

# SwarmUS: An open hardware and software on-board platform for swarm robotics development

É. Villemure, P. Arsenault, G. Lessard, T. Constantin, H. Dubé, L.-D. Gaulin, X. Groleau, S. Laperrière, C. Quesnel and F. Ferland

**Abstract**—Real life implementations of distributed swarm robotics are rare. The standardization of a general purpose swarm robotics platform could greatly accelerate swarm robotics towards real life implementations. The SwarmUS platform is an open-source hardware and software on-board embedded system designed to be added onto existing robots while providing them with swarm features, thus proposing a new take on the platform standardization problem. These features include a distributed relative localization system based on Ultra-Wideband, a local communication system based on Wi-Fi and a distributed coordination system based on the Buzz programming language between robots connected within a SwarmUS platform. Additionally, a human-swarm interaction mobile application and an emulation of the platform in the Robot Operating System (ROS) is presented. Finally, an implementation of the system was realized and tested on two types of robots : a TurtleBot3 Burger and two Pioneer 2DX.

**Index Terms**—Swarm Robotics, Software-Hardware Integration for Robot Systems and Localization.

## I. INTRODUCTION

Swarm robotics is the study of multi-robot systems orchestrated with swarm intelligence to perform certain tasks. Even if the entertainment industry has shown centrally controlled swarms of drones in action [1], the main focus of the swarm robotics field revolves around distributed system to remove any single point of failure. Those distributed systems show promising robustness, fault-tolerance and flexible characteristics compared to single robots when achieving specific tasks [2]. However, swarms of robots are rarely seen in real life applications. According to Nedjah and Junior [3], it is mainly caused by a lack of standard methodologies and the use of generic hardware and software platforms in swarm robotics development. Establishing those standards could greatly accelerate research towards real life applications. Multiple robotics platforms are available such as E-Puck [4], Kilobot [5], and R-One [6]. However, these platforms were not designed to test real-life scenarios where a task is performed in an environment not suitable for small wheeled robots. Additionally, there is a lack of compatibility between the communication system and the coordination system of the majority of those available platforms [3]. Therefore, it

All authors are with the Department of Electrical Engineering and Computer Engineering of the University of Sherbrooke (2500 boulevard de l'Université, Sherbrooke, Québec (Canada), J1K-2R1). F. Ferland is an IEEE member. Contacts for the authors: {Etienne.Villemure, Philippe.Arsenault2, Gabriel.Lessard, Thierry.Constantin, Hubert.Dube, Louis-Daniel.Gaulin, Xavier.Groleau, Samuel.Laperriere, Charles.Quesnel, Francois.Ferland}@usherbrooke.ca.

hinders the development of heterogeneous swarms by the community.

The open source SwarmUS platform proposes a different approach to the hardware standardization problem: giving existing non-swarm dedicated robot models capabilities to work as a distributed swarm. These provided swarm capabilities are comprised of a distributed coordination system, a local communication network and a relative localization system. This paradigm takes advantage of the broad spectrum of robot models that have all the necessary sensors, computation power, and actuators to perform more realistic swarm applications. It allows the development of distributed heterogeneous swarm while using a common hardware and software platform for the swarm related features. Additionally, researchers can retrofit this platform on the robots they already own. Therefore, it facilitates the inclusion of new researchers in the swarm robotics community while taking advantage of their expertise on their own equipment.

To the best of our knowledge, this idea of adding swarm capabilities to generic robots has been explored only once in the literature [7]. The SwarmUS platform differs from that work mainly from the custom hardware platform, the use of Wi-Fi instead of Zigbee for higher bandwidth, the use of Buzz [8] as a swarm coordination system, its integration with the Robot Operating System (ROS) [9] and the included relative localization feature. We believe that the addition of these features may offer a viable solution towards widespread adoption and standardization.

This paper is organized as follows: Sect. II presents the SwarmUS platform and describes each components of the system from a hardware and software perspective. Sect. III presents results from an experimental validation of the main features of the platform as observed from its integration on three robots (two modified Pioneer 2DX and one modified TurtleBot3 Burger) executing a "follow the leader" task. Section IV discusses the current performances and limitations of the original prototype.

## II. SYSTEM DESCRIPTION

The SwarmUS platform is an open hardware and software on-board system composed of multiple software libraries and two types of electronic circuit boards: the *Hiveboard* and the *Beeboard*. Fig. 1 shows the high-level architecture of the SwarmUS platform and how the libraries and circuit boards can be interfaced with a robot or an Android phone for Human-Swarm Interaction (HSI). Integrating the SwarmUS platform on a robot grants it three general swarm features:

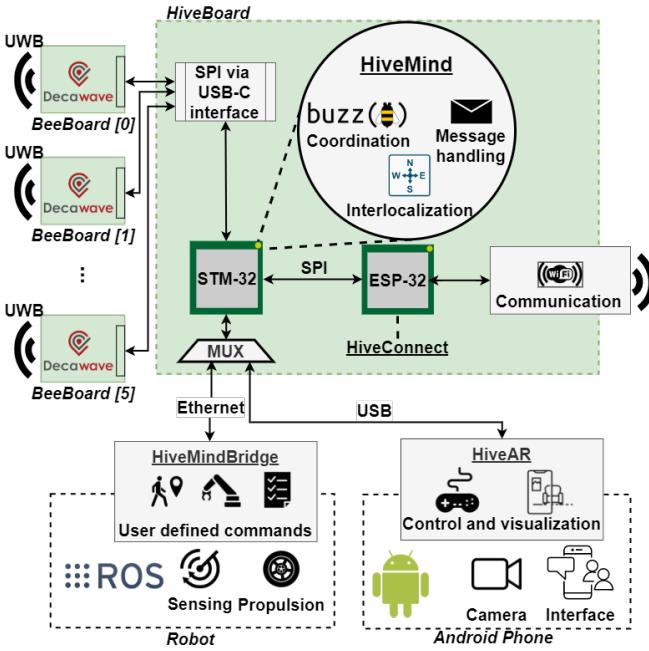


Fig. 1: High-level diagram of the SwarmUS platform illustrating the relations between the *Hiveboard*, the *Beeboards*, the host devices (Example: robot or phone) and their internal systems.

coordination, communication, and relative localization. Both coordination and communication features are implemented on the *Hiveboard*, a custom made 8-layer printed circuit board (PCB) which is the central hardware component of the platform.

The coordination feature manages the swarm intelligence and the behaviors of an individual robot with all the other robots equipped with the platform. The implementation of the swarm behaviors is handled by the Buzz programming language [8] which was specifically designed to program a swarm of robots. Each *Hiveboard* has a Buzz Virtual Machine (BVM) where the Buzz scripts are executed.

The communication feature creates a local network between all the *Hiveboards* using Wi-Fi. This feature supports the necessary needs in communication for the coordination feature while also enabling robot to robot communication.

The SwarmUS platform can create a standalone swarm of *Hiveboards*. However, since the *Hiveboard* has two wired interfaces (USB and Ethernet) to communicate with external devices, a swarm can be formed from any such devices. These devices are assigned as hosts, and can be heterogenous robots or Android phones or tablets. An Android device can thus be connected through USB to a *Hiveboard*, join the swarm and then be used to visualize and send commands to the swarm, enabling HSI. In the scope of this project an Android application, named *HiveAR*, has been developed to configure and monitor the state of the swarm.

Because of the level of abstraction between the *Hiveboard* and the host device, the swarm development can be disconnected from the development of robot features like sensing

and navigation. Therefore, the focus of the swarm robotics community could be put around standardizing a swarm platform without worrying about the different mechanical base and computational capabilities of robots.

The last feature, the relative localization system, allows robots equipped with the system to measure their relative distance and bearing from each other. This type of relative measurement is mandatory for multiple swarm behaviors [10] [11]. These measurements are obtained through the Ultra-Wideband (UWB) technology of the *Beeboards*. UWB offers more range and has more environmental flexibility (ex: not dependent on ambient lighting) compared to other local measurement systems used in swarm robotics [12]. Additionally, the system is independent from a centralized system like GPS or overhead cameras for smaller robots to work, giving more robustness to the swarm. As shown in Fig.1, at least one *Beeboard* needs to be connected to a *Hiveboard* via a USB-C interface in order to provide those measurements. A single *Beeboard* connected to a *Hiveboard* gives the relative distance between every other *Beeboards* in line of sight. If there is at least three *Beeboards*, it can also give the bearing angle. Up to six *Beeboards* can be connected to a single *Hiveboard* to enhance the measurements.

The *Hiveboard* contains an STM32 microcontroller unit (MCU) which is responsible of the coordination feature, thus executing the BVM and the Buzz scripts. It also has a message management system to handle the messages from the other *Hiveboards* coming from the Wi-Fi network and from the connected host device. Lastly, it also contains the relative localization algorithm to calculate the range and the bearing of other robots based on the *Beeboard* data. All the software components, including the low level hardware interfaces, are combined under the same firmware called the *HiveMind*. The *Hiveboard* also has an ESP32 module interfacing with the STM32 to manage Wi-Fi networking. The ESP32 module has its own firmware called the *HiveConnect*.

To enable communication with the *Hiveboard*, a C++ library for the host devices, the *HiveMindBridge*, has been developed. In the scope of the project, the *HiveMindBridge* has only been implemented in a ROS package for devices with an Ethernet interface, but it could be implemented on any C++ platform.

To ease development with the SwarmUS platform, the same code that runs on a *Hiveboard* can be cross compiled as a ROS node on any Linux-based system. The low level hardware interfaces, the Wi-Fi network and the relative localization system can be all emulated using ROS nodes, ROS topics, and the Gazebo simulator [13] to provide a fully emulated system. Therefore, it facilitates the transition from simulation development to real world integration, which is an important feature for a standardized platform [2].

In the end, once the SwarmUS platform is connected to a host device, the developer only needs to write the Buzz scripts in the *HiveMind*. There, the developer needs to define callback functions called between the *Hiveboard* and its host device in the *HiveMind* and the *HiveMindBridge*. Finally, they only need to ensure that the host device reacts desirably

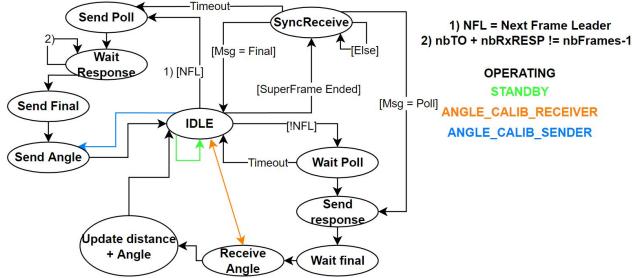


Fig. 2: Finite state machine of the localization system

to these callbacks by performing the desired actions.

All the hardware and software components of SwarmUS are open source under the MIT License [14] and can be found on GitHub<sup>1</sup>, along with full documentation<sup>2</sup>.

### A. Localization

The relative localization feature of the SwarmUS platform is provided by the active *Beeboards* antennas. Each *Beeboard* harbors a Decawave DW-1000 integrated circuit with a Chiolas ANT110 UWB antenna and can be connected to the *Hiveboard* via an USB-C connector supporting an SPI bus. A system with a *Hiveboard-Beeboard* combination can localize another similar system using the two-way ranging (TWR) technique [15].

TWR consists of exchanging three messages between an initiator and a responder. As the messages travel from one to another, the time of arrival is recorded and passed on with the messages. Since multiple robots are using TWR in the same air space, message collision might happen. To prevent this, the SwarmUS platform has a scheduling mechanism where each system has a specific time slot to send its TWR message. The management of all these messages and the synchronization of the SwarmUS platform is done via a finite state machine (Fig. 2). The synchronization is performed for every robot that joins the swarm and happens during the "SyncReceive" step in FSM the diagram. However, the number of time slots in the schedule is set during compile time so the maximum scale of the swarm must be known beforehand.

To determine the bearing measurement, a pair of *Beeboard* antennas are necessary to measure the phase difference of arrival. When a signal is received by two synchronized *Beeboards*, a quadrature system as described in [16] is used to extract a phase difference. From this difference, an angle can be extracted in a  $[-90, 90]^\circ$  range as it can be seen in Fig. 3. Since a single pair cannot tell the difference between a front and back arrival, a third antenna must be introduced to create an equilateral triangle formation to break the mirror effect, therefore creating three pairs of antennas. Furthermore, a single antenna pair has difficulties when the absolute value of the angle is high because the calculation

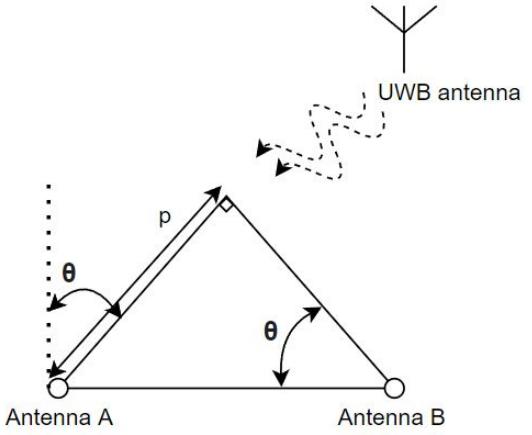


Fig. 3: Geometric relation between the difference of phase ( $p$ ) of two antennas and the bearing angle ( $\theta$ )

of the angle is based on the  $\arctan^3$  function.

It is imperative to characterize the angular response with a test bench to compensate any misplacement in the position of the antennas and to verify if there are unreachable or inaccurate zones of reception. Doing so can reveal blind spots of each antenna pair that can be covered by better placed pair. In the SwarmUS platform, each antenna pair has a weight corresponding to the certainty of the measurement. The certainty is based on the value angle returned and a line of sight quality indicator (as provided by an internal registry of the DW-1000), any measure outside  $[-30,30]^\circ$  or  $[150,210]^\circ$  is considered more uncertain. Proceeding this way, the false values caused by measurement errors are minimized and the best values are maximized before the angle is returned.

### B. Coordination

The SwarmUS platform leverages the Buzz [8] language to implement swarm behaviors via a virtual machine (BVM). Since this programming language was tailor-made for controlling robotic swarms, it contains key data structures and functionality to manage groups of robots. For instance, it can easily be used to manage a shared data table updated by agents in the swarm called the *stigmergy*. It also provides a *neighbor* type used to store positional information on the robots in the swarm as well as communicate with *broadcast* and *listen* methods.

Buzz runs in the BVM, which has a base version to run on Unix systems and also a more lightweight version, the Bitty Buzz Virtual Machine<sup>4</sup> (BBVM), targeted for microcontroller like ARM Cortexes. The *HiveMind* firmware is built around the BBVM, both for the firmware and its emulation to limit discrepancies between the two implementations. The integration of the BBVM in the *HiveMind* firmware relies on providing implementations to core systems of the BBVM,

<sup>3</sup>[https://swarmus.github.io/SwarmUS-doc/sections/reference/Interloc/how\\_it\\_works/angle/](https://swarmus.github.io/SwarmUS-doc/sections/reference/Interloc/how_it_works/angle/)

<sup>4</sup><https://github.com/buzz-lang/BittyBuzz>

<sup>1</sup><https://github.com/SwarmUS>

<sup>2</sup><https://swarmus.github.io/SwarmUS-doc/>

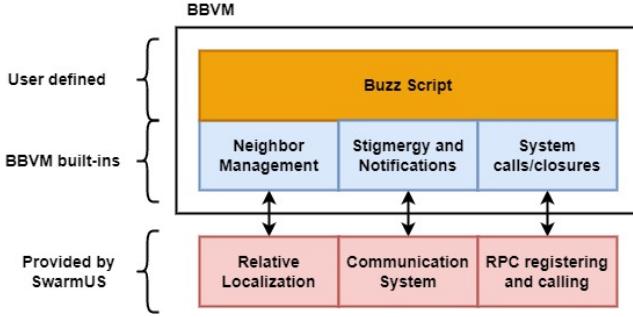


Fig. 4: The components constituting the BBVM. The Buzz script is defined by the user, the parts in blue are core systems in the BBVM and the parts in red are the system powering those functionalities on the SwarmUS platform.

like relative localization and communication systems. Figure 4 presents the links between the functionalities available in Buzz and their link to the components of the system.

The firmware also links together the Buzz script and the *host* robot. The robot exposes a manifest of actions, like movement or interactions with the environment, that it can perform. This manifest takes the form of a list of callbacks, transmitted between the host and the *Hiveboard*. The buzz script controls the robot and this abstraction allows for easy deployment on heterogeneous robots that have different capabilities. This manifest of actions can also be used to help the swarm task allocation process based on robot capabilities [17].

### C. Communication

The communication system for the platform revolves around a Wi-Fi network, provided by one of the *Hiveboards* or by an external access point. Every board is configured to join the same configured network to then exchange data.

The communication scheme supports broadcast and unicast communication. The broadcast is mainly used to update the *stigmergy* from Buzz whereas the unicast is used to send commands. These commands are sent by other agents and can either target a given agent's *Hiveboard* or host.

The two microcontrollers on the *Hiveboard* will then dispatch the messages to the proper target with a messaging system (Fig 5) built using Google's Protobuf<sup>5</sup>. The system is built around Remote Procedure Calls (RPCs) to either invoke an action or update information, either coming from the rest of the swarm, the host or other systems like the relative localization feature.

### D. Human-Swarm Interaction (HSI)

The HSI interface, called *HiveAR*, revolves around controlling multiple agents, either as a group or individually. The interface was built as an Android application that connects to a *Hiveboard*, making it a member of the swarm. Like shown in Fig. 6a and 6b, it can view the available actions of an agent, call those actions with parameters and broadcast

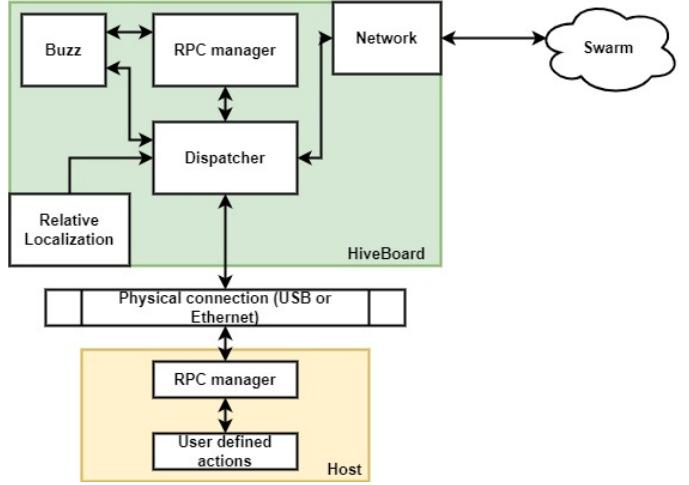


Fig. 5: Communication datapath between a host, a *Hiveboard* and the swarm. Both host and *Hiveboard* have a dispatcher to forward calls to the proper target. The network box represents the ESP32 and the *HiveConnect* firmware, which also has an internal dispatcher to translate between agent id and IP addresses for unicast calls.

commands to the swarm. It is also responsible to configure some parameters for the *Hiveboard* like its ID and network configuration. It leverages Augmented Reality (AR) features to easily select an agent to view its command, to call them, and to monitor the status of the agent as well as track its movement. To localize the robot, the AR feature uses the cellphone's camera to find April Tags<sup>6</sup> placed on the robots as it can be observed in Fig. 6c.

## III. EXPERIMENTAL VALIDATION

Six *Hiveboards* and eighteen *Beeboards* were produced for the experimental validation. The size and top layer of these 8-layer PCB are shown in Fig. 7.

<sup>6</sup><https://github.com/AprilRobotics/apriltag>

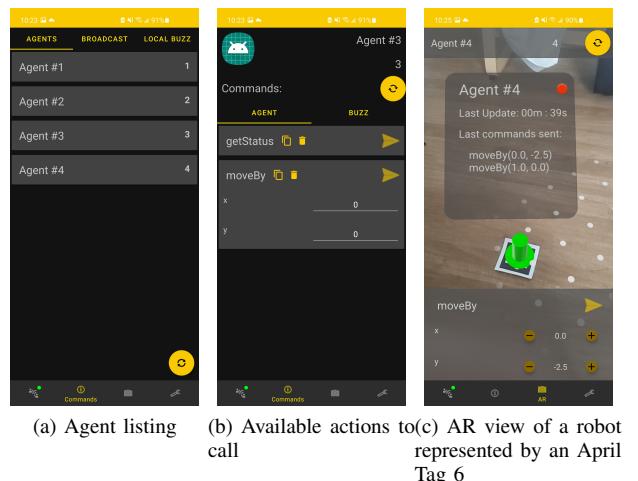


Fig. 6: The *HiveAR* interface

<sup>5</sup><https://developers.google.com/protocol-buffers>

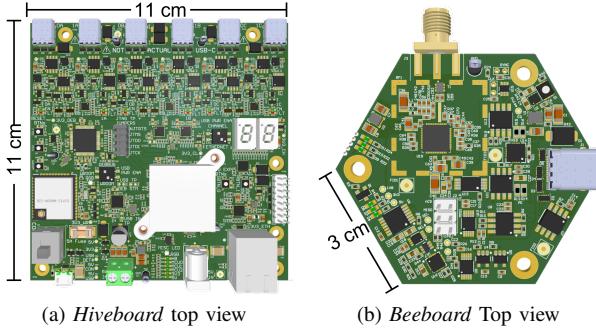


Fig. 7: SwarmUS boards

The integration of the SwarmUS system was done on two types of robots : a TurtleBot3 Burger and two Pioneer 2DX. The hardware and software of the differential drive Pioneer 2DX have been updated in the scope of this project. The two Pioneer 2DX are equipped with a A2M8 RPLidar lidar, a D400 series Realsense Camera, a 2x30A RoboClaw as their motor controller and a Raspberry Pi 4 4GB as their main computer. The TurtleBot3 Burger is a differential drive robot with a small footprint of 13.8 by 17.8 cm equipped with a LDS-01 lidar and has been updated to have a Raspberry Pi 4 4GB. On the software level, these robots run on a ROS architecture for safe navigation towards goal positions received by the *Hiveboard*. Except for the nodes interfacing with sensors and motors, the main implemented nodes are *HiveMindBridge* and *movebase*<sup>7</sup>, responsible for path planning and obstacle avoidance.

Figure 8 illustrates a CAD model of the integration of the SwarmUS platform on one of the Pioneer 2DX. The *Beeboards* are mounted on a custom 3D printed support<sup>8</sup> placing their UWB antennas 2.7 cm apart in a triangle formation. Out of multiple conducted experiments to find the best antenna formation, this triangle pattern showed the best relative localization performances and is used throughout all this section. Fig. 8 also shows the *Hiveboard* supported by another 3D printed support beneath the Pioneer's lidar to save space and to lift the lidar over the *Beeboards* antennas.

Since the TurtleBot3 Burger is a scalable layered robot, the integration of the SwarmUS platform on the TurtleBot3 Burger required to add a new layer equipped with a *Hiveboard* and *Beeboard*.

A DC/DC converter was added on each robot to produce a 5 V bus from their 11.1 V LiPo battery. This converter powers directly the *Hiveboard*, which then powers the three *Beeboards*. To reduce power consumption, the *Hiveboard* automatically shutdowns the internal components of communication channels that aren't connected to a *Beeboard*.

In the *Hiveboard* and three *Beeboards* configuration, the SwarmUS platform consumes an average of 7 W while executing its Buzz script and localizing another platform.

<sup>7</sup>[http://wiki.ros.org/move\\_base](http://wiki.ros.org/move_base)

<sup>8</sup>All the 3D models are open source and can be found in the main SwarmUS repository.

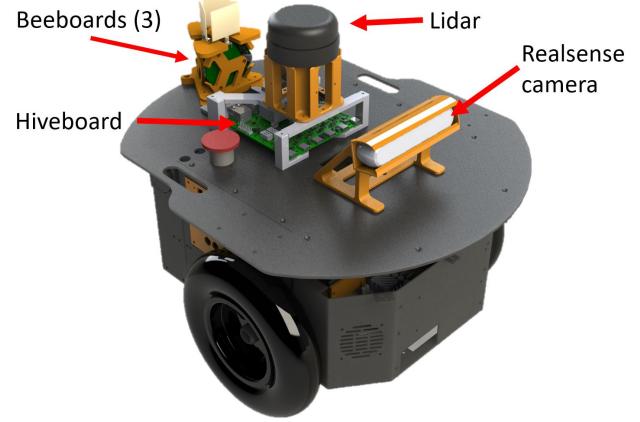


Fig. 8: A Pioneer 2DX with a SwarmUS system

This measurement was taken by the displayed voltage and current value on an external power source powering the *Hiveboard* at 5 V.

Each kit costs around 900 USD for parts, cables and antennas included. The assembly cost is omitted since multiple assembly processes might be used to reduced cost or speed up the process. However, the boards were designed in such a way that only the top side is populated by components, making assembly by hand easier.

The following subsections describe tests that were performed to evaluate each feature provided by the SwarmUS platform.

#### A. Localization

To characterize the relative localization system, multiple measurements of the position and bearing were made. All the tests were made in a line-of-sight environment without any obstacle between the systems. For the distance, 200 measures were taken every 50 cm from 0.5 m to 9 m (Fig. 9a). The average absolute error was 11 cm and the standard deviation was 3.43 cm. For the bearing, a test bench composed of a turning table controlled by a step motor was created. An encoder at the output of the stepper motor measured the bearing angle. The table had step increments of 3.51° and was placed 2.5 m from a transmitting antenna. For each step, 100 measures of the angle were made. The average absolute error measured was 17° and the standard deviation was 1.7° (Fig. 9b and 9c).

Since the localization system uses time frames to exchange information, the larger the swarm grows, the slower the refresh rate of the localization system will be, following a decreasing exponential rule. The refresh rate depends on multiple factors such as the number of antenna used, the transmission speed, the speed of the SPI bus between the microcontroller and DecaWave modules, the Rx and Tx timestamp size, the size of each frame individually and the idle time between each frame (Fig. 9d). It should be noted that the maximum number of robots configured for the localization system limits the refresh rate of measurements and not the actual number of robots.

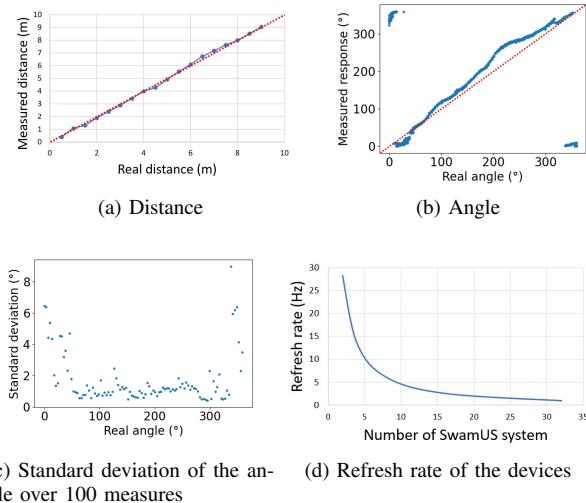


Fig. 9: Relative localization measurements

### B. Coordination

A qualitative evaluation of the coordination system was performed with a "Follow the Leader" task. This behavior was implemented by having a vector pushing the heading of the robot towards the leader. Although *movebase* is used for obstacle avoidance, this ROS node cannot properly avoid other robots because they can't fully see each other with their lidar. For this reason, a second vector pushing robots from each other was added to avoid other robots. The leader ID was maintained in the stigmergy to be changed dynamically.

At first, the validation was performed in a simulation of nine robots using the cross compiled version of the *HiveMind* in ROS and was used as a milestone before testing on real robots. The simulation was done with Gazebo, with ROS nodes and topics emulating the communication and relative localization systems. The simulation allowed to test the behavior of the swarm, independently of the hardware. The validation was executed by changing the leader of the swarm, at run time, using the simulated HSI from the Android Emulator. The simulated robots converged towards the dynamically changing leader as expected. However, some collisions occurred with nine robots, suggesting that the implemented Buzz behavior was not fully tuned. Since the Buzz-based collision avoidance algorithm was relatively simplistic, a full integration with the navigation stack and proper obstacle avoidance could have given better results.

For real world validation, the same "Follow the leader task" was executed with two Pioneer 2DX and an handheld *Hiveboard* and *Beeboard* assembly acting as the leader. One of the robot, without any other obstacles around it, was successfully able to follow the handheld leader, thus confirming that the platform was controlling the robot.

When testing with two Pioneer 2DX and the handheld leader, more emphasis was put on tuning the collision avoidance mechanism compared to the simulation. The imperfections of the relative localization system lead to different

behaviors on the robots, mostly affected by the refresh rate. With a better mechanism for collision avoidance, where robots only avoid other robots in their heading direction, a swarm of two robots were able to follow a third leader agent without collisions. The Turtlebot was first included in the experimental swarm, but it wasn't successfully in achieving the task for mechanical integration reasons.

### C. Communication

For the communication system, there are two important metrics: bandwidth and latency. The bandwidth limits the size and the frequency of updates on the swarm and the latency affects the frequency of updates due to the design of the communication system. Starting with the bandwidth, it limits the number of bytes per second that can be transmitted on the network. For a system running with Buzz, the most common used of the bandwidth is to share stigmergy updates between agents. The bandwidth usage is given by the following equation:

$$\text{Bandwidth} = \text{Size} \cdot \text{Frequency} \cdot N^2 \quad (1)$$

where Bandwidth is the a number of bytes transmitted per second in the whole network, Size is the amount of bytes per message used to share stigmergy updates, and  $N$  the number of agents. The bandwidth is proportional to the square of  $N$  as each stigmergy in each robot is updated and propagated from that robot to the others. Thus, the size of the stigmergy depends on the application as well as the required update frequency. Measuring with iperf<sup>9</sup> in a  $125 \text{ m}^2$  room, the lowest bandwidth measured was between 1.9 and 10.5 MB/s. This metric can be used as an indication of the feasibility of an application with a given stigmergy and swarm size. However, the bandwidth is heavily affected by distance, so measurements should be made in the target environment before deployment to validate the actual available bandwidth.

Concerning latency, due to the multilevel nature of the communication system, each message must pass by multiple fixed length queues before being sent. For each level, for example between the two microcontrollers or the network, there is a inherent latency between the insertion in the queue and the message being sent and then removed. Pushing in this queue faster than the latency is a cause of message loss due to overflow. The current highest latency being 9 ms, the frequency of stigmergy updates is limited to approximately 100 Hz. This latency is introduced by the implementation of the communication protocol between the two microcontrollers.

### D. Hardware Integration

Since the SwarmUS platform aims to be installed on generic robots, it is important to validate how well it gets integrated on them. First, since the "Follow the Leader" task has been executed and worked on the Pioneers, the whole system is functional: the *HiveMind* sends the desired messages to the *HiveMindBridge* which in return cause

<sup>9</sup><https://iperf.fr/>

the desired actions of the robot. Also, the platform was adequately powered from the robot and didn't suffer from any power loss.

However, mechanical issues appeared during the platform's integration on the TurtleBot3 Burger. The weight added by the DC/DC converter, the *Hiveboard*, three *Beeboards* and their mechanical supports made the robot unable to cross obstacles as low as 1 mm without falling. On the other hand, the Pioneer 2DX, being bigger robots, had no stability or weight issues.

Another mechanical limit that has been observed with the space taken by the platform on the robot. The USB-C cables that go to the three *Beeboards* can take around the double of the space taken by the board themselves. This unexpected occupied space is taken by the stiffer USB-C cables that are mechanically constrained to not follow a direct path between the *Hiveboard* and its respective *Beeboard*. Placing other useful components for the robot in that space is difficult without adding worrying constraints on the USB-C cable.

Finally, the mechanical and electrical components close to the *Beeboards* antennas changed the response of the localization system. For instance, on the TurtleBot, the Raspberry Pi computer and lidar were close to these antennas and their interference in the localization system rendered a portion of the 0-360° circle unstable. On the Pioneer, the antennas were further away from other components and didn't suffer the same issue.

#### IV. DISCUSSION

The performance of the localization system can vary depending on the specific implementation. As the standard deviation is lower than the average absolute error, the implementation of the each system could be significantly improved by correctly mapping the system response to experimental results. Furthermore, fusion of the relative localization measurements of the surrounding agents or even sensor fusion with the robots sensors could improve the precision of the system. Changes in the environment and to the line-of-sight also changes the response of the system, increasing the absolute error average and the standard deviation. More tests should be conducted on the system to have a better understanding of how objects in the line-of-sight affects the system's response. However, it should be a good rule of thumb to install the *Beeboards* somewhere on a robot so no components are near or potentially in its line of sight.

The system also has a limitation regarding the scale of the swarm, like any other system that needs to share radio air space. The refresh rate of the localization system exponentially decrease with each new system that wants to be localized, the maximum being 28 Hz for two robots. Therefore, a smaller swarm can benefit from a fast refresh rate, where a bigger swarm might need to estimate these distances and bearings by other means between each measurement. A future improvement can be made to the scheduling mechanism by changing dynamically the number of time slots in the schedule to reflects the actual number of robots, thus always keeping the refresh rate to its optimal

value. It is also important to mention that the radio used for localization has a range of 9 m, other methods of localization could be used for longer distances. On the other side, the actual limitation to the number of robots simultaneously using the localization system is also linked to the area they occupy, so the 9 m range become less limiting for larger robots.

Furthermore, some improvements could be made to the deployment mechanism of the Buzz script. Currently, when making changes to the Buzz script, the whole *HiveMind* firmware needs to recompiled and flashed to every *Hiveboard* constituting the swarm. This makes development and deployment on a large swarm a tiresome experience [2]. This could be alleviated with an Over-The-Air update system via the network to update the Buzz script of every agents in the swarm without having to individually flash them.

An important feature that showed promising results is the simulation of the whole platform in ROS. We only had access to three robots during the development of the platform, but the simulation capabilities enables the development of Buzz scripts on a bigger scale. The cross-compilation feature made the transition between the design and the real implementation effortless. To enhance the realism of the simulation, the imprecision of the relative localization could be added to the simulated measurements. Overall, the ROS integration and simulation of the platform is a key feature towards a standard platform.

Currently, the communication system relies on having one device managing the Wi-Fi network for the entire swarm, which implies that the whole network can fail if this node is unreachable. The communication system could be improved by having an election mechanism in the swarm to dynamically choose an agent as a router or integrate the ESP-WIFI-MESH library<sup>10</sup>, which would be relatively straightforward and improve the robustness of the platform. Additionally, the management of the messaging system could be optimized to reduce the 9 ms latency in each communication, thus increasing the bandwidth of the communication system.

On the hardware side, some areas of improvements could be made. Firstly, a new integrated circuit by Decawave, the DW-3000, could upgrade the *Beeboards* by simplifying the circuits around the UWB chip and rendering the *Beeboards* more compact. Secondly, the hardware used to transmit the data and the synchronized clock needed between the *Beeboards* consumes around 35% of the total power. This power consumption could be diminished by choosing better suited components for the application, a choice we did not have while the silicon crisis was in motion [18]. Even if the total consumed power of 7 W is not an issue for mid-to-large size robots, the usage of lighter or even less components could diminish the power consumption without compromising the system accuracy as the clock distribution jitter is not the largest noise contributor in the distance and bearing measurements. For example, the redriver DS90LV001TLD/NOPB was added to overcompensate the signal integrity in the USB-

<sup>10</sup><https://www.espressif.com/en/products/sdks/esp-wifi-mesh/overview>

C cable and could be removed without significant impact to the performance of the clock distribution system. Lastly, components could be put on the underside of the boards, making the board much more compact. While it would complicate its assembly by hand, it would ensure a smaller footprint of the whole system. For instance, the TurtleBot3 Burger could benefit from such a reduction in the board footprint.

These straightforward improvements to the original prototype would thus reduce cost, space and power consumption for the overall platform, making it attractive to larger scale production and adoption as a viable solution to the swarm hardware standardization problem. The current design of the SwarmUS generic platform could fix three out of five key swarm robotics standards that needs to be established and followed by each swarm platform according to [3]. Those standards are 1) a common minimum processing power, 2) a common communication interfaces and 3) a standard localization system that each swarm robotics platform should have. The SwarmUS platforms ensures those standards whatever the robotics platform because of its fixed hardware. However, further research should be made to ensure that the current performance of the SwarmUS system is sufficient. The two other standards according to [3] are the standard sensors that a swarm platform should have and the standard tasks a swarm should be able to perform. We believe these standards to be dependent of the robot used and therefore out of scope for the SwarmUS platform alone.

## V. CONCLUSION

Distributed swarm of robots shows promising features compared to centralized or single robots in the accomplishment of specific tasks. However, it has been demonstrated that the lack of standards around swarm robotics platform hinders their arrival as viable solution to real life problems [3]. This paper presents the SwarmUS platform has a potential solution to the standardization problem by including many of the necessary features required by a swarm robotics platform on any model of robots. Even if the system has been successfully implemented on only two robots, the original hardware prototype performed correctly and showed promising results that could predict good performance of the platform on a larger swarm. Multiple solutions have been suggested to improve the prototype towards a generic swarm robotics platform that can be adopted by the swarm robotics community at large.

## ACKNOWLEDGMENTS

The SwarmUS project was supported by the ASEQ foundation and the Université de Sherbrooke. Part of the work was also supported by a NSERC Discovery grant. The team wants to acknowledge the precious exchanges and advices given by Pr. François Michaud, Jonathan Bouchard and the *Groupe de recherche en appareillage médical de Sherbrooke* (GRAMS) laboratory.

## REFERENCES

- [1] M. Waibel, B. Keays, and F. Augugliaro, “Drone shows: Creative potential and best practices.” 10.3929/ethz-a-010831954: ETH Zürich, Jan. 2017.
- [2] M. Dorigo, G. Theraulaz, and V. Trianni, “Swarm Robotics: Past, Present, and Future [Point of View],” *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, Jul. 2021.
- [3] N. Nedjah and L. Junior, “Review of methodologies and tasks in swarm robotics towards standardization,” *Swarm and Evolutionary Computation*, vol. 50, 2019.
- [4] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptoez, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, “The e-puck, a robot designed for education in engineering,” in *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1. IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.
- [5] M. Rubenstein, C. Ahler, and R. Nagpal, “Kilobot: A low cost scalable robot system for collective behaviors,” in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3293–3298.
- [6] J. McLurkin, A. McMullen, N. Robbins, G. Habibi, A. Becker, A. Chou, H. Li, M. John, N. Okeke, J. Rykowski, S. Kim, W. Xie, T. Vaughn, Y. Zhou, J. Shen, N. Chen, Q. Kaseman, L. Langford, J. Hunt, A. Boone, and K. Koch, “A robot system design for low-cost multi-robot manipulation,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 912–918.
- [7] M. Chamanbaz, D. Mateo, B. Zoss, G. Tokí, E. Wilhelm, R. Bouffanais, and D. Yue, “Swarm-enabling technology for multi-robot systems,” *Frontiers Robotics AI*, vol. 4, no. APR, 2017.
- [8] C. Pincioli and G. Beltrame, “Buzz: An extensible programming language for heterogeneous swarm robotics,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 3794–3800.
- [9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.
- [10] A. N. Bishop, P. N. Pathirana, and A. V. Savkin, “Target tracking with range and bearing measurements via robust linear filtering,” in *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE, 2007, pp. 131–135.
- [11] N. Shiell and A. Vardy, “A bearing-only pattern formation algorithm for swarm robotics,” in *10th International Conference on Swarm Intelligence, ANTS 2016, September 7, 2016 - September 9, 2016*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9882 LNCS. Brussels, Belgium: Springer Verlag, 2016, pp. 3–14.
- [12] A. Kohlbacher, J. Eliasson, K. Acres, H. Chung, and J. C. Barca, “A low cost omnidirectional relative localization sensor for swarm applications,” in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Feb. 2018, pp. 694–699.
- [13] N. Koenig and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Oct. 2004, pp. 2149–2154 vol.3.
- [14] “MIT License | Software Package Data Exchange (SPDX).” [Online]. Available: <https://spdx.org/licenses/MIT.html>
- [15] Decawave, “APS013 APPLICATION NOTE : The implementation of two-way ranging with the DW1000,” 2015. [Online]. Available: [https://www.decawave.com/wp-content/uploads/2018/10/APS013-The-Implementation-of-Two-Way-Ranging-with-the-DW1000\\_v2.3.pdf](https://www.decawave.com/wp-content/uploads/2018/10/APS013-The-Implementation-of-Two-Way-Ranging-with-the-DW1000_v2.3.pdf)
- [16] I. Dotlic, A. Connell, H. Ma, J. Clancy, and M. McLaughlin, “Angle of arrival estimation using decawave DW1000 integrated circuits,” in *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*. Bremen: IEEE, Oct. 2017, pp. 1–6.
- [17] I. Budinská and Š. Havlík, “Task allocation within a heterogeneous multi-robot system,” in *2016 Cybernetics & Informatics (K&I)*, Feb. 2016, pp. 1–5.
- [18] J. Voas, N. Kshetri, and J. F. DeFranco, “Scarcity and Global Insecurity: The Semiconductor Shortage,” *IT Professional*, vol. 23, no. 5, pp. 78–82, Oct. 2021.