Technical Feasibility

November 12, 2021

Team Poseidon Way-Finding

Sponsor: Michael Leverington

Faculty Mentor: Han Peng

Team:

Fernando Diaz

Ulugbek Abdullayev

Brandon Jester

Jonathan Gomez

1.0 Introduction

Robots can do fast calculations and accomplish billions of commands per second. Although there are many autonomous movement and obstacle avoidance modules, many of the solutions are not concrete. Movement and object avoidance is essential to robotics, as it may open the door to many possibilities. Such as autonomous navigation vehicles, capable of transporting passengers without human assistance. But basic movement and object avoidance must be mastered before progressing to more complex structures.

Dr. Leverington, the client, is a professor of computer science at Northern Arizona University (NAU), and his goal has been to forge the minds of future computer scientists. His business has involved teaching students to problem solve and teaching them to solve otherwise complex problems. His motto relies on his ability to forge young minds to wield the powers of technology, mainly computer programming.

The client has focused on developing a flexible, cost-effective robotics platform to be used within college-level programs for educational purposes. Dr. Leverington made the Thirty Gallon Robot to accomplish that robotics platform, initially known as the robot-assisted tours or RAT. The Thirty-Gallon refers to the tank which encases the robot's components.

The problem is that the client's robot does not currently move without user input. Previous Capstone Projects made it move with a joystick controller only. Therefore, the problem is making a robot with a simple movement and obstacle avoidance module. The project requires that the robot autonomously goes from one end to the other end of the long hallway of the second floor of the engineering building and to come back to the original starting point. Furthermore, the robot must be able to do this, all while avoiding obstacles.

Movement is challenging to implement because computers are only able to acquire and execute data. Robots are essentially dumb, as they are not able to rationalize the data presented to them. Instead, they must rely on instructions provided by the programmers who built their software. Furthermore, a robot must manage relevant information to help it avoid obstacles in its path.

- Movement is not as straightforward, given that the robot must determine the route to traverse. In this case, the long hallway of the second floor of the engineering building at NAU.

- Obstacle avoidance is also challenging to implement because not only are there moving obstacles. But unforeseen occurrences can arise, such as doors being opened or detecting stairs.

It is necessary to build a robot modular enough to be used by the next generation of students studying at NAU. A fully working movement and obstacle avoidance module is the team’s aim, a robot that can move through the long hallway of the second floor of the engineering building. The robot should be able to: move through the long hallway of the second floor of the engineering building and have an obstacle avoidance module to evade obstacles.

2.0 Technological Challenges

The problem requires a robot to move straight down a long hallway while avoiding any obstacles it might encounter. While it may seem straightforward, the reality is that there are many moving parts in achieving this. One is having the robot move, and the central part is using sensors and programming to help it avoid obstacles. Furthermore, autonomy is not always easy to program, and as such, the group must acquire a deep understanding of the topic to complete this project.

The technological challenges faced are as follows:

- The robot will require a main framework or architecture on which the capstone team will build a codebase that will make up a movement module.
- The robot will need an obstacle avoidance system.
- The robot must stop when it has reached the end of the long hallway and come back.

Overcoming these challenges would produce the basis of an operational robot capable of giving tours and bring this project one step closer to being fully realized.

3.0 Technology Analysis:

This section will discuss the two significant challenges identified: a software framework and an obstacle avoidance module. The capstone team will present possible solutions and possible alternatives, along with the chosen approaches.

3.1 Software Framework:

The robot will require a framework or software architecture for interacting with the raspberry pi, the sensors, and the motors. This framework will organize the scripts and modules and allow the Raspberry Pi to receive data from sensors, compute it, and execute commands to control the robot's movement. For the framework, there are two significant characteristics required:

- Ease of Use - The programming framework should also be easy to use and understand to make software development quicker. It must allow for the creation of demos and prototypes without using too much time.
- Packages Available/Ease of Expansion - In the future, this project will require more advanced modules such as localization and navigation. Some programming frameworks come packaged with more tools and libraries that provide better starting points for these modules. It is a requirement that the framework has more access to these tools to speed up implementation time.

3.1.1 Alternatives:

ROS for Robotics: ROS or robot operating system is a framework that contains tools and libraries meant to make it easier to build robotics applications. ROS can be used as the central framework to make the robot run. It is a proven technology used for both high complexity and low complexity projects alike. Many of the libraries allow for quicker development and faster prototyping. In addition, given that it is an open-source platform, more solutions are available, allowing collaboration with other robotics experts. ROS supports the development of robotic applications using python and C++. Also, the framework is present in the industry through its brach ROS-Industrial, which brings ROS to an industry standard that requires a more robust framework (Description). Because of its many tools and libraries, ROS will allow future generations of students to add new modules to the framework more easily.

Plain Python/C++ (No Framework): It is possible to achieve the same goals without using a framework made for robotics. This approach involves using Raspberry Pi with native Python tools and libraries. This approach would possibly involve using the raspberry pi's GPIO pins for connecting to sensors and various other components like motor drivers. It could also use the USB ports to hand these tasks off to an Arduino board. Using serial connections with the Raspberry Pi and Python will establish communication between the two boards. Data collected from a graphical user interface(GUI) or keyboard input using the Raspberry Pi will send data to an Arduino to control the motor drivers and make the motors move using C++.

3.1.2 Analysis:

For analysis, the performance of Python and C++ were compared, along with ROS. In addition, how complementary the three tools are with each other was investigated. Although Python is slower than C++, as observed by Naser Tamini, who saw that "C++ is much faster than Python in running the same algorithm and instructions. It is not a surprise to most programmers and data scientists, but the example shows that the difference is significant" (Tamini). Mixing both languages with ROS will ensure the project will not be constrained by only one programming language. Furthermore, combining C++ with Python can complement each other's weaknesses that the languages have. Such as the pros and cons found in Ricardo Tellez's article where a disadvantage of Python is that it can be slow vs. C++'s advantage, which is fast when working ROS (Tellez).

3.1.3 Chosen Approach:

Technology	Ease of Use	Package Availability	Total
ROS	6	8	14
Python/C++	7	5	12

Table 3.1: Software framework Comparison, On a scale of 1-10, 10 being the best for a given characteristic

The chosen solution for the framework of this project is ROS, based on the total for the characteristics in Table 3.1. The table found that the ease of use of ROS to be just slightly more complex to use than plain Python and C++ programs. ROS is relatively more difficult to execute because it changes how scripts are written and performed at runtime. Still, it also is made much more modular as it splits sensors and receivers into separate components called nodes and utilizes a publisher and subscriber model (Holland). Nodes can be implemented into any other ROS package, making them valuable. For package availability, Python has many libraries for robotics, but ROS has many more libraries tailored explicitly to creating robotics applications. ROS has access to thousands of packages that involve “specific functionality for hardware abstraction, machine-to-machine communications, device drivers, package management, testing and visualization” (Holland). Because there are more packages intended for robotics, it is rated higher than native Python and C++ libraries.

3.1.4 Proving Feasibility:

The capstone team will prove the feasibility of this framework by utilizing the ROS directories and package for the project. The capstone team will create a simple dual-node keyboard publisher and motor driver subscriber to take in keyboard input and translate it to movement.

3.2 Obstacle Avoidance

With the programming framework now chosen, the robot will need an obstacle avoidance system. Along the robot’s path, to prove functionality, the capstone team will introduce obstacles in the way of its end goal. The robot will require an obstacle avoidance system that will detect obstacles in front of itself, efficiently move around them, and keep moving forward. Lack of obstacle avoidance can damage people, damage to the robot, or property damage. First, the robot needs a hardware sensor that will gather data to be used to detect obstacles. Then, an algorithm will compute this data into commands to move the robot around the barrier. Alternative hardware options for detection are listed below, and for these hardware sensors, the essential characteristics identified are:

- Cost: One of the project’s primary motivations is a cost-effective solution to building an autonomous robot. Thus, cheaper sensors are a more desirable solution.
- Data: A relatively efficient and comprehensive obstacle avoidance system can be completed with only small amounts of data. However, having access to a lot of information can make the system much more robust and could be combined with artificial intelligence algorithms to improve decision making (“LiDAR vs. Cameras”).
- Complexity: The system’s complexity is associated with the time it will take to implement. A more straightforward approach would be quicker to understand and create.

3.2.1 Alternatives:

Xbox Kinect: Having a central camera positioned, the Kinect can give the robot more data. Additionally, Kinect has the technology to allow the robot to detect depth and distances to objects in front of it. Cameras can give access to much more information than other sensors, allowing for better-informed decision-making. Of course, visual data is much harder to process and understand. However, it could be a better solution for future-proofing the robot.

LiDAR: LiDAR or Light Detection and Ranging is a technology where a sensor will output invisible light rays that hit a surface and bounce back to the sensor. The sensor then uses the speed of light, and the time it took for the light to be reflected to calculate the distance of the object it hit. Using LiDAR, the robot can find and analyze the length of obstacles and prevent the robot from colliding with them. LiDAR is a possible solution for obstacle avoidance and can also have applications for a future navigation module.

SONAR: SONAR or Sound Navigation and Ranging is the same concept as LiDAR. Sound waves are projected from a sensor, hit an object, and bounce back to the sensor. The sensor will find the distance between it and the object using the speed of sound. The concept is nearly identical to LiDAR; thus, their characteristics are almost the same. Because of their similarities, the same principles as LiDAR can be used and allow the robot to avoid obstacles by analyzing the proximity of nearby obstructions.

3.2.2 Analysis:

For this project's purposes, Xbox 360 Kinect sensors, one-directional lidar sensors, and one-directional sonar sensors specifically were compared for each category. There are different types of sensors; some provide more functionality for increased prices. The Kinect sensor can give much more data than LiDAR, such as words on a sign or colors of objects, while LiDAR can only identify the location of an object and its distance ("LiDAR vs. Cameras"). A Kinect sensor can receive this information along with the space to objects. And similarly, for the complexity of the data, Kinect is also more complex because LiDAR, as mentioned, provides location and distance. At the same time, the Kinect offers this and more in the form of RGB values for each frame.

3.2.3 Chosen Approach:

Technology	Cost	Data	Complexity	Total
Kinect	7	10	4	21
LiDAR	5	3	8	16

SONAR	9	3	8	20

Table 3.2: Obstacle Avoidance Comparison. On a scale of 1-10, 10 being the best for a given characteristic.

The Kinect is the chosen solution based upon the characteristics identified in Table 3.2. For cost comparison, the team calculated the average cost of each sensor, along with the necessary amount required. For LiDAR and SONAR, the price individually is much lower, but more than one sensor is needed to be feasible. For data comparison, as mentioned previously from the research done, the Kinect has access to distance data along with image data, giving it the highest score based upon the relevant data required for an obstacle avoidance module. This is also similar to the complexity rating; Kinects access more data, therefore it is more difficult to process and understand. The Kinect, while being a little more complex, has the most information while at a reasonable price point.

3.2.4 Proving Feasibility:
In order to demonstrate that the Kinect system is feasible, the capstone will give a simple command to move the robot forward down a straight hallway. Objects will be placed in front of its path and it will be tested whether it can correctly detect them and avoid them while still making it to the correct end position.

4.0 Technology Integration

With the necessary tools established, their integration will be discussed. The robot will move and avoid obstacles with the ROS (Robot Operating System) architecture integrated with the raspberry pi and additional sensors. The Xbox Kinect will enhance the robot's sensory capabilities by providing distance and image data that will aid in recognizing further barriers in its path.

Per client requirements, Raspberry Pi 4B will be the robot's central controller and serve as a processing unit for movement and obstacle avoidance. The goal is to integrate the ROS framework, Raspberry Pi, and Kinect sensors to provide movement and obstacle avoidance. To ensure the robot reaches the destination safely, the team will implement the coordinates of the 2nd-floor's long hallway of the Engineering building into the ROS program.

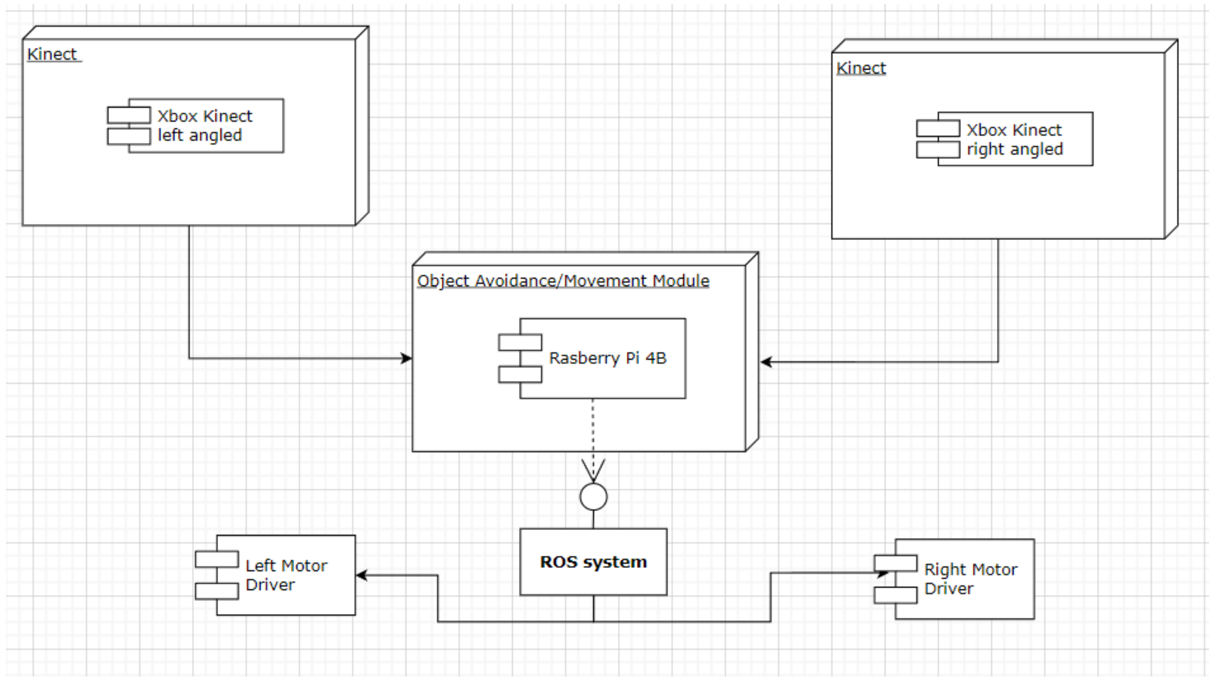

Figure 4.2 Systems Diagram

Figure 4.2 illustrates how different modules of the system will communicate with each other. By using the Kinect sensors, ROS will compute object avoidance mechanisms through the Raspberry Pi 4B module. According to Core Electronics, the Raspberry Pi can use GPIO pins to connect the robot’s motors via the breadboard circuit connection (Clinton). Python movement commands have been given to the robot to ensure the motor driver boards’ functionality. Observed results will confirm that both motor drivers are fully functional, proving the current movement implementation practical. Due to the robot's altitude from the ground, some minor objects can be hard to detect with one Kinect sensor positioned in front. Therefore, two Xbox Kinect sensors will be installed: one placed at a 30-degree angle to the right and the other placed at a 30-degree angle to the left. Both Kinect sensors will be connected to the Raspberry Pi 4B using the USB3.0 connection to feed live image data to the ROS system. At a given moment, the robot will need to avoid obstacles using the Kinect system installed. The sensors must feed visual data to the program as the robot is initialized to proceed to a specific location and detect obstacles along the way. If no challenges are present, the robot will move to the coordinates given and check whether it reached the end of the hallway using visual image detection through Kinect. The Kinect installed on the robot will constantly publish data to the ROS system. The obstacle avoidance node will subscribe to this data, process it, and decide whether the robot will need to reroute or keep going in the same path. The ROS program and Kinect will work in parallel to each other as the robot is in motion.

The integration described above is the current solution to avoiding obstacles the thirty-gallon robot might encounter while moving to and from its target location.

5.0 Conclusion:

The capstone team will develop a movement and obstacle avoidance module for the Thirty Gallon Robot for this project. The module will focus on having the robot move from one end of the long hallway of the second floor of the engineering building to the other and back again. Furthermore, the module will avoid any obstacles it may find in its path. Unfortunately, this is much easier said than done, and even a simple movement system comes with a series of challenges to overcome during development. The robot must be capable of not only moving straight down a long hallway but also maneuvering around any obstacles it detects. Finally, when the robot reaches the end of the hallway, it needs to stop and turn around. From this, two main problems were identified: Software framework and obstacle avoidance. The technical feasibility compared ROS (Robot Operating System) with native Python and C++ libraries for a software framework. ROS was the chosen solution because of its modularity and its access to robotics-specific packages and libraries. Three alternative sensors were compared for the obstacle avoidance system. LiDAR was one option that provided simplistic data but at a high cost. SONAR provided comparable data with a lower price. The Kinect sensor was the chosen solution because it collects similar distance data to LiDAR and SONAR with image data. This extra data can allow for more advanced and informed decision-making at a price point between LiDAR and SONAR.

Works Cited

Clinton. “How to Control a Motor with the Raspberry Pi - Tutorial.” *Core Electronics*, 23 Nov. 2018, https://core-electronics.com.au/tutorials/how-to-control-a-motor-with-the-raspberry-pi.html.

“Description - Ros-Industrial.” *ROS*, https://rosindustrial.org/about/description.

Holland, Sam. “Robot Operating Systems: What They Are and How to Use Them.” *Electronics Point*, https://www.electronicspoint.com/research/robot-operating-systems-what-they-are-and-how-to-use-them/.

“Lidar vs. Cameras for Self Driving Cars - What's Best?” *AutoPilot Review*, 3 July 2021, https://www.autopilotreview.com/lidar-vs-cameras-self-driving-cars/.

Oliver, Ayrton, et al. “Using the Kinect as a Navigation Sensor for ... - Auckland.” *Using the Kinect as a Navigation Sensor for Mobile Robotics*, https://www.cs.auckland.ac.nz/~burkhard/Publications/IVCNZ2012_OliverEtAl.pdf.

Mishra, Ruchik, and Arshad Javed. “ROS Based Service Robot Platform.” *IEEE Xplore Temporarily Unavailable*, https://ieeexplore.ieee.org/document/8384644.

Tamimi, Naser. “How Fast Is C++ Compared to Python?” *Medium*, Towards Data Science, 13 Jan. 2021, https://towardsdatascience.com/how-fast-is-c-compared-to-python-978f18f474c7.

Tellez, Ricardo. “Learn Ros with Python or with C++? Pros & Cons.” *The Construct*, 28 Jan. 2021, https://www.theconstructsim.com/learn-ros-python-or-cpp/.