Design of an Autonomous Shop Floor Robot (GOFR) with ROS

Eli Westbay, Isaiah Storey, Francis Nkrumah, Jr., Mert Bal, and Reza Abrishambaf
Miami University Regionals

Abstract

This paper presents the design of an autonomous, mobile guided-object fetching robot (GOFR) using the robot operating system (ROS) focusing on architecture of hardware components, electronic communication protocols and software algorithms used for control and decision-making.

The GOFR has been designed by a team of undergraduate engineering technology students as part of their capstone project course utilizing their technical skills and knowledge on mechanical design, instrumentation, communication, and control systems. The GOFR robot is a multi-purpose mobile platform with object recognition and navigation capability intended for material transport applications in computer integrated manufacturing shop floor environments. Key features for functionality including sensing, communication, control and flexible mechanical movements have been developed using technologies such as LIDAR technology, Pixy2 Object-tracking camera, infra-red sensors, linear actuators, microcontrollers, and Mecanum wheels for omnidirectional movement respectively. The GOFR has been tested in an integrated robotics laboratory for basic functionality to be used as an autonomous, automated guided vehicle. This paper presents the design methodology used for developing the GOFR as well as the preliminary results of the initial implementation. The paper also discusses the experiences and lessons learned during development and testing stages.

1. Introduction

Automated guided vehicles (AGVs) are deployed in several different application domains and the range of vehicle types has increased alongside customers' needs. Primary applications are in manufacturing, warehousing, automotive, chemical, paper-print, food, and healthcare industry. The variety of applications specify the general system requirements, such as size, load-capacity, load mechanism, navigation constraints, the number of deployed vehicles, and type of the environment. In the following, we discuss the different requirements.

The robot operating system (ROS) is employed to implement a navigation system which fulfills the requirements of an industrial material transportation system. A software system must be designed that is capable of coordinating all vehicles in the system, ensuring a conflict free traveling and performing material handling actions. To do so, methods are required to localize the vehicle, perceive the environment, and plan optimal paths through the environment. Additionally, a hardware abstraction layer is required to enable the communication with sensors and actuators.

A driver software on top must interface with high-level applications. This enables an evaluation of sensor readings and to control the robot system. ROS is an open-source middleware for robotic platforms. ROS provides all necessary features of an operating system and enables the development of applications in C++ and Python. ROS is published under the BSD license, allowing for commercial use of ROS [1]. It provides a large open-source robotics library with state-of-the-art solutions for robotics tasks such as localization, path planning, image processing, collision avoidance and motion control. The ROS runtime environment manages the execution of applications and the inter-process communication.

One of the basic goals of ROS is to enable small, mostly independent programs called nodes that all run at the same time and can be executed on different machines. That concept allows for modularization and a computational distribution [2]. An autonomous mobile robot (AMR) has been designed and demonstrated in [3]. It is based on ROS considering electronics and communication aspects of the designed architecture. The robot has been tested under various conditions to identify the issues and problems. An assisted-living low-cost robot has been designed and implemented using ROS in [4]. It was developed for Human-Robot Interaction (HRI) to provide services to elderly people. ROS has been used to develop AGVs as well. In [5], a decentralized method has been shown in order to implement an AGV controlling system. The proposed system is proposed to overcome the issues and challenges of the centralized system.

In this paper, a senior design group of students designed, developed and implemented a GOFR in pursuit of making an AGV. It is capable of receiving and delivering pallets autonomously. It is based on ROS and Lidar technology for localization, linear actuators, a Pixy2 Object-tracking camera for pallet locating and docking, and Mecanum wheels for omnidirectional movement. The paper is structured as follows: Section II describes the problem, Section III demonstrates the robot design, Section IV shows the final results, and Section V concludes the paper.

2. Problem Statement

The lab in Phelps Hall at Miami University Hamilton campus has several robotic cells, and because they are stationary robots, they are all isolated from each other. For these robots to work together, a robot that can move from station to station is required. A robot that can autonomously move to where it is needed without intervention (usually to pick up and deliver parts) is called an automated guided vehicle or AGV. Although AGVs are already prevalent in many industries, this AGV is custom designed to work with the Phelps Lab. Instead of recreating the code for the robot’s automation, we will be implementing ROS, which will oversee all tasks and operations that are required by the AGV by communicating with the drive control systems, sensing systems, localization systems, and material handling systems. Each of these systems will be designated a node. A node is an executable program that can gather information and send messages to other nodes. The drive control system will control the speed and direction of the AGV.

In cases where a workstation is connected directly to the AGV via Bluetooth, the workstation can request a job in a similar way as the webserver. Once the GOFR receives the request and is at the destination, the drive control system will switch over to docking mode. Docking mode does not use lidar, instead it will use the Pixycam to navigate. First the robot will adjust its height, using the main linear actuator and linear rails, so that it matches the dock’s height using a time-of-flight sensor pointing towards the ground. Then it will find the color-coded targets and slowly move in keeping itself aligned with the target. Once the target is a certain distance away, it will correct itself for the final alignment. Then it will move forward, until the proximity sensor in front detects an object. At this point the forks are in the pallet. It will then lift the forks until the pallet clears the base and then switch back into long range mode.

The GOFR has the ability to be used as a shop-wide AGV with multiple waypoints, to service more working centers within the lab. It will be designed to assist the other team's project along with all the robots in the lab. A special attachment has been designed so that the AGV can carry a piece of plywood to be delivered to the CNC router. But also use customized pallets to retrieve and drop off parts to different robots in the lab.

3. Robot Design

A. Frame Design

The initial design of the GOFR closely mirrored the design of the previous team. The GOFR was to have two powered wheels in the rear with two casters in the front, as shown in Figure 1.

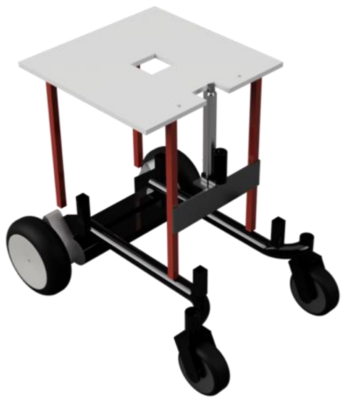


Fig. 1. Base platform.

One big advantage to this design was ease of integration. Using this design would mean there would have been less work ahead in making a functional autonomously guided vehicle. This was very appealing at first, but the drawbacks outweigh the advantages. One of the drawbacks of the caster wheel design is complexity in movements. For a robot of this design to move to a point that is not directly in front of it or behind it, it must use a series of complex turns and maneuvering to reach its objective. For example, if the wheelbase in Figure 2 needs to travel from point A to point C it must first traverse to point B, then to C. In most cases, this will not be an issue, as the robot would just need enough space to maneuver. Unfortunately, there are many spatial restraints in the area where the GOFR is to be implemented. In order to conform to the environment in which the GOFR was to operate, it needed to move more efficiently.

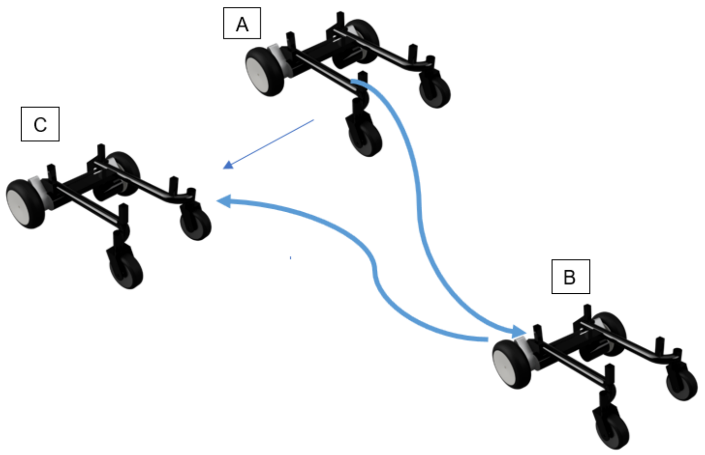


Fig. 2. Traveling issue from point A to point C.

After researching different drivetrains, the Mecanum wheel drivetrain was selected. The Mecanum wheel is depicted in Figure 3.

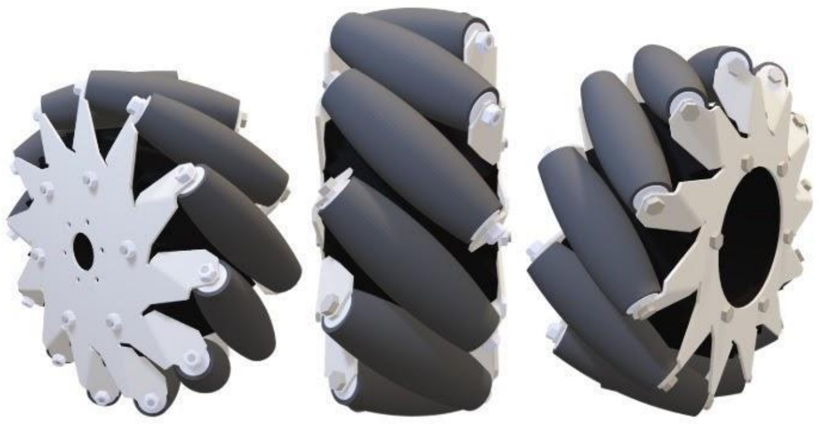


Fig. 3. Mecanum wheels.

In this case, each wheel consists of 12 individual rollers biased at 45° with respect to the endplates that sandwich the rolls. The four Mecanum wheels are driven by four independent motors. This allows for omni-directional motion or motion in any direction as shown in Figure 4. Instead of making several turns to get to the same point as the robot in Figure 2, the Mecanum wheels would allow a robot to traverse directly to that point. Realizing this is the direction in which the GOFR should go, drastically changed the initial design. The Mecanum drive train requires four independently driven Mecanum wheels, therefore four motors. The initial design of the GOFR (Figure 1) could not support the additional motors that were necessary to implement the Mecanum wheel design.

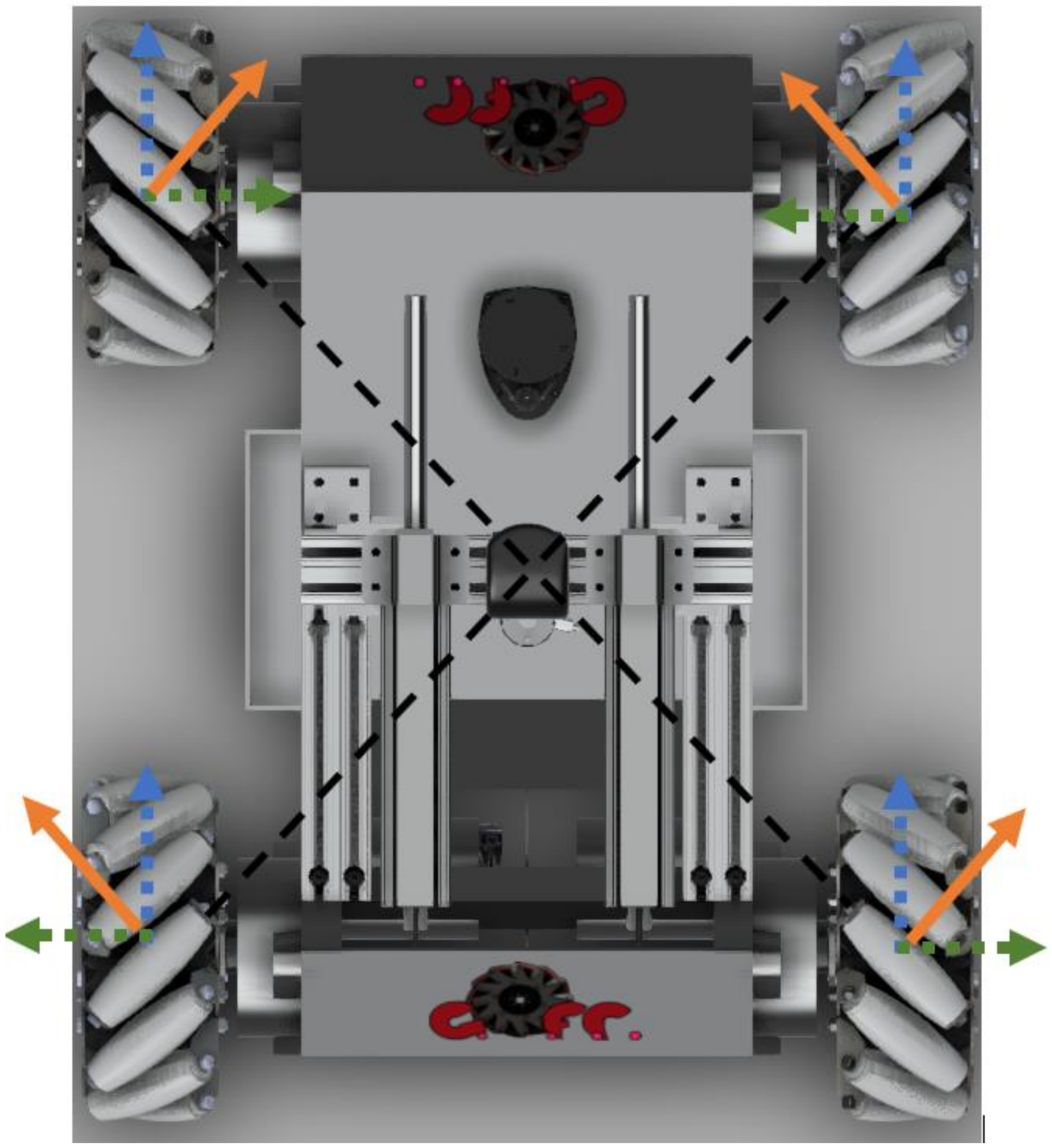


Fig. 4. Mecanum wheels and their moving directions.

B. Electrical Design

While the differential drive and the tablet axis are formed by simple DC kit motors and motor controllers, the 24 V battery originates from an e-bike. For the tablet axis, belts and pulleys from 3D printer supply were chosen. Selecting the processing unit was also fundamental since it should be low-cost, energy-efficient but performant. That's why we picked an octa-core computing device Odroid XU4 with eMMC flash storage. We found that it is capable of running the software that the robot needs with minimal power consumption. Other components include DC-DC converters and a Wi-Fi bridge. As it became clear very soon, that the outer shell of the robot is also costly (e.g., when 3D-printed or molded in small quantities), a design was chosen that limits covers to the mobile base, while the structure to support the tablet is made of metal tubes. To keep the vertical axis simple, all cabling between tablet and robot base was avoided. The tablet only connects to charging contacts in its lower end position. Communication is solved via Wi-Fi. MobiKa's sensors consist of a low-cost LIDAR sensor, which provides distance data in a 360◦ angle in a horizontal plane and a 3D sensor, which looks downward in a 45◦ angle along the front of the robot and allows detecting persons, tables and small obstacles on the ground. In future, the camera will be attached on an additional axis which will enable it to look forward (e.g., to recognize faces of a standing person) and also backward (e.g., for docking a battery charger). For safety reasons, the robot is also equipped with a small bumper to detect collisions. The electrical diagram is shown in Figure 5.

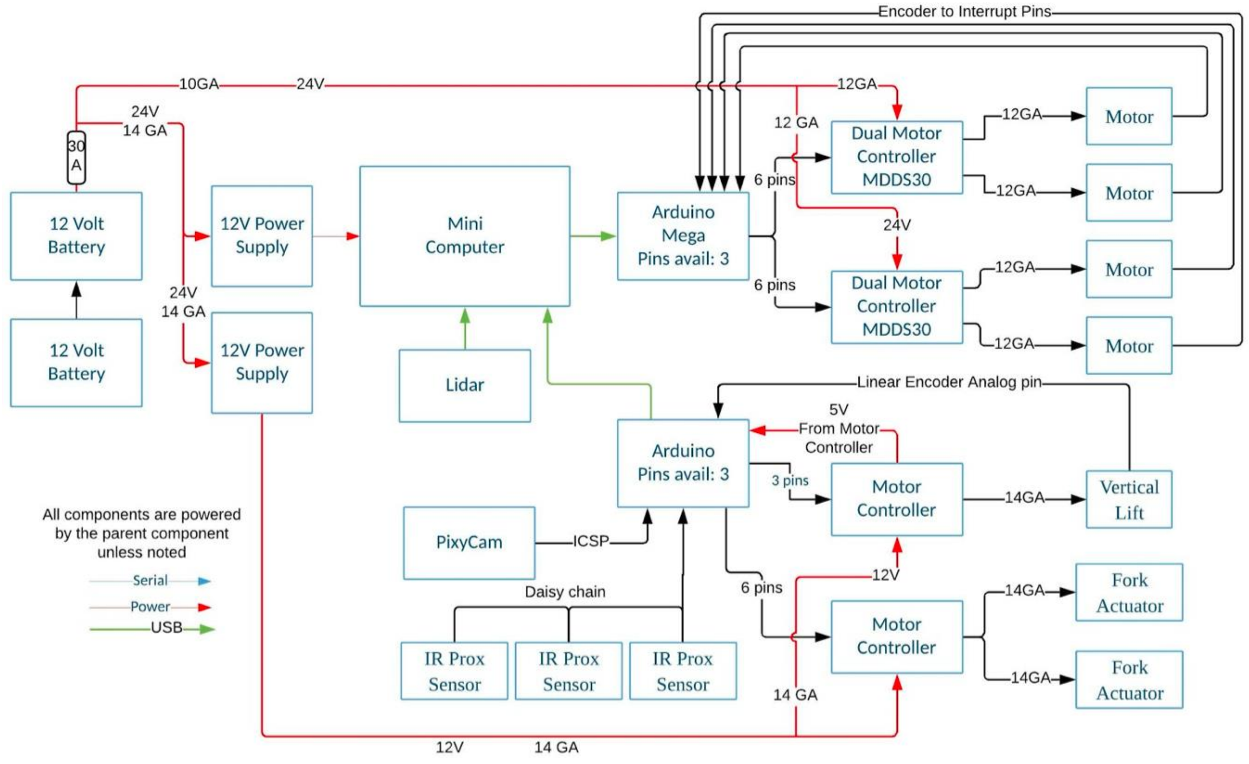


Fig. 5. Electrical diagram of the GOFR.

C. *Software Design*

To solve the automation requirement, an ROS was implemented instead of recreating the code for the robot's automation. ROS oversees all tasks and operations that are required by the AGV by communicating with the drive control systems, sensing system, localization systems, and material handling systems. Each of these systems are composed of nodes within a package. A node is an executable program that can gather information and send messages to other nodes [6]. The programs can be either C++ or Python. This adds to the compatibility of the operating system. Most of these systems will use multiple nodes to operate making it unwieldy to execute the nodes individually. The launch file takes care of this by batch loading these nodes along with any parameters required. It will even launch other launch files, allowing for a complex system that is maintained by ROS.

The drive control system controls the speed and direction of the AGV based on the information it receives from other systems. It runs a node called gofr_core that uses the arduino serial port to subscribe to the cmd_vel topic which publishes linear and rotational speeds. It then converts the speed data into motor signals and then publishes encoder readings to the encoder topic. A flow chart of the signals that are sent to the motors can be seen in Figure 6. A node converts the encoder data into odometry data. This system is a part of the GOFR_bringup package in ROS. This package handles all the initializations for the GOFR and the LIDAR as well.

The sensing system will monitor all sensor data, like the ultrasonic sensors for side and rear object detection, and the sensors found on the material handling system. The localization system will oversee scanning, mapping, and path planning. Finally, the material handling system will monitor and prioritize the job queue. These systems will be made up of multiple nodes. For instance, the sensing system will require a node for each sensing module that is installed on the robot. These nodes will communicate with each other and ROS master using a subscribe and publish routine (Figure 6).

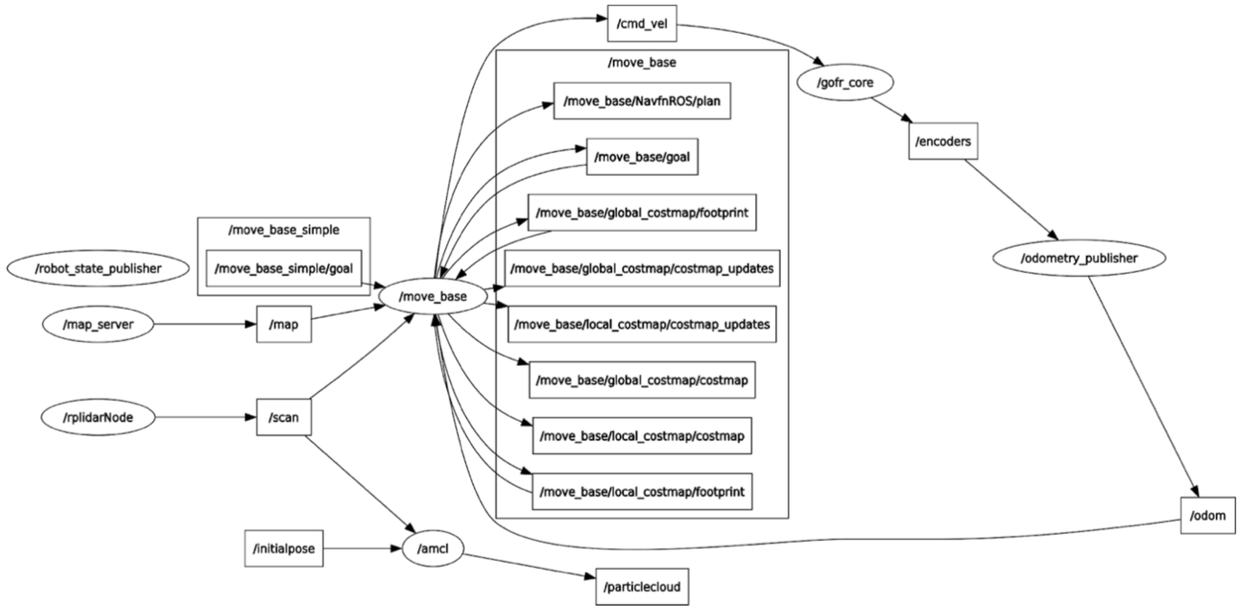


Fig. 6. ROS master based on publish and subscribe.

This same routine will also be used to receive job requests. The job request function is crucial to this project because this is what makes the GOFR versatile. Without it, it would just be a robot hardwired to go from point A to point B. This function can be accessed from an app or website. Once the user sends a job request, a job-planning node within ROS will receive it, prioritize it, and then store it until the path planning node requests the next job. Once this happens the job node will send a one-time message back, stating the new location. This type of message is called a service message, it is only sent when requested as this helps reduce the load on the processor.

An example of the map and lidar data can be seen in Figure 7. This is a map that was captured by the GOFR and it is what the Navigation system uses. The light gray space is considered unoccupied space whereas the black and dark gray areas are considered occupied and unknown spaces, respectively. The GOFR is represented by the gray rectangle which serves as a footprint for the software to use when calculating paths. The green arrows represent where the algorithm thinks the GOFR could be based on the current data. The GOFR is considered to be in the center of the cluster. The denser the arrow grouping, the more likely the GOFR is actually in that location. The blue and pink colors near the map's walls represent where the algorithm thinks the walls are. It is like a heat map where the pink is higher probability than the blue. This updates and becomes more accurate as the GOFR changes positions.

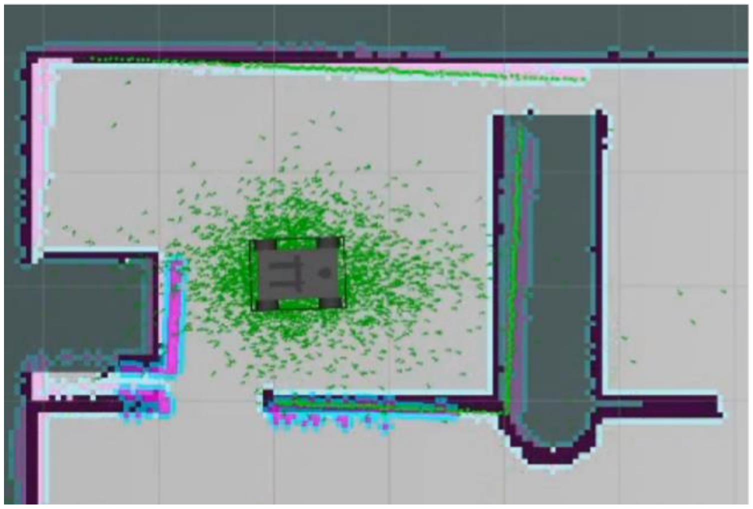


Fig. 7. The map of lidar data.

4. Final Result

The final result is shown in Figure 8. Most of the testing and programming took place on a smaller robot with Mecanum wheels. Autonomous motion as well as mapping of an entire room was achieved.

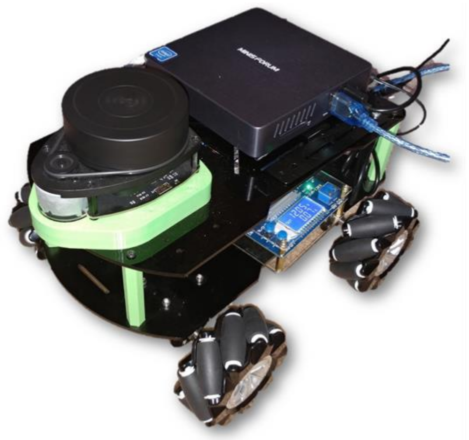


Fig. 8. The GOFR.

The implementation of the Mecanum wheel drive train on the GOFR itself was also successful. A manual version of the GOFR was completed. Initially the motors were allowed to operate at full speed. Without testing in this manual mode, a full speed autonomous GOFR would have been dangerous. The four-motor drive allowed the GOFR to reach very high speeds and in some instances it became unpredictable. This manual mode allowed the GOFR team to determine safe operating speeds and to clamp the motors' speed down, well out of the danger zone. These precautions ensure that the GOFR is ready to operate fully autonomous.

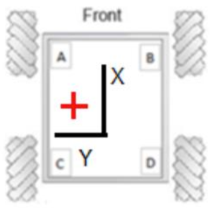
In the end the GOFR was tested via remote control, making full use of its omnidirectional capabilities as well as operating the linear actuators. In order to make full use of the omnidirectional capabilities using a remote control, it was necessary to implement several equations. These equations (Table I) are what drove the speeds and directions of each wheel. For example, in order to move the GOFR at a 45° angle (with respect to horizontal) while staying forward-facing, it is necessary to go through each equation. Since the GOFR is to remain forward facing, W will be 0 for each equation. A 45° angle is achieved in this frame when X is positive, Y is negative and both are equal in magnitude. So if an arbitrary speed is chosen, say 10m/s, Wheels B and C will not move. Wheels A and D, however, will both move at 20m/s. This will propel the GOFR forward-facing at a 45° angle.

Table I. Speed and direction equations.

$Wheel_A = X - Y - W$ $\quad$ $Wheel_B = X + Y + W$

$Wheel_C = X + Y - W.$ $\quad$ $Wheel_D = X - Y + W$

where X is forward speed, Y is lateral speed, and W is the angular velocity



5. Conclusions

A group of undergraduate senior design students of the Department of Engineering Technology designed and implemented an AGV called GOFR. The project allowed for a deeper view into robot automation with ROS, a more in-depth perspective on the advantages of omni positional drive systems, like Mecanum wheels, and how to design them, and it has created many new opportunities for future students to learn new skills while studying engineering technology.

References

[1] Fazlollahtabar, H and Saidi-Mehrabad, M, "Autonomous Guided Vehicles: Methods and Models for Optimal Path Planning", *Springer*, Boston, MA. 2015.
[2] P. Goebel, "ROS By Example", Vol. 2, Lulu, 2014.
[3] M. Köseoğlu, O. M. Çelik and Ö. Pektaş, "Design of an autonomous mobile robot based on ROS," 2017 *International Artificial Intelligence and Data Processing Symposium (IDAP),* 2017, pp. 1-5, doi: 10.1109/IDAP.2017.8090199.
[4] Florenz Graf, Çağatay Odabaşı, Theo Jacobs, Birgit Graf, Thomas Födisch, "MobiKa - Low-Cost Mobile Robot for Human-Robot Interaction", *28th IEEE International Conference on Robot and Human Interactive Communication*, October 14-18, 2019, New Delhi, India. Piscataway, NJ, USA.
[5] R. Walenta, T. Schellekens, A. Ferrein and S. Schiffer, "A decentralised system approach for controlling AGVs with ROS," *2017 IEEE AFRICON*, 2017, pp. 1436-1441, doi: 10.1109/AFRCON.2017.8095693.
[6] Evangelos Matsinos, "Modelling of the motion of a Mecanum-wheeled vehicle", *Institute of Mechatronic Systems*,

Zurich University of Applied Sciences, 2012.