

Cheap Autonomous Rovers for Multi-Agent Applications

*Freedom Rover Units

1st Jordy A. Larrea Rodriguez

Department of Electrical and Computer Engineering

University of Utah

Salt Lake City, USA

Jordy.larrearodriguez@gmail.com

2nd Brittney L. Morales

Department of Electrical and Computer Engineering

University of Utah

Salt Lake City, USA

brittneymrls@gmail.com

3rd Misael Nava

Department of Electrical and Computer Engineering

University of Utah

Salt Lake City, USA

misaelnava812@gmail.com

Abstract—The state of the art in autonomous swarms employs a decentralized model consisting of multi-agent networks. These robotic collaborative systems hold the potential to adapt to new environments and optimize individual performance to specific tasks without having to deal with global systems prone to single points of failure. Our team’s focus lies therein in developing a multi-agent system capable of a decentralized network. The swarm will incorporate three two-wheel differential drive rovers interfaced through the Robot Operating System (ROS) via wifi telecommunication through the micro-ROS agent service. The central bay system consists of a single laptop to communicate objectives for the agents to complete (carefully designed demos). Our development stack will leverage ROS for project management, simulation capabilities, navigation libraries, and native server-client model in robotics applications. The rovers AI will incorporate simultaneous localization and mapping (SLAM) techniques for RT positioning based on a priori grid or map; thus, facilitating navigation through improved state space mapping. Thus, we introduce FRU-bot, an autonomous platform capable of deployment in decentralized and centralized systems.

Index Terms—Decentralized Communication, Swarm Communication, Multi-agent, ROS, Gazebo

I. INTRODUCTION

[Brittney]

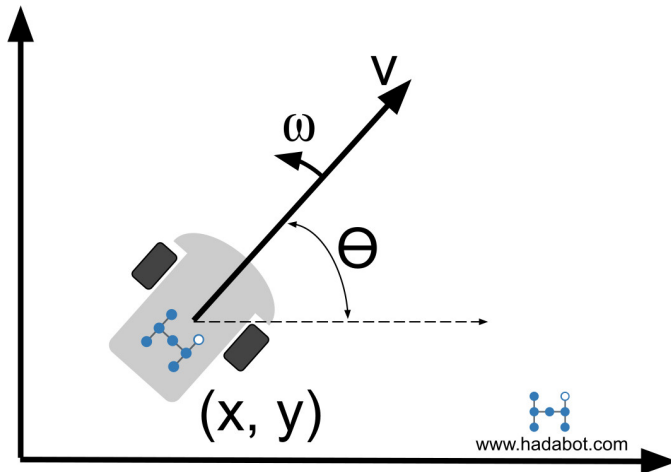


Fig. 1. Graphical representation of 2D robot pose information standard to 2 wheel robots [1].

II. BACKGROUND

A plethora of elements must come together to successfully deploy a swarm of decentralized agents. Our system requires a working knowledge in both the physical and abstract. The agents are subject to their physical characteristics: i.e., their circuitry, power consumption, mechanical components, sensors, processing power, and geometry to navigate and respond to an environment. The complexity that drives cyber-physical systems is what largely makes robotics difficult. An agent must be able to function perfectly in a world with a continuous state space. Today, researchers are seeing a high degree of success with robots that incorporate mathematical and data driven modeling that leverages the superior computational capabilities of modern processors to solve complex continuous problems in robotics.

A. Common Robot Autonomous Platforms

Common mobile platforms for autonomous robots encompass a range of designs tailored to diverse applications within the field of robotics. Wheeled platforms, such as differential drive and omnidirectional robots, are popular choices due to their simplicity and efficiency in navigating various environments. Differential drive platforms use two powered wheels, allowing the robot to turn and move forward or backward by adjusting the speeds of the wheels. Omnidirectional platforms, on the other hand, utilize special wheel configurations to achieve unrestricted movement in any direction. Lastly, tracked platforms offer enhanced traction and stability, making them suitable for challenging terrains.

One notable platform is the TurtleBot 3, a popular open-source robot platform widely used for research and education. TurtleBot 3, developed by ROBOTIS, is a compact and modular robot equipped with various sensors, including a 360-degree lidar sensor, a camera, and inertial measurement units (IMUs). Its differential drive system allows for omnidirectional movement, making it versatile for navigation in dynamic environments. The TurtleBot 3 is often employed in the development and testing of autonomous navigation algorithms, mapping techniques, and obstacle avoidance strategies. Its affordability, ease of use, and vibrant community support make it an attractive choice for researchers, students, and hobbyists interested in exploring the intricacies of autonomous robotics.

B. Robot Localization

Normally, robots use a wide range of sensors to inform themselves about the surrounding environment. Our agents, for example, rely on a 6 axis inertial measurement unit (IMU) and rotatory encoders to produce an $\{X, Y, \theta, \dot{X}, \dot{Y}, \dot{\theta}, \ddot{X}, \ddot{Y}\}$ state configuration at every time step by considering physical properties and mechanics (Fig. 1). However, a frequent problem with using rudimentary techniques solely for odometry is that error propagated at each time step due to environmental conditions, sensor specifications, electrical noise, and rounding error can deviate a naive agent during localization. The problem persists even with the use of a known map and highly accurate sensors. Consider how a human might traverse

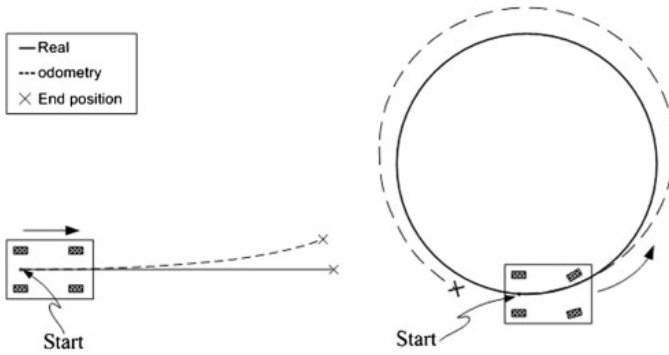


Fig. 2. Graphical representation of basic odometry error as a robot follows a reference trajectory [20].

a city by using a map. The human could ensure their own position by simply pondering their place in respect to features of the map like buildings, intersections, or street signs. Even if the human absent mindlessly navigated the city, they could triangulate their location by simply calculating their positions based on observations (such as the coffee shop across the street). Our robots, incorporate a Kalman ExFor an agent, SLAM leverages known knowledge of poses at past time steps and emissions or observations (sensor readings) made as the agent accumulates error through basic odometry to calculate probabilities of where the agent can be in space. Any number of sensors that detect 'emissions' from the environment can be used for SLAM; however, 2D lidar and cameras are regularly used for the high degree of precision awarded. Agents are capable of both learning a new environment and updating a known map with SLAM; thus, justifying our deployment of SLAM to improve the navigation capabilities of each agent.

C. Robot Kinematics

iJordy

D. ROS2 and Open Source Frameworks

iJordy

E. Robot Description and Transformation Trees

iJordy

F. Gazebo for Robot Simulation

iJordy

G. Autonomous Navigation

iJordy

H. Multi-Agent Swarm Systems

iJordy

III. SCOPE OF PROJECT AND DESIGN

The scope/goal of the project is to create a small group(our swarm) of semi-decentralized rovers that achieve a common goal. Our swarms will consist of an aim of 3+ rovers which although does not constitute an ideal amount for a swarm, however this will be enough for a proof of concept for the future that will use a larger group of individuals. As for the common goal, the team of robots should be able to work together to put on a little synchronized dance performance or to push a block from point A to another point in space. While the concept of these rovers being decentralized means that they are able to act independently with assistance from a central leader. This now leads into the question of how these autonomous rovers are going to be made and how they will be aware of their environment. There are multiple ways we can get the rovers to perceive their surroundings, in the case of the project there are three options. One option is to use a 2D Light Detection and Ranging (lidar) sensor. The lidar will give the robot an image of the objects currently around it and the distance from the unit itself. With the implementation of lidar it will open up the avenue to accurately use SLAM(Simultaneous Localization and Mapping). For the actual build of the rovers the plan is to use a premade chassis for the outer shell of the robot. It will also use two stepper motors and a caster wheel to enable movement. As for powering/controlling the system, it will use a custom PCB power module for powering the different components with another custom PCB motor driver to control the stepper motors.

IV. METHODS AND PLANNING

A. Initial Timeline and Project Tasks

Most milestones will be collaborative, meaning they are not going to be directly assigned to people on the team. The only exception is the first milestone where the team members that still do not have a good understanding of ROS will work on becoming familiar with it. However, mini-milestones will be assigned as the team gets to them. An example of this is when it comes to testing parts each person will choose which part they will test. The same will go with coding aspects of the project and these assignments will be documented to keep track of who has worked on what. Here is the initial timeline for the project broken down into individual tasks:

- May: Assemble Rover/PCB/Get Better understanding of ROS
 - Main goal is to make the custom PCB for power module and motor drivers
 - Buy parts: Motor, Chassis, LiPo Battery, development boards, environment sensing parts
 - Test PCB and parts to make sure they are functional and understand how to interface with them
 - Become familiar enough with tools to begin code development
- June: Setup ROS/Code Framework
 - Use ROS to create initial code for the multi-agent server to assist Rovers

- Also use ROS to manipulate and control motors to allow Rover to move
- Begin interfacing with sensors and other components
- July: Setup ability to see environment
 - Setup Lidar/Computer Vision
 - Get Rovers to build map of their environment SLAM
 - Ensure rovers can move through environment without colliding
- August: Get Rover to move in consistent manner and Implement Agent AI
 - Finish up main part of Rover framework
 - Get the Rovers to move in a set pattern
 - Work on the AI for rovers
- September: Setup central server/Test Communication
 - Finish creating ROS multi-agen server
 - Get the server to communicate with Rovers
 - Test reaction time between server and Rovers
- October: Rover will work together on a task
 - Get Rover to push box
 - Get Rovers to dance around
 - Or get Rovers to complete another demo that is list later
- November/December: Debug and finish up Project
 - These are free months to use to buffer if the project falls behind or need extra time to debug

B. Project Resources

1) *Software*: To finish the project on time we will need to deploy the use of external tools to assist in its completion. Luckily there are many tools at disposable to get the rovers on their wheels and working together. The first and most important resource the project will use is ROS. ROS is a tool that is made to interface well with robotics, offering a multitude of tools to help assist in the development process. Some aspects ROS assists in are “services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management” [8], essentially the operating system of ROS help handle some challenges one would face in a typical embedded process. There are different versions of ROS, in our case, we will be deploying ROS2 Humble Hawksbill. As for the reason why the team is deploying Humble, it is supported on Ubuntu Linux Jammy Jellyfish version 22.04 [9], which is the environment currently being used for the project. Another advantage of ROS is that everything is abstracted to a concept of a Node.

The official ROS documentation refers to Nodes as having a “modular purpose” [10] and being capable of sending/receiving data from other nodes via Topics and Service. These Nodes are developed in C++ or Python. It is easy to get lost in all the specifics of ROS, luckily the developers do have tutorials and other documentation on the tool itself for individuals to get started(plus ROS is open source meaning there

are alternatives even if the documentation is not useful [8]). With the assistance of ROS implementing control/deploying the rovers will be simplified to focus on other aspects of the project. Although using ROS enables easier deployment of the tools needed for the project, simulations are an integral part of ensuring that the rovers operate as intended. To create simulations of the swarm, Gazebo will address this issue. Gazebo is 3D simulation software that integrates well with ROS [10]. Like the ROS documentation Gazebo also offers some tutorials on getting started with it and how to operate it with ROS. Gazebo will allow us to test the rover code before programming and physically testing the rover themselves. This now leads to the question of how the rovers physically look and operate.

2) *Hardware*: For the hardware, let’s first look at the brains of the rover, the development board. For the project, the rovers will use either the ESP32-DevKitC-32E and or ESP32-DevKitC-32UE by Espressif (the difference between the two is 32E uses a PCB antenna and 32UE uses an IPEX antenna [11]). To list some of the specs of the device it has 448 KB ROM, 520 KB SRAM, it has WiFi, Bluetooth, UART, SPI, I2C, GPIO, ADC, and a lot of other features [11]. However, this is all trivial when other boards also offer these features. Ultimately, the reason we are going to use the ESP32 is due to ROS being used to develop the rovers. ROS does not work with any microcontroller and needs specific ones to be able to operate. Essentially, these boards take advantage of “FreeRTOS, one of the three RTOSes officially supported by the micro-ROS project, which is natively used by this family of boards, and supports the latest Foxy release of ROS 2” [12] and this does work with ROS 2 Humble version as well [13]. For what is needed for the project these boards are perfect and will allow us to control the swarm of rovers via WiFi. There is also a chance that the project will also deploy a Raspberry PI 3 or 4 to be the central server, but given that at the time of writing, there is a supply chain issue with them, the central server will run on our machines/laptops. Now let us look at the rest of the body of the rover. One of the challenges of getting the rover to work well together is to ensure that the rovers move predictably and accurately. To get the rovers to move there are a couple of items that are needed to do this correctly. The motors that are going to be used are simple DC Electric TT gear motors, these are cheap motors that are generally used in Arduino projects. However, the issue with these motors is that they do not have any form of encoding/giving feedback for knowing motor speed. To solve the problem of a lack of feedback, use of IR infrared slotted optical optocoupler module photo interrupter sensor, these are rotary encoders that can be attached to the motors for information on the motor speed and direction. Finally, with our custom H-bridge motor driver, we can use a PWM from the ESP32 to control the speed and direction of the motors. The three of these combined will allow us to accurately control and move the rovers despite the difference between motors. Then with the integration of a 2D lidar sensor, the rover will have the ability to perceive the world around them and avoid obstacles and other rovers.

All of this, including the micro-controller, is being powered through a Lipo battery that can provide enough current and power to the system. Here is a wiring schematic of the rovers.

Fig 3. displays the hardware we will be using for our rovers and approximate cost for a total of 3 rovers. The total comes to be \$344.65 and each rover comes to be \$114.88 all before tax and shipping cost. As of right now this isn't all of our parts and we will be adding and removing components as we test each of them, as we might find a less expensive component or a better alternative component for our needs throughout the summer.

V. DISCUSSION

Testing will be done regularly from the first-day our components arrive to assure quality and functionality. As we assemble each component together, testing will help in debugging and localizing the issue in the system. Creating tests for each component helps with creating a good understanding on how that device individually works as well as how it interacts with other devices and software. On the topic of software, testing is of critical importance, without testing the software integration could lead to unforeseen behavior within the system as a whole. If our tests were done correctly, on demo day it will present our flawless creation, the freedom rover units to the world.

VI. DEMO DESCRIPTION

Our goal is to demonstrate one of these three demos on demo day. The First demo shows each drone/rover coordinating with one another to align themselves to make a letter. An example of this demo: from the server we give the task of creating the letter 'L', with 5 drones scattered on the map, each rover will represent a pixel and will move to that spot, as seen in Fig.4. If there were two rovers that are the same distance away from a spot, one rover will have higher priority over the other and the rover with the higher priority will get that spot. As for the other rover it will go to the next closest open spot, as seen in Fig.5. Another option for the First demo, is that each rover will have multiple different LED lights to make a light show on the ground.

The Second demo is to have the rovers coordinate with one another to move an object from one location to another. An example: 3 rovers scattered around the map, as seen in Fig. 6. The map is 5' by 5', where (0,0) is located at the bottom left corner and (5,5) is located at the top right corner. Each rover get the task to move a cardboard box from (1,5) and (4,5) to (0,5). All 3 rovers will communicate and align themselves around the closest box, to move it to the said location, which is seen in Fig. 6. Once done, they will wait for their next instruction which may be to move it to a different location.

As for our last demo, the Third demo shows drones moving in a line around the map. For example: there are 2 tasks, the first task is to follow behind another drone and the second task is to move around the map. The second task will be given to one rover only and the first task will be given to all the other drones on the map. In this demo, there are 5 drones

scattered throughout map and drone named #2 will get the second task. As for the other Drones: #1,#3,#4,#5, they will get the first task. Let's say the order of drone conga line is : #2,#3,#1,#5,#4. If we were to separate #1 from the conga line and place it elsewhere on the map, it will find its way behind the last rover in the conga line which in this case is #4. Thus conga line will look like: #2,#3,#5,#4,#1.

As a team we have decided to focus on demonstrating the second demo for demo day in December of 2023. Our goal is to demonstrate the use of a decentralized system, where each rover will be able to do their individual task and when needed to communicate with each other to complete the set task as a swarm. Presenting the second demo, showcases this achievement.

VII. MITIGATION AND RISKS

As with any project, there are challenges and risks that need to be addressed as the project progresses. The first is the issue of what happens when one breaks or gets to us broken. To solve this issue spare parts will be ordered as a precaution when the team has more money available. Parts are going to be tested to ensure they are working properly and safe handling of said parts to avoid accidental breaking. The simulation aspect of the rover brings with it a unique set of risks, if the teams spend too much time in simulation, unforeseen issues may occur when trying to translate/test in bulk with the physical rovers. To avoid this, incremental testing will be done to ensure that each step of the process is working. Time is also a big risk, with a project as large as this one, the team plans on meeting and working over the summer to ensure that the project will be finished in time. Finally, there is the issue of the custom H-bridge and power circuit PCB. If those fail and cannot find the issue with them we may resort to premade circuit board vendors from places like Adafruit and Digikey. Overall, the biggest issue/risk of the project is taking good care of the parts and ensuring they work because devices such as the 2D Lidar and Lipo batteries are expensive. We would rather not have to buy more than what we need.

ACKNOWLEDGEMENT

Big Thanks to Prof. Daniel Drew (University of Utah) for valuable discussion and insight; Many Thanks Prof. Erik Brunvand (University of Utah) for feedback and insight. Thanks to GAZEBO tutorials for making it easy to follow instructions for setup.

REFERENCES

- [1] Jack, "Implement ros2 odometry using vscode running in a web browser," 05/05/2023 2022. [Online]. Available: <https://blog.hadabot.com/implement-ros2-odometry-using-vscode-in-web-browser.html>
- [2] W. Chen, J. Liu, H. Guo, and N. Kato, "Toward robust and intelligent drone swarm: Challenges and future directions," *IEEE Network*, vol. 34, no. 4, pp. 278–283, 2020.
- [3] R. Cooley, S. Wolf, and M. Borowczak, "Secure and decentralized swarm behavior with autonomous agents for smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*. IEEE, Conference Proceedings, pp. 1–8.

- [4] D. S. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, vol. 2, pp. 189–200, 2021.
- [5] M. S. Güzel, V. B. Ajabshir, P. Nattharith, E. C. Gezer, and S. Can, "A novel framework for multi-agent systems using a decentralized strategy," *Robotica*, vol. 37, no. 4, pp. 691–707, 2019.
- [6] S. Engebråten, K. Glette, and O. Yakimenko, "Field-testing of high-level decentralized controllers for a multi-function drone swarm," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*. IEEE, Conference Proceedings, pp. 379–386.
- [7] K. Lee, W. Chung, and K. Yoo, "Kinematic parameter calibration of a car-like mobile robot to improve odometry accuracy," *Mechatronics*, vol. 20, no. 5, pp. 582–595, 2010.
- [8] "Wiki," Aug 2018. [Online]. Available: <http://wiki.ros.org/ROS/Introduction>
- [9] "Installation." [Online]. Available: <https://docs.ros.org/en/humble/Installation.html>
- [10] "Understanding nodes." [Online]. Available: <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>
- [11] "esp32-wroom-32e_esp32-wroom-32ue_datasheet_en_2023," 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
- [12] F. Finocchiaro, "micro-ros porting to esp32," Aug 2020. [Online]. Available: <https://micro.ros.org/blog/2020/08/27/esp32/>
- [13] P. Garrido, "micro-ros humble hawksbill release," May 2022. [Online]. Available: https://micro.ros.org/blog/2022/05/31/humble_release/
- [14] R. C. Cardoso and A. Ferrando, "A review of agent-based programming for multi-agent systems," *Computers*, vol. 10, no. 2, p. 16, 2021.
- [15] T.-K. Hu, F. Gama, T. Chen, Z. Wang, A. Ribeiro, and B. M. Sadler, "Vgai: End-to-end learning of vision-based decentralized controllers for robot swarms," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Conference Proceedings, pp. 4900–4904.
- [16] S. Khaliq, S. Ahsan, and M. D. Nisar, "Multi-platform hardware in the loop (hil) simulation for decentralized swarm communication using ros and gazebo," in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, Conference Proceedings, pp. 310–315.
- [17] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and ai for uavs: Opportunities and challenges," *IEEE Internet of Things Journal*, 2022.
- [18] A. Pandit, A. Njattuvetty, and A. K. Mulla, "Ros-based multi-agent systems control simulation testbed (mascot)," *arXiv preprint arXiv:2212.12657*, 2022.
- [19] S. Qamar, S. H. Khan, M. A. Arshad, M. Qamar, and A. Khan, "Autonomous drone swarm navigation and multi-target tracking in 3d environments with dynamic obstacles," *arXiv preprint arXiv:2202.06253*, 2022.