Noname manuscript No.
(will be inserted by the editor)

HeRo 2.0: A Low-Cost Robot for Swarm Robotics Research

Paulo Rezeck · Héctor Azpúrua · Maurício F. S. Corrêa · Luiz Chaimowicz

Received: date / Accepted: date

Abstract The current state of electronic component miniaturization coupled with the increasing efficiency in hardware and software allow the development of smaller and compact robotic systems. The convenience of using these small, simple, yet capable robots has gathered the research community's attention towards practical applications of swarm robotics. This paper presents the design of a novel platform for swarm robotics applications that is low cost, easy to assemble using off-the-shelf components, and deeply integrated with the most used robotic framework available today: ROS (Robot Operating System). The robotic platform is entirely open, composed of a 3D printed body and open-source software. We describe its architecture, present its main features, and evaluate its functionalities executing experiments using a couple of robots. Results demonstrate that the proposed mobile robot is capable of performing different swarm tasks, given its small size and reduced cost, being suitable for swarm robotics research and education.

Paulo Rezeck and Hector Azpúrua
Graduate Program in Computer Science,
Computer Vision and Robotics Laboratory,
Department of Computer Science,
Universidade Federal de Minas Gerais, Brazil.
E-mail: {rezeck, hector.azpurua}@dcc.ufmg.br

Hector Azpúrua is also with
Instituto Tecnológico Vale,
Ouro Preto, MG, Brazil.
E-mail: hector.azpurua@itv.org

Maurício F. S. Corrêa
Computer Vision and Robotics Laboratory,
Department of Computer Science,
Universidade Federal de Minas Gerais, Brazil.
E-mail: mauricio.ferrari@dcc.ufmg.br

Luiz Chaimowicz
Computer Vision and Robotics Laboratory,
Department of Computer Science,
Universidade Federal de Minas Gerais, Brazil.
E-mail: chaimo@dcc.ufmg.br

Keywords Swarm robotics · Mobile Robot · Autonomous Robot

1 Introduction

Robotic swarms are potentially becoming well suited for a wide range of real-world problems with a high societal and economic impact. The requirement of distributed and decentralized processing relying only on local information brings several practical advantages over other robotic systems allowing scalability, resiliency, and adaptability. This aspect further leverages the use of swarm robots in agriculture, the mining industry, warehouse management, and robotics education.

In spite of the increasing application potential of real-world robot swarms, several challenges are still open, ranging from efficient processing and communication to robust locomotion and sensing. In addition, one of the main challenges in employing swarm-based solutions in the real world is the development of capable yet affordable robotic platforms.

Moreover, despite the existence of off-the-shelf solutions and some open software and hardware efforts, the cost of the platforms and the logistics make it difficult for many researchers or educators to acquire or reproduce most of them. In order to alleviate such issues, the robotic platform presented in this work takes advantage of the recent technological advancements to use mass-produced components that are smaller, affordable, and long-term available. In addition, the design and assembly process follow new trends, such as the *Maker Movement* and *Do It Yourself*, which allow others to reproduce and customize the robot using additive manufacturing.

In this sense, we assume the following requirements as an objective to build a swarm-capable robotic platform:

- **Affordability**: Robots should be as inexpensive as possible since most swarm teams may have tens or hundreds of robots;
- **Small and yet capable**: Robots should be small and equipped with some form of sensing capability to allow interaction with their environment; Also, they should have a long-term power autonomy since the swarm may need to operate long enough for the collective behavior to emerge;
- **Reliability**: Robots should be highly fault-tolerant;
- **Scalability**: They should be able to successfully perform different tasks even when the number of robots increases. In this sense, communication capabilities should support a large number of robots;
- **Easily reproducible**: Robots must be easily assembled and must not use hard-to-acquire or extremely hard-to-assemble components;
- **Easily programmable**: Robots should be easily programmable and compatible with modern robotic frameworks and development pipelines.

Satisfying these conditions in a single design is a difficult challenge. The design choices concerning one requirement, such as size, produce additional constraints to others, such as sensing and powering. Consequently, the design process should simultaneously take all of these constraints and find convenient design solutions for multi-purpose applications.

Assuming the above requirements for a capable swarm robotic device, we present the design of HeRo[1], a significant low-cost robot (18 USD) composed of a 3D printed body and off-the-shelf components (Fig. 1). The robot is entirely open-source and carries a diverse set of sensors that makes it suitable for a wide range of swarm applications and education efforts. The platform is deeply integrated with the highly popular Robot Operating System (ROS) framework for quick prototyping, allowing remote and local robot control using standard programming interfaces. In order to facilitate the development of swarm algorithms, we also provide a simulated robot model with a realistic test environment in Gazebo. This work is an evolution upon the first, simpler, version of the HeRo platform presented in [30].

The real-world sensorial and locomotion performance of the HeRo platform was evaluated in comparison to other popular commercial robot platforms. Some of the metrics for comparison are odometry accuracy, range, and quality of IR sensors (when used as range sensors), power consumption and autonomy, communication robustness, and scalability potential. Further, we also evaluated the HeRo performance in some cooperative tasks such as flocking, transportation, and mapping. Besides being relatively small in size, results show that the HeRo platform is significantly capable, making it cost-effective and suitable for swarm applications.

[1] HeRo 2.0: `https://verlab.github.io/hero_common`

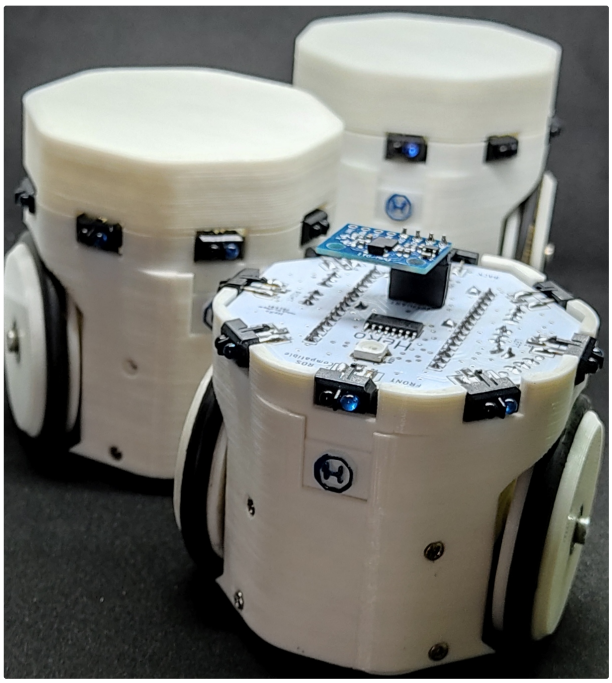


Fig. 1: Design of the proposed open swarm robotic platform. The body of the robots was designed for and fabricated using additive manufacturing.

The remainder of this paper is structured as follows. A review of the literature of small robot platforms and systems is presented in Section 2. The mechanical and electrical design, as well as the software and communication architecture, are presented in Section 3 and Section 4, respectively. The robot's performance with respect to a set of metrics is evaluated experimentally in Section 5. In Section 6, we present the use of HeRo in some swarm applications. Finally, Section 7 brings the conclusions and directions for future work.

2 Related Work

Swarm robots have some elementary features that differentiate them from other types of platforms, such as simplicity, the capacity to scale and cooperate, size, and communication capabilities, among others [23]. Especially, some critical aspects for a swarm robot are footprint and cost, as those two aspects will facilitate the scalability of a real-world swarm system. Modern robotic systems also leverage the ecosystem and modularity of the Robot Operating System (ROS) [27] to improve the development environment and allow realistic simulations.

In this sense, a wide range of small and relatively simple robots have been proposed for swarm applications. Most of these platforms are open or have open-source parts, while only some of them are closed source or only available commercially. In this section, we present

the most prevalent and relevant platforms for general swarm experimentation and highlight the most important pros and cons of each one. We divide them according to their locomotion mechanisms and restrict this comparison to small robots (less than 10 cm), which are generally more suitable for swarm robotics. Table 1 presents a summary of this comparison.

2.1 Vibration-based platforms

Recently, robots using vibration-based motion mechanisms have become more common. In general, such mechanism can be easily coupled to the robot, but requires an extra effort in the robot's motion control algorithms. In addition, it requires a smooth experimentation surface and may have a relatively slow movement. Moreover, there is no real form of odometry, making it challenging to move precisely over long distances or perform for a long time if this information is necessary.

The **Kilobot** [31], developed at *Harvard University* - USA, is one of the most popular swarm robots. It is an open-source platform with parts costing only 14 USD. But it is also produced and distributed as a commercial product for 100 USD. The robot has an ATmega328 (8-bit at 16 MHz) microcontroller and is equipped with an ambient light sensor on top and an IR sensor on the bottom used for proximity readings and communication. The robot has an alternative moving principle based on two vibration motors, reducing cost and size, but it also limits the robot's maximum speed up to 1 cm/s. An overhead controller device is used to communicate via IR with all robots enabling remote control and uploading the robot's firmware over the air. Even with a relatively high commercial cost and limited sensing, research groups were able to successfully carry out experiments with up to 1000 robots [33], showing that Kilobot may be an interesting platform for swarm applications.

The **Droplet** [14, 9] is another vibration-based small robot developed at the *Correll Robotics Lab* at the *University of Colorado Boulder*, USA. Despite being slightly larger than Kilobot, this robot features improvements in the mechanism of locomotion and sensing. The robots carry six IR sensors for proximity, bearing, and robot-to-robot communication. For locomotion, it uses three vibration motors to allow omnidirectional control of the robot, which is very convenient given its low speed (1 cm/s). An Xmega128a3u (16-bit at 32 Mhz) microcontroller is also an improvement over Kilobot, allowing control, data processing, and general-purpose computation. In addition, the Droplet can also perform continuous experiment runs due to a powered floor equipped with alternating positive charge and ground stripes. Besides powering, this feature is also suitable for data transmission, enabling programming an entire swarm directly via the floor. The commercial cost of this robot is similar to the Kilobot (100 USD), but it still requires a powered floor mechanism to power the robots.

2.2 Wheel-based platforms

Although vibration-based locomotion does not require any complex mechanism to actuate the robot, such approach proves unsuitable for precise movements over long distances, mainly due to their nonlinear behavior and excessive slippage towards undesired directions. On the other hand, wheel based systems are more practical to control and efficient given that the torque generated by the motor acts directly and roughly linearly on the wheel. Below we list some wheel-based robotic platforms.

Khepera [22, 21] is one of the early small robots developed in the mid-1990s at the *LAMI laboratory at École Polytechnique Fédérale de Lausanne* (EPFL), Switzerland. The original version of Khepera is a small (5.5 cm) differential wheeled mobile robot that has been used by researchers of several universities for different applications. Two DC brushed servo motors with incremental encoders actuate and control the robot's wheels to reach up to 100 cm/s. A Motorola 68331 (32-bits at 16 MHz) microcontroller running μKOS RTOS serves as the robot's main processor, enabling motion control, sensing, and communication. In addition to eight IR sensors used to estimate distance and ambient light, the robot also allows extra modules that expand its functionality. Some examples are gripper-like manipulation, vision, and robot-to-robot communication modules. Over the years, several versions of Khepera have been developed, improving unit processing, locomotion, and sensing but requiring an increase in size. The latest version, the Khepera IV [34], is still a differential wheeled mobile robot with a diameter of 14 cm. This robot houses twelve IR sensors, five ultrasound sensors, two microphones, encoders, an inertial measurement unit (IMU), and a camera. The main processing unit is a Gumstix embedded computer running GNU/Linux, and Bluetooth allows robot-robot communication or communication with a remote server. Its commercial version retails for 3180 USD.

Alice [7] is another small robot developed for swarm applications at the *Autonomous Systems Lab at École Polytechnique Fédérale de Lausanne* (EPFL), Switzerland. Alice is a two-wheeled differential drive robot made of a light plastic chassis with PCB on top. The robot has a small footprint of 2.2 cm and uses two high-efficiency swatch motors for locomotion reaching up to 40 cm/s. A low-power PIC16F877 (8-bits at 4 MHz) microcontroller controls the robot and executes other applications. Alice has various built-in sensory modules such as 4 IR sensors mounted around the robot for obstacle detection and short-range robot-to-robot communication. An IR receiving on top allows the robot to receive

Table 1: Comparison of popular swarm robotics platforms.

Robot	Cost (USD)	Size (cm)	Motion/Speed (cm/s)	Communication	Autonomy (h)	Builtin Sensors	Open Source	ROS Enabled	FW Programming
Kilobot [31]	100/14*	3.3	vibration, 1	IR	3 – 24	proximity, light	✓	-	OTA
Droplets [14]	100	4.4	vibration, 1	IR	∞	distance, light, bearing	✓	-	wired
Khepera I [22]	N/A	5.5	wheel, 100	IR/RF	1–	distance, light, encoder	-	-	wired
Alice [7]	N/A	2.2	wheel, 40	IR/RF	1 – 10	distance, light	-	-	wired
AMiR [3]	85*	7.5	wheel, 10	IR	2	distance, light, bearing	-	-	wired
E-puck [20]	975	7.0	wheel, 13	Wi-Fi/Bluetooth	1 – 3	distance, light, camera, mic, imu	-	✓	OTA
Jasmine [12]	120*	3.0	wheel, 30	IR	1 – 2	distance, light, bearing	✓	-	wired
GRITSBots [26]	50*	3.0	wheel, 25	RF	1 – 10	distance, bearing, imu	✓	-	OTA
Zooids [16]	50*	2.6	wheel, 44	RF	1 – 2	touch sensor	✓	-	wired
mROBerTO [13]	60*	1.6	wheel, 15	Bluetooth	1 – 6	distance, light, imu, camera	✓	-	OTA
WsBot [17]	17*	3	wheel, 3.5	Wi-Fi	4	-	-	✓	wired
MicroMVP [38]	90*	8.0	wheel, 25	Zigbee	1 – 2	-	✓	✓	wired
Cellulo [24]	140*	7.5	wheel, 18	Bluetooth	2	touch, visual odometry	-	-	OTA
Colias IV [11]	100*	4.0	wheel, 35	Bluetooth	1 – 3	distance, light, camera, mic, imu	✓	-	wired
Mona [1]	120	6.5	wheel, 15	RF	∞	distance, light, encoder	✓	✓	wired
HeRo	**18***	**7.3**	**wheel, 25**	**Wi-Fi**	**3 – 9**	**distance, light, encoder, imu**	✓	✓	**OTA**

*parts only

external commands, and a radio frequency (RF) module is used for remote communication. In addition, the robot supports different expansion modules, such as a gripper module and a linear camera. The first design of Alice used two watch batteries allowing the robot to operate for up to 10 hours. Further evolution of the platform allows the use of solar panels.

AMiR [3] is a two-wheeled differential drive small robot developed at the *University Putra*, Malaysia. It is an open platform, and the components required to assemble it cost about 85 USD. The robot's footprint is 7.5 cm, and two micro DC internal gear motors actuated the robot with a maximum speed of 10 cm/s. An ATmega168 (8-bits 8 MHz) microcontroller is used as the main processor to control all functions such as communication, trajectory, and perception, among others. The robot carries 6 IR sensors enabling proximity and bearing estimation and also short-range robot-to-robot communication. The robot uses a 3.7 V 200 mAh lithium battery allowing it to operate up to 2 hours. In addition to the physical platform, AMiR has been successfully simulated in Player/Stage and has been used by several researchers and robotics educators [2, 4].

The **E-puck** [20] is one of the most successful small-size commercial robots. Initially designed for education it has also been used for swarm robotics research. The E-puck is a two-wheeled differential drive robot, and its retail cost is about 975 USD. The robot has a small footprint of 7.0 cm and uses two planetary-geared step motors for actuation, reaching up to 10 cm/s. The latest version of the e-puck is powered by an STM32F4 (32-bits at 180 MHz) microcontroller, and an Espressif ESP32 is used as Wi-Fi/Bluetooth module. The robot hosts various built-in sensors, including microphone arrays, proximity sensors, a 640×480 pixels camera, and an inertial motion unit. The robot can be programmed through a serial BUS or Bluetooth interface, and Wi-Fi is used for communication. In addition, it can be extended with other sensing modules, such as bearing and an omnidirectional camera module, and even processing modules using Raspberry Pi. The robot uses a 3.7 V 1200 mAh lithium battery allowing it to operate up to 3 hours. A growing user community provides software, documentation, and discussion groups favoring the platform's integration with various simulators and robotics frameworks, such as Gazebo and ROS. Despite its benefits, the commercial version of the basic E-puck is quite expensive, making it not affordable for large swarms.

Jasmine [12] is another widely used two-wheeled differential drive small robot. Developed at the *University of Stuttgart*, Germany, its parts cost about 120 USD. The Jasmine robot has a small footprint of 3 cm and uses two micro DC internal gear motors to actuate the wheels, reaching a maximum speed of 30 cm/s. The third version of the robot is equipped with an ATmega168 (8-bits at 20 MHz) microcontroller, and uses 6 IR sensors for proximity and bearing estimation, light measurements, and communication with other robots. The robot also has LEDs on top, allowing status monitoring or debugging. In addition, many customized boards can extend the robot's capabilities, including improved sensing and connectivity. The current version of Jasmine uses a 3.7 V 250 mAh lithium battery that has enough capacity for running time up 2 hours. Moreover, the robots can autonomously recharge the bat-

tery by touching a pair of metal contacts (power and ground) attached to the wall for convenience. Thus, the robot detects when its battery needs to be recharged and moves autonomously to the dock without human intervention.

GRITSBot [26] is a small robot developed at *Georgia Institute of Technology*, USA. GRITSBot is part of *Robotarium*, a project to make multi-agent experiments more accessible to the research community, opening up a showcase testbed to the general public [25,37]. The GRITSBot is another wheeled differential-drive robot composed of three modular layers that house five functional robot blocks. The motor layer is responsible for controlling the two stepper motors and odometry estimation. The mainboard houses an Atmega328 (8-bit at 16 MHz) microcontroller, the wireless communication module, the battery charging circuit, and the power supply. A Nordic nRF24L01 microchip serves as a low-power consumption communication module operating at 2.4 GHz. This module enables robot-to-robot communication, over-the-air firmware reprogramming, and remote control from a server. The sensor layer includes six infrared distance sensors, an accelerometer, and a gyroscope. A 400 mAh LiPo battery supplies the robot allowing long-time power autonomy up to five hours. The robots can also move autonomously to a power source and automatically recharge the battery conveniently.

Zooid [16] is a small robot platform designed for swarm applications available at an approximate cost of 50 USD. This robot is an open-source open-hardware platform created as a joint work between the Shape Lab at *Stanford University* (USA) and the Aviz team at *Inria* (France). The motors are mounted in a non-collinear fashion, allowing a small footprint of only 2.6 cm. Even though the motors do not rotate around the same axis, the robot has a similar net force and moment as a robot with colinear motors. An STM32F4 (32-bit at 48 MHz) microcontroller manages the overall logic computation and communicates wirelessly with the main master computer using an nrf24L01 2.4 GHz radio chip. In addition, the robot is equipped with touch sensors for tactile swarm applications and some on-top photodiodes used for localization. A projector-based tracking system is used for robot position tracking. This device projects a sequence of gray-coded patterns onto a flat surface, enabling the robots to use their photodiodes to decode the gray code into position and orientation. Unlike classical camera-based systems, this projector-based tracking system does not add any latency from networking for the local feedback control on each robot, making position control more stable. However, this localization system costs approximately 700 USD and archives similar resolution compared to overhead-camera localization systems.

The **mROBerTO** [13] is a small footprint (1.6 cm) robot developed at *University of Toronto*, Canada. Despite the small footprint, the robot features several built-in sensors such as distance, ambient light, IMU, and camera, making it interesting for swarm applications. In addition, the robot supports extensions such as a module with 8 IR sensors for obstacle detection. The robot's mainboard is a Nordic nRF51422 microchip composed of an ARM Cortex-M0 (32-bits at 16 MHz) with built-in Bluetooth Smart and ANT+ capability. The nRF51422 board supports over-the-air programming, saving time when setting up several of robots simultaneously. Regarding the robot actuation mechanism, the first version of mROBerTO did not require wheels and used the motor shafts directly in contact with the floor surface to move the robot. Despite being a compact actuation mechanism, allowing the robot to reach speeds of up to 15 cm/s, it requires a smooth contact surface for proper robot control. In more recent versions, the authors improved the actuation mechanism to utilize small stepper motors with wheels [8].

The **WsBot** [17] is another small footprint (3.3 cm) robot developed at *Universidade Tecnológica Federal do Paraná*, Brazil. This robot has a design similar to mROBerTO but at an assembly cost of only 17 USD. In addition to being extremely inexpensive, one may easily assembly this robot using only off-the-shelf parts. The WsBot is envisioned for demonstrations of applications in Industry 4.0, so the robot has a built-in wireless charging system allowing automatic battery recharges for continuous operation. Two micro DC motors with small wheels drive the robot allowing it to reach a speed of up to 3.5 cm/s. An Espressif ESP8266 (32-bits - 160 MHz) microchip, with built-in Wi-Fi, enables remote control of the robot using a server executing ROS. The robot does not feature any built-in sensor or extension boards. Instead, a global localization system based on an overhead camera and fiducial markers is used for robot close-loop control.

MicroMVP [38] is a small robot developed at MIT, USA. It has an open-source design utilizing 3D printing technology, and it is also extremely simple and easy to assemble. MicroMVP was designed to use only off-the-shelf components, and it is composed of an ATmega32U4 (8-bit at 16 MHz) microcontroller with built-in xBee support and two geared motors. As WsBot, MicroMVP does not provide built-in sensors reducing its applicability as a swarm-capable robot. It also uses an overhead camera and fiducial markers attached on top of the robot to localize them, serving as closed-loop control. However, MicroMVP uses more expensive components reaching an assembling cost of 90 USD.

Cellulo [24] is one of the world's first tactile small robot platforms developed at *École Polytechnique Federale de Lausanne* (EPFL), Switzerland. It combines autonomous capabilities with haptic-enabled multi-user tactile interaction allowing research on rehabilitation, gaming, and human-computer interaction. The robots are designed to be small, sturdy, low-cost, and simple to

operate. The current Cellulo robot is equipped with a self-localization system based on an activity sheet and a downward-facing camera, holonomic motion, six capacitive touch buttons, Bluetooth communication, and a low-cost PIC32MZ (32-bit at 200 MHz) microcontroller. The localization system [10] enables the user to estimate the global pose of many robots and also is robust against kidnapping and occlusions (usually due to user manipulation).

Colias is a novel alternative to AMiR developed at the *University of Lincoln*, UK for swarm robotic applications. Colias sensor unit is based on extension boards to achieve better modularity. In this way, each part has different features and functions that can work independently. The mainboard uses an ATmega168 (8-bit at 8 MHz) microcontroller to control the motors and power management. This board houses IR sensors that provide proximity measurements used for obstacle detection. The motion is produced by two differential-driven wheels reaching a maximum speed of 35 cm/s. The new Colias IV [11] is additionally powered with a high-level ARM Cortex M4 microcontroller running at 180 MHz, two digital microphones, one 9-axis motion sensor, and a tiny VGA camera to enable visual tasks. A Bluetooth extension module enables Colias IV to communicate with a remote host device such as a laptop or a smartphone, receiving motion commands or sending sensor data.

Mona [1] is a small open-source robot built as a customized design of Colias. It has also been designed as a modular platform, allowing additional modules, such as wireless communication or a vision board. Mona is mainly designed to investigate the feasibility of the proposed Perpetual Robotic Swarm [5]. The robot is specially designed to use an inductive charging approach and several additional functions such as a radio frequency (RF) transceiver and battery level monitoring module. This perpetual charging interface allows for large-scale, long-term autonomy robotics research. Mona has also been developed to be compatible with several standard programming environments, and it has been successfully used for both education and research at the *University of Manchester*, UK. The robot has been produced as a low-cost platform for robotic education and swarm research in collaboration with a commercial partner. It retails at 120 USD per robot, and it remains fully open-source (hardware and software).

2.3 Design Choices

In Table 1, we can see that some design choices are common to the majority of platforms, such as the use of wheels for locomotion and the presence of distance sensors as part of the robot's sensor suite. On the other hand, some important features are present in only a few designs, such as Wi-Fi communication and ROS compatibility, probably because these technologies have become more affordable and available recently. In HeRo, we tried to mix the well-tested and common solutions with some novel enhancements, making the robots more capable yet simple and reliable. The next section summarizes HeRo's main features, which will be detailed in the remainder of this paper.

2.4 Proposed platform: **HeRo**

In this paper, we present the project and implementation of a novel small robot for swarm robotic applications. The current proposal is an evolution upon the first, simpler version of the HeRo platform which was briefly presented in [30]. A summary of the characteristics of all HeRo versions is described in Table 2. In this improved version, the mainboard uses an Espressif ESP8266 (32-bits 160 MHz) microcontroller to perform the motors' control and acquire and process sensor data. This microcontroller houses a built-in Wi-Fi module, allowing the robots to communicate among themselves robustly and reliably, using TCP/IP protocols. The locomotion system consists of using two differential-driven wheels reaching a maximum speed of 25 cm/s. The board houses a set of sensors such as 8 IR sensors that provide light and proximity measurements for obstacle detection, an inertial motion unit for improved odometry and general use, and two rotary encoders for localization and motion control. The mainboard is also modular, allowing the user to attach several other components such as a camera, motors, displays, and transistors for communication or localization. To facilitate programming, HeRo supports FOTA (Firmware Over-The-Air) using a Wi-Fi interface. Such technology allows the users to upload their codes on many robots remotely. Moreover, HeRo is also a ROS-compatible robot and communicates using a TCP/IP connection with a remote computer executing ROS. Since the robot's autonomy is an important factor considering the time and number of experiments, HeRo provides a long-time autonomy using a powerful Li-Po battery. The main contributions of the proposed platform compared to its counterparts are its balance between low cost and capacity, simplicity in terms of assembly, and seamless integration with ROS allowing easy programming.

3 Mechanical and Electrical Design

This section presents the mechanical and electrical design of our swarm robot. All decisions considered the maximum use of commercially available components for ease of production and assembly and minimum possible price without sacrificing the processing power and sensing capabilities. Therefore, in the following, we present

Table 2: Characteristics of the different HeRo versions.

	HeRo v0.1	HeRo v1.0	**HeRo v2.0**
Board	Arduino Nano	ESPressif ESP8266 - ESP12	ESPressif ESP8266 - ESP12
MCU	Atmel Atmega328 8-bit @ 16 MHz	Tensilica LX106 32-bit @ 80/160 MHz	Tensilica LX106 32-bit @ 80/160 MHz
Communication	RF nrf24l01 2.4 Ghz	Wi-Fi 802.11bgn	Wi-Fi 802.11bgn
Actuation	Servo Motors	Servo Motors	Servo Motors
Footprint	10 cm	8 cm	7.3 cm
Sensors	None	3 x infrared sensors	8 x infrared sensors, encoders and IMU
Battery	3.7 V 1000 mAh Li-Po	3.7 V 1000 mAh Li-Po	3.7 V 1800 mAh Li-Po
Cost*	9 USD	14 USD	18 USD

* parts only

the best-suited system for HeRo after evaluating multiple microcontroller boards, wireless technologies, sensors, actuators, and model designs for additive manufacturing.

3.1 Mechanical Design

One of the primary steps in developing a mobile robot is modeling its mechanics. The process defines the kinematic model of the robot, its actuation mechanism, and also its structural design.

3.1.1 Kinematic Model

After reviewing the literature, we observed that most robots proposed for swarm robotics are based on the differential-driven model. In a nutshell, a differential wheeled robot is a mobile robot whose movement is based on two separately driven wheels placed on either side of the robot body. The robot changes its direction by varying the relative speeds of its wheels, and therefore it does not require an additional steering motor. We also decided to implement such model since it is very suitable for designing a small, low-cost robot that requires good maneuverability and speed using a simple actuation mechanism.

3.1.2 Actuators and Encoders

A convenient and affordable way to actuate the wheels of a differential robot is the use of geared DC motors. Besides actuating the robot, these motors enable the use of encoders for computing odometry, which is important for localization and closed-loop motion control.

Although it is intuitive to use geared DC motors with encoders, such components can significantly increase the cost of the robot. Thinking of a low-cost solution, we decided to use small continuous servo motors to actuate the robot's wheels. Such motor is similar to geared DC motors with an h-bridge component, allowing motor speed control.

Besides having reasonable precision for speed control, continuous SG90 servo motors contain a built-in microchip that controls the motor speed and direction using only pulse-width modulation (PWM) signals. In addition, this motor has a significant torque of 1.8 kgf/cm, which allows us to use a 5 cm diameter wheel to reach a maximum linear speed of 25 cm/s with 0.3 kgf/cm of torque without losing traction.

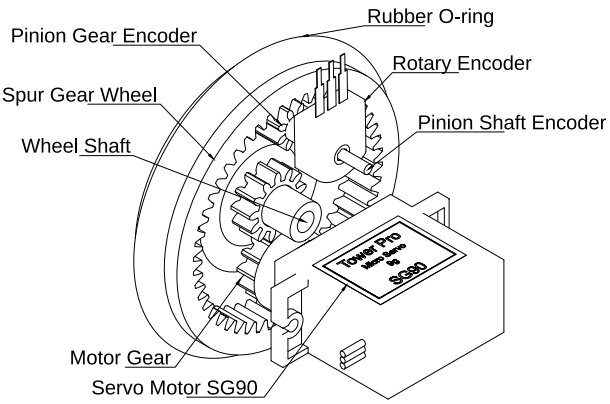


Fig. 2: Torque transmission mechanism from the motor to wheel and wheel to the rotary encoder.

Instead of directly connecting the wheel to the motor shaft, we attach the wheel to the robot chassis and use a gear mechanism (1:1) to transmit torque from the motor to the wheel – this further reduces backlash and wheel misalignment that impact the encoder readings.

Since such motor does not have a built-in encoder, we took advantage of larger wheels to design a mechanical transmission system (1:6) between the wheel and a mechanical rotary encoder. By considering the low cost, availability, and compact form, we selected Kailh rotary encoders. This encoder is widely used on mouse devices as a step counter for the scroll button, and the simplest models, like the ones we use, can count 48 steps per cycle. However, performing the wheel-encoder transmission (1:288) increases the wheel position measurement to 1.25° degrees of resolution, which means that the robot detects a wheel step of 0.54 mm when it is moving. Fig. 2 illustrates the motor-wheel and the wheel-encoder transmission system. The motor, rotary encoder, and wheel shaft are fixed to the robot chassis, and the other parts are moving.

Table 3: General specifications of the robot.

Specification	Value
Size	$0.068 \times 0.073 \times 0.076$ $(L \times W \times H)$ m
Weight	0.156 Kg
Moment of Inertia	$I_{xx} = I_{yy} = 1.27e^{-4}$ and $I_{zz} = 1.04e^{-4}$ Kgm^2
Wheels Distance	0.0631 m
Wheel Diameter	0.0492 m
Linear Speed	0.25 m/s

3.1.3 Structural Design

After defining the actuation mechanisms, we proceed with designing the robot's chassis. To facilitate assembly and further extensions, we design the robot's chassis to be modular and 3D-printable so one can easily print it using a conventional 3D printer. Overall, the robot structure comprises four main parts: the motor and board chassis, cover, and the e-Hat module.

The **motor chassis** supports both motors and the wheels shaft – where the wheels are attached. As the robot has two actuated wheels, it has only two contact points on the ground. To better adjust the balance and alignment of the robot, we created two screwable castor wheels. These castor wheels are attached to the motor chassis and allow us to fine-tune the robot's balance. On top of the motor chassis, we attach the **board chassis** that holds the encoders, battery, and the main processing board.

Considering further extensions, we take advantage of a modular chassis to coin the concept of **e-Hat**. Such part is attached on top of the robot and works as a shield extending the robot's sensorial or acting capabilities. For instance, we developed an e-Hat with an IMU sensor. Other components, such as a camera, sonar, actuator, or even a UWB transceiver for indoor localization, can also be used.

Finally, we design a **cover** part to prevent dust accumulation inside the robot, protecting the main processing board and gears. In addition to protection, this part also enhances the robot's visual aesthetic. Fig. 3 shows an expanded view of the robot's design, and Table 3 shows some specifications of the robot. An interactive CAD visualization is available at A360 platform[2].

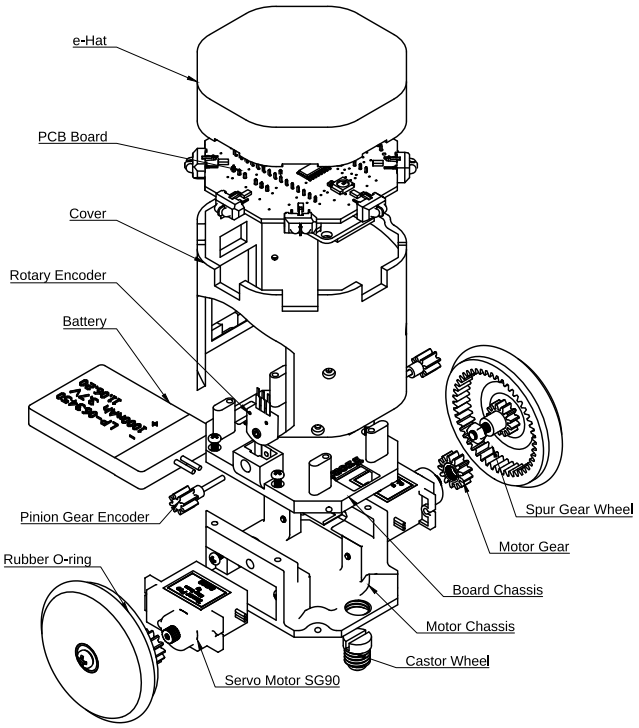


Fig. 3: An expanded view of the robot's components and body parts.

3.2 Electrical Design

In addition to the robot's mechanical design, we also present its electrical design. This process defines the electronic components built in the robot, such as processing unit, sensing, and power management.

3.2.1 Microcontroller

One of the major decisions concerning the robot's electrical design is the selection of an appropriate microcontroller. This component defines the robot's computational capacity and the number and variety of components we can use.

After considering many alternatives, we select the Espressif ESP8266 as the main processing unit. This microcontroller is remarkably inexpensive given its excellent processing power (32-bit 160 MHz) with 4 MB of memory. Also, it has a built-in Wi-Fi microchip that provides a fast IEEE 802.11 connection with a full TCP/IP stack. So, the robots can communicate among them or with a remote computer using robust and scalable protocols. Moreover, it is efficient, easy to program, and widespread in Maker communities, allowing others to easily develop customized modules for the robot.

3.2.2 Sensing

A important requirement for a small robot used in swarm experiments consists of measuring distances to neighboring robots and obstacles. In HeRo, we chose to use infrared sensors for this due to their size and cost. We arrange eight IR transmitters and receivers (TCRT5000) around the circumference of the robot with 45° increments. We selected such sensor because it is cost-effective and has a reasonable resolution and range.

Although it is not common using this sensor for long distances (> 10 cm), and even the manufacturer defines the sensor's maximum range[3] to only 2 cm, we found

[2] Robot Design CAD: `https://a360.co/3lWHiv0`

[3] TCRT5000: `www.vishay.com/docs/83760/tcrt5000.pdf`

a strategy to increase its range without a considerable decrease in accuracy. By properly operating how the emitter LED activates, we use a technique known as Pulsed Over-Current Driving LED [18] to increase the detection range to up to 20 cm. In short, when voltage is applied to the poles of the IR emitter LED, for a short time, the resistance is low (low conductor temperature), which allows a high current flow (up to 3 A for $t < 25\mu$s), causing the LED to emit IR light with high intensity. If we keep the LED activated, this resistance tends to increase and stabilize, reducing the intensity of the light (60 mA max). In our setup, we generate pulses with a duration of 100 microseconds at 0.2% duty cycle. Thus, as the overcurrent pulse duration is short enough and the blanking duration for cooling off long enough, even the cheapest and most commonplace IR LEDs can be driven with extreme currents. This technique may reduce the IR LED life span, but probably the LED should work for more than a year.

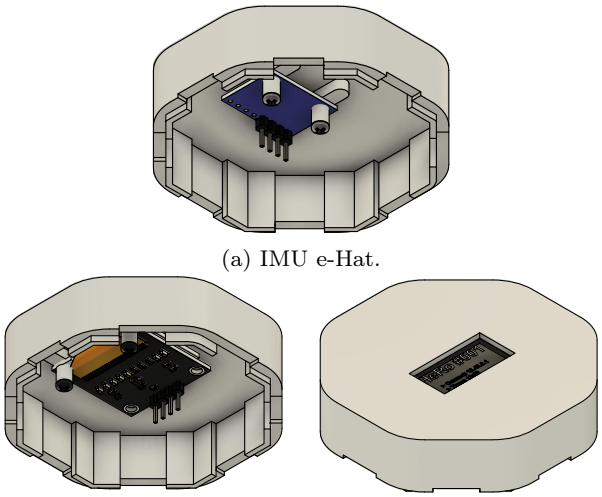
To control the IR LED activation, we use a MOSFET component. Due to the limited number of ADC pins on the microcontroller – it only has one pin with 10-bits of resolution – we have to include an 8-channel analog multiplexer enabling the microcontroller to read all the eight IR phototransistor. This setup allows us to precisely measure distances, avoiding environment light interference since we can measure it by using only the IR receivers.

As mentioned earlier, the robot has two pairs of rotary encoders coupled to the wheels by a drive mechanism. An encoder is a sensor that generates digital signals in response to the motion, providing information about position, velocity, and direction. As the typical mouse wheel internally works as a precise encoder, we took advantage of this inexpensive component (less than USD 0.10) and used it as a robotic sensor. It comprises a conductive disc and three contacts that generate two square waves in quadrature when the encoder shaft rotates, enabling counting 48 pulses per shaft revolution and also identifying the turn direction.

In addition to the encoders, the robot also houses two WS2812b RGBA LED indicators used for status monitoring or debugging. These addressable LEDs have an IC built right into the LED, allowing communication via a one-wire interface (it uses one digital pin to control multiple LEDs in series). We can also control the brightness, and the color of each LED individually, which allows us to produce unique and complex effects for status in a simple way.

3.2.3 E-Hat

Besides the built-in features, the robot functionality can be extended using e-Hats. This robot module works as a shield, allowing users to create/customize specific modules for their applications. This module is coupled to a 4-pin bus on top of the robot configured to use I2C or UART protocols. In addition to communication, the bus also provides 5V (800 mA) to power the module. In this paper, we developed two e-Hats for demonstration and experimentation (see Fig. 4). The first one consists of an e-Hat IMU composed of an MPU6050 sensor with gyroscope and accelerometer that can be fused into the velocity and position estimation to account for odometry errors, such as the slip produced by the wheels. The second one is an e-Hat display that can be used either as a user interface or as part of a location system based on camera and fiducial markers.



(a) IMU e-Hat.

(b) Display e-Hat.

Fig. 4: Example of multiple e-Hat versions for the HeRo platform.

3.2.4 Power Supply

Since the robot's autonomy is an important factor considering the time and number of experiments, HeRo uses a 3.7 V 1800 mAh Li-Po battery. The battery voltage is regulated by a MT3608 DC-DC step-up module, managing the board power supply to 5 V. These components enable the robot to perform up to 3 hours of experiments, considering the continuous use of all components. The motors are directly powered by a step-up power module avoiding any voltage drop impacting the robot's speed. In addition to this module, we also use a TP4056 module to recharge the battery using a USB cable.

3.2.5 Assembly

Because most of the robot's parts are off-the-shelf components, we decided to simplify the mounting and wrap them right on a PCB board. To increase reproducibility, we carefully design this PCB board so that even novice users can assemble it. As an alternative, the user

Table 4: Parts cost per robot unit.

Parts	Quantity	Cost (USD)
Servo Motors SG-90	2	2.06
Mouse Encoder 48 PPR	2	0.10
ESP8266 Nodemcu	1	2.50
Rubber O Ring 38mm	2	0.10
IR TCRT5000	8	0.68
LED RGB WS2812b	2	0.51
IMU MPU6050	1	0.85
LI-PO Battery 3.7 V 1800 mAh	1	5.85
PCB board and Components	1	4.30
3D Printer Parts (PLA) and Fastening	1	1.50
Total		**18.72 USD**

can also assemble it in several specific PCB manufacturers, which nowadays attend at a highly affordable cost. Fig. 5 shows the proposed front and back views PCB board's design. A complete tutorial for the robot assembly can be found on the project's website[4].

3.3 Part Costs

After defining the mechanical and electrical components of the robot and its assembly process, we can estimate its cost. Table 4 gives a summary of the cost of the components used in HeRo. All part prices assume retail buying from standard part distributors on the Internet. We expect this cost would be greatly reduced if parts are acquired in bulk directly from manufacturers.

Finally, after carrying out the robot assemblies, we arrived at the result shown in the Fig. 6.

4 Software and Communication Architecture

In addition to the physical robot, we also describe a computational framework to facilitate its use in swarm applications. In this section, we present a software and communication architecture that enables the programming of multiple robots. Moreover, we also present a simulated environment, useful in the early stages of application development.

4.1 Software Architecture

A typical robot swarm dilemma is how to program multiple robots quickly, easily, and efficiently. A practice that has become very common in the experimentation stage is to use a master-slave architecture in which robots (slaves) remotely communicate with a computer (master) running the user application. This strategy is highly efficient as it does not require the user to burn the firmware every time he needs to change his application. Despite being convenient, remotely executing the application on a computer does not always capture effects that impact the application at the deployment level, e.g., local communication issues, low processing capacity, and other errors. Thus, strategies that use FOTA (Firmware Over-The-Air) technology enable users to remotely load their applications on the robot and run it directly on it.

In this work, we propose a flexible architecture to use any of the programming practices mentioned above. Our architecture is composed of a firmware compatible with ROS (Robot Operating System) and FOTA. In the following, we detail the software architecture.

4.1.1 Firmware

The firmware is one of the fundamental parts of the robot since it computes the control of the motors and access to the sensor's data. For HeRo, we chose to implement the firmware using the Arduino IDE. Such platform is easy to use and widespread in makers' communities, making the firmware easier to follow and modify. It also provides many libraries enabling us to control the microcontroller ports and handle actuators, sensors, and TCP sockets.

The firmware is built on top of the rosserial framework allowing the robots to be compatible with the ROS middleware. Rosserial comprises different tools, including a protocol for wrapping standard ROS serialized messages and multiplexing multiple topics and services over network sockets. Such framework abstracts several communication concepts, allowing a compact and efficient implementation. In practice, the user only needs to configure a few communication parameters so that the robots can connect using TCP/IP networking with a remote computer running ROS. To facilitate this configuration process avoiding keep reprogramming the firmware), we implemented a remote configuration mode for the robot using a web interface (Fig. 7a). To open this interface, the users must first turn on the robot in configuration mode. This mode creates an access point where the user connects using a computer or a smartphone. By using a browser, the user can access the robot webpage and set up its name, access point credentials where the ROS server is running, ROSMaster IP address, and port. Once the robot is properly configured, it automatically connects to the ROS server, and then the user access the robot's features through topics and services (Fig. 7b). This entire process can be done in less than a minute, and the configuration remains saved even if the robot is turned off.

In addition to providing a master-slave communication architecture, the firmware is also composed of some basic modules that compute the robot's kinematic control, odometry, and sensor data. Next, we describe all these modules present in the firmware.

[4] Tutorial: `https://verlab.github.io/hero_common`

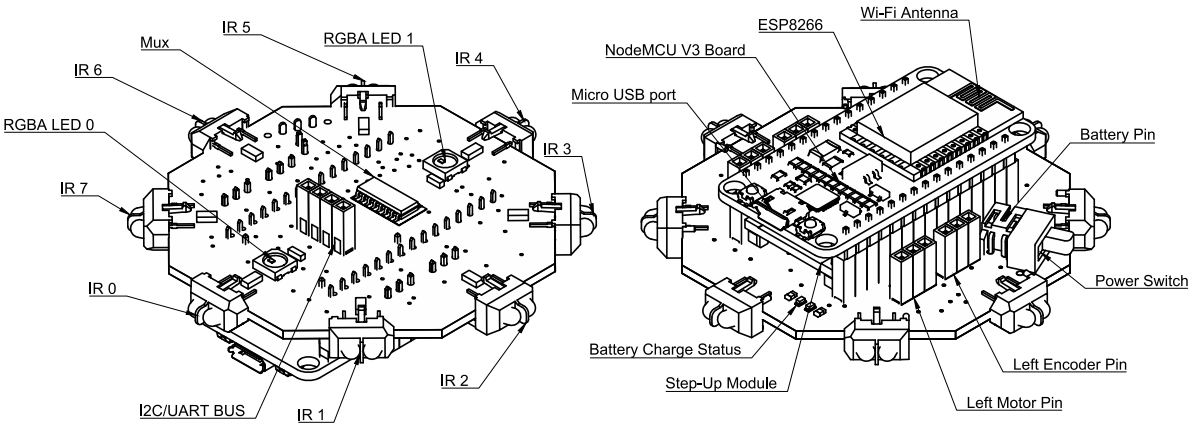


Fig. 5: Overview of HeRo's PCB board.

Fig. 6: Top, bottom, front, and left views of the HeRo swarm platform.

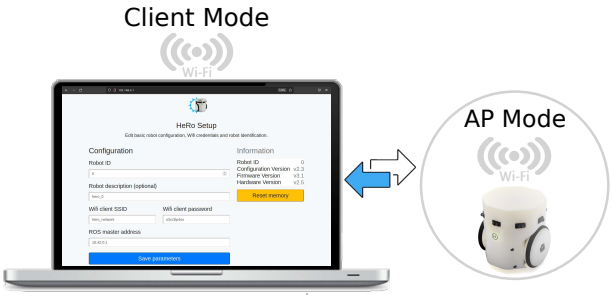
Sensors In its basic form (without the e-Hat), the robot has eight infrared transceivers and two quadrature mechanical rotary encoders as sensors. The eight infrared sensors are mounted around the robot to provide a complete field of view of the environment. These are connected to a 10-bits ADC port on the microcontroller through an 8-channel analog multiplexer allowing the estimation of the distance to an obstacle as well as the ambient light once we can control the infrared emitter. Obstacle detection and distance estimation use fundamental principles of electromagnetic radiation and its reflection. Mathematically, the reflected signal intensity measured with a sensor is modeled by [6]:

$$s(d, \gamma) = \frac{\alpha}{d^2} \cos(\gamma) + \beta, \tag{1}$$

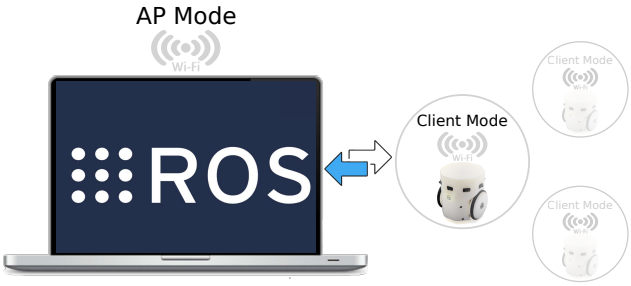
where $s(d, \gamma)$ is the output value of the sensor, d is the distance to the object, and γ is the angle of incidence with the surface; the model variable α includes several parameters such as the reflectivity coefficient, output power of the emitted IR light and the sensitivity of the sensor and it is estimated empirically; β is the offset value of the amplifier and ambient light effect and it is measured regularly after performing the Equation 1.

As mentioned, HeRo has two quadrature encoders attached to each wheel. A quadrature encoder, also known as an incremental rotary encoder, is commonly used to measure the speed and direction of a rotating shaft. The encoders' channels are connected to the microcontroller's interrupt pins. Each pulse calls an interrupt routine in the microcontroller, increasing an independent counter variable to estimate how far each wheel has turned. To estimate the velocity of the wheel, we measure the frequency of the pulses. The output of the encoders is used as an input to a controller for closed-loop motion control and for localization.

Motion Control We previously defined the robot as a two-wheel differential-drive mobile robot composed of two servo motors, each with a quadrature encoder. Due to non-holonomic constraints, the robot cannot move along its wheel axis concerning its body reference, but it can change its direction by varying the relative instantaneous speed of its wheels and hence does not require an additional steering motor. One of the most suitable ways of controlling this robot's motion in 2D space consists of controlling its linear and angular speed. By assuming classical kinematics modeling for a differential-drive mobile robot [32], one can determine the velocity of the robot in its own reference frame or in the inertial frame, as shown in Fig. 8. Formally, the instantaneous



(a) Remote configuration mode.



(b) ROS communication mode.

Fig. 7: Robot firmware modes: (a) remote configuration and (b) ROS communication mode. The first one helps the user configure the robots to connect to a server running ROS without requiring reprogramming the robot. After properly setting up the robot, it connects automatically with the ROS server allowing the user to send and receive commands through ROS topics and services.

velocity, expressed with respect to the robot body frame and the inertial frame, is given by:

$$\mathbf{v}^R(t) = \begin{bmatrix} v_x(t) \\ 0 \\ \omega(t) \end{bmatrix}, \tag{2}$$

$$\mathbf{v}^I(t) = \begin{bmatrix} v_x(t)\cos(\omega(t)) \\ v_x(t)\sin(\omega(t)) \\ \omega(t) \end{bmatrix}. \tag{3}$$

Despite one may control the robot's velocity in any reference frame, in this case, we find controlling it regarding the robot frame more convenient as it can simplify the control problem and make it easier to achieve the desired motion. From now on, we will describe how we control the instantaneous velocity of the robot in its own frame. That is, we want to control the linear speed $v_x(t)$ along the X_R-axis and its angular speed $\omega(t)$ around the Z_R-axis for $\mathbf{v}^R(t)$.

By assuming such velocities as inputs, we must map them into wheel velocities so that we can control the

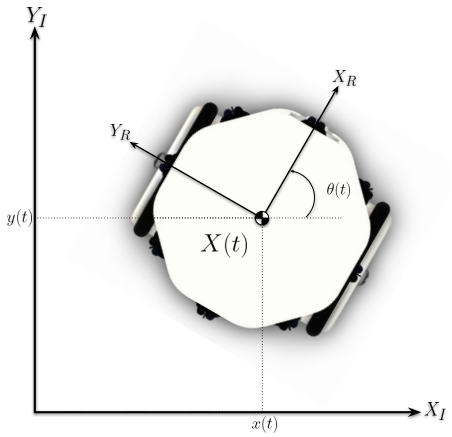


Fig. 8: Robot represented in an inertial reference frame.

robot. The problem of mapping the relationship between robot velocity and wheel speed is called inverse kinematics. That is, given the linear speed $v_x(t)$ and angular speed $\omega(t)$, we can compute the desired left speed $v_l(t)$ and right speed $v_r(t)$, to produce the specific motion of the robot (see Fig. 9). The equation below describes the inverse kinematic model concerning the robot reference frame:

$$\begin{bmatrix} v_l(t) \\ v_r(t) \end{bmatrix} = \begin{bmatrix} \frac{2v_x(t) - l\omega(t)}{2} \\ \frac{2v_x(t) + l\omega(t)}{2} \end{bmatrix}, \tag{4}$$

where $v_l(t)$ and $v_r(t)$ are the tangential speeds of the left and right wheels; $v_x(t)$ and $\omega(t)$ are the linear and angular speeds of the robot in its own reference frame; and l is the distance between the left and right wheels.

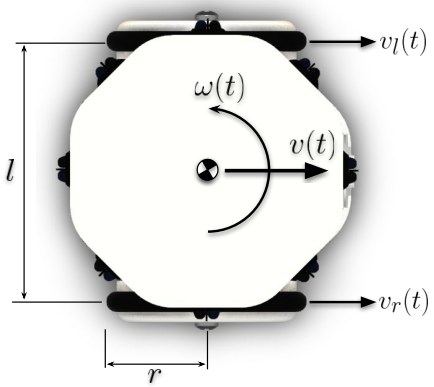


Fig. 9: Robot local reference frame.

After computing the desired tangential speed on each wheel, we need to control the motors so that they maintain these speeds. The proportional integral derivative (PID) controller is the most common control al-

gorithm used for this application. It can correct the present error through proportional action, eliminate steady state offsets through integral action, and better estimate future trends through a derivative action. The mathematical model of a PID is defined by

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}, \tag{5}$$

where $u(t)$ is the control signal to each motor, that is, PWM signals; dt is the control loop interval time; $e(t)$ is the error regarding the desired and current tangential speeds of each wheel; and K_p, K_i and K_d, all non-negative, denote the coefficients for the proportional, integral, and derivative terms, respectively.

To estimate the error $e(t)$, we subtract the desired tangential speed from the current tangential speed estimated by the encoders. That is, we count how far each wheel has turned and compute the rate for a loop interval. Formally, the current tangential speed for both wheels is computed by

$$\bar{v}_l(t) = \frac{\Delta s_l}{dt}, \tag{6}$$

$$\bar{v}_r(t) = \frac{\Delta s_r}{dt}, \tag{7}$$

where Δs_l and Δs_r are the distance each wheel has traveled for a time interval dt, respectively.

Moreover, we use a simple Kalman filter to reduce the noise of the reading and improve the quality of both estimated speeds. Formally, we compute the following process for each wheel measurement:

$$K = \frac{\sigma_e^-}{\sigma_e^- + \sigma_m}, \tag{8}$$

$$\hat{v} = \hat{v}^- + K(\bar{v} - \hat{v}^-), \tag{9}$$

$$\sigma_e = (1 - K)\sigma_e^- + |\hat{v}^- - \hat{v}|q, \tag{10}$$

where σ_e is the estimation uncertainty adjusted by the filter; σ_m is the measurement uncertainty, that is, how much we expect the estimated speed can vary; K is called Kalman gain; $\bar{v}$ is the current measured speeds, *i.e.*, $\bar{v}_l(t)$ and $\bar{v}_r(t)$; $\hat{v}$ is the filtered speed; q is the process variance, that is, how fast the measurement moves; and finally, the superscript $(^-)$ indicates previous values of a variable.

Finally, we summarize the robot velocity control as a block diagram, depicted in Fig. 10.

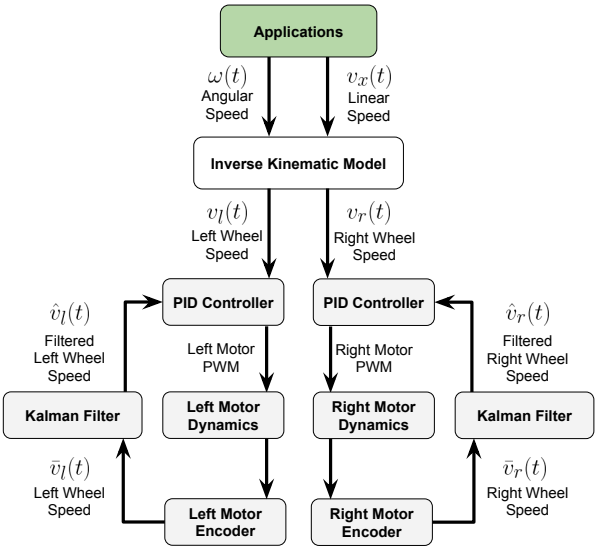


Fig. 10: Diagram illustrating the velocity control of a differential robot.

Localization Odometry is the most used method for determining the position of a mobile robot concerning an inertial reference frame (see Fig. 8). In most practical applications, odometry provides easily accessible real-time positioning information in-between periodic absolute position measurements. Several different types of sensors are commonly used for odometry. This work addresses odometry by placing encoders on each wheel and counting how far each wheel has turned. Using these two measurements, we can estimate how far the robot has moved forward and its heading. The distance traveled by the robot is the average of how much each wheel has turned and is presented in Equation 11. On the other hand, the heading of the robot is estimated (assuming insignificant wheel slip) from the difference of these displacements over the distance between the wheels and is presented in Equation 12.

$$\Delta L = \frac{r(\Delta s_r + \Delta s_l)}{2}, \tag{11}$$

$$\Delta\theta = \frac{r(\Delta s_r - \Delta s_l)}{l}, \tag{12}$$

where Δs_r and Δs_l represent how much each encoder has turned in the loop time interval; r is the wheel radius; and l represents the distance between the wheels of the robot, as shown in Fig. 9.

Once we have computed how far the robot has traveled and turned, we can integrate this information to estimate its current pose regarding the inertial reference frame. Considering the pose of the robot at time t in a plane is given by the state vector

$$\mathbf{X}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix}, \tag{13}$$

the pose of the robot after a loop time interval dt is given by

$$\mathbf{X}(t+dt) = \mathbf{X}(t) + \begin{bmatrix} \frac{\Delta L}{\Delta\theta}(\sin(\theta(t)+\Delta\theta) - \sin(\Delta\theta)) \\ \frac{\Delta L}{\Delta\theta}(\cos(\theta(t)+\Delta\theta) - \cos(\Delta\theta)) \\ \Delta\theta \end{bmatrix}, \tag{14}$$

where $\Delta\theta$ is the variation of the robot orientation in Z_I-axis in a time interval, that is, $\Delta\theta = \theta(t+dt) - \theta(t)$.

Note that if the robot moves in a straight line, the change in angle $\Delta\theta$ is zero, and the odometry model (14) becomes undefined since $\frac{\Delta L}{\Delta\theta}$ is undefined. In this case, a different model should be used to compute the odometry, such as using the change in distance ΔL. Thus, in order to approach this special case, we test this condition, and if it occurs, the following model is computed for the odometry,

$$\mathbf{X}(t+dt) = \mathbf{X}(t) + \begin{bmatrix} \Delta L \cos(\theta(t)) \\ \Delta L \sin(\theta(t)) \\ \theta(t) \end{bmatrix}. \tag{15}$$

4.2 Communication Architecture

In order to provide communication between a workstation and the robots, we implement HeRo as a ROS-compatible robot by connecting them using TCP/IP.

The Robot Operating System [27] is an open-source, meta-operating system for robotic applications. It provides similar services expected from a typical operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple platforms.

The communication is conducted by a publish/subscribe model, where topics made up of predefined message structures can be communicated between multiple nodes (processes) in the network. These topics, for example, odometry, can be accessed by any node in the network, allowing for easy scalability of publishers and subscribers. In this way, robots can readily communicate with other robots in the network in a well-defined way.

However, most swarm robots, including HeRo, are unable to process a full-fledged native ROS instance given the restricted CPU resources. To integrate these functionalities to less powerful microcontrollers without a complete instance of ROS, we implemented the communication module over the rosserial protocol, which has been proved as a reliable and scalable communication method for swarm systems [36]. Rosserial[5] is a protocol for wrapping standard ROS serialized messages and multiplexing multiple topics and services over a network socket. In short, the rosserial nodes convert data from normal structured XMLRPC protocol handled by TCP natively in ROS to serialize data out to the microcontroller. This node also deserializes data from the microcontroller back into the correct message structures to be sent around the conventional ROS network.

While the robot is compatible with ROS 1 using the rosserial framework, it has not yet been possible to make it fully compatible with ROS 2. One challenge is that the rosserial framework has not been ported to ROS 2, and microROS[6] (rosserial alternative in ROS 2) does not have support for the microcontroller used by the robot (ESP8266). As an alternative for ROS 2 users to be able to interface with the robot, we provide a containerized environment using the Docker platform[7]. This way, ROS 2 users can use packages such as ROS-Bridge that allow ROS 2 to interface with the ROS 1 package. Figure 11 shows an overview of the communication architecture.

In theory, the network's bandwidth limits the number of connected robots: as more robots are added, more connections are made, taking up capacity. However, we did not observe any overhead communicating with multiple robots, even using a consumer-grade wireless network router. A typical network addresses 254 devices, but network techniques (e.g., subnets) allow increasing this limit as much as we need. A complete study showing the reliability and scalability of using this protocol for swarm robots is present in [36].

4.3 Simulation and Visualization

The execution of simulations plays an essential role in robotics research as a tool for quick and efficient testing of new concepts, strategies, and algorithms. Moreover, good visualization tools are very important during the experiments to better track and observe the robot execution. In this sense, we also developed a simulation model of HeRo that can be used together with Gazebo and RViz.

[5] Rosserial: `http://wiki.ros.org/rosserial`
[6] microROS: `https://micro.ros.org`
[7] Docker: `https://www.docker.com`

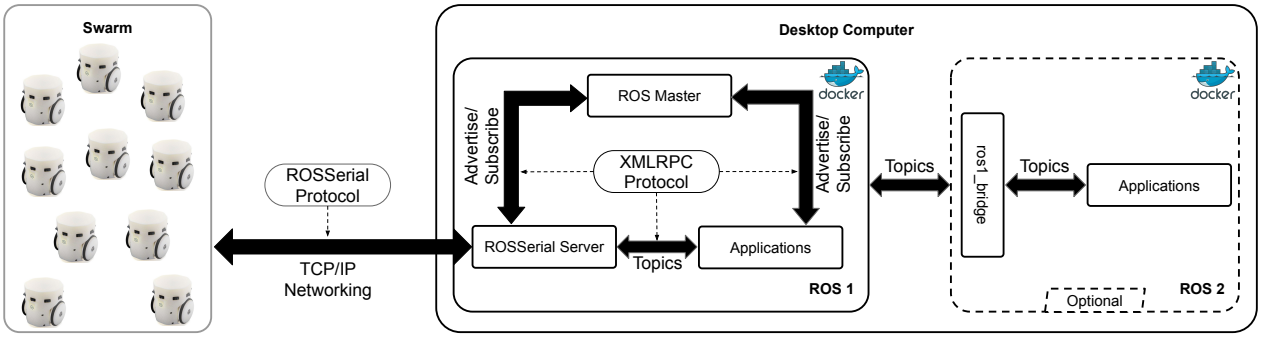


Fig. 11: An overview of the communication process. The robot's microcontroller acts as a bridge to the sensors and actuators and then rosserial acts as another bridge from the microcontroller to ROS. ROS 2 users can optionally instantiate a ROS 1 bridge interface and interact with the robots. The system infrastructure is organized in Docker containers, which promotes its installation and use.

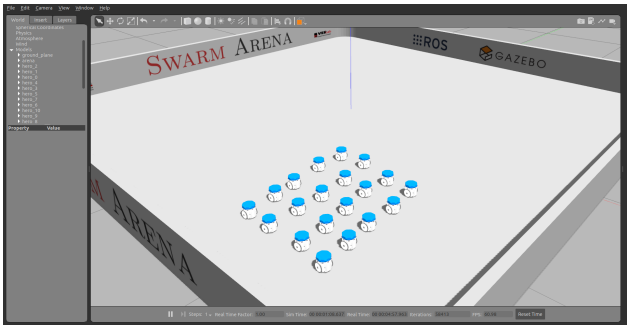


Fig. 12: Multiple instances of HeRo being simulated in the Gazebo simulator.

4.3.1 Gazebo Simulator

In order to make simulations with our robots in ROS, we decide to use Gazebo since it is fully integrated with ROS. Gazebo [15] is a multi-robot simulator for complex indoor and outdoor environments. It is suitable for simulating a population of robots, sensors, and objects in a three-dimensional world. ROS and Gazebo use the 3D model of a robot or its parts, whether to simulate or visualize them, through the XML files, called Unified Robot Description Format (URDF). This file describes all the structures of the robot, such as its parts, joints, dimensions, and texture, among others.

After describing the robot using the URDF file, creating a simulated model using the built-in plugins provided by the Gazebo is straightforward. However, this approach is inefficient when simulating multiple robots and requires a high computational cost. To reduce consumption and increase the number of simulated robots, we designed a compact plugin that implements all the robot's functionalities. In this way, we optimized the maximum processing performed by each simulated robot without overloading Gazebo's physics engine. Fig. 12 shows multiples instances of HeRo being simulated in the Gazebo simulator.

4.3.2 Robot Visualization Tool

In addition to the simulation, it is also essential to have a robot visualization tool that shows the state of sensors and actuators during the experiment. In ROS, we can visualize the robot's state using the RViz visualization tool. This tool provides 3D visualization of the robot by loading the URDF file and can project sensors data obtained by the ROS topics such as odometry, laser, and IMU using plugins. Note that RViz is not a simulator but only a visualization tool. In this way, the robot visualized in this tool can be real or simulated depending only on who publishes the information. Fig. 13 shows an example of viewing a real robot in Rviz. In the image, we can see the 3D model of the robot overlaying a colored axis that indicates the robot's pose relative to an initial frame (colored axis in the background of the scene). The sequence of small axes indicated the temporal pose of the robot computed by the odometry. Colored spheres around the robot can move closer or further away from the robot and indicate the readings of the distance sensors. On the right, we can follow the linear velocity of each robot's wheel.

4.4 Programming

The communication architecture defined for our robot allows it to be programmed in two different ways: using the ROS framework or the OTA firmware technology.

In the first mode, we can program and run applications on a server, which communicates and controls each robot in a decentralized way. In other words, each algorithm is executed in a process on the server, and this process has access via Wi-Fi with its respective robot. This mode is very convenient and scalable in the early stages of testing with multiple robots. Furthermore, using the ROS framework for implementation, we have a series of tools and may use different programming languages.

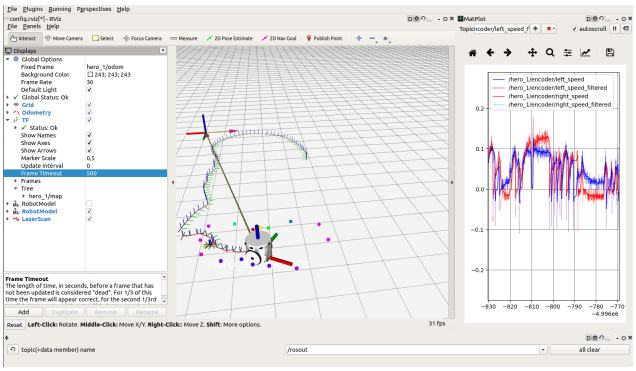


Fig. 13: RViz showing a single HeRo robot. RViz is a 3D visualization tool for ROS allowing control and observe the current state of the robot.

On the other hand, processing algorithms remotely is not always suitable for robot swarm applications. In this case, the algorithm must run directly on the robot, maintaining the convenience of programming the robots simultaneously. In this programming mode, we use the OTA technology to burn the firmware in several robots using Wi-Fi. This process uses the Arduino IDE to implement and compile the application and then uses the command line to transmit the binary code for robots. Despite being convenient, this mode is limited in terms of the availability of high-level tools. In addition, it requires using a programming language compatible with the microcontroller, in this case, C/C++.

5 Performance Evaluation

This section presents a series of experiments that evaluate our robot's performance as a capable swarm robot. Initially, we analyzed motion control and evaluate the robot's odometry in comparison to the *E-puck*, a popular commercial swarm robot. We also evaluate and discuss the performance and scalability of communication when using ROS and make an analysis of the robot's energy consumption when demanding different types of applications.

5.1 Motion Control Analysis

In this experiment, we evaluated the robot velocity control, which consists of ensuring that the robot reaches a desired velocity concerning its own reference frame by controlling the wheel speeds. As previously detailed, the wheel speed control uses a PID controller. The feedback information consists of the current wheel speed, estimated by the encoder's readings and filtered by the Kalman filter. The parameters used by both methods were all obtained empirically and defined as: $K_p = 1200$, $K_i = 2300$, and $K_d = 0.1$ for the PID parameters, and 0.02 m/s of sensor noise (measurement uncertainty) and 0.2 as process variance for the Kalman filter.

One way to evaluate the controller's performance is to check its response time and residual ripple. Thus, we observe the controller's behavior when starting with the robot halted, and then we set a desired linear and angular speeds to $v_x = 0.0$ m/s and $\omega = 3.17$ rad/s so that the robot turn in place for 4.5 seconds and then stop. Fig. 14 shows the performance of the robot.

As expected, we observe a similar response time for the left and right wheels, reaching the desired tangential speed ($v_l = -0.1$ m/s and $v_r = 0.1$ m/s) after approximately 1.5 s. Although we can make the system more responsive, we opted for a more conservative controller with no overshoot to avoid sudden movements make it difficult to control the robot. After both wheels reach the desired speeds, we measure a mean absolute error of 0.00033 ± 0.0023 m/s for the left wheel and 0.00010 ± 0.0014 m/s, which is remarkable considering the low-cost components used within the robot. Moreover, by controlling the speed for each wheel, the robot reached the desired linear and angular speeds with a mean absolute of 0.00022 ± 0.0010 m/s for the linear speed and 0.00358 ± 0.0530 rad/s for angular speed.

5.2 Localization

In this experiment, we evaluate the odometry of our robot and compare the results with the one obtained by the E-puck [20]. To better analyze the capabilities of these robots, we implement the same odometry model and use the same experimental setup.

This comparison is interesting because E-puck uses relatively expensive stepper motors against the inexpensive servo motor used in our robot. The E-puck computed its odometry by counting steps commanded to each motor, reaching a maximum resolution of 1024 steps per wheel revolution, which is more than the provided by our encoders (288 steps per revolution). However, there is no feedback when the wheel steps, so it probably produces more false-positives counts.

To measure the pose estimation accuracy for both robots, we use the OptiTrack tracking system[8] as a ground truth reference. The trajectory performed by both robots is a rectangular shape (1.3x1.1 m), delimited by four points. We control each robot to move to the four points consecutively until it completes three loops. Both robots traveled equal distances while maintaining the same velocity to keep the comparison as reliable as possible.

Besides comparing the odometry produced by both robots, we extended another HeRo with an IMU e-Hat. Combining these two sensors improved our robot's

[8] OptiTrack: `http://optitrack.com/`

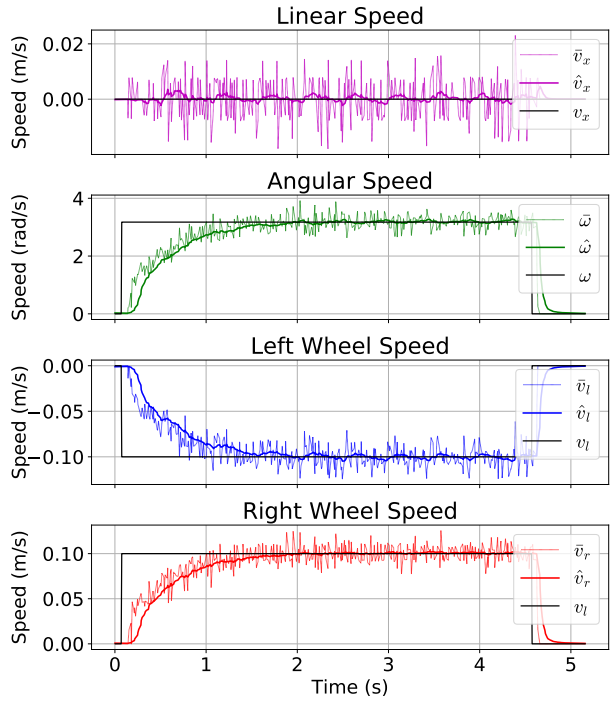


Fig. 14: Analysis of the robot speed control showing the accuracy in reaching the desired linear v_x and angular ω speeds concerning the robot reference frame. To reach such motion, the robot computes the desired speeds on each wheel, v_l, and v_r, and then uses a PID controller to control the motors. The current wheel speeds are computed from the encoder's readings, $\bar{v}_l$ and $\bar{v}_r$, and filtered, $\hat{v}_l$ and $\hat{v}_r$, to reduce noise.

orientation estimate and, consequently, improved the robot's odometry. This IMU is composed of a gyroscope and an accelerometer, and it has a built-in MPU (motion processing unit) that combines both sensors, generating an orientation estimation. In this case, the orientation estimation provided by the IMU and the odometry drift (no zero mean noise), but the IMU orientation estimations drifts are smaller. In this way, we replaced the orientation of the odometry with the one provided by the IMU. Fig. 15 shows the trajectories performed by (a) an E-puck, (b) a HeRo without e-hat, and (c) a HeRo using e-hat with a gyroscope and accelerometer. A video of this experiment is available on Youtube[9].

As observed, the HeRo's odometry is comparable to the E-puck's. Given that E-puck is one of the most robust and well-used robots for swarm experimentation, we believe that our robot also proves to be an attractive solution. Moreover, the components used by HeRo are highly affordable when compared to E-puck. Furthermore, using the module with inertial sensors improved the robot's orientation, making the localization more robust, allowing its use in other applications.

[9] Odometry Comparison: `https://youtu.be/9s6Fg20u0pc`

5.3 Distance Sensor

This experiment assesses the performance of the IR sensor concerning the distance estimation to a white obstacle. Before evaluating the performance of the distance sensor, we need to characterize the sensor (convert the analog signal to distance).

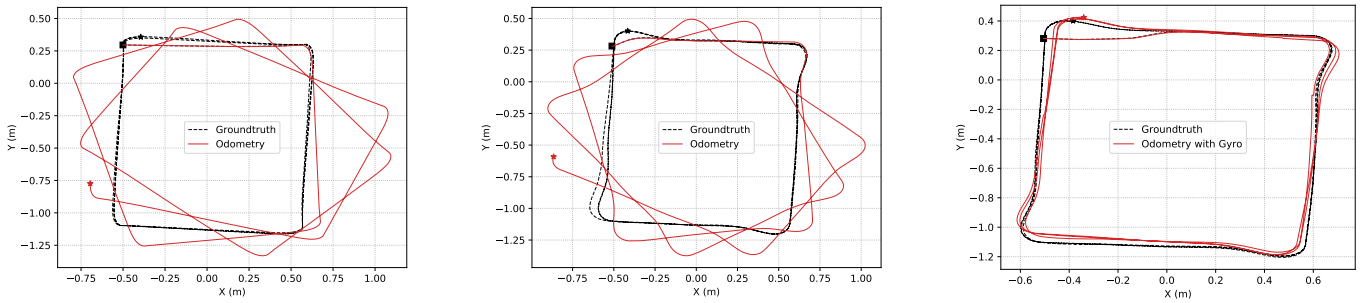
To convert the infrared sensor readings to distance, we first take the sensor readings using a 10-bits ADC input for various object distances, ranging from 0 to 40 cm, in one-centimeter intervals. To remove light interferences, we first read the IR sensor without activating the IR emitter and then turn the emitter on and take another reading. The difference between these two readings returns a more robust measurement of the effective light intensity reflected by the obstacle.

Fig. 16a shows these measurements assuming log-scale for y-axis. As observed, it seems possible to detect objects within the range of 30 cm, but to better estimate the distance, we decided to limit such a range to 20 cm. After collecting these measurements, we perform the distance sensor calibration solving the Equation 1. Fig. 16b shows the distance estimates for the object after the calibration process.

5.4 Communication

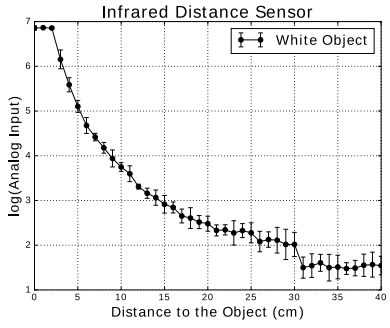
Communication mechanisms is swarm robots must be scalable to accommodate a large number of robots. This experiment evaluates the communication scalability regarding the bandwidth (the maximum amount of data that can travel through a channel) when using ROS to program the robots. We estimate the number of robots supported in the network by computing the total bandwidth used by one robot and the maximum bandwidth supported in the network.

Table 5 shows the measured bandwidth of each topic (communication channel) between the robot and a server executing ROS. We assume that these topics publish or subscribe to messages at specific frequencies, set as the default rate of HeRo processing. We captured these measurements using the *rostopic tool* which provides the packet size of a single message considering: an overhead of 20 Bytes for the TCP packet (for the Wi-Fi data connection) and 8 Bytes for *rosserial* serialization; and the message data size, depending on each type of topic. In addition to packet size, the *rostopic tool* also provides the actual bandwidth for each topic, allowing us to compute the total bandwidth used by one robot (44 KBps).

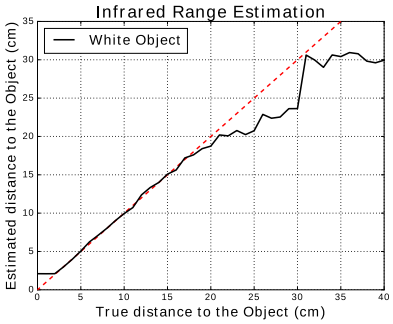


(a) E-puck odometry. (b) HeRo odometry. (c) HeRo + IMU odometry.

Fig. 15: Trajectory performed by: (a) E-puck, (b) HeRo and (c) HeRo using e-Hat with inertial sensors.



(a)



(b)

Fig. 16: Analysis of the infrared distance sensor. (a) Shows the readings obtained from a single IR sensor as a function of the distance to the white target; and (b) shows the estimated distance after calibrating the sensor for a maximum range of 20 cm.

Assuming the Wi-Fi module used in our robot can handle at least 1 MBps[10], we are only using 4.2% of the maximal capacity. Moreover, the robot connects to a consumer-grade wireless network route in infrastructure mode that provides a maximum bandwidth of 150 Mbps (or 18 MBps). Considering that one HeRo uses only 44 KBps to communicate, theoretically, we estimate that almost 420 robots are supported in this network. Despite typical Dynamic Host Configuration Protocol (DHCP) can not address all these IPs, one may use other ways to avoid this limitation, such as subnetwork or using multiple routers.

[10] Datasheet: `www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf`

Table 5: Maximum amount of data that can travel through a ROS topic. These topics are operating at different frequencies, set as the default rate of HeRo processing.

ROS Topics	Frequency (Hz)	Packets Size (KB)	Bandwidth (KBps)
/imu	30	0.320	8.60
/laser	20	0.130	3.20
/odom	30	0.730	18.55
/encoder	30	0.100	3.15
/led	2	0.016	0.321
/cmd_vel	20	0.048	0.967
/tf	30	0.068	8.542
Total			**43.33**

5.5 Power Consumption

Another critical concern is the robot's power autonomy, which defines its operating time. This experiment analyzes the power consumption of the components, establishing the power autonomy of our robot. To better understand the consumption of the robot, we measure the current (mA) used by the robot in three typical situations: (i) when sensors and communication are active, (ii) with the indicator LEDs turned on, and (iii) with the motors active. Table 6 shows the consumption (in mA) of the robot for these combinations.

For the first case, we observe the effect of the frequency of publication in the robot's power consumption. Thus, we measured the current for three different frequency rates (5, 20, and 40 Hz) and noticed that these rates have a minimal impact on consumption. For the second case, we kept the communication frequency at 20 Hz, and turned on the two indicator LEDs, and changed its white light intensity from half to full. We noticed that the light intensity used by the LEDs significantly impacts the consumption (almost 50 mA). In the last case, we kept the communication frequency rate and turned on the two motors. To observe the impact

of the robot's velocity on the power consumption, we measured the current for two different velocities. As expected, the motors are the most power-consumer components in the robot, and their velocity proportionally affects the power consumption.

In order to analyze the power consumption in a general way, we assumed a typical use for these three combinations, in which we retained the sensors and communication at 20 Hz; indicator LEDs with half intensity; and motors reaching a speed of 10 cm/s. Thus, we observed the consumption reaching 550 mA with a slight deviation of 25 mA. Suppling the robot with a 3.7 V 1800 mAh Li-Po battery, we estimate the minimum and maximum autonomy of 3 h and 9 h, respectively.

Table 6: Power consumption of HeRo considering a voltage of 3.7 V. Typical power consumption is the average of 30 samples.

Mode	Min	Typical (mA)	Max
Sensing & Communication at 5 Hz	152	161 ±9	180
Sensing & Communication at 20 Hz[1]	153	175 ±16	205
Sensing & Communication at 40 Hz	170	183 ±9	205
Sensing & Communication[1] & LEDs (50%)[2]	187	205 ±14	245
Sensing & Communication[1] & LEDs (100%)	225	247 ±2	292
Sensing & Communication[1] & Motors (10 cm/s)[3]	455	512 ±30	584
Sensing & Communication[1] & Motors (25 cm/s)	613	660 ±25	717
Typical Use[123]	396	550 ±47	628

[1] Typical frequency rate used for sensing and communication.
[2] Common brightness used in the LEDs indicators (White color).
[3] Common velocity performed by the robot during the experiments.

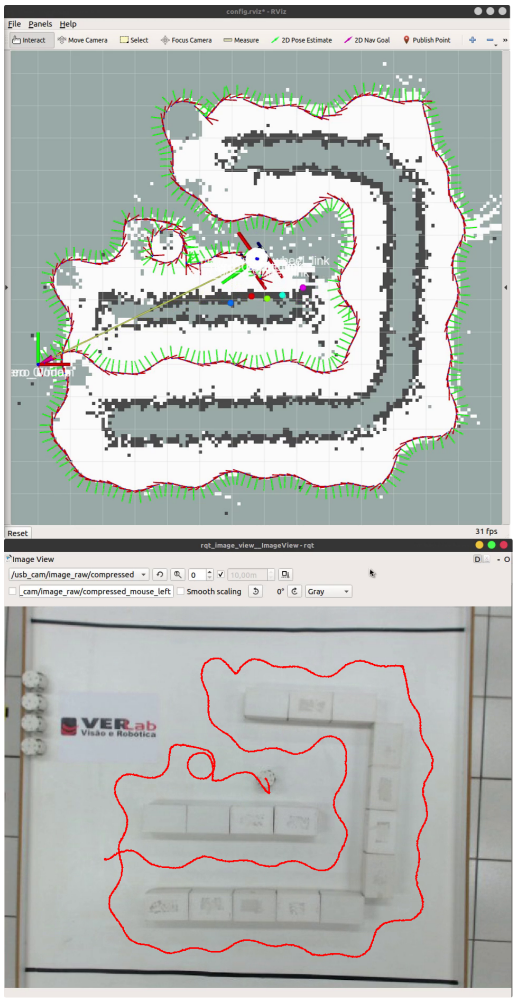


Fig. 17: An occupancy map produced by the robot using only eight IR sensors and odometry. At the top, it is the RViz visualization tool showing the map computed by the robot. The sequence of axes (x-red, y-green) shows the trajectory computed by the odometry. At the bottom is a top-view image showing the environment and the ground truth trajectory performed by the robot.

6 Applications

In addition to evaluating the robot's performance, we also demonstrate its capability executing a set of different applications.

6.1 Mapping

This experiment shows the capacity of a single real robot to perform a mapping task. We configured the robot to communicate with a remote computer running ROS. Then we used the Gmapping[11] package, a laser-based SLAM (Simultaneous Localization and Mapping) algorithm provided as a standard ROS package for mapping an environment. We set up an environment composed of white cardboards forming a scene similar to a hall contained in an area of 1.20×1.20 m. Fig. 17 shows the occupancy map produced by the robot. The sequence of axes (x-red, y-green, z-blue coming out of the figure) projected onto the map represents the temporal pose of the robot computed from the robot's odometry. At the bottom it is a top-view image showing the environment setup and the ground truth trajectory performed by the robot. A video showing the execution of this experiment is available on Youtube[12].

Overall, the mapping task is challenging and requires sophisticated sensors such as LiDAR and more accurate localization methods. However, we obtain interesting results using only a small robot with eight IR sensors and wheel odometry. Even though the excellence of the mapping is in the SLAM algorithm, we corroborate the robot's capability to perform this task satisfactorily, which makes it interesting for cooperative mapping tasks.

[11] GMapping: `http://wiki.ros.org/gmapping`

[12] Mapping Performance: `https://youtu.be/_RWCCI8BI1s`

6.2 Decentralized coverage

This experiment shows five robots performing a covering task in a small bounded environment (0.8×1.20 m). Unlike the previous experiment, which requires more computational processing, this one implemented the coverage method directly in the robot's firmware without requiring it to communicate with a server running ROS. The coverage method consists of randomly navigating the robots through the environment and avoiding obstacles and collisions with other robots using only local sensing. Fig. 18 shows a sequence of images captured by an overhead camera. Each of the five robots is programmed to emit a different color, making them easier to see and identify. The lines in the images represent the path performed by each robot. A video of this experiment is available on Youtube[13].

In this experiment, the coverage area is relatively small, given the number of robots. We set a small area to verify that the robots could perform maneuverability and deal with possible interference between the IR sensors. As a result, we verified that the robots were navigating through the environment, avoiding collisions for long periods, which shows that the operation of multiple robots in a small area is feasible.

6.3 Flocking Behavior

Another experiment demonstrates the robot's capacity to perform a flocking behavior. We used five robots initially randomly distributed in an environment in this experiment. The flocking algorithm implemented in this experiment is decentralized and only requires the relative position and velocity among the neighbors' robots (see [29] for more details). Since we do not yet have a set of sensors onboard the robot that estimates such information, we set a remote server executing ROS to emulate such a sensor. In short, we use an overhead camera and the Apriltag tracker algorithm [35,19] to locate the robots in the scene and then compute their relative positions and velocity to provide to the algorithm. Fig. 19 shows a sequence of images captured by the overhead camera. The figure shows the initial configuration of the robots. After a runtime, the robots manage themself to aggregate and navigate the environment as a group. A video of this experiment is available on Youtube[14].

In this experiment, we verified the robot's motion control response when reproduced with other robots. Indeed different robot requires specific calibration parameters for PID. Concerning the flocking task, it is required cohesive and aligned navigation between agents. If the robot's motion control is not properly adjusted, this may result in incorrect group navigation. As an outcome, we verified that the robots are well calibrated and capable of performing the flocking task, resulting in aligned and cohesive navigation between the robots.

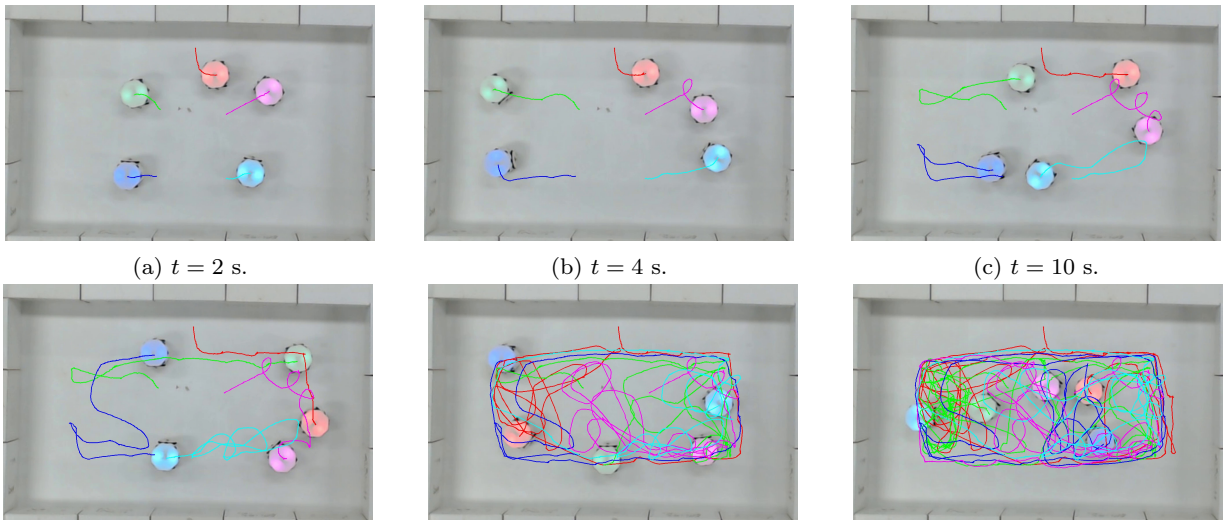
6.4 Cooperative Transportation

Finally, we conducted experiments evaluating the robot's performance in a cooperative transportation task. In this task, the swarm must coordinate to push an object toward its goal location, taking advantage of multiple robots' forces applied to the object. The strategy implemented in these experiments is described in [28] and does not require prior knowledge of the shape and location of the object, only its target. So, the robots can navigate through the environment, form groups, and when they detect the object, they can move around it looking for contact positions that allow the object to be pushed towards its objective. Despite being a decentralized strategy and not requiring global information on the swarm or the object, robots must estimate their neighbors' relative position and velocity and distinguish between object and obstacles detection. As in the previous task, we used the overhead camera location system to emulate such sensors. Fig. 20 shows a sequence of snapshots taken by the overhead camera. In the sequence, the robots group and coordinate to transport the object to its goal. A video of this experiment is available on Youtube [15]. As expected, we verified that the robot could properly control its motion while interacting with objects. Moreover, it also has enough traction to start pushing it, which allows the robots to perform cooperative transport tasks.

7 Conclusion

This work presented HeRo, a novel open-source swarm robotic platform that is cost-effective, capable, and scalable, developed using off-the-shelf components and additive manufacturing. This robot was specially designed for swarm applications, given its small size, sensing, and networking capabilities. The proposed robot has WiFi communication, over-the-air firmware upgrades and is fully compatible with ROS, facilitating the development of newer functionalities. We also presented a simulation environment for the HeRo platform using Gazebo, a popular simulation engine for continuous testing and quick prototyping. Experiments evaluating the sensor accuracy, odometry, autonomy, swarm communication, and control show a performance in par or superior to other commercial or more expensive platforms. Mapping, decentralized coverage, flocking behavior and transportation tasks performed with a group of HeRo robots validate the robot's capacities for real-world swarm applications and educational use.

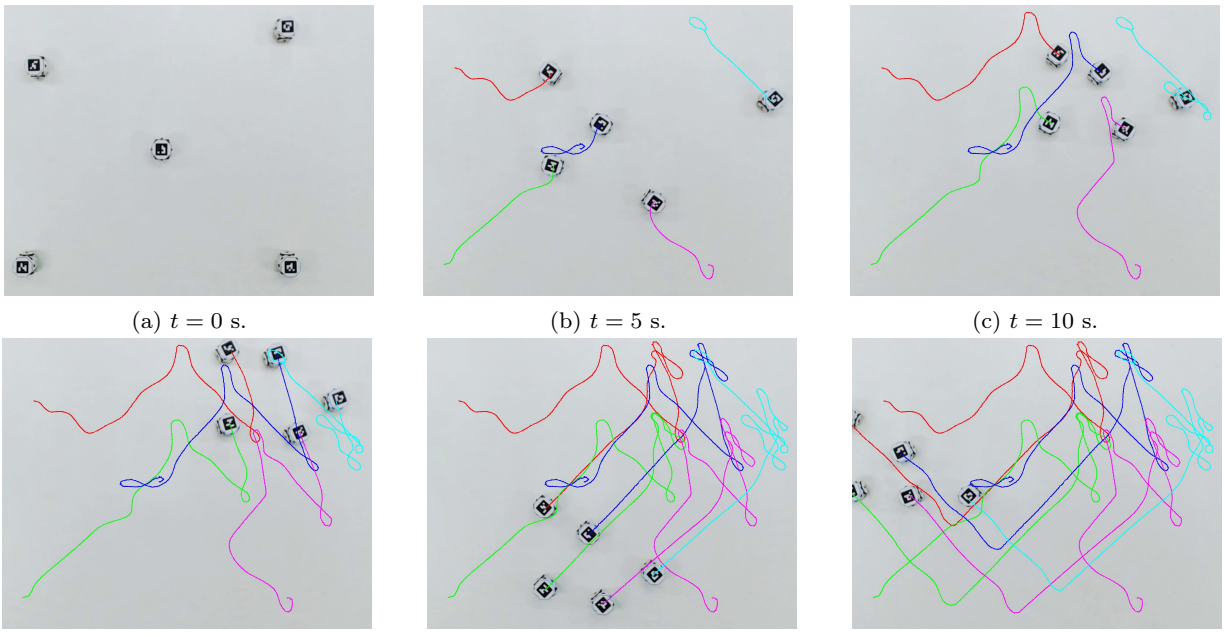
[13] Decentralized Coverage: `https://youtu.be/KmQXBcXKBtE`
[14] Flocking Behavior: `https://youtu.be/u7iioSKtHU8`
[15] Cooperative Transport: `https://youtu.be/hAS7FKYkKWQ`



(a) $t = 2$ s. (b) $t = 4$ s. (c) $t = 10$ s.

(d) $t = 20$ s. (e) $t = 80$ s. (f) $t = 120$ s.

Fig. 18: Snapshots of an experiment showing five HeRo robots performing decentralized coverage.



(a) $t = 0$ s. (b) $t = 5$ s. (c) $t = 10$ s.

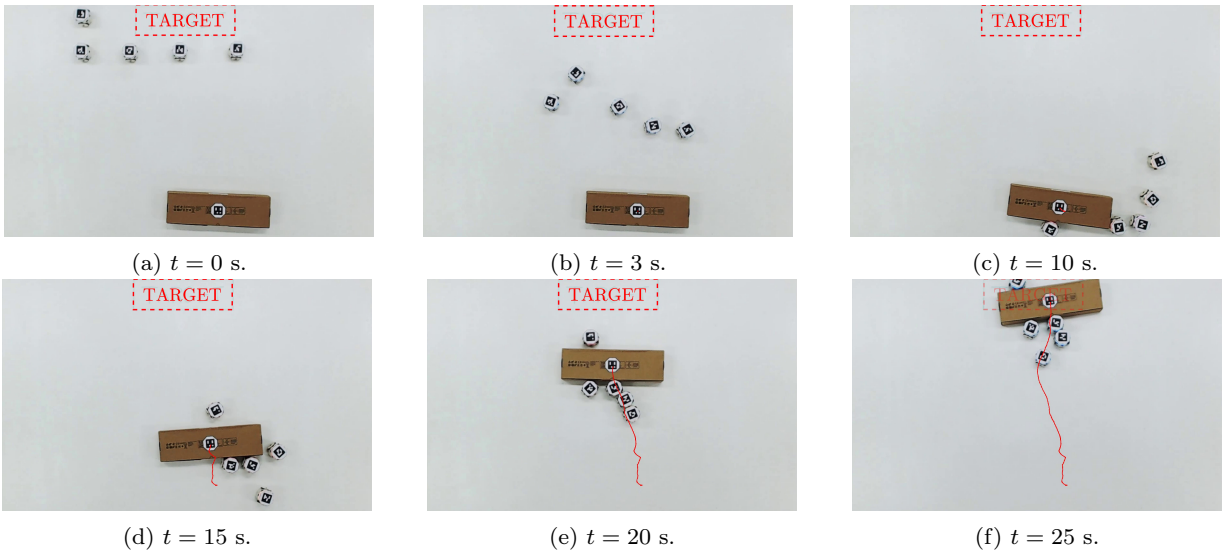
(d) $t = 15$ s. (e) $t = 25$ s. (f) $t = 30$ s.

Fig. 19: Snapshots of an experiment showing five HeRo robots performing flocking behavior.

7.1 Limitations

Despite the robot's remarkable performance, there are still some issues that impact its use and maintenance. Below we list and discuss some of these points.

- **Reproduction/Assembly**: although the robot has a simple mechanical design, the use of additive manufacturing technologies such as conventional 3D printers does not always allow a proper fit of the parts. Then, it requires manual adjustments or finishing during assembly, demanding time and effort. This process is essential for the robot's transmission mechanisms, impacting wheel movement and encoder readings if left unattended.
- **Robot calibration**: the robot is designed to use affordable parts and components that have been available for several years. As expected, low-cost components also impact robot performance, requiring the user to calibrate the IR sensors and motors occasionally. As each component has different charac-



(a) $t = 0$ s. (b) $t = 3$ s. (c) $t = 10$ s.

(d) $t = 15$ s. (e) $t = 20$ s. (f) $t = 25$ s.

Fig. 20: Snapshots of an experiment showing five real HeRo robots transporting an object toward its goal location.

teristics, the calibration process is required for each of the eight IR sensors and the two servo motors.
- **Wireless recharge**: another point is the lack of convenience in charging the battery of each robot by plugging in a cable. Recently, wireless charging modules have become commonplace, but they still come at a high cost compared to the robot's cost. Although the idea of having an automatic recharge system is interesting, due to the cost and size, we decided to wait and deal with the manual recharge of the robots.
- **Mechanical wear**: finally, another point impacted by the use of low-cost components is their durability. Although the rotary encoder is an interesting solution, some low-cost models has a short lifespan for our application, requiring replacement after months of use. In addition to the encoder, we also have to check the gear mechanism since we use ABS/PLA material that wears out with use and storage.

7.2 Future Work

We are continuously working on improving HeRo capabilities. In a near future, we intend to develop newer expansions to the platform in the form of e-Hats, improve the internal filters for localization and improve the assembly process of the platform. We will also validate newer forms of localization using UWB or other indirect wireless methods. A full migration to ROS2 will also be performed to maintain the software stack of the robot up to date with current robotics advancements.

Conflict of interest

The authors have no conflicts of interest to declare.

References

1. Arvin, F., Espinosa, J., Bird, B., West, A., Watson, S., Lennox, B.: Mona: an affordable open-source mobile robot for education and research. Journal of Intelligent & Robotic Systems pp. 1–15 (2018)
2. Arvin, F., Samsudin, K., Ramli, A.R., Bekravi, M.: Imitation of honeybee aggregation with collective behavior of swarm robots. International Journal of Computational Intelligence Systems **4**(4), 739–748 (2011)
3. Arvin, F., Samsudin, K., Ramli, A.R., et al.: Development of a miniature robot for swarm robotic application. International Journal of Computer and Electrical Engineering **1**(4), 436–442 (2009)
4. Arvin, F., Turgut, A.E., Bellotto, N., Yue, S.: Comparison of different cue-based swarm aggregation strategies. In: International Conference in Swarm Intelligence, pp. 1–8. Springer (2014)
5. Arvin, F., Watson, S., Turgut, A.E., Espinosa, J., Krajník, T., Lennox, B.: Perpetual robot swarm: long-term autonomy of mobile robots using on-the-fly inductive charging. Journal of Intelligent & Robotic Systems **92**(3-4), 395–412 (2018)
6. Benet, G., Blanes, F., Simó, J.E., Pérez, P.: Using infrared sensors for distance measurement in mobile robots. Robotics and autonomous systems **40**(4), 255–266 (2002)
7. Caprari, G., Siegwart, R.: Design and control of the mobile micro robot alice. In: Proceedings of the 2nd International Symposium on Autonomous Minirobots for Research and Edutainment, AMiRE 2003: 18-20 February 2003, Brisbane, Australia, pp. 23–32. CITI (2003)
8. Eshaghi, K., Li, Y., Kashino, Z., Nejat, G., Benhabib, B.: mroberto 2.0–an autonomous millirobot with enhanced locomotion for swarm robotics. IEEE Robotics and Automation Letters **5**(2), 962–969 (2020)
9. Farrow, N., Klingner, J., Reishus, D., Correll, N.: Miniature six-channel range and bearing system: algorithm, analysis and experimental validation. In: 2014 IEEE

International Conference on Robotics and Automation (ICRA), pp. 6180–6185. IEEE (2014)
10. Hostettler, L., Özgür, A., Lemaignan, S., Dillenbourg, P., Mondada, F.: Real-time high-accuracy 2d localization with structured patterns. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 4536–4543. IEEE (2016)
11. Hu, C., Fu, Q., Yue, S.: Colias iv: The affordable micro robot platform with bio-inspired vision. In: Annual Conference Towards Autonomous Robotic Systems, pp. 197–208. Springer (2018)
12. Kernbach, S.: Swarmrobot. org-open-hardware microrobotic project for large-scale artificial swarms. arXiv preprint arXiv:1110.5762 (2011)
13. Kim, J.Y., Colaco, T., Kashino, Z., Nejat, G., Benhabib, B.: mroberto: A modular millirobot for swarm-behavior studies. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2109–2114. IEEE (2016)
14. Klingner, J., Kanakia, A., Farrow, N., Reishus, D., Correll, N.: A stick-slip omnidirectional powertrain for low-cost swarm robotics: Mechanism, calibration, and control. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pp. 846–851. IEEE (2014)
15. Koenig, N.P., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IROS, vol. 4, pp. 2149–2154. Citeseer (2004)
16. Le Goc, M., Kim, L.H., Parsaei, A., Fekete, J.D., Dragicevic, P., Follmer, S.: Zooids: Building blocks for swarm user interfaces. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology, pp. 97–109. ACM (2016)
17. Limeira, M.A., Piardi, L., Kalempa, V.C., de Oliveira, A.S., Leitão, P.: Wsbot: A tiny, low-cost swarm robot for experimentation on industry 4.0. In: 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), pp. 293–298. IEEE (2019)
18. Lin, M.S., Chen, C.L.: An led driver with pulse current driving technique. IEEE transactions on power electronics **27**(11), 4594–4601 (2011)
19. Malyuta, D.: Guidance, navigation, control and mission logic for quadrotor full-cycle autonomy. Master's thesis, ETH Zurich (2018)
20. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th conference on autonomous robot systems and competitions, LIS-CONF-2009-004, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco (2009)
21. Mondada, F., Franzi, E., Guignard, A.: The development of khepera. In: Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop, CONF, pp. 7–14 (1999)
22. Mondada, F., Franzi, E., Ienne, P.: Mobile robot miniaturisation: A tool for investigation in control algorithms. In: Experimental robotics III, pp. 501–513. Springer (1994)
23. Olaronke, I., Rhoda, I., Gambo, I., Oluwaseun, O., Janet, O.: A systematic review of swarm robots. Curr. J. Appl. Sci. Technol. **39**, 79–97 (2020)
24. Özgür, A., Lemaignan, S., Johal, W., Beltran, M., Briod, M., Pereyre, L., Mondada, F., Dillenbourg, P.: Cellulo: Versatile handheld robots for education. In: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, pp. 119–127. ACM (2017)
25. Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., Egerstedt, M.: The robotarium: A remotely accessible swarm robotics research testbed. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 1699–1706. IEEE (2017)
26. Pickem, D., Lee, M., Egerstedt, M.: The gritsbot in its natural habitat-a multi-robot testbed. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 4062–4067. IEEE (2015)
27. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: Ros: an open-source robot operating system. In: ICRA workshop on open source software, 3.2, p. 5. Kobe, Japan (2009)
28. Rezeck, P., Assunção, R.M., Chaimowicz, L.: Cooperative object transportation using gibbs random fields. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 9131–9138 (2021). DOI 10.1109/IROS51168.2021.9635928
29. Rezeck, P., Assunção, R.M., Chaimowicz, L.: Flocking-segregative swarming behaviors using gibbs random fields. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 8757–8763 (2021). DOI 10.1109/ICRA48506.2021.9561412
30. Rezeck, P.A.F., Azpurua, H., Chaimowicz, L.: Hero: An open platform for robotics research and education. In: 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), pp. 1–6 (2017). DOI 10.1109/SBR-LARS-R.2017.8215317
31. Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., Nagpal, R.: Kilobot: A low cost robot with scalable operations designed for collective behaviors. Robotics and Autonomous Systems **62**(7), 966–975 (2014)
32. Siegwart, R., Nourbakhsh, I.R., Scaramuzza, D.: Introduction to autonomous mobile robots. MIT press (2011)
33. Slavkov, I., Carrillo-Zapata, D., Carranza, N., Diego, X., Jansson, F., Kaandorp, J., Hauert, S., Sharpe, J.: Morphogenesis in robot swarms. Science Robotics **3**(25) (2018)
34. Soares, J.M., Navarro, I., Martinoli, A.: The khepera iv mobile robot: performance evaluation, sensory data and software toolbox. In: Robot 2015: second Iberian robotics conference, pp. 767–781. Springer (2016)
35. Wang, J., Olson, E.: Apriltag 2: Efficient and robust fiducial detection. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4193–4198. IEEE (2016)
36. West, A., Arvin, F., Martin, H., Watson, S., Lennox, B.: Ros integration for miniature mobile robots. In: Annual Conference Towards Autonomous Robotic Systems, pp. 345–356. Springer (2018)
37. Wilson, S., Glotfelter, P., Wang, L., Mayya, S., Notomista, G., Mote, M., Egerstedt, M.: The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multi-robot systems. IEEE Control Systems Magazine **40**(1), 26–44 (2020). DOI 10.1109/MCS.2019.2949973
38. Yu, J., Han, S.D., Tang, W.N., Rus, D.: A portable, 3d-printing enabled multi-vehicle platform for robotics research and education. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 1475–1480. IEEE (2017)