Requirements Document

Version 2.0

November 23, 2021

Team Poseidon Way-Finding

Sponsor: Michael Leverington

Faculty Mentor: Han Peng

Team:

Fernando Diaz

Ulugbek Abdullayev

Brandon Jester

Jonathan Gomez

Accepted as baseline requirements for the project:

Client: _____ ***Date:*** _____

Team Lead: _____ ***Date:*** _____

1.0 Introduction

Robotics has been an area of interest for almost as long as computer science has been. The concept that machines could be given a task and complete it more efficiently than a human could has been even more prevalent in the past few decades than ever. This trend will only continue as technology and automation become more pervasive in our everyday lives. Because of this, robotics has been a continuously growing sector of computer science and will remain integral for decades to come. In the past, the main inhibiting factor to robotics was its power, cost, and complexity. Electronic components needed for robots have become significantly cheaper while simultaneously becoming more powerful. Because of the costs, in the past, there has been a severe lack of learning opportunities for students to use a physical robot until now. Robotics in classrooms has been too expensive to create and use. However, it has become feasible to create fully autonomous robots that remain inexpensive.

The client, Dr. Michael Leverington, is a lecturer 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. Dr. Leverington is interested in robotics and has seen this decrease in cost and lack of educational opportunity and came up with a solution to it. His answer is to develop a flexible, cost-effective robotics platform in college-level programs for educational purposes.

To accomplish that robotics platform, Dr. Leverington made the thirty-gallon robot, initially known as the robot-assisted tours or RAT. The thirty-gallon refers to the tank which encases the robot's components. The thirty-gallon barrel uses a wooden dolly as the base and has access to components such as two motors and a Raspberry Pi. The components in total cost approximately $1000. For students to use the thirty-gallon robot as an educational tool, it needs to be autonomous and support programmability. A student could create their robotics application or program and get hands-on experience with an actual robot; their program would use the robot's movement and navigation. The student would not have to worry about implementing these advanced modules.

2.0 Problem Statement

Now that the thirty gallon robot has been introduced, there lies a problem with its current implementation that will need to be fixed. The problem is that the thirty-gallon robot only has the hardware implementation completed, and there is no software to control the robot currently. The complete end goal solution of the thirty-gallon robot involves full autonomous movement and navigation. It will also require a proof of concept that supports programmability; so, the robot will be able to give tours of the engineering as fully functional proof. The project has been broken up into smaller milestones. Our problem will focus solely on the initial modules required to get the thirty-gallon robot to this end goal. Therefore, there are two features or modules needed:

- Autonomous Movement: The robot will need to move independently of human input or interference. In this case, it will drive down the long hallway of the second floor of the engineering building and back.

- Obstacle avoidance: The robot must be able to detect obstacles through a sensor. Once an obstacle is detected, the robot must reroute itself to avoid it or stay on the path.

These two modules make up the fundamental movement and basic programmability. While the robot will not be capable of full navigation, it can follow a preprogrammed or dynamic path. To ensure that it gets there safely, it will use its obstacle avoidance obstacle to prevent crashing and keep on the correct path. More advanced modules for the robot, such as navigation, can not be implemented without the fundamental movement of the robot. Therefore, our solution is integral to getting the thirty-gallon robot to its end goal of fully autonomous movement, navigation, and programmability. This project will then be given to future capstone, which will be able to fully implement the thirty gallon robot.

3.0 Solution Vision

As mentioned in the problem statement, this project's solution is the first step to the full thirty gallon robot. Because of this, for our solution, we are focused on making the thirty gallon robot capable of fundamental movement. It will be capable of movement given some input or programmed path, and it will support basic programmability. For the current module, there are two key features that this solution will require:

- The robot will be able to move autonomously down a straight hall and back
- The robot will be able to avoid obstacles while moving

To realize these features, it must be understood how the components will interact with each other. The robot has two motors attached to wheels that serve as the movement and steering. These motors each connect to a motor driver that supplies power to the motor and controls its speed. The motor drivers then link to a Raspberry Pi computer that computes the logic for the robot and sends signals to the motor driver to create movement.

The Raspberry Pi allows users to connect and control electronic components such as motors and sensors. These components communicate with each other through the Raspberry Pi, the central computer for the robot. The system will be using Robot Operating System (ROS) for the Raspberry Pi's main software framework and architecture. ROS is not an operating system but is an open-source framework that contains tools and libraries specifically designed to make building and implementing robotics applications easier. Using ROS changes how the components communicate with each other by separating them into topic publishers and subscribers. In the ROS framework, publishers gather data and send it to a topic; subscribers can follow this topic and receive data from it.

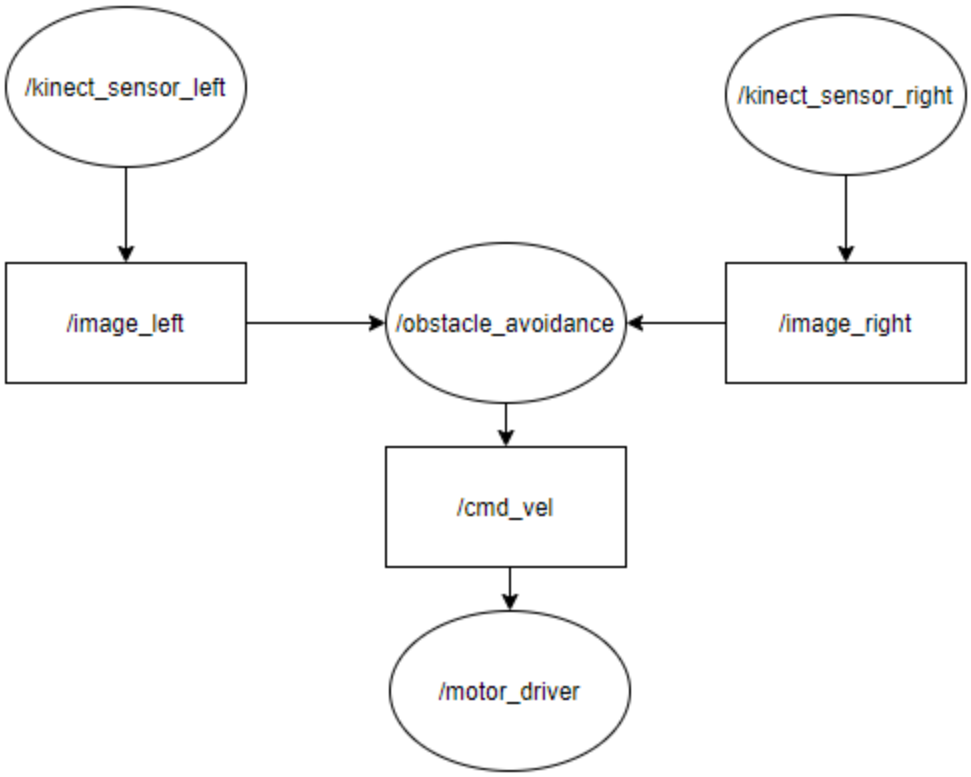

Figure 3.1: Diagram of the architecture of central ROS nodes

From *Figure 3.1*, the leading system architecture uses ROS to publish data to other nodes that will take in the data and compute it into movement commands. From the diagram, the node */kinect_sensor_left* is a Kinect sensor rotated 30 degrees left from the forward direction of the robot. This sensor publishes its image data to the */image_left* topic. The obstacle avoidance node takes in this data and, based upon the algorithm used, will decide whether there is an obstacle in the way and, if so, will publish movement commands to the command velocity topic. Finally, the motor driver controller node will translate this data into movement commands for the robot's motors. By using this system, autonomous movement and obstacle avoidance can be achieved for the thirty gallon robot. This solves the issue of creating the fundamental movement for the robot. As explained, the end goal of the thirty gallon robot is to be fully autonomous. These two modules are required in order to create later modules and meet this end goal.

4.0 Project Requirements

Now with the solution outlined, we had to gather our requirements and specify what the client wanted from this solution. From the discussions and meetings with the client, three domain-level requirements have been outlined:

- **DR1. Autonomous Movement:** The client has specified that the robot will require an autonomous movement module for the first high domain-level requirement. This means that the robot will be able to move through some program without any human input. This movement could be a specific path or a dynamically generated path.

- **DR2. Obstacle Avoidance:** The second domain-level requirement is for the robot to have an obstacle avoidance system in place. If the robot encounters a wall or obstacle, it will reroute itself to move around or away from the obstacle. It will then continue down its current path. The client required that the robot use a sensor to detect obstacles and avoid them autonomously.

- **DR3. Returning To Start:** For the final domain-level requirement, the robot will be able to recognize the end of its path, turn around and return to its starting position. While not made a specific requirement, the client expressed that in order to achieve this, the solution should be contained to the robot itself. This means that an object placed by the team to detect the end of the path would not be used.

To fully understand the functional requirements, each requirement is broken down into smaller low level requirements Each of the following high-level requirements will be broken down into these low level requirements to demonstrate a complete understanding. These requirements also have specific performance requirements to which these functional requirements' performance will be measured and quantified. There are also some environmental requirements regarding what hardware will be used that the client requires.

4.1 Functional Requirements

Each Domain Level Requirement (DR#) will be followed by ones of its lower level functional requirement (FR#) and the number of the functional requirement. The requirement will be explained, and if it has any sub requirements they will be noted by bullet point.

DR1-FR1. The computer for the robot must be able to send movement commands to the motor drivers housed on the robot.

- Program held on the raspberry pi will send commands using python to the motor drivers through its GPIO pins.
- These commands must also be able to be sent independently of each motor. The robot must be able to move one motor faster than the other in order to achieve turns and rotations.

DR1-FR2. The robot must be able to follow a pre-programmed or pre-planned path.

- Programs could be written and executed on the robot, and the robot will perform the given path..

DR1-FR3. The robot's movement must not be controlled by any human input.

- A program could be executed on the robot, and no human can directly impact the robot's movement. Does not include indirect impact such as obstacle avoidance.

DR2-FR1. The obstacle avoidance system will be able to detect obstacles

- The robot will use a sensor to gather information and determine if there is an obstacle in front of it.

DR2-FR2. The robot must be able to reroute around obstacles or away from them.

- Upon detecting an obstacle in the path movement commands will be sent to move the robot around until obstacle is no longer in view
- Upon detecting a wall, movement commands will be sent to keep the robot in the center of the path
- Robot must continue on path after rerouting.

DR2-FR3. The robot will wait for wait for obstacles to pass if they are moving

- Upon detecting obstacle and slowing down, the robot will wait briefly to see if the object has moved, and if so will wait for it to pass

DR2-FR4. Robot will not hit or bump into obstacles

- Robot will stop before it hits the obstacle once detected

DR3-FR1 Robot can recognize and identify the end of the path

- Sensor on robot will take in information, and will interpret through the robot's computer whether the end of the path has been reached

DR3-FR2 Robot will return back to its original starting position

- Robot will perform a 180 degree turn at the end of the path
- Robot will move down its path again going the opposite direction

DR3-FR3 Robot can recognize the original starting position

- Sensor will detect the end of path or original start position
- Robot will stop upon detection.

4.2 Performance Requirements

DR1-PR1. The robot must maintain a speed of an average human walking (~1.4 meters per second)

- Will slow down for obstacle avoidance

DR2-PR1. Obstacles must be detected from a distance between one to three meters

DR2-PR2. Robot must be able to detect obstacles that are ½ meters tall and above

DR2-PR3. Robot will stop around one meter to one foot away from the obstacle it detects

- Needs enough space in order to clear the obstacle or move around it.

DR2-PR4. Robot will be able to move around obstacles within 20 seconds after initial detection

DR3-PR1. Robot will detect the end of the path or its original starting point within five seconds of reaching the designated end point.

4.3 Environmental Requirements

Along with requirements on how the robot needs to function, the Client has provided a list of intrinsic requirements that must be followed during the development of the project. The client has set a set of hardware requirements that must be followed. There are three major requirements identified for hardware:

- Thirty Gallon Robot Housing
- Raspberry Pi
- Budget Limit

Thirty Gallon Robot Housing

The Thirty Gallon Robot was designed and built by previous Electrical Engineering capstone students and is intended to be the robot used in the final version of the thirty gallon robot project. The robot that was provided includes the thirty-gallon barrel as the housing, wooden dolly for the base, and significant hardware components such as the raspberry pi, two motors, and two motor drivers. Any additions in terms of modules or hardware components must be contained inside the provided thirty-gallon robot. Any electrical components will also need to be powered off of the batteries onboard the robot.

Raspberry Pi

A Raspberry Pi was requested to be used by the client Dr. Leverington, specifically the Raspberry Pi model 4B, with 8 gigabytes of ram. The Raspberry Pi is affordable and is a device that will not break the budget. Raspberry Pi's start at 35 dollars before tax, kits range at 170 dollars; it is not an extremely expensive machine. In addition, it allows for

multiple devices to be connected to it, from keyboards to monitors. Also, the 40-pin GPIO header enables the Raspberry Pi to be connected to the robot's motors and control them. This Raspberry Pi will serve as the computer of the thirty gallon robot. It is where the codebase will be held and where most of the computation for movement and obstacle avoidance will be computed. Functionality can be extended and made modular with other microcontrollers and microcomputers, but the Raspberry Pi will remain the main computer for the robot.

Budget Limit

The total cost of the robot will be cheap enough that other organizations such as colleges would be able to purchase and build this same robot. Therefore, a budget limit has been set for the new parts that can be purchased for the movement and obstacle avoidance modules. The budget limit has been set at $300.

5.0 Potential Risks

With requirements established, there lies some risks for potential errors that this solution will face. For each risk identified, its likelihood and its impact or severity on the project will also be explained. For every risk, a possible preventive solution is also presented. For the thirty gallon robot, three major risks has been identified:

- Hardware Malfunction
- Robot Moved By Object
- Miscalculation

Hardware Malfunction: The robot has many hardware components like batteries, motors, motor drivers, and the Raspberry Pi and arduino boards. A hardware malfunction could take the form of dead batteries while the robot is running, or one of the motor driver boards stops working. The effects of this error occurring can be low to moderate. There is a possibility that upon a malfunction, the hardware remains broken. In this circumstance, the program would be unable to run, and the hardware would need to be replaced causing a possible setback. Preventive measures include regular checks on the hardware to ensure that the motors and motor drivers still function, that batteries are charged, and wires are connected correctly.

Robot Moved By Object: While the robot is quite heavy, and won't necessarily be picked up, there is a possibility that someone or something can push it or move it. Granted, the likelihood of this occurring is very small, and the severity of it is small as well. For determining the end point and returning, the robot will be using the Kinect sensors. These are unaffected by movement as every frame can be analyzed regardless of location; and past locations do not have an effect on the analysis of another frame. However, the robot will not have a high level understanding of the path and its rotation on that path. It will rely primarily on moving forward until it hits the end point. So, if the robot was rotated it would move in the wrong direction or get stuck. For mitigation strategies, warning signs or warning lights should be integrated. These make the robots' existence more apparent.

Miscalculation: Bugs are bound to occur within the software. With the thirty gallon robot, the severity of a miscalculation or logical error occurring could be high. The severity depends on the circumstances however. If there is a miscalculation for the obstacle avoidance module, it could misread objects as open spaces. This could result in the robot bumping into it, possibly causing damage to both the object and the robot.

In the worst circumstances a human would get mistaken by the obstacle avoidance and possible harm could occur. The damage caused would also be related to how fast the robot would be moving, and at max speed is about as fast as a human can walk. The mitigation strategies involve making sure the robot slows down upon detecting any people or objects in front. Also, automated and manual testing will be conducted. One benefit of ROS as well is it's access to tools and specifically simulators. The obstacle avoidance code can be used in these simulated environments and can be bug fixed this way.

6.0 Project Plan

Now that key requirements have been established, and the solution towards the enlisted problems above have been discussed, the team was able to definitively plan on the ordered timeline for the project (Figure 6.0 below).

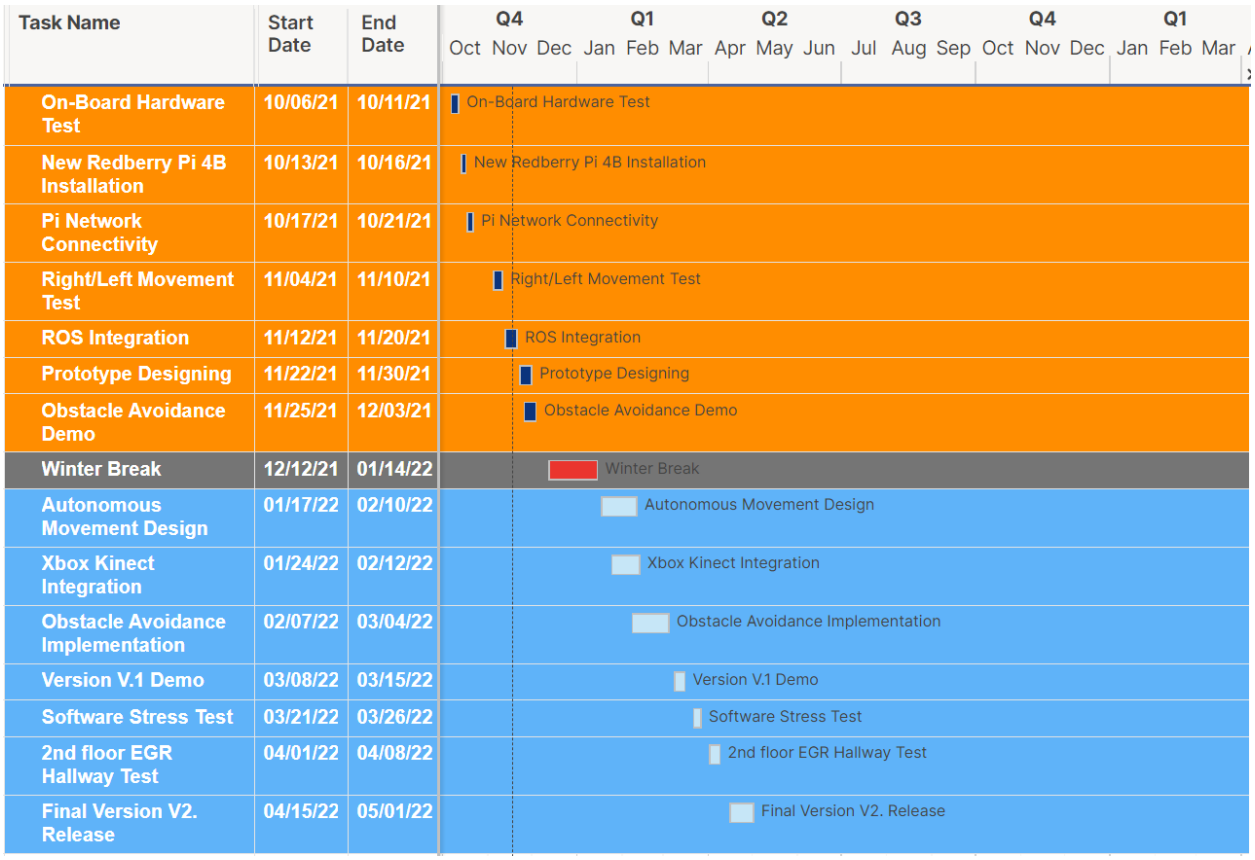

Figure 6.1: Gantt chart timeline for the project progression.

Having successfully completed the initial hardware test onboard and installed the new Raspberry Pi 4B as illustrated in *Figure 6.1*, the team is currently integrating ROS and python movement libraries into the robot. Once the ROS and basic movement algorithms have been programmed, within a few days, the team will start designing the initial prototype for the robot in which simulated obstacle avoidance will be demonstrated. From this point, the ordered plan of remaining functional requirements is listed as follows:

1. Prototype/Obstacle Avoidance Demo - November 2021

As winter break approaches, the team is planning to design an initial prototype in which simulated obstacle avoidance will be demonstrated. Few “fake” obstacles will be backed within the software to test the reactivity of the robot towards certain obstacles. Structured work on this prototype will run through the end of the first half of the capstone.

2. Autonomous Movement Design - January 2022

   Since making the robot move autonomously is one of the main goals set for the project, the team developed the initiative to get started on the task early in the second part of the capstone.

3. Kinect and Object Avoidance Integration - February 2022

   Installing Kinect is crucial for obstacle avoidance, as obstacles in the way of robots will be detected through the kinect sensors. Therefore, these two functionalities will be set forward to start around the same time so that there is plenty of time to work on the architectural setup.

4. V.1 Version Demo and Software Stress Test - March 2022

   Once the initial software version has been deployed, it is not far away from reality that potential risks might concur. As per precautionary measures, the team will stress test the software for possible outcomes.

5. Hallway Test and Final Release - April 2022

   As the final deadline approaches for the project compilation, the team is looking to test the robot in the second floor long hallway for the engineering building. Once the concept has been proven that the robot indeed is capable of moving autonomously from the start of the hallway to the end and return to the starting point, while avoiding obstacles, the final version of the software will be released.

7.0 Conclusion

Robotics is an ever-expanding field with limitless possibilities and the power of fast computations. Robotics components have gotten cheap and more powerful, yet there are few classroom uses of robots. Dr. Leverington noticed these two trends and came up with the thirty gallon robot as the solution. The thirty gallon robot is going to be an inexpensive autonomous robotic platform for use within college level programs for educational purposes. This robot needs to be autonomous and programmable so that students would be able to develop their own robotics applications and programs; this enables more practical and physical learning. Dr Leverington also hopes to make this solution extendable to other colleges and organizations in the future. Because the robot will be inexpensive it should be reasonably priced for these groups. Using the thirty gallon robot as a recipe, these organizations would be able to create their own autonomous robotic platform for use in education. The full thirty gallon robot project requires full autonomous navigation, and as a proof-of-concept that it supports programmability, will be given the task of giving tours of the engineering building. For our solution, we are implementing the first steps to this, autonomous movement and obstacle avoidance. In order to complete this, the onboard motors and motor drivers can be used to move the robot. Along with this, a Kinect sensor can be used to detect obstacles in the way. By implementing these two modules, it brings the thirty gallon robot project closer to its end goal of full autonomy.

The current schedule for team Poseidon Wayfinding is to ensure that the first steps towards a complete autonomous tour-giving robot are taken. The team is looking to implement an initial movement algorithm integrated within ROS to make the robot move autonomously in the 2nd floor long hallway of the engineering building while avoiding obstacles within the capabilities of harming the robot movement. The software and hardware architecture of the robot is as follows:

- The Raspberry Pi 4B model will be the base computer of the robot. The team will install ROS on this computer to send movement signals to the left and right driver boards. The autonomous movement module will control power and speed for the motors through this program, in which the rate of the robot is pre-determined to human walking speed. Consequently, the python algorithm will send initial directions and power to the motors through the driver boards.

- With two Xbox Kinect sensors positioned left and right-angled on the robot to ensure that any significant objects (with potential threat for the robot) within the range of 1-3 meters are being recognized.

The team's goal is to ensure that the architecture above is accomplished promptly. Therefore, the next big step towards achieving this goal is to create a mock-up version that will showcase the movement and obstacle avoidance features. Overall, team Poseidon Wayfinding is very optimistic about the prospects of this development. A well-structured timeline will help the team keep on track.