] and neural networks [27].

#### 2.1.2. Communication Protocols

Indoor Wireless Communications are technologies that allow devices to be uniquely identified and exchanged over a limited distance of 10 m to 100 m [9]. There are several criteria for selecting Bluetooth, UWB, ZigBee, or Wi-Fi technology. The most commonly used technologies are Bluetooth or WI-FI because most robots have it by default. ZigBee is identified as an alternative for users looking to work with a more significant number of devices, a low transfer rate, and requiring less energy consumption. Finally, UWB technologies are less present as a combination of high transfer rate and low normalized energy consumption.

#### 2.1.3. Applications

Applications such as ROS, Kafka, and MAVLink are versatile tools to incorporate most devices, infrastructures, programming languages, and technologies. For example, ROS is defined as a robotic middleware to help manage the complexity and heterogeneity inherent in distributed systems [1]. Apache Kafka is a distributed messaging system widely used in big data applications [2] and MAVLink is a communication protocol used for the bidirectional communications between drones and ground stations over a wireless channel [4]. In this sense, by having approved research platforms, it is possible to leave aside hardware problems and focus on algorithms' performance.

There are different examples of applications that use ROS in its architecture as in [28–30]. In [31] Apache Kafka was used to parallelize a set learning architecture. In contrast, in [29], small drones are used to collect data to test obstacle avoidance algorithms.

Finally, MAVLink being a communication protocol, can be found implemented in several tools such as ROS. In [32] MAVLink was used with ROS to communicate with the pixhawk drone and perform autonomous flights. In [33], they used the protocol to communicate with real drones to validate the performance of the proposed architecture.

The element relationships of the Robotic System Architectures with the software developed are shown in Table 1.

**Table 1.** Comparison between similar applications and tools.

Software (Tool)	Computer Architecture	Supported System	Comm. Devices	Infrastructure
ROS [28]	x64 ARM	GNU/Linux	Wi-Fi	Network Camera FCU
Kafka, ROS [29]	ARM	GNU/Linux	4G	Network VPN Cloud Service Cameras
MAVLink [34]	ARM	Android 4.3	Zigbee	X-bee Modules Tablet Camera
Proposed architecture	x64... STM, EV3	Linux... $\mu$ C Based	Wi-Fi	Network UWB

Table 1 shows that the proposed architecture incorporates non-compliant devices such as EV3 devices and UWB technology as localization, communication, and infrastructure devices.

Some architectures propose using cloud services to improve the computational capacity and intelligence in vehicles such as AGVs [35]. Some examples suggest using architectures based on the edge-fog model as seen in [30]. The authors used the publish-subscribe model of ROS to perform 3D scenario reconstruction through SLAM and proof of concept presented in [36]. In the case AVG [37] the authors give an application-oriented to AGV collaboration, while in [38] they use ROS. Other similar examples can be seen in [39–41].

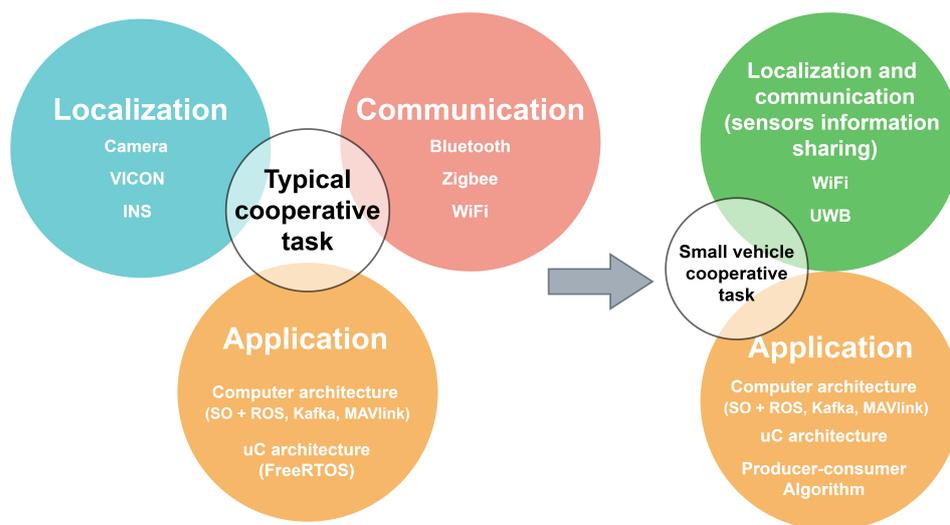
#### 2.1.4. Cooperative Tasks

Cooperative tasks have been investigated in various scenarios with software-oriented architectures [42,43]. Information sharing using algorithms among aerial or ground vehicles has been studied. These algorithms are proposed to solve the path planning problem [44], for the surveillance of multiple moving ground targets [45] or intercepting intruders [46]. Tools of general-purpose have proposed the simulation of these algorithms. However, real implementation of the hardware of these algorithms requires a particular adaptation. The adaptation to solve this problem is shown in Section 3.

### 3. Customized Architecture for Small Vehicles

The previous section showed a typical software-oriented architecture with the most common technologies. In this section, an alternative hardware-oriented architecture will be developed. Figure 1 shown a general comparison of layers and technologies employed.

The small vehicle platform allows test different autonomous navigation strategies, lowering the risk and cost of large-scale testing. The platform enables destructive testing at a meager cost, infrastructure and energy consumption reduced.



**Figure 1.** Comparison of software-oriented architecture (left-side) and proposed architecture (right-side).

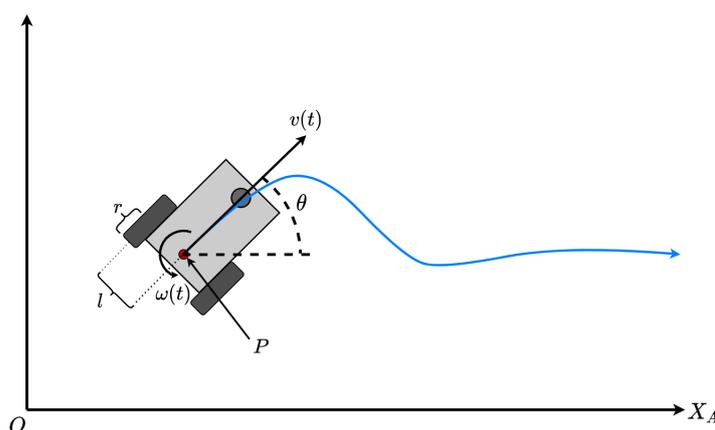
### 3.1. Small Vehicles Models

#### 3.1.1. Ground Robot

We used a robot as a mobile reference point within the workspace. The robot with 3-DOF  $(x, y, \theta)$  is of a differential configuration, so it has two independent, active wheels and a fixed wheel as a support point. The odometry of the vehicle uses encoders and a gyroscope as a relative reference system. The vehicle’s operation is executed intrinsically in the controller using the dynamic model proposed in [47].

Figure 2 shows the robot with its differential system variables as it travels along a given path in an absolute coordinate system  $X_A Y_A$ . The mobile is composed of two wheels with diameter  $\phi$ , each placed at a distance  $L$  from the intermediate point  $P$ . The velocities of the wheels are denoted by  $v_d$  (dextram = right) and  $v_s$  (sinestram = left) respectively, so that the system inputs are:

$$[L, \phi, \theta, v_s, v_d] \in \mathbb{R}^5. \tag{1}$$



**Figure 2.** Ground vehicle model.

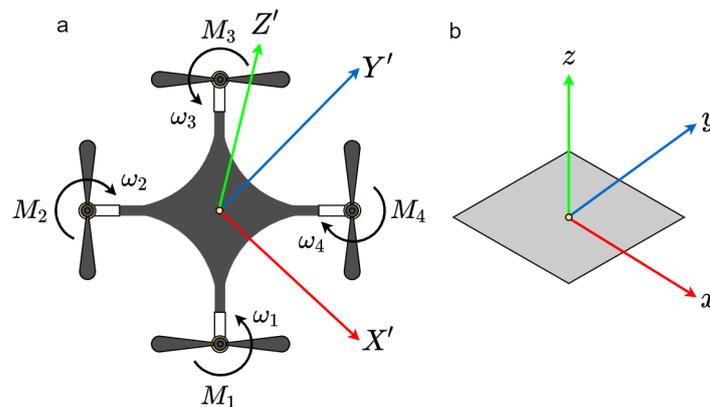
According to the model of direct kinematics presented in [47], the robot motion is given by the following matrix equation in (2).

$$\xi_A = \begin{bmatrix} (\omega_d + \omega_s) \cos \theta \\ (\omega_d + \omega_s) \sin \theta \\ \omega_d - \omega_s \end{bmatrix}, \tag{2}$$

### 3.1.2. Aerial Vehicle

A 6-DOF nano-quadcopter ( $x, y, z$ , pitch, roll, yaw) to follow the ground reference robot's trajectory was used. The internal odometry consisted of INS, whose IMU integrates an accelerometer, gyroscope (in three axes) and a barometric pressure sensor. Additionally, the INS data are merged with the height estimation obtained employing an optical flow sensor and the absolute positioning system's data. The operation of the vehicle is done in an autonomous, semi-autonomous way or by using teleoperation.

To operate the vehicle, we assumed the 12-state model, studied in [48,49] and validated in [50]. According to Figure 3, the model requires two reference systems: the non-inertial one  $X'Y'Z'$  located at the center of gravity of the quadcopter, orientated as shown, and the inertial one  $x, y$ , and  $z$  relative to the center of the earth. Of the 12 parameters, the variables  $x, y$  and  $z$  represent the coordinates of the center of gravity,  $u, v$  and  $w$  are the linear velocities along the  $X', Y'$  and  $Z'$  axes, and  $\psi, \theta$  and  $\phi$  represent the rotation angles (pitch, roll and yaw) with the corresponding angular velocities. All variables are expressed within the inertial reference system. The model is non-linear and is described in [50,51].



**Figure 3.** Nano-quadcopter. (a) Body frame. (b) Inertial frame.

### 3.2. Sensor Information Sharing

A particular research objective was proposed a 3D global positioning platform oriented to devices with limited hardware. The low-cost UWB sensors were used to perform simultaneous localization and information exchange between the small vehicles.

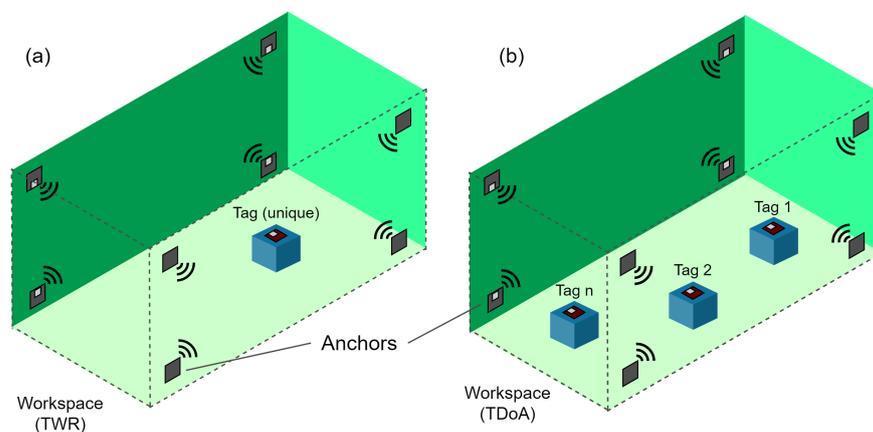
The system consists of anchors and tags that communicate with UWB pulses. The anchors within the workspace serve as absolute reference points for the tags. Likewise, the tags are used as a mobile reference point to triangulate the anchors' signals and, simultaneously, as a communication interface to other external devices. Two-Way Ranging (TWR) algorithms are used to detect a single tag, and a Time Difference of Arrival (TDoA) algorithm is used to detect the position of one or more tags inside the same workspace. The diagram of the operation is shown in Figure 4.

### 3.3. Producer-Consumer Algorithm

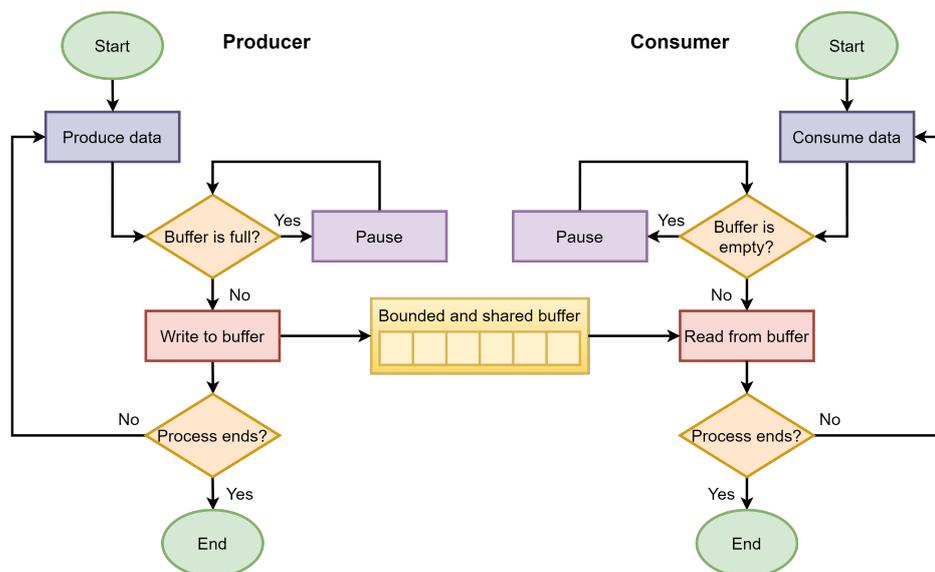
A reduced algorithm based on an analogy to the producer–consumer problem was another particular research objective. This consideration helps to perform tasks to share information with two or more complex sensors to localize small vehicles. Furthermore, this algorithm is reduced to achieve accessible communication among multiple robots and an extra path planning task was executed at the same time.

Collaboration in navigation among autonomous robots represents a problem of synchronization between the transmission and reception of positioning and movement data. In this problem, there appear two parallel processes (producer and consumer), which exchange information by a finite memory buffer [52]. Here the producer is in charge of generating and inserting data into the buffer, trying not to saturate it. Otherwise, the

consumer extracts the data from the buffer one by one, preventing the buffer from emptying. Applications of this problem are commonly in concurrency, and message passing, although trivial, is extremely useful when needed [53]. Figure 5 shows the flowchart of the consumer–producer problem.



**Figure 4.** Different configurations of the absolute positioning system mounted within a workspace: (a) existence of only one tag: a location algorithm based on “two way ranging” TWR is used. (b) Two or more tags are present: a location algorithm based on “time difference of arrival” TDoA is used.



**Figure 5.** Flowchart of the producer–consumer problem.

The producer–consumer problem is somehow analogous to the problem of cooperative follow-up of a trajectory. In said problem, one or more dependent robots follow the trajectory generated by a master robot until finally coinciding in (almost) the same point. The master robot draws the trajectory in the workspace, so it operates as a producer of information on points. On the other hand, the dependent robots receive their information on the trajectory needed to get to the master, beginning from their starting points. It is, though, possible to view the dependent robots as consumers of information.

The FIFO (First In-First Out) buffer can be modelled as an ordered set  $D$  of  $q_i$  information elements. The state of this kind of memory is determined through the sequence  $\{q_0, q_1, \dots, q_m\}$ , ordered by the moment insertion of its elements by (3) and (4). In this

buffer, a new information element  $q_0$  is always placed at the end of its queue by the insertion instruction of an  $I(q_0, D)$ :

$$I(q_0, D) = q_0 \cup D. \quad (3)$$

The operation raises all elements' position in the buffer by one place and increases, consequently, the cardinality by one unit:  $|I(q_0, D)| = |D| + 1$ . On the other hand, the instruction of an extraction procedure  $E(D)$ , removes, when executed, the first element  $q_m$  from the queue of the  $m$  data elements in  $D$ :

$$E(D) = D \setminus q_m. \quad (4)$$

It is clear that this command lowers the cardinality of the set  $D$  by one unit:  $|E(D)| = |D| - 1$ .

For implementation of the algorithm in the robots, we used the following general considerations. Let  $W \subset \mathbb{R}^3$  be the workspace and  $R$  the set of positions of the  $i$  robots, with  $i \in \mathbb{Z}^+, i = 1, \dots, n$  and  $n > 1$ . The value of the instantaneous position at time  $t$  of each robot is represented by  $R_i(t) = (x_i(t), y_i(t), z_i(t))$ . The trajectory  ${}_{t_0}^{t_f}R_i$  of the robot  $i$  between the starting point  $R_i(t_0)$  and the endpoint  $R_i(t_f)$  are characterized by a sequence of positions  $R_i(t_j)$  which corresponds to the equidistant sequence of sampling moments  $t_j = t_0 + j \cdot \Delta t$ , within the interval between the initial  $t_0$  and final time  $t_f$ , separated by a sampling interval  $\Delta t$  between each moment, such that:  ${}_{t_0}^{t_f}R_i = \{R_i(t)\} \forall t \in [t_0, \dots, t_f]$ . To the master robot we assign, as identifier, the number 1. In this way the path of this robot is symbolized with  ${}_{t_0}^{t_f}R_1$ . The elements of the trajectory are then successively in  $D_1$  loaded elements with the insertion command:

$$I(R_1(t_j), D_0). \quad (5)$$

The follower robots  $R_s$ , with  $s \in \{2, 3, \dots, n\}$ , reconstruct the path by successively extracting the leading elements in the  $D_1$  buffer:

$$R_s \leftarrow (D_1). \quad (6)$$

The Algorithm 1 for the case of the master robot (producer) can be summarized as follows:

---

#### Algorithm 1 Producer algorithm

---

- 1: **procedure** PRODUCER( $D_1, trajectory$ )
  - 2:   Initialize  $|D_1| = 0$  (empty)
  - 3:    $R_M$  starts *trajectory* at  $R_1(t_0)$
  - 4:   **while** *trajectory* is not finish **do**
  - 5:      $R_M$  inserts its position in memory  $D_1$
  - 6:     Delay  $\Delta t$
- 

The Algorithm 2 for the case of the follower robot (consumer) can be summarized as follows:

---

#### Algorithm 2 Consumer algorithm

---

- 1: **procedure** CONSUMER( $R_M, D_1$ )
  - 2:   **if**  $R_M$  starts *trajectory* at  $R_1(t_0)$  **then**
  - 3:     **while** position  $R_1(t_j)$  is changing or  $D_1$  is not empty **do**
  - 4:       Extract an element from  $D_1$  and send it to robot  $s$ , as its next position  $p_s$
  - 5:       Move to  $p_s(x_s, y_s, z_s)$
-

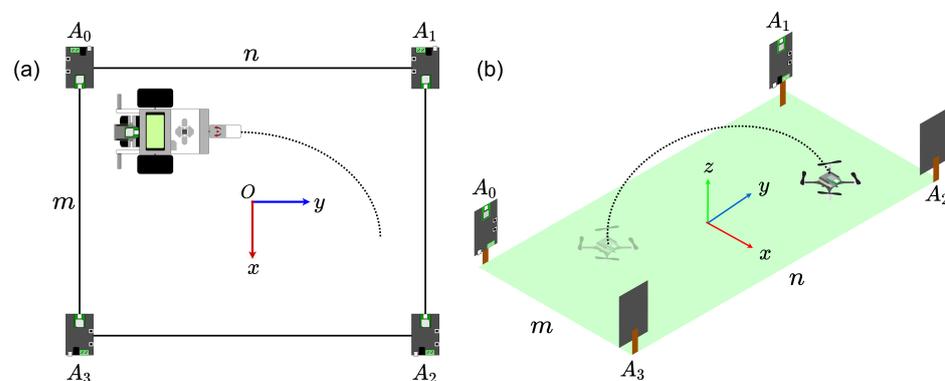
## 4. Experiments, Results and Discussion

### 4.1. Experiments

The experiments performed were divided into two parts: (1) the characterization of the sensor information sharing with small vehicles and (2) intercepting and landing of the aerial vehicle on the ground vehicle with shared information using the producer–consumer algorithm.

#### 4.1.1. Experiment 1: Characterization

This experiment aimed to characterize the absolute positioning system and, after that, estimate the tracking accuracy in 2D and 3D. The system was placed within a controlled environment, and functionality tests were carried out within the fixed position's polygon, defined by the anchors. The 2D tracking tests consisted of following prior defined trajectories using the ground vehicle. The corresponding robot was tagged to know in real-time its position in the  $xy$  plane, left side of Figure 6. In the  $xyz$  space, we used a nano quadcopter with a tag integrated, as shown on the right side of Figure 6. The data obtained was used to reconstruct the trajectory and to know the accuracy of the system.



**Figure 6.** Diagram of the characterization experiment. (a) In 2D. (b) In 3D.

We used a Loco Positioning System (LPS), which employs a UWB DMW1000 module from Decaway and whose precision is estimated by the manufacturer to be within  $\pm 0.1$  m. The LPS is composed of a Loco Positioning Node (anchors) and a Loco Positioning Deck (tags). For triangulation of the signal, the system supports configurations with 4, 6 and 8 anchors and uses the TWR algorithm (a single tag) and TDoA in the existence of two or more tags. A minimum of four anchors must triangulate the signal within a 3D space, so we selected this configuration during our experiments. Four anchors (numbered from 0 to 3) were placed at a distance 0.16 m above the floor, forming a rectangle with dimensions  $m$  by  $n$ . We decided to place the absolute reference system's origin in the center of this  $m \times n$  rectangle. Table 2 shows the values of  $m$ ,  $n$  and the coordinates of each anchor.

**Table 2.** Positions of the anchors employed to characterize the positioning system.

Size		Anchor's Coordinates ( $x, y, z$ )			
m [m]	n [m]	Anchor 0 [m]	Anchor 1 [m]	Anchor 2 [m]	Anchor 3 [m]
1.86	5.0	(−0.93, −2.50, 0)	(0.93, 2.50, 0)	(−0.93, 2.50, 0)	(0.93, −2.50, 0)
1.90	5.0	(−0.95, −2.50, 0)	(0.95, 2.50, 0)	(−0.95, 2.50, 0)	(0.95, −2.50, 0)
5.20	1.9	(−2.50, −0.95, 0)	(2.50, 0.95, 0)	(−2.50, 0.95, 0)	(2.50, −0.95, 0)
2.00	2.0	(−1.00, −1.00, 0)	(1.00, 1.00, 0)	(−1.00, 1.00, 0)	(1.00, −1.00, 0)

A LEGO EV3 ground robot was used in its differential configuration. We employed a gyroscope with an accuracy of  $\pm 3$  degrees and an optical encoder with an accuracy of  $\pm 1^\circ$  to measure the wheels' rotation. The controller of the EV3 ground robot contains an ARM 9 processor with 64 MB of RAM and an SD card reader on which the GNU/Linux-based EV3dev Operating System (OS) was installed. The trajectories and odometry of the EV3

land mobile as described in [54] were implemented using Python as the programming language and the EV3dev OS libraries.

The airborne vehicle used is a Crazyflie 2.1 quadcopter, which has as main controller an STM32F405 and uses the nRF51822 for communication with the Crazyradio-PA telemetry system. The architecture of the Crazyflie quadcopter allows for the stacking of accessories, known as decks. We used the Flowdeck V2 as an optical flow sensor and the LPS Deck as a tag, compatible with LPS. The Crazyflie quadcopter's architecture is open to implement tools and algorithms made by the community of developers. Crazyflie quadcopter uses the Kalman Extended Filter, as proposed by [55,56] for the fusion of data of the IMU, the Flowdeck V2 and the LPS. The implemented control algorithms are based on [57,58].

For the individual tests, we worked with the TWR mode. We assigned tag 1 to the Crazyflie quadcopter and tag 2 to the LEGO EV3 robot. As a test to determine the UWB triangulation system's accuracy, we implemented a desired trajectory in the workspace for both robots. A computer was interfaced with a Crazyradio-PA to acquire the position data of each tag. For this purpose, a circle of 0.61 m radius and different heights, 0.02 m in the EV3 robot and 0.5 m in the Crazyflie quadcopter.

#### 4.1.2. Experiment 2: Intercepting and Landing Task

The experiment's objective was to validate the operation of the proposed consumer-producer algorithm when solving the combined task of intercepting and landing, where the Crazyflie 2.1 quadcopter must intercept the LEGO EV3 lander and land on top of it. The initial position of the EV3 robot was  $x = 0$  m,  $y = 0.66$  m and  $z = 0.02$  m, and its given path was an arc with a radius of 0.5 m and an arc length of also 0.5 m. The initial position of the Crazyflie quadcopter was at  $x = 0$  m,  $y = 0$  m and  $z = 0$  m, and during the trajectory tracking phase, the quadcopter was asked to maintain a height of 0.5 m.

Figure 7 shows the flowchart of both the producer and consumer algorithms implemented on a computer to manage the dataflow of the EV3 robot, the Crazyflie quadcopter and the TDoA mode of the LPS. In this case, process A (producer) is the EV3 robot, and process B (consumer) is the Crazyflie quadcopter. The process starts when the computer has verified the communication between the EV3 robot and the Crazyflie quadcopter. The computer sends then to the EV3 robot the information about the trajectory to be followed. When the EV3 robot starts to move, it sends its position data to the computer. In parallel to the EV3 robot following the demanded path, the computer starts the producer-consumer algorithm and instructs the Crazyflie quadcopter to start its takeoff. When the Crazyflie quadcopter reaches the reference height of 0.50 m, it starts receiving the buffer position data of the EV3 robot from the computer.

The EV3 robot sends its position data continuously and independently to the  $D_1$  buffer in the computer until it reaches the endpoint of the given trajectory. The Crazyflie quadcopter consumes the position data added by the EV3 robot in the buffer  $D_1$  and computes its movements by comparing its position with the buffer's positions. At the moment, the Crazyflie quadcopter does not encounter any more position data in the buffer  $D_1$  it starts a landing maneuver and touches down on top of the EV3 robot.

The details about the algorithm's implementation can be viewed in the block diagram in Figure 8. It illustrates the relationship of the tools used between the computer, the EV3 robot and the Crazyflie quadcopter during the algorithm's execution. The white arrows indicate the internal communication flow, while the dotted arrows represent external communication flow.

The algorithm was implemented on a Python script using multi-threaded programming and executed on a computer with a Core i7 processor with 8 GB of RAM. The Python script sends orders over the local network to the EV3 robot using the MQTT protocol through an MQTT broker installed in the EV3dev OS in the EV3 robot. MQTT was used because it is ideal for machine-to-machine (M2M) communication due to this protocol's lightness. On the other hand, the EV3 robot was connected with a wireless network card.

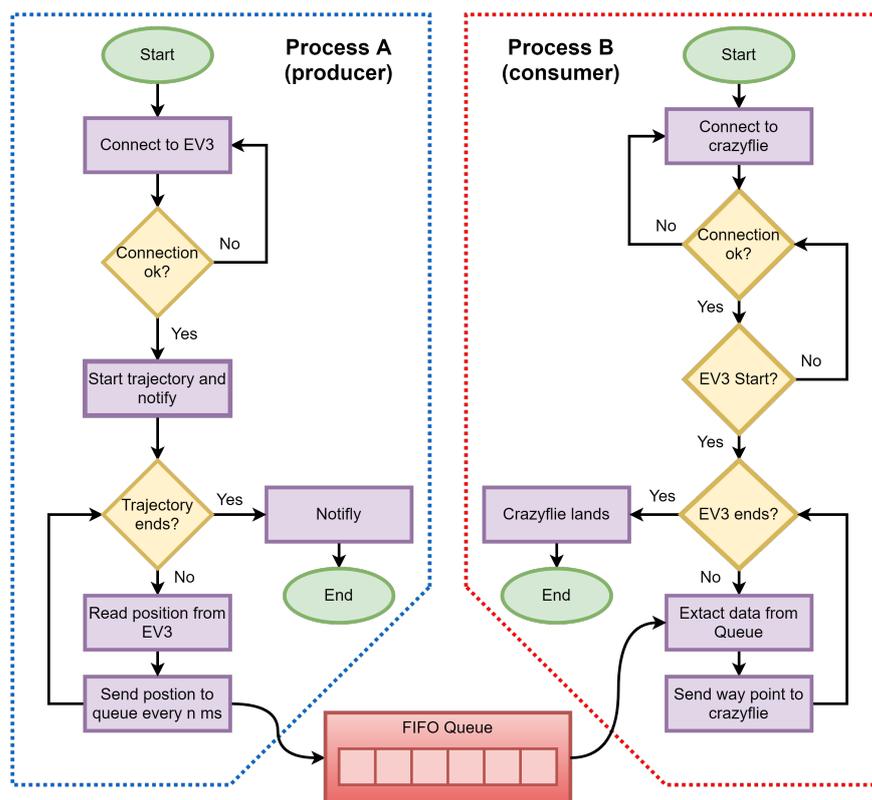


Figure 7. Flowchart of the process A (producer) implemented for the mobile robot EV3 and of the process B (consumer) implemented for the Crazyflie quadcopter.

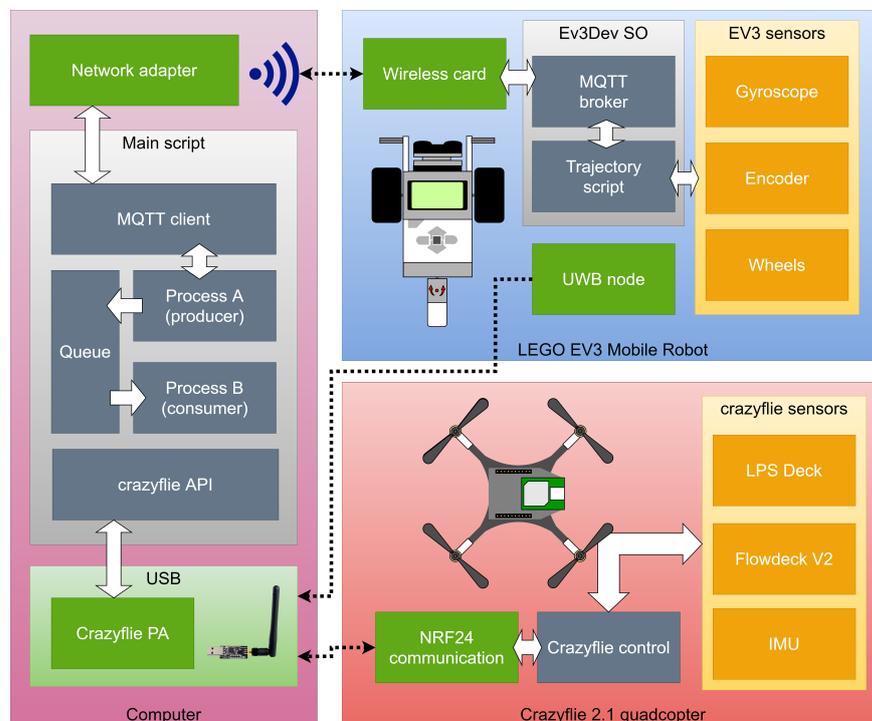


Figure 8. Block diagram illustrating the interaction between the components during the execution of the algorithm.

The communication between the UWB tags and the computer was done using the Crazyradio-PA telemetry system, which communicates wirelessly, using the NRF24 protocol. One Crazyradio antenna can communicate with up to 8 tags.

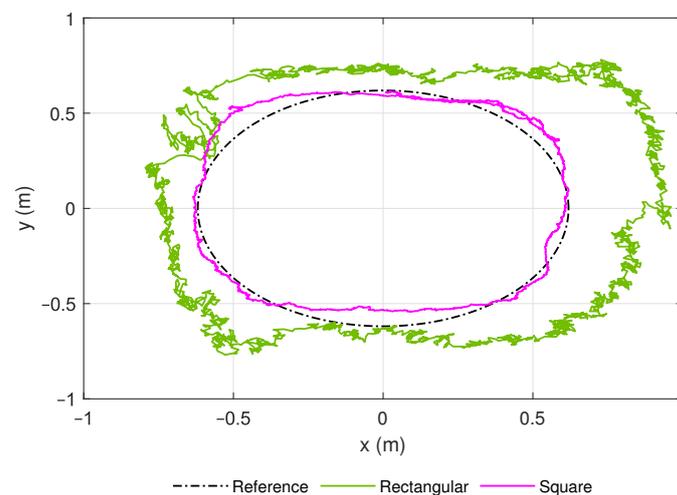
#### 4.1.3. Experiment 3: Performance

This experiment aimed to evaluate the scaling of the system about the size of the consumer's buffer. The proposal consisted of generating a constant amount of information of 1.92 kbps by the producer. The tested trajectory was the same as in the previous experiments with a speed of 0.1 m/s. Buffer size started with 0, increments of 1 and up to 14 times (24 Bytes up to 360 Bytes). The units of measurement used were: the consumption rate in Kbps, the kinetic energy in  $\text{Kg} \cdot \text{m}^2/\text{s}^2$ , and the processing time of the information in the queue with s. The break-even point to determine the system's efficiency was to compare the increase of the energy and the processing time.

An additional test was to eliminate the information sent by the Ev3 robot to destabilize the operation of the quadcopter to 50% of the task.

#### 4.2. Results and Discussion

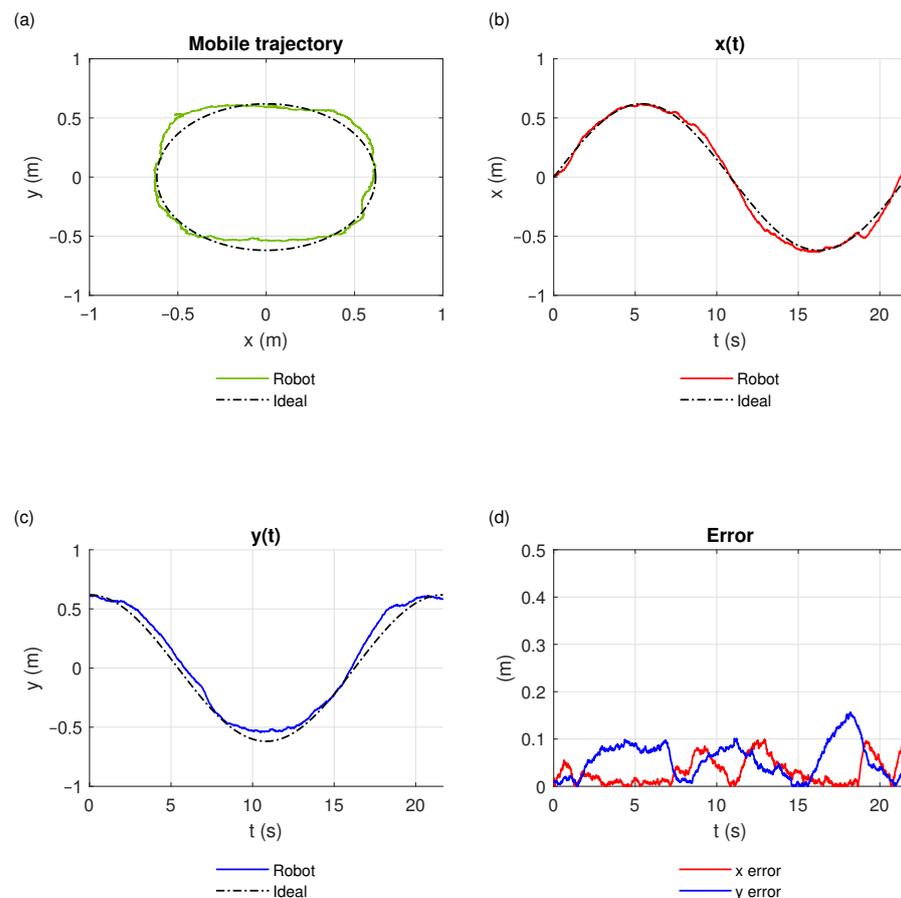
During the characterization process for the anchor positions, as presented in Table 2 of Section 3, it was observed that a square-shaped distribution of the anchors in the workspace leads to a better performance than a rectangular one (Figure 9). Nevertheless, the manufacturer recommends a rectangular layout. The best configuration was a square workspace of 2 m by 2 m and with the anchors at its vertices after trials.



**Figure 9.** Performance of anchors in rectangular (green) and square (pink) configuration concerning a reference path.

Figure 10 shows the results obtained after characterization and tuning of the  $2 \text{ m} \times 2 \text{ m}$  workspace. Figure 10 shows the ideal trajectory (dashed line) and the one performed (green line) by the EV3 robot (green line). We found that the data obtained were consistent; however, when the robot moved near the anchors, the error increased significantly up to 0.1 m. Figure 10b,c show the EV3 robot's displacement on each of the  $x$  and  $y$  coordinate axes. The path carried out by the EV3 robot in the  $x$ -axis is much closer to the given one than in the  $y$ -axis, due to the tag's orientation. However, there are probabilities of perturbations that increase the error during some moments, for example, in  $t = 19.7 \text{ s}$ .

Figure 10 shows the RMS-error calculation of the path in both coordinate axes. In the  $x$ -axis, the error is within a range below 0 to 0.1 m as claimed by the manufacturer of the UWB system; however, in the  $y$ -axis, the maximum error went up to 0.16 m. The average RMS-error in the  $x$ -axis was 0.042 m, again by a factor of 2/3 lower than the RMS-error of 0.062 m in the case of the  $y$ -axis.

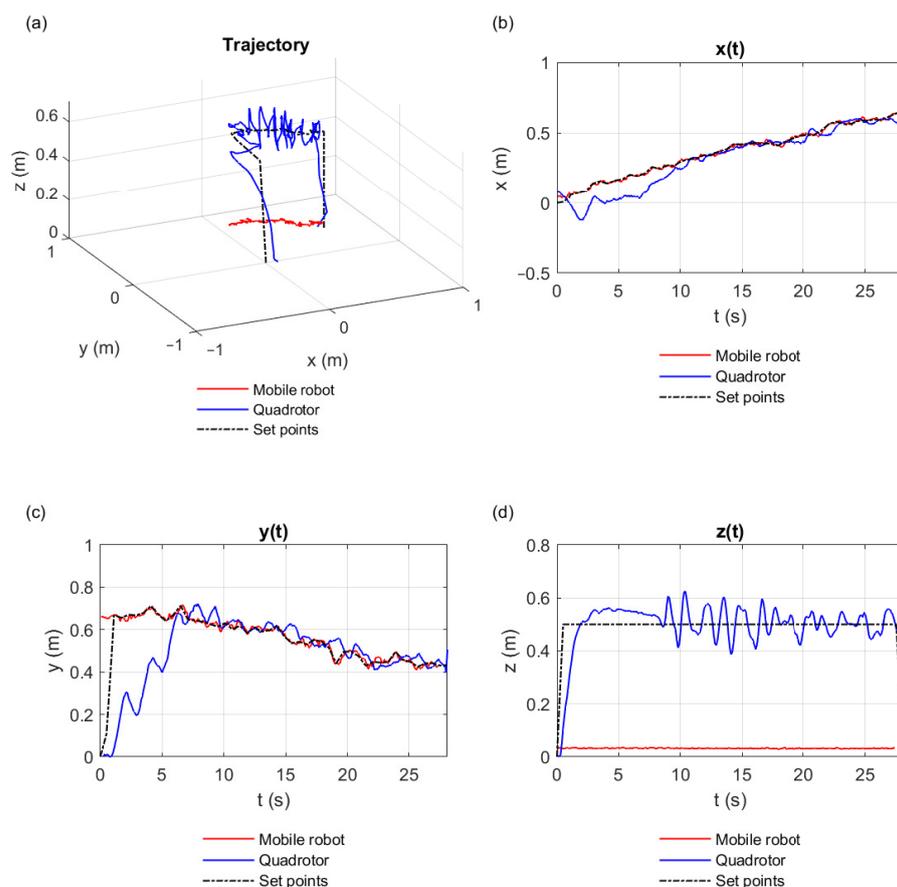


**Figure 10.** Results of the characterization run for a  $2 \times 2$  m workspace. (a) Trajectory performed by the robot. (b)  $x$ -axis component of the movement. (c)  $y$ -axis component of the movement. (d) Quadratic error on  $x$  and  $y$  between the performed path and the given ideal trajectory.

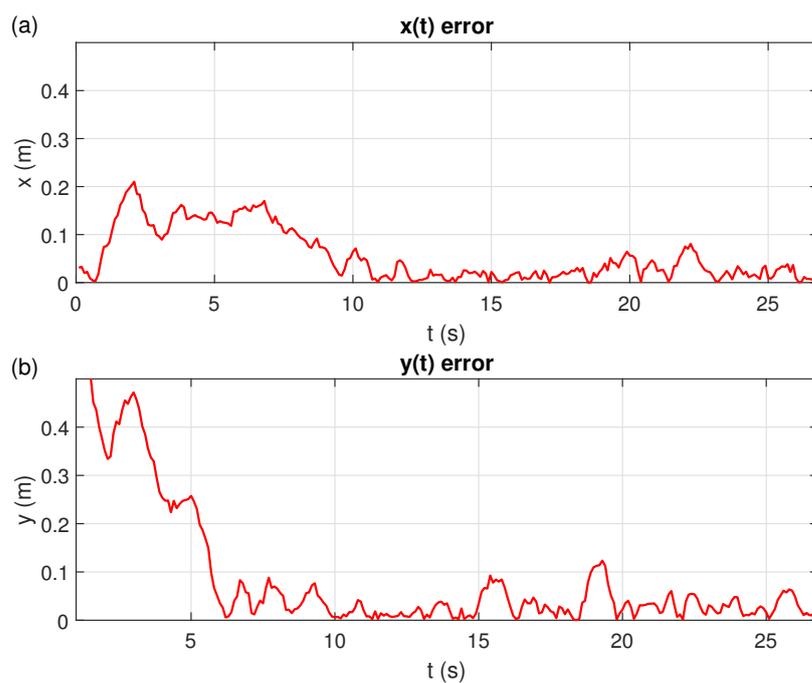
The producer-consumer algorithm always kept the equilibrium of data. For this reason, the Crazyflie quadcopter intercepted the EV3 robot with acceptable performance. Figure 11 illustrates the best result we obtained when executing the algorithm; that is, it represents the sequence of the points  $R_1(t_j)$  that were stored in  $D_1$ 's queue, coming from the EV3 robot, which was later on converted into set points for the Crazyflie quadcopter. Figure 11a shows the trajectories performed by the EV3 robot (red line) and by the Crazyflie quadcopter (blue line) in 3D perspective. The ideal trajectory for the quadcopter is shown by the black dashed line. The projections of the trajectories on the  $x$ ,  $y$ , and  $z$  axes are similarly represented in Figure 11b–d.

Figure 12 shown the RMS-positioning-error of the EV3 robot and the Crazyflie quadcopter from the ideal path. We restricted the evaluation of the EV3 robot's performance to the  $xy$ -plane. In Figure 12a, it is seen that the RMS-error in  $x$  is always smaller than 0.22 m. When  $(t_f - t_0) < 10$  s the error is greater than 0.1 m, for  $10 \text{ s} < (t_f - t_0) < 19$  s you get an average RMS-error of 0.04 m, and in the interval of  $19 \text{ s} < (t_f - t_0) < 23$  s the RMS-error is greater than 0.04 m but always less than 0.1 m, due to the proximity to the anchor three.

Figure 12b shown that the RMS-positioning-error in  $y$  is greater than the one in  $x$ . This error is because the distance between the two robots at the beginning is 0.61 m and is interpreted as an error. The Crazyflie quadcopter manages to reduce its RMS-positioning-error to less than 0.1 m after the first 6 seconds. Once a level of proximity under 0.1 m is reached, the average RMS-error in  $y$  drops down to a level of 0.032 m.

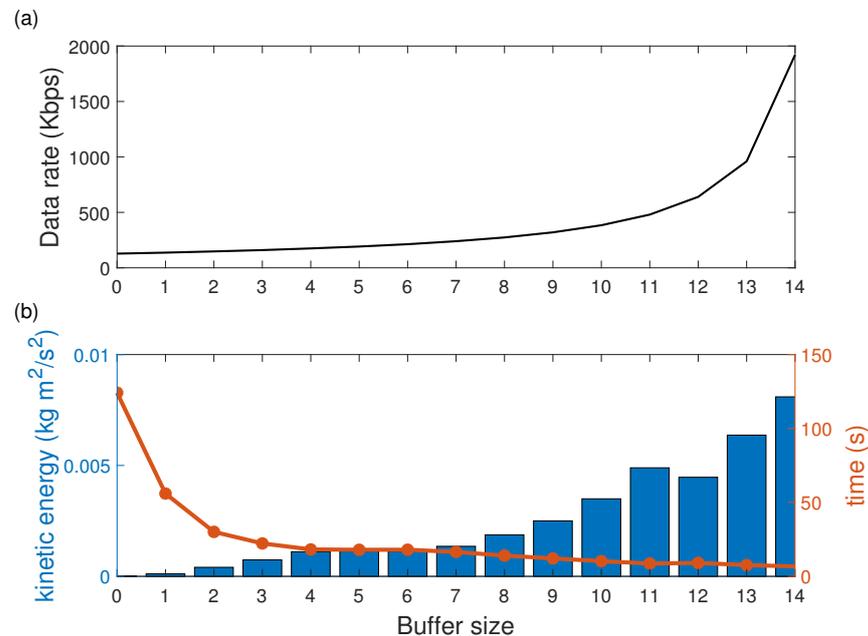


**Figure 11.** The paths taken by the EV3 (blue line), and the Crazyflie quadcopter robot (red line) compared to the ideal path (black dashed line). (a) The trajectory performed in 3D perspective. (b) Displacement in x. (c) Displacement y. (d) Displacement in z.



**Figure 12.** RMS-positioning-error between the EV3 robot and the Crazyflie quadcopter. (a) x-axis error. (b) y-axis error.

Figure 13a shows the performance results of the system about the scalability of the consumer buffer. The exponential behavior of the data rate with three segments can be observed. A moderate slope was observed in the first segment from 0 to 8 times the buffer size, from 8 to 12 a steeper slope and after 12 a sudden increase.



**Figure 13.** Buffer size vs. (a) consumption data rate and (b) kinetic energy and execution time comparison as a system's performance evaluation.

The results have allowed us to verify that the system can be scaled in terms of buffer size to increase the consumption data rate. The limitation of this increase is affected by the response of the quadcopter. Another scenario would be to increase the number of quadcopters sharing the information; in this sense, the rate is divided among the robots cooperating in the same task. The maximum operating limit of the platform is 6.8 Mbps and a packet length of 1023 bytes.

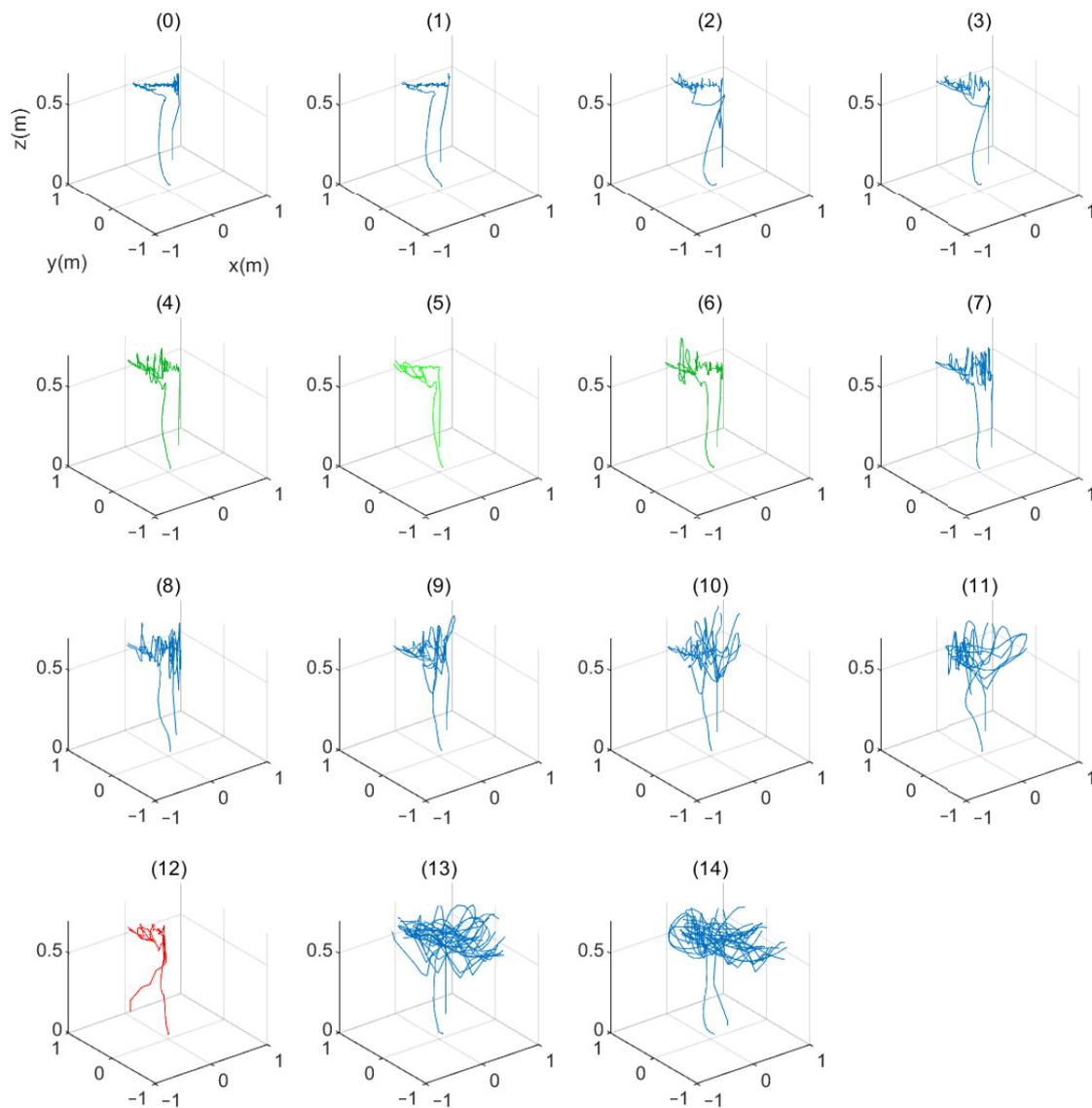
The results to find the balance point between the energy consumed by the quadcopter and the execution time are shown in Figure 13b. It was observed that with higher efficiency of the consumption data rate by the quadcopter, the performance and energy increases; however, for a rate lower than three and higher than 8, the results showed an excess or lack of time.

From the results, we can deduce that there is a relationship between the energy consumption of the quadcopter and the information efficiency. For this, by repeating the experiments, a buffer with the size of 6 is the optimal value for executing the task.

In buffer size 12, a perturbation was tested by eliminating information from the producer. Thus, it was verified that the lack of information decreased the energy consumption; therefore, the relationship between the quality of the information and the energy consumption was verified.

Figure 14 shows the results of the trajectories based on the scalability of the buffer size. A more stable behavior was observed when the information to be consumed is higher Figure 14(0–8). However, when the data is missing, there is a more unstable behavior as is shown in Figure 14(12).

Finally, a video of the experiments can be seen at <https://rodrigo vazquezlopez.github.io/Cooperation/> (accessed on 23 April 2021).



**Figure 14.** Different tests made using buffer sizes from 0 to 14.

## 5. Conclusions and Future Work

The research highlighted the advantages of software-oriented robotic system architectures; however, the disadvantages with small vehicle resources were shown. The proposed hardware-oriented design succeeded in solving a small vehicle system's particular problem without losing the advantage of compatibility with widely used tools.

The producer-consumer algorithm successfully solved the problem of synchronization of messages shared by two or more vehicles. This algorithm was implemented in real hardware, while other similar algorithms only were simulated. This research also solved the problem of intercepting an airborne vehicle's trajectory to a level below 0.04 m and landing it safely on top of the ground vehicle. The EV3 robot and the Crazyflie quadcopter are affordable small-scale platforms and offer the experimenter reduced scale equipment that avoids unnecessary exposure of people to risks.

It was possible to reduce the number of anchors from 6 and 8, as mentioned in similar publications, to only 4. The square-shaped workspace gives better results than the recommended rectangular one. Furthermore, in parallel, perpendicular or diagonal paths, the RMS-positioning-error is reduced, compared to the one in curved paths, i.e., when the tags rotate in the plane, has to be interference that increases the error. The positioning error

increased due to the proximity between a tag and anyone of the anchors. Even under these higher noise conditions, the results were satisfactory; that is, an average error of 0.04 m was achieved.

The LPS platform configuration has the following advantages: Experimental test in small environments of only 2 m × 2 m × 1 m (width, length and height), the platform is portable and does not require complex installations. Due to the ongoing epidemic, it is impossible to work in research laboratories in many places in the world. Therefore, the possibility of having a compact platform, like the one presented here, allows continuing developing research in our home-laboratories.

Performance tests showed the ability to scale the memory size and thus process more information. However, the efficiency of the system does not improve proportionally. A balance point was found to define a trade-off between data, power consumption, and processing time. Therefore, we can suggest considering the efficiency relationship between the size of the vehicles and their information management systems with the system's scalability.

For further development, a new intelligent and robust algorithm will be proposed to be executed intrinsically in the airborne and terrestrial robot and achieve the algorithm's implementation with a more significant number of cooperating robots.

**Author Contributions:** Conceptualization, R.V.-L., J.C.H.-L. and J.S.-G.; methodology, R.V.-L., J.C.H.-L. and J.S.-G.; software, R.V.-L.; validation, P.v.B., J.S.-G.; formal analysis, R.V.-L., J.C.H.-L. and J.S.-G.; investigation, R.V.-L., J.C.H.-L. and J.S.-G.; resources, D.L.M.-V.; data curation, D.L.M.-V.; writing—original draft preparation, R.V.-L., J.S.-G. and P.v.B.; writing—review and editing, R.V.-L., J.S.-G. and P.v.B.; visualization, P.v.B.; supervision, J.C.H.-L.; project administration, D.L.M.-V.; funding acquisition, D.L.M.-V. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding and the APC was funded by Sistemas Ciberfisicos research project by UAM-L.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** This study was supported by the Secretaría de Investigación y Posgrado of the Instituto Politécnico Nacional under the research grant 20211103. R. Vazquez-Lopez acknowledges financial support from Becas Nacionales-CONACYT. J. C. Herrera-Lozada, D. L. Martinez-Vazquez and J. Sandoval-Gutierrez thank the support from the Sistema Nacional de Investigadores-CONACYT.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Koseglu, M.; Celik, O.M.; Pektas, O. Design of an autonomous mobile robot based on ROS. In Proceedings of the 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 16–17 September 2017; pp. 1–5. [\[CrossRef\]](#)
2. Aung, T.; Min, H.Y.; Maw, A.H. CIMLA: Checkpoint Interval Message Logging Algorithm in Kafka Pipeline Architecture. In Proceedings of the 2020 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 4–5 November 2020; pp. 30–35. [\[CrossRef\]](#)
3. Sanchez-Lopez, J.; Molina, M.; Bavle, H.; Sampedro, C.; Suárez Fernández, R.; Campoy, P. A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework. *J. Intell. Robot. Syst. Theory Appl.* **2017**, *88*, 683–709. [\[CrossRef\]](#)
4. Atoev, S.; Kwon, K.; Lee, S.; Moon, K. Data analysis of the MAVLink communication protocol. In Proceedings of the 2017 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 2–4 November 2017; pp. 1–3. [\[CrossRef\]](#)
5. Evers, R.; Masters, P. The application of low-altitude near-infrared aerial photography for detecting clandestine burials using a UAV and low-cost unmodified digital camera. *Forensic Sci. Int.* **2018**, *289*, 408–418. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Zahran, S.; Masiero, A.; Mostafa, M.M.; Moussa, A.M.; Vettore, A.; El-Sheimy, N. Uavs enhanced navigation in outdoor gnss denied environment using uwb and monocular camera systems. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *XLII-2/W13*, 665–672. [\[CrossRef\]](#)

7. Akçakoca, M.; Atıcı, B.; Gever, B.; Oğuz, S.; Demir, M.; Saldıran, E.; Yuksek, B.; Koyuncu, E.; Yeniceri, R.; Inalhan, G. A simulation-based development and verification architecture for micro uav teams and swarms. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019. [[CrossRef](#)]
8. Dirir, A.; Elsayed, H. Object Tracking Framework for Unmanned Aerial Vehicles. In Proceedings of the 2019 IEEE Global Conference on Internet of Things (GCIoT), Dubai, United Arab Emirates, 4–7 December 2019; pp. 1–6. [[CrossRef](#)]
9. Lee, J.; Su, Y.; Shen, C. A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In Proceedings of the IECON 2007—33rd Annual Conference of the IEEE Industrial Electronics Society, Taipei, Taiwan, 5–8 November 2007; pp. 46–51. [[CrossRef](#)]
10. Pinto, M.F.; Honório, L.M.; Marcato, A.L.M.; Dantas, M.A.R.; Melo, A.G.; Capretz, M.; Urdiales, C. ARCoG: An Aerial Robotics Cognitive Architecture. *Robotica* **2021**, *39*, 483–502. [[CrossRef](#)]
11. Xu, C.; Wang, X.; Duan, S.; Wan, J. Spatial-temporal constrained particle filter for cooperative target tracking. *J. Netw. Comput. Appl.* **2021**, *176*. [[CrossRef](#)]
12. Onyekpe, U.; Kanarachos, S.; Palade, V.; Christopoulos, S.R. Vehicular localisation at high and low estimation rates during gnss outages: A deep learning approach. In *Deep Learning Applications, Volume 2*; Wani, M.A., Khoshgoftaar, T.M., Palade, V., Eds.; Springer: Berlin/Heidelberg, Germany, 2021; Volume 1232, pp. 229–248. [[CrossRef](#)]
13. Brena, R.; García-Vázquez, J.; Galván-Tejada, C.; Muñoz-Rodríguez, D.; Vargas-Rosales, C.; Fangmeyer, J. Evolution of Indoor Positioning Technologies: A Survey. *J. Sens.* **2017**, *2017*. [[CrossRef](#)]
14. Mohr, H. UAV Implementation of Distributed Robust Target Location in Unknown Environments. In Proceedings of the 2020 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2020; pp. 1–10. [[CrossRef](#)]
15. Gargioni, G.; Peterson, M.; Persons, J.; Schroeder, K.; Black, J. A full distributed multipurpose autonomous flight system using 3D position tracking and ROS. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14 June 2019; pp. 1458–1466. [[CrossRef](#)]
16. Yashin, G.; Trinitatova, D.; Agishev, R.; Ibrahimov, R.; Tsetserukou, D. AeroVr: Virtual reality-based teleoperation with tactile feedback for aerial manipulation. In Proceedings of the 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2–6 December 2019; pp. 767–772. [[CrossRef](#)]
17. Wu, R.; Pandurangaiah, J.; Blankenship, G.; Castro, C.; Guan, S.; Ju, A.; Zhu, Z. Evaluation of Virtual Reality Tracking Performance for Indoor Navigation. In Proceedings of the 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), Portland, OR, USA, 20–23 April 2020; pp. 1311–1316. [[CrossRef](#)]
18. Shin, J.M.; Kim, Y.S.; Ban, T.W.; Choi, S.; Kang, K.M.; Ryu, J.Y. Position tracking techniques using multiple receivers for anti-drone systems. *Sensors* **2021**, *21*, 35. [[CrossRef](#)] [[PubMed](#)]
19. Adjei, H.; Oduro-Gyimah, F.; Shunhua, T.; Agordzo, G.; Musariri, M. Developing a bluetooth based tracking system for tracking devices using arduino. In Proceedings of the 2020 5th International Conference on Computing, Communication and Security (ICCCS), Patna, India, 14–16 October 2020; pp. 1–5. [[CrossRef](#)]
20. Lie, K.; Yeo, K.; Ting, A.N.; Chieng, D.T. Indoor Tracking with Bluetooth Low Energy Devices Using K-Nearest Neighbour Algorithm. In Proceedings of the 2020 IEEE 10th Symposium on Computer Applications Industrial Electronics (ISCAIE), Shah Alam, Malaysia, 18–19 April 2020; pp. 155–159. [[CrossRef](#)]
21. Shi, G.; Ming, Y. Survey of indoor positioning systems based on ultra-wideband (UWB) technology. *Wirel. Commun. Netw. Appl.* **2016**, *348*, 1269–1278. [[CrossRef](#)]
22. Silvia, Z.; Martina, C.; Fabio, S.; Alessandro, P. Ultra Wide Band Indoor Positioning System: Analysis and testing of an IPS technology. *IFAC-PapersOnLine* **2018**, *51*, 1488–1492. [[CrossRef](#)]
23. Lutz, P.; Schuster, M.; Steidle, F. Visual-inertial SLAM aided estimation of anchor poses and sensor error model parameters of UWB radio modules. In Proceedings of the 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2–6 December 2019; pp. 739–746. [[CrossRef](#)]
24. Cao, Y.; Yang, C.; Li, R.; Knoll, A.; Beltrame, G. Accurate position tracking with a single UWB anchor. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 2344–2350. [[CrossRef](#)]
25. Goma, M.; Silva, O.D.; Mann, G.; Gosine, R.; Hengeveld, R. ROS Based Real-Time Motion Control for Robotic Visual Arts Exhibit Using Decawave Local Positioning System. In Proceedings of the 2020 American Control Conference (ACC), Denver, CO, USA, 1–3 July 2020; pp. 653–658. [[CrossRef](#)]
26. Ledergerber, A.; D’Andrea, R. Ultra-wideband range measurement model with Gaussian processes. In Proceedings of the 2017 IEEE Conference on Control Technology and Applications (CCTA), Maui, HI, USA, 27–30 August 2017; Volume 2017, pp. 1929–1934. [[CrossRef](#)]
27. Zhao, W.; Goudar, A.; Panerati, J.; Schoellig, A.P. Learning-based Bias Correction for Ultra-wideband Localization of Resource-constrained Mobile Robots. *arXiv* **2020**, arXiv:2003.09371.
28. Pinto, M.F.; Honório, L.M.; Melo, A.; Marcato, A.L.M. A Robotic Cognitive Architecture for Slope and Dam Inspections. *Sensors* **2020**, *20*, 4579. [[CrossRef](#)] [[PubMed](#)]
29. Munro, A.; Clayton, G. Drone Swarms, Communications Performance and Big Data. In Proceedings of the 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), Honolulu, HI, USA, 22–25 September 2019; pp. 1–5. [[CrossRef](#)]

30. Silva, L.A.Z.d.; Vidal, V.F.; Honório, L.M.; Dantas, M.A.R.; Pinto, M.F.; Capretz, M. A Heterogeneous Edge-Fog Environment Supporting Digital Twins for Remote Inspections. *Sensors* **2020**, *20*, 5296. [[CrossRef](#)]
31. Wallace, C.; Ackels, S.; Benavidez, P.; Jamshidi, M. Real-Time Distributed Ensemble Learning for Fault Detection of an Unmanned Ground Vehicle. In Proceedings of the 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary, 2–4 June 2020; pp. 463–468. [[CrossRef](#)]
32. Angel, M.; Abdulla, A.K.; David, M.; de la Escalera, A. 3D Trajectory Planning Method for UAVs Swarm in Building Emergencies. *Sensors* **2020**, *20*, 642. [[CrossRef](#)]
33. Koubâa, A.; Qureshi, B.; Sriti, M.F.; Allouch, A.; Javed, Y.; Alajlan, M.; Cheikhrouhou, O.; Khalgui, M.; Tovar, E. Dronemap Planner: A service-oriented cloud-based management system for the Internet-of-Drones. *Ad Hoc Netw.* **2019**, *86*, 46–62. [[CrossRef](#)]
34. Basso, M.; Zacarias, I.; Tussi Leite, C.E.; Wang, H.; Pignaton de Freitas, E. A Practical Deployment of a Communication Infrastructure to Support the Employment of Multiple Surveillance Drones Systems. *Drones* **2018**, *2*, 26. [[CrossRef](#)]
35. Damacharla, P.; Mehta, D.; Javaid, A.Y.; Devabhaktuni, V.K. Study on State-of-the-art Cloud Services Integration Capabilities with Autonomous Ground Vehicles. In Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, IL, USA, 27–30 August 2018; pp. 1–5. [[CrossRef](#)]
36. Salmeron-Garcia, J.J.; van den Dries, S.; Diaz-del Rio, F.; Morgado-Estevez, A.; Luis Sevillano-Ramos, J.; van de Molengraft, M.J.G. Towards a cloud-based automated surveillance system using wireless technologies. *Multimed. Syst.* **2019**, *25*, 535–549. [[CrossRef](#)]
37. Okumuş, F.; Dönmez, E.; Kocamaz, A.F. A Cloudware Architecture for Collaboration of Multiple AGVs in Indoor Logistics: Case Study in Fabric Manufacturing Enterprises. *Electronics* **2020**, *9*, 2023. [[CrossRef](#)]
38. Okumuş, F.; Kocamaz, A.F. Cloud Based Indoor Navigation for ROS-enabled Automated Guided Vehicles. In Proceedings of the 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 21–22 September 2019; pp. 1–4. [[CrossRef](#)]
39. Cardarelli, E.; Digani, V.; Sabattini, L.; Secchi, C.; Fantuzzi, C. Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses. *Mechatronics* **2017**, *45*, 1–13. [[CrossRef](#)]
40. Indri, M.; Lazzero, I. A new HW/SW architecture to move from AGVs towards Autonomous Mobile Robots. In Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019; pp. 1551–1554. [[CrossRef](#)]
41. Yilmaz, A.; Temeltas, H. ROS Architecture for Indoor Localization of Smart-AGVs Based on SA-MCL Algorithm. In Proceedings of the 2019 11th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkey, 28–30 November 2019; pp. 885–889. [[CrossRef](#)]
42. Li, J.; Dong, W.; Sheng, X.; Xu, S. Visual Servoing of Micro Aerial Vehicles with the Cooperation of Ground Vehicle. In Proceedings of the 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), Portland, OR, USA, 20–23 April 2020; pp. 150–155. [[CrossRef](#)]
43. Destiarti, A.; Kristalina, P.; Sudarsono, A. Mobile cooperative tracking with RSSI ranging in EKF algorithm for indoor wireless sensor network. In Proceedings of the 2016 International Conference on Knowledge Creation and Intelligent Computing (KCIC), Manado, Indonesia, 15–17 November 2017; pp. 60–66. [[CrossRef](#)]
44. Yulong, D.; Bin, X.; Jie, C.; Hao, F.; Yangguang, Z.; Guanqiang, G.; Lihua, D. Path Planning of Messenger UAV in Air-ground Coordination. *IFAC-PapersOnLine* **2017**, *50*, 8045–8051. [[CrossRef](#)]
45. Marzoughi, A.; Savkin, A. Autonomous navigation of a team of unmanned surface vehicles for intercepting intruders on a region boundary. *Sensors* **2021**, *21*, 297. [[CrossRef](#)]
46. Huang, H.; Savkin, A.; Li, X. Reactive autonomous navigation of UAVs for dynamic sensing coverage of mobile ground targets. *Sensors* **2020**, *20*, 3720. [[CrossRef](#)]
47. Siegwart, R.; Nourbakhsh, I.R.; Scaramuzza, D. *Introduction to Autonomous Mobile Robots*; MIT Press: Cambridge, MA, USA, 2011.
48. Greiff, M. Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation. Ph.D. Thesis, Department of Automatic Control, Lund University, Lund, Sweden, 2017. Student Paper.
49. Luis, C.; Ny, J.L. Design of a Trajectory Tracking Controller for a Nanoquadcopter. *arXiv* **2016**, arXiv:1608.05786.
50. Hustiu, S.; Lupascu, M.; Popescu, S.; Burlacu, A.; Kloetzer, M. Stable hovering architecture for nanoquadcopter applications in indoor environments. In Proceedings of the 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 10–12 October 2018; pp. 659–663. [[CrossRef](#)]
51. Budaciu, C.; Botezatu, N.; Kloetzer, M.; Burlacu, A. On the Evaluation of the Crazyflie Modular Quadcopter System. In Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019; Volume 2019, pp. 1189–1195. [[CrossRef](#)]
52. Stallings, W. *Operating Systems: Internals and Design Principles*, 8th ed.; Prentice Hall: Hoboken, NJ, USA, 2012.
53. Dijkstra, E.W. Information streams sharing a finite buffer. *Inf. Process. Lett.* **1972**, *1*, 179–180. [[CrossRef](#)]
54. Feng, L.; Borenstein, J.; Everett, H.R. *Where Am I: Sensors and Methods for Mobile Robot Positioning*; University of Michigan: Ann Arbor, MI, USA, 1994.
55. Mueller, M.; Hamer, M.; D’Andrea, R. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadcopter state estimation. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; Volume 2015, pp. 1730–1736. [[CrossRef](#)]

- 
56. Mueller, M.; Hehn, M.; D'Andrea, R. Covariance correction step for Kalman filtering with an attitude. *J. Guid. Control Dyn.* **2017**, *40*, 2301–2306. [[CrossRef](#)]
  57. Smeur, E.; Chu, Q.; Croon, G.D. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *J. Guid. Control Dyn.* **2016**, *39*, 450–461. [[CrossRef](#)]
  58. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525. [[CrossRef](#)]