Engineering Design II

Main Project Proposal

R.A.V.I

Robot Aid for Visually Impaired

Group 6 - Team Members:

Victor Baldoceda	Vbaldoceda2015@fau.edu
Joseph Burton	josephburton2017@fau.edu
Nelson Ordonez	nordonez2018@fau.edu
Joshua Sojka	jsojka2014@fau.edu
Alejandro Moracen	amoracen2017@fau.edu

Advisors: Dr. Hari Kalva / Dr. Hanqi Zhuang

Department of Electrical Engineering and Computer Science

Date: 07 May 2020

Summary:

The visually impaired tend to need some type of service dog to help them move around. The use of guide dogs can be costly and tedious to maintain. An intelligent guide robot could be a cheaper and easier to maintain alternative that can help the visually impaired.

Table of Contents

List of Figures

Introduction

1.1 Statement of Problem and Overview of the Solution

People who are blind face a major handicap when compared to everyone else in life. A lot of them must learn how to get around either by using tools like a rod or just by their sense of touch. Some of these individuals find themselves lucky enough to get some type of service dog to help them move around. The problem with this is that the use of an animal can be very expensive and tedious to maintain. An intelligent guide robot could be a cheaper and easier to maintain alternative that can help the visually impaired. This guide bot will make use of two main object detections methods that will work parallel to one another. The first one is a Light Detection and Ranging (LIDAR), this is a remote sensing system which uses light as pulsed lasers to measure multiple distance points. This can be used to have an understanding of the surroundings of the guide bot. The information will be sent to the processor, which will use AI to have obstacle avoidance with the height of objects in mind. The second system is a camera with AI, which will be able to distinguish objects and recognize specific objects. This system will also provide feedback on objects being detected around to the user via speaker/ Bluetooth hearing device. The guide bot itself will have a bar so the user can know where the guide bot is and will also have controls. The guide bot will have a set of rules not yet determined on how to avoid, slow down, or wait, depending on the input combination from the object detection, object recognition, and user command. This whole system will also have an app for the rest of the user inputs. The app will be voice-controlled to help visually impaired users with sound feedback to communicate.

1.2 Significance

The aim is to make the use of some guiding mechanism widespread. In other words, we want everyone who needs a guide dog to be able to have the option of getting a guide bot instead. Getting a guide dog is a luxury that not many can afford, with the cost of a good guide dog going into the tens of thousands of dollars range. On top of that, the dog must be fed and cared for at the owner's expense. The guide bot would be a way of getting rid of all this, and hopefully, be more capable of guiding an individual.

1.3 Goals and Objectives

The purpose of the project is to design and develop an autonomous robot that is able to guide blind users in unknown indoor and outdoor environments. The system will be equipped with visual sensors and laser range finders.

The specific objectives are:

1. Design, construct and implement a Power Supply System. (Josh, Victor)
2. Design and implement a Sensing System. (Josh, Nelson)
3. Design and implement a Communication System. (Victor, Nelson)
4. Design and implement an AI System. (Joseph, Alejandro)
5. Design and implement an App System. (Joseph, Alejandro)

1.4 Literature Survey:

Patent CN201410756510.1: The basic idea is to build an intelligent guide dog that should be affordable and durable for the general public to promote the use of [1]. This intelligent dog would use ranging and data processing in order to fill the role of a traditional guide dog. The patented idea is very similar to our design. They chose to use an ultrasonic ranging system speed, an image acquisition system, an image processing system, a data processing system, a decision processing system, the information feedback system, the motion control system, and storage systems. Many of the systems mentioned should be able to be performed by a simple micro, while the ranging for ours would be based on LiDAR.

Patent US8195353B2: The patent was assigned by a company named Hitachi, Ltd. The parent company of Hitachi, Ltd is Hitachi Group. This is a Japanese company based out of Chiyoda, Tokyo. This patent was started on 12-18-2016, and the application was filed by Hitachi Ltd on 12-17-2007. This patent was approved on 06-05-2012, and it expires 3-18-2031 [2]. The goal of this patent is to have a robot device and guide system for guiding an unguided person, such as a visitor to a certain destination. This patent claims that a robot will be able to detect the unguided person's position and also calculate the distance to the destination.

Another feature that this will have is that the robot will have a guide robot device where the position of the unguided person can be received and transmitted. This robot is also able to identify the unguided person and display information for the recognized unguided person. This

robot will take images of the unguided person for identification purposes and also have an image storage unit; these images will also be accessible through the guide control unit. This robot has an alternative recognition system that uses voice recognition to verify the unguided person. The display will also show the unguided person the direction of the path with an arrow on the map. From reading this patent, this robot is made to be a guide robot for a large facility. For example, at a factory where you can pass the guard checkpoint, then a guide robot can recognize you and lead you to the facility you need to go to. This patent is similar to the functions we need to use to lead a visually impaired person. However, someone the guiding strategies are a little dated, which is good for us because we can take this and improve on it by adding LIDAR—more advanced image recognition using AI. We can take advantage of better batteries to have more torque, so the guide robot can also carry the load from the user, such as bags or other personal belongings.

Patent US20180178390A1: The invention created by Yang Suonho and Choi Heaimin was filed on 12-22-2017, granted on 11/23/2019, and assigned to LG Electronics Inc [3]. The patent describes the design of a Guidance Robot device that will provide a route guidance service to a user in a public, private, or commercial place like an airport. The robot may include a driving unit, a body, and an image and voice recognition unit to determine when a person is close to the robot. A reader is provided inside the robot, and an object having a readable code needs to be inserted for the robot to compute the route. The device includes lidar, which is a laser radar used for autonomous driving. Also, the device uses a camera to detect objects and compute the distance between the Guidance Robot and the object. Compared to the Guidance Robot, our project, the Smart Service Bot, will provide a safe route to help a visually impaired person navigate through rooms and public places. Similar to the Guidance robot, we plan on using a combination of Lidar and cameras with AI object recognition to detect and avoid objects. However, our project will have an app that connects to the robot and lets the user send the GPS data needed to compute the route to the chosen destination. Furthermore, using the camera in the robot and image recognition, the user can get a short description of the environment at any time.

Patent CN105796289A: The idea of the Blind Guide Robot is very uplifting and awesome since it is being used to help individuals that are impaired (blind). This robot uses a walking speed sensor to collect data of the impaired user and figures out how fast or slow the user is walking and then matches its speed to the users, so it doesn’t get away from the user [4]. Also, the robot voice recognition, so the blind person can control the device by voice. It also has a camera, infrared, laser sensors, and radar to detect objects and avoid them so the user will also prevent the obstacles. This patent is relative to my group's project idea. Our idea is to recreate a guide dog for the blind using multiple sensors and artificial intelligence to get the user to their destination safely. This blind guide robot idea is similar since it’s a guide for the blind by using AI and speed sensors. The idea of velocity sensors is good for us to also use, since we are

making a platform that will be in front of the person, guiding them. For our design, we haven't discussed how fast the system will go, so using velocity sensors on our project will help our system tell the speed of the user and go that same speed, so it is not guiding the user to fast or slow.

Patent BR102015003790A2: The patent describes a low-cost guide dog for the visually impaired like ours. The aim of the patent is the same as ours. They wish to assist in the navigation of the visually impaired. The patent itself outlines a self-contained design consisting of an ultrasonic sensor array, a voice recognition receiver board, speakers, and a GPS [5]. Movement by the wheels is coordinated by the GPS and sensor array. On the detection of an object, the system will stop and notify the user of an obstacle before proceeding forward. The control for the system is entirely voice-based with commands forward, left, right, and stop. For the movement itself, the robot has one motor for each of its four wheels. Finally, the patent also mentions but does not elaborate on prediction software for traffic lights, signaling busses, and an app. Overall, the concept itself is like ours. Both the patent and our project include the idea for voice control, the use of sensors to control the device and guide dog collar. Some key differences for the project include the use of ultrasonic sensors over LiDAR and how the app and camera are integrated into the project. Specifically, we offload our voice functionality to the app for use with already available smartphone functions. The camera is used for object identification over simple monitoring on the user. Our method of charging the device differs as well. A simple removable or solar powered battery is used in the patent. We plan on creating a docking station to recharge the device. The use of an onboard LCD for status of the device and speakers on the device are ideas we could use to improve our overall design.

System Design

2.1 Project Requirements

This product is meant to help the visually impaired. On App Launch, the user will have an introductory set up with the help of another person. In this meeting, they will be introduced to all the voice commands and Android Speech to Text. The user or another person will provide the User's height for the purpose of Collision Detection and Navigation. After setup, the App will be fully voice controllable. At user discretion, the app can link to either external speakers on the device or Bluetooth hands-free headset for the user feedback information. This feedback information includes when the user attempts to set up an action, on completion of an action, and notification of any caution areas. The device will be able to help users navigate through rooms and avoid obstacles for both the device and the users. This will be done with a combination of LIDAR and cameras with AI object recognition. This sensor data will be used to make the decision to either keep going forward, slow down, turn left, turn right, or slowly back up. Once the user has arrived at the destination, the feedback will alert them. Another feature that will run parallel with the program will be battery monitoring so that the user is aware of the battery life left and range.

Functional Requirements:

A. The Associated app shall provide secure, personalized accounts to each user.
B. The System shall be able to detect and avoid objects in its vicinity.
C. The System shall be able to let the user know when to stop or proceed.
D. The System shall be able to recognize and distinguish differences between objects.
E. The System should be able to recognize when to stop or avoid an object completely.
F. The System shall be able to find an efficient path to a destination.
G. The Associated app shall be able to recognize user commands.

Usability Requirements:

A. The device shall require minimal input from the User to function properly
B. The battery lifetime is at least 2 hours.
C. The device should be light enough for an average adult to carry it if necessary.
D. The System shall function within reasonable terrain and distance.
E. The system should respond to user commands either with an app or remote.

Safety Requirements:

A. The system shall monitor the battery percentage.
B. The system will have good durability.
C. The system shall not accidentally harm the user or others around the device.
D. The system shall not engage any unnecessary actions without user consent.
E. The System shall only communicate with its user through secure means

2.2 Product Design

Functional Requirement Description:

Requirement	Subsystem	Description
Functional A	App System	The app will use a set of IDs and pairing to allow the use of mobile assistance in controlling the associated device. The options include device settings and commands.
Functional B	Sensor Systems	This system will use LiDAR, light detection, and ranging systems. This will be able to give them microprocessor data showing objects within range. The system will be able to tell where the objects are and help with Accident prevention. When assisting the user with the setup, A minimal height will be entered to help the user not run into obstacles.
Functional C	Sensor Systems	The device should be strong enough to exert some kind of tugging or pulling onto the user so that the user knows when to start walking along with the device. If this tug isn't felt, then this will alert the user that he/she should come to a stop and wait until it feels the pulling again.
Functional D	Sensor Systems/AI System	The system will use a combination of camera and AI so that it can tell when a person is walking in front of the device so that it can come to a complete stop and wait for pedestrians to cross before continuing its path. The system will use a combination of LiDAR and Cameras to detect impending objects before attempting to recognize non-impeding ones. If an area is deemed unsafe (Caution signs, Warnings, traffic cones, police tape, etc.), the user will be alerted by the device.
Functional E	Sensor Systems/AI Systems	The device should be smart enough to recognize the size and type of object that it has in its path. Using the information, it should be able to determine whether it should completely stop and wait, change directions completely, or just make a minor weave in order to continue on its path.
Functional F	Sensor Systems/AI Systems	Once it knows where it's going, the device should be able to be smart enough to recognize the shortest path to the desired destination in order to not consume its own battery life. This will also help for the user's own sake; the person using the device will not get tired by walking more than he/she should go to a certain place.
Functional G	App System	In use with Google Assistant API, the app recognizes user commands in setting the device and preferences. The app should be able to distinguish common commands from each other.

Usability Requirement Description:

Requirement	Subsystem	Description
Usability A	Sensor System/App System	Using a balance of user commands and input, the User will require minimal setup and control in order to operate the device. The app will enable Bluetooth connection and pairing automatically at launch. Google Assistant will be activated to allow more advanced controls of the app with google accessibility as a backup.
Usability B	Power System	Achievable through weight and battery supply. The batteries must be able to power the Jetson Nano. This battery system will need a higher current to supply the higher power required by the Jetson Nano. Therefore, a great choice of battery system will be 18650's due to their abilities to be combined in series and parallel in order to achieve the desired voltage and Ah. In order to keep all the batteries charged at the same voltage and prevent current overloads, a BMS (Battery Management System) will be used.
Usability C	All Systems	The expected maximum weight should be at most 20 pounds. An average adult should be able to carry the device if it runs out of power. If the device comes out too heavy, this will hinder the usability of the system. The handle is of negligible weight
Usability D	Power/ AI system	Aware of the battery drain, the device will make the user aware of the maximum distance it can travel without its next recharge.
Usability E	App System	A set of controls is available on a harness attached to the device. The controls can be used to override the navigation stack of the robot.

Safety Requirement Description:

Requirement	Subsystem	Description
Safety A	Power System	The user will be made aware of a low battery around 20% and then again be notified when it's around 10% and needs to be charged.
Safety B	Housing System	In order to keep the machine stable, we plan on making a lower housing battery tray for the batteries at the bottom. This tray will need to be water-resistant to avoid problems. By having the weight at the bottom, this serves a dual purpose by making the system very stable.
Safety C	App System/ Sensor System	The use of detection will result in a safe stoppage of movement, and interaction with the device will be kept to a set of accessible controls. The user will not interact with the carrier itself. Instead, a handle provides physical notification of motion and houses the actual physical controls for the device. Sensors on the device will stop or slow-motion based on a few key classifications from the AI (whether the object has motion and size).
Safety D	App System/ AI system	The device will not function without user commands from the app or controller. Settings that would cause the device to leave the user a disabled by default.
Safety E	App System	Through Bluetooth and App level encryption, Functions, and commands from the user will be made secure.

2.3 Block Diagram

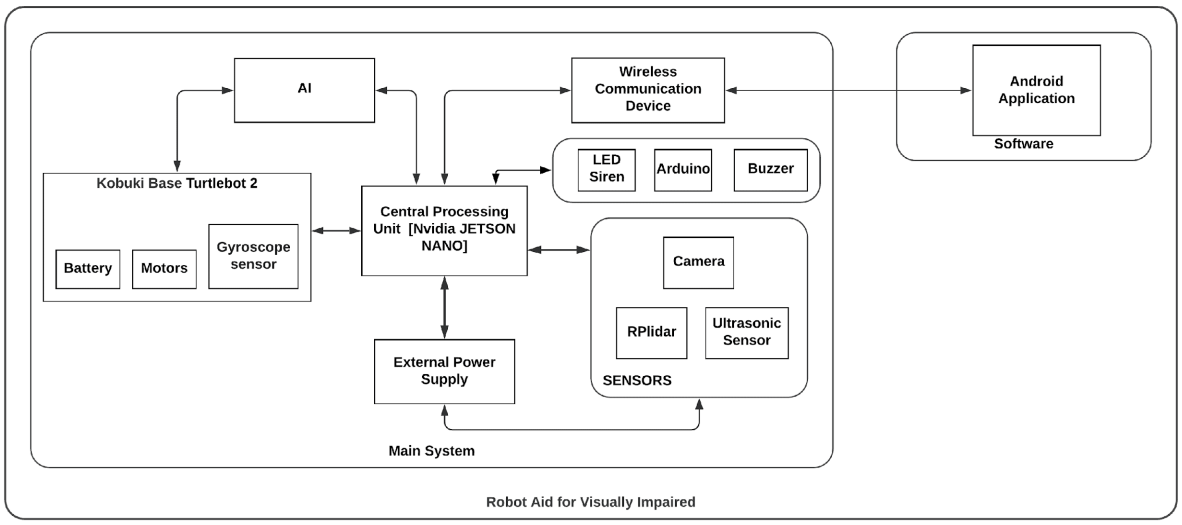

Figure 1: Block Diagram

The block diagram can be broken down into five subsystems. There is the communication system, power system, sensing system, feedback system, and user interface. The CPU has two-way communication with the AI and the sensing system, receiving and sending data and waiting for commands. The Power system is the external power supply, which consists of 12 18650 Lithium ion cells which can output 12V and 10 Ah. This battery pack provides power to each of the subsystems that make up the entire project. The CPU has two-way communication with the wireless communication system (Bluetooth) sending the data to the user interface of the application and waiting for the user to give a command. Once the power gets low, it will send data to the application to let the user know that it needs to be recharged. The CPU then processes all the data coming from the sensors, user interface, and AI to make decisions on whether to move or stop. Another sub system we have is the sensing system. We are using three sensors. First, we have the RPLiDAR which uses laser-based optical ranging, which will tell the bot whether there is an object in its path. The sensor is positioned on the top of the robot, and it provides the robot 360 degrees of object detection. Next, we have the Kinect camera which is used for color and depth. Lastly, we have the ultrasonic sensor which is used as a depth sensor to make sure the bot stops before a certain depth comes upon it and it also sends a signal from the Arduino to the vibrating motors in the handle and siren on the bot the depth is in front of the bot for the user to know.

2.4 State Diagram

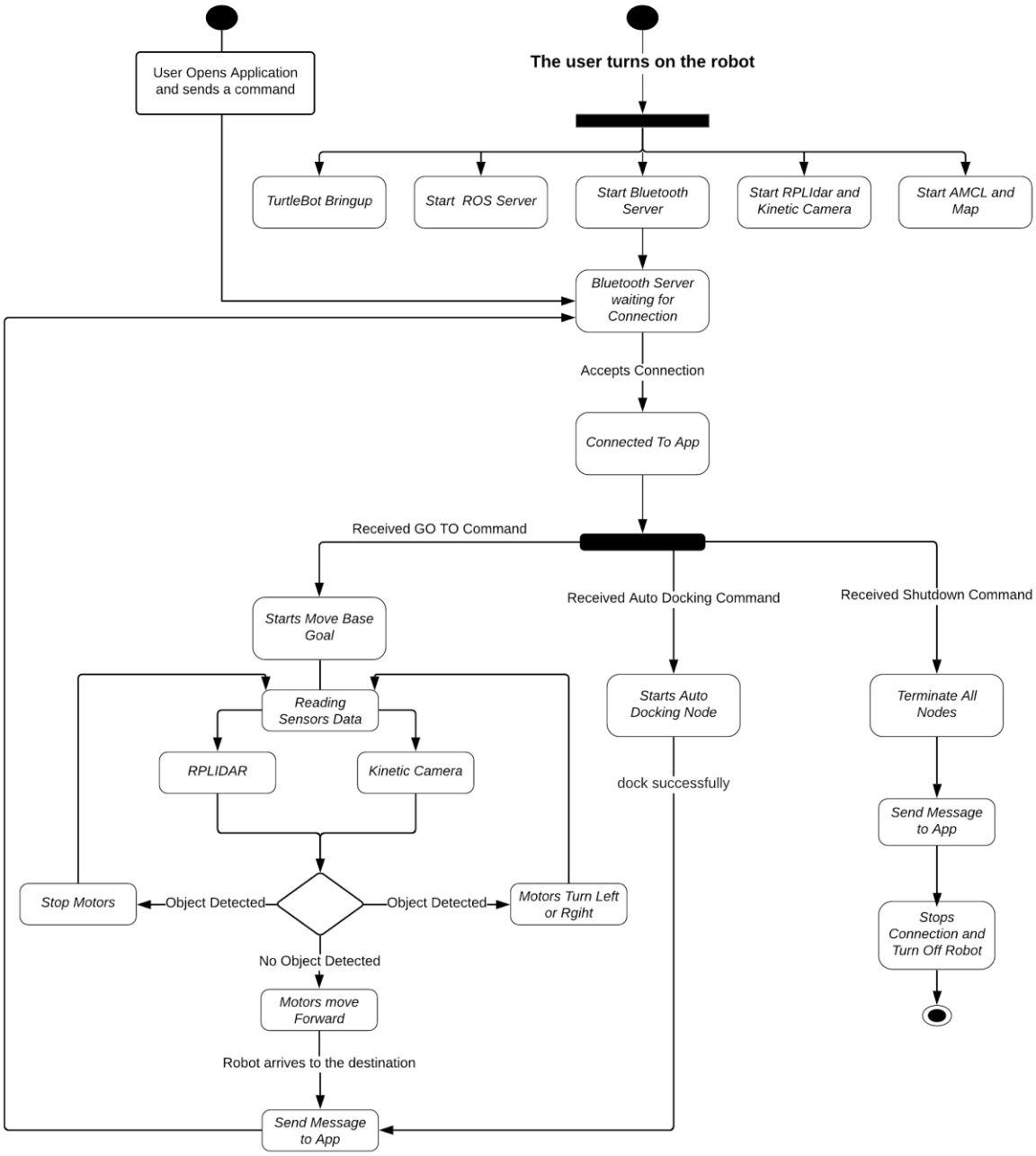

Figure 2: State Diagram

2.5 Use Cases

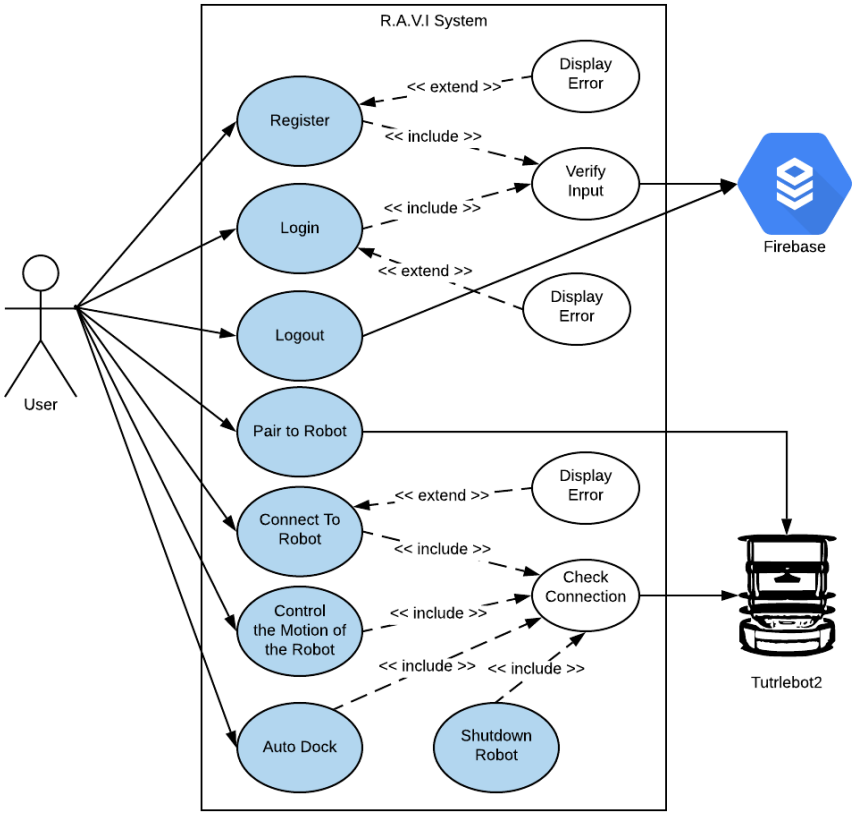

Figure 3: Use Cases

- Register: Registration allows users to create a new account using Firebase as the backend database. The app would display an error message if the email was already used.
- Login: Allows an authorized user to log in to the application successfully.
- Pair To Robot: The users can start scanning for the robot and complete the pairing process.
- Connect To Robot: Allows users to initiate a connection to the robot using its unique Mac address. Displays an error message if the connection failed.
- Control the motion of the Robot: Allows users to choose between predefined destinations in the UI. Users can use voice commands or click a button in the UI.
- Logout: Allows users to sign out.
- Auto Dock: Allows users to send a command to the robot to dock automatically.
- Shutdown: Allows users to turn off the Robot.

Use Case: User Creates Account

1. User Opens the application
2. The application displays the registration form
3. User enters email, name, and password
4. The application reads the content to the user
5. User confirms credentials and clicks Register
6. App create a new account by sending the user's email address and password to Firebase
7. The application redirects the user to the login screen.

Use Case: User Logs In

1. User carries out the Registration
2. The Application displays the Login Form
3. User enters email and password
4. The application verifies the credentials
5. The application brings the user to the Dashboard.

Use Case: Pair to Robot

1. User carries out the Log in
2. User selects the Pair to Robot screen in the bottom navigation menu
3. The application brings the user to the Pairing screen
4. Users select “Start Scanning.”
5. The application shows the Robot’s MAC address if found
6. User selects the Device
7. The application finished the pairing process.

Use Case: Connect To Robot and Control the motion of the Robot

1. User carries out Log in
2. User chooses between the predefined destinations
3. The application initiates a Bluetooth connection
4. The application sends the destination to the server
5. The server executes the Move Base Goal Node
6. RPLidar and Kinetic camera begin the startup
7. Robotics System begins motion
8. Sensor System detects an object in the path
9. Repeat from step 6 until the destination is reached or motion is stopped

System Implementation

3.1 Hardware

Nvidia Jetson Nano (All)

Figure 4: Nvidia Jetson Nano

The Nvidia Jetson Nano was chosen to be the central controller for this project because of its computing power and its ability to process images. The Jetson Nano is a CUDA-capable Single Board Computer (SBC) from Nvidia. The Nano is the latest in Nvidia's line of Jetson embedded computing boards [16], used to provide the brains for robots and other AI-powered devices. It is designed to perform fast, deep learning inferences on a small size board. The Jetson Nano runs the ROS packages and handles the Bluetooth communication between the robot and the app.

RPLIDAR A2 2D 360 Degree (Alejandro Moracen and Joseph Burton)

Figure 5: RPLIDAR A2 [17]

The RPLIDAR A2 uses laser-based optical ranging, which will tell the bot whether there is an object in its path. This sensor emits output as PWM, which will then be analyzed by a microcontroller. The sensor is positioned on the top of the robot, and it provides the robot 360 degrees of object detection. Some of the specification and reasons why we choose this sensor are:

- Range radius: 0.14m - 12m/18m (6 meters on other sources)
- Ultra-thin: 4cm
- Sample rate: 2000-8000 times (measurement frequency) (typical - 4000)
- Scan Rate: 5 – 15Hz (scan frequency) (typical 10Hz)
- Angular resolution: 0.45 – 1.35 (typical 0.9)
- Work of years: 5
- Low noise brushless motor
- Wireless power and optical communication technology.
- Light Source: Low power infrared laser light.
- Laser Safety Standard: Class 1 (safety to human and pets)
- Laser triangulation ranging principle, and high-speed RPVision range engine measures distance data 8000 times per second and has an excellent performance in a long distance.
- It performs a clockwise 360-degree omnidirectional laser range scan.
- ROS packages “rplidar_ros and rplidar_python” are available for RPLIDAR A2.

Kinect Camera (Alejandro Moracen and Joseph Burton)

Figure 6: Kinect Camera

Kinect sensor is the default 3D sensor that comes with the included hardware of the Turtlebot robot. Some of the specifications are:

- Sensor
  - Color and depth-sensing lenses
  - Voice microphone array
  - Tilt motor for sensor adjustment
- Field of View
  - Horizontal field of view: 57 degrees
  - Vertical field of view: 43 degrees
  - Physical tilt range: ± 27 degrees
  - Depth sensor range: 1.2m - 3.5m

Turtlebot2 (Victor Baldoceda)

Figure 7: Turtlebot2

Turtlebot, as described on the official website, is a low-cost, personal robot kit build on top open-source robotic software [18]. The main features and parts are the followings [19]:

- All Turtlebots use ROS - Robotic Operating System
- Maximum translational velocity: 70 cm/s
- Expected Charging Time: 1.5/2.6 hours (small/large battery)
- Docking: within a 2mx5m area in front of the docking station
- PC Connection: USB or via RX/TX pins on the parallel port
- Kobuki Base
- Kinect Mounting Hardware
- TurtleBot Structure
- TurtleBot Module Plate with 1-inch Spacing Hole Pattern.
- Battery: Lithium-Ion, 14.8V, 2200 mAh (4S1P - small), 4400 mAh (4S2P - large)
- Docking IR Receiver left, center, right

Battery (Joshua Sojka and Victor Baldoceda)

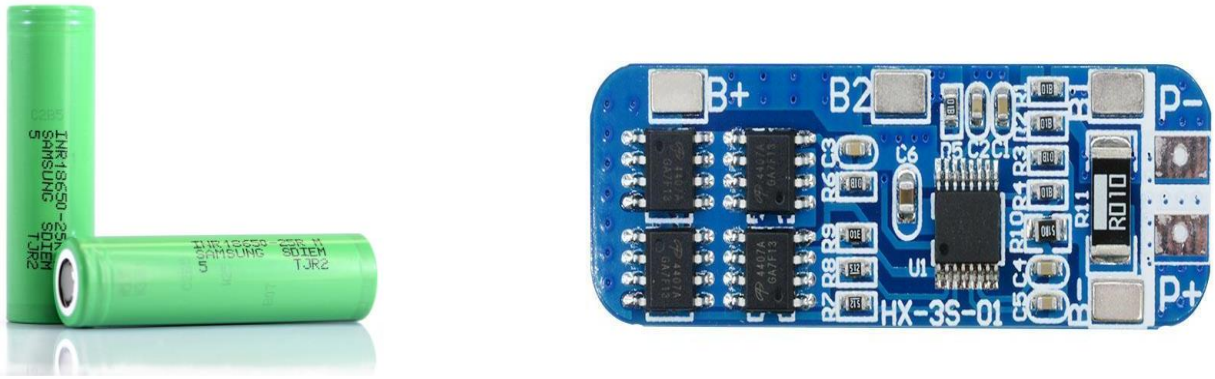

Figure 8: Battery Components

For the power supply of this project, we needed something that could provide high current drainage but was not too heavy that it would affect the movement of the robot. To solve this problem, we decided to use Lithium-ion 18650 cells. These batteries checked both of the requirements that were sought after. We used a 3s4p setup combined with a 12V 10A battery management system. We have 12 Samsung 25R 18650 batteries. They are connected in 3 sections in series, and each section has 4 in parallel. This is all connected to the BMS 3s, producing 12V and max of 10A. The BMS is connected to the KANCRO DC-DC USB Charging Module. This charging module changes the 12V 10A to 5V 5A to power up the Jetson Nano. The Jetson Nano tells the charging module how much amps it needs depending on how much it is processing. The battery will also power up the RPLIDAR, and LED Siren.

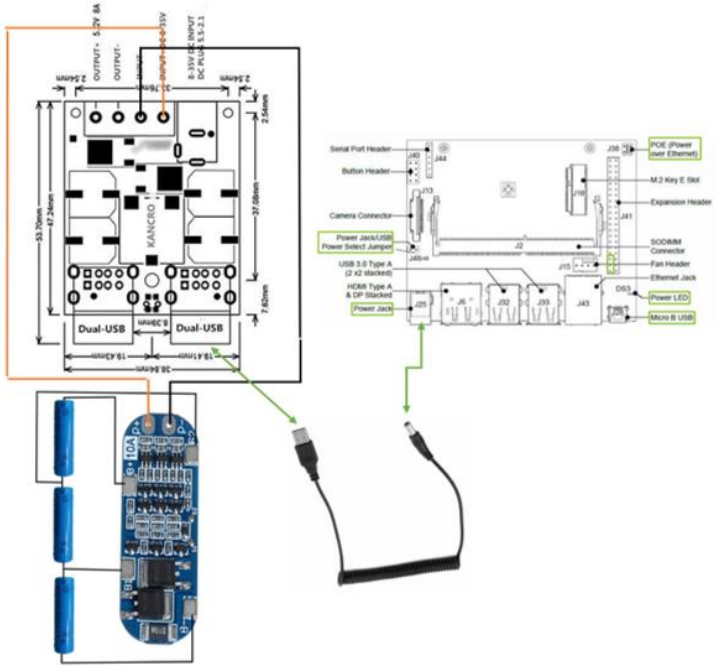

Figure 9: Battery Schematics

Voltage Regulator (Joshua Sojka)

Figure 10: Voltage Regulator

Since our battery provides 12 volts, we needed a way to step that voltage down in order to provide power to most of the devices of the project, such as the Jetson Nano. This voltage regulator drops the voltage from our power supply from 12 volts to 5 volts, and each USB outlet could provide up to 4 amps. This allows us to provide power to multiple devices from our supply safely.

Arduino (Nelson Ordonez)

Figure 11: Arduino

In order to split the work, an Arduino was used to drive the feedback motors and sensors for the project. The Arduino provides an easy to use platform for people with little experience to use. In this case, the Arduino Uno was used to control what we call the trench detection system. The trench detection consists of an ultrasonic sensor, vibration motors, a buzzer, and the Arduino platform.

Ultrasonic Sensor (Nelson Ordonez)

Figure 12: Ultrasonic Sensor

What the Ultrasonic sensor does is continuously monitor the distance from where it is mounted to the floor below it. From the Arduino code, we set the distance to a certain threshold. If the distance from the sensor to the floor falls below the threshold, then a signal will be sent to the motors.

Buzzer and Vibration Motors (Nelson Ordonez and Victor Baldoceda)

Figure 13: Buzzer and Vibration Motors

From the outgoing signal, we control both our buzzer and vibration motors. Since the user is said to be someone who has difficulty seeing, then we need to provide other forms of stimulus. Both the buzzer and vibration motors provide a physical and audible stimulus for the user to know when there is a path that is a potential hazard.

Siren (All)

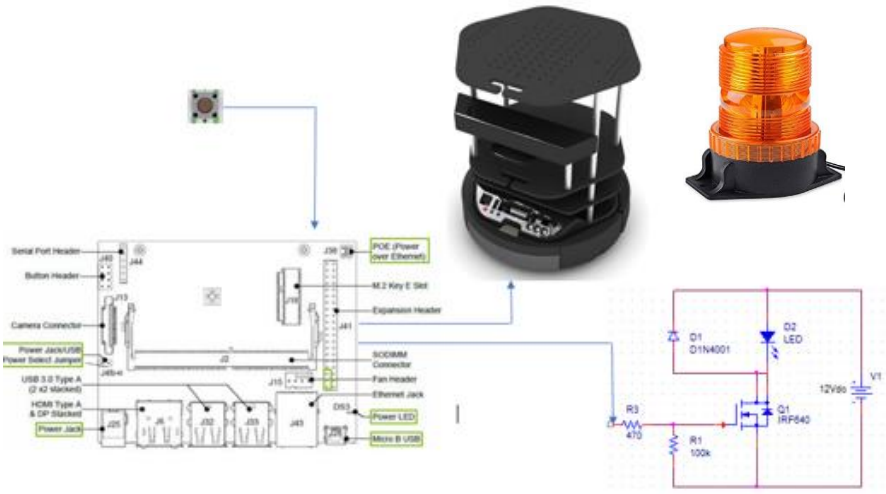

Figure 14: LED Siren Controller

To control the movement and the LED siren installed onto the bot, a button was installed on the handle. When the button is pressed, and a waypoint is selected, the Jetson Nano controls the movement of the robot. The logic behind this movement also controls the LED siren, which is hooked up to a MOSFET switch circuit, which controls the voltage across the LED to be 0V or 12V depending on the input that is received at the gate.

3.2 Software

Android Application (Alejandro Moracen)

The Android Application was designed and developed following the guidelines [6] for implementing key elements of accessibility so that visually impaired users can successfully utilize our app. For example, the UI has widgets that are easier to tap and have a larger touch target size. Also, I included the attribute content Description that describes the element's purpose to the user. Each description is unique; therefore, the user can recognize that the focus is on an element that has already had focus. Navigating the app with a screen reader such as Talkback, allows the user to explore the interface by moving their fingers over the screen to hear the description set by the attribute content Description. This provides a sense of the entire interface, and the user can quickly learn how to navigate the app.

The user interface allows for various application features, including using voice commands to connect and control the robot and receive data from the robot. The user will be able to access their dashboard with their credentials. All the users' accounts are managed using Firebase as the backend service. When new users create an account, this data becomes a node in the existing JSON structure of the Firebase Realtime Database. Once the robot is turned on, the app will initialize a connection using a Bluetooth socket. The users can choose between the predefined destinations in the UI.

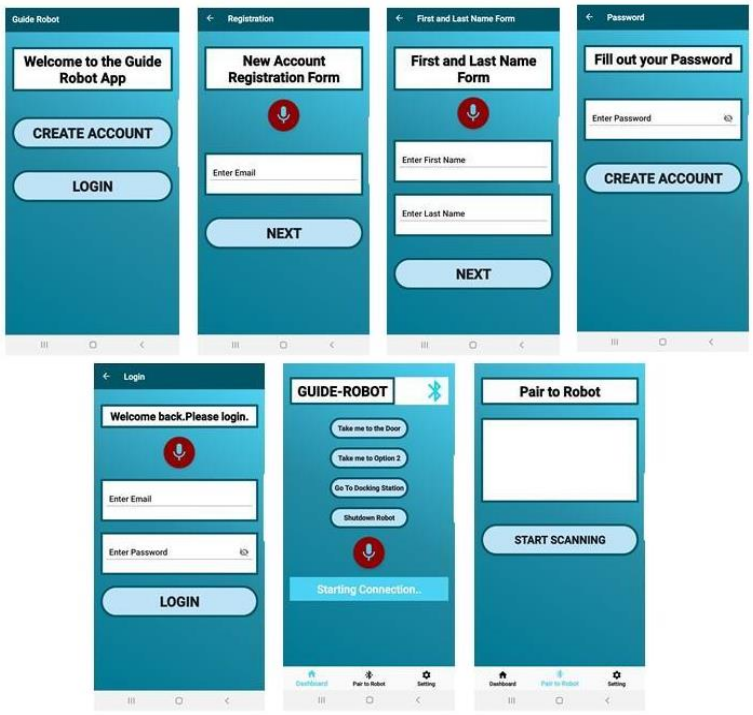

Figure 15: Android Application

ROS (Alejandro Moracen)

In order to control the Turtlebot2 to drive around an indoor location. We chose ROS (Robotic Operating System) because it is compatible with Turtlebot2, is an open-source, meta-operating system [7], there are many resources available that contains all the necessary libraries and tools needed to complete the project, and ROS provides the services required to write code that can control the robot's behavior. We installed ROS in our Jetson Nano board. After installing the latest version of ROS Melodic that runs on Ubuntu 18 Bionic, the operating system used by Nvidia Jetson Nano. We encountered the problem that the Turtlebot packages to support Turtlebot2 have not yet been released into ROS Melodic. For this reason, we had issues connecting the Jetson Nano with the Turtlbot2. However, we were able to build all the packages from sources [8] and added more dependencies if running catkin_make returned any error.

ROS provides basic commands for fast navigation between ROS packages [9]. These shell commands called rosbash contain useful bash command-line tools. For example, roscd, rospd, rosls, rosed, roscp, and rosrun

- rospack: information about packages
- roscd: for moving fast between ROS packages
- rosls: allows to ls directly in a package by name

ROS software is organized in packages [10]. These packages might contain ROS nodes, libraries, datasets, configurations files, etc.

- Nodes use ROS to exchange messages with each other. Nodes can be written in python (rospy) or C++ (roscpp).
- Messages are sent between nodes by publishing them to topics. Messages can be standard primitive types or more complex data types.
- Topics are ROS nodes publish or subscribe to other topics to send or receive the message.
- Roscore is a combination of ROS Master and ROS Parameters needed for ROS nodes communication.

Mapping (Alejandro Moracen and Joseph Burton)

The map was created by implementing Hector SLAM (Simultaneous Localization and Mapping) [11]. SLAM technique provided us the ability to keep track of the current position of the robot and, at the same time, build a map of an unknown environment. The process involves driving the robot around the environment with the RPLIDAR. The advantage of using Hector SLAM is that we can create a map without odometry.

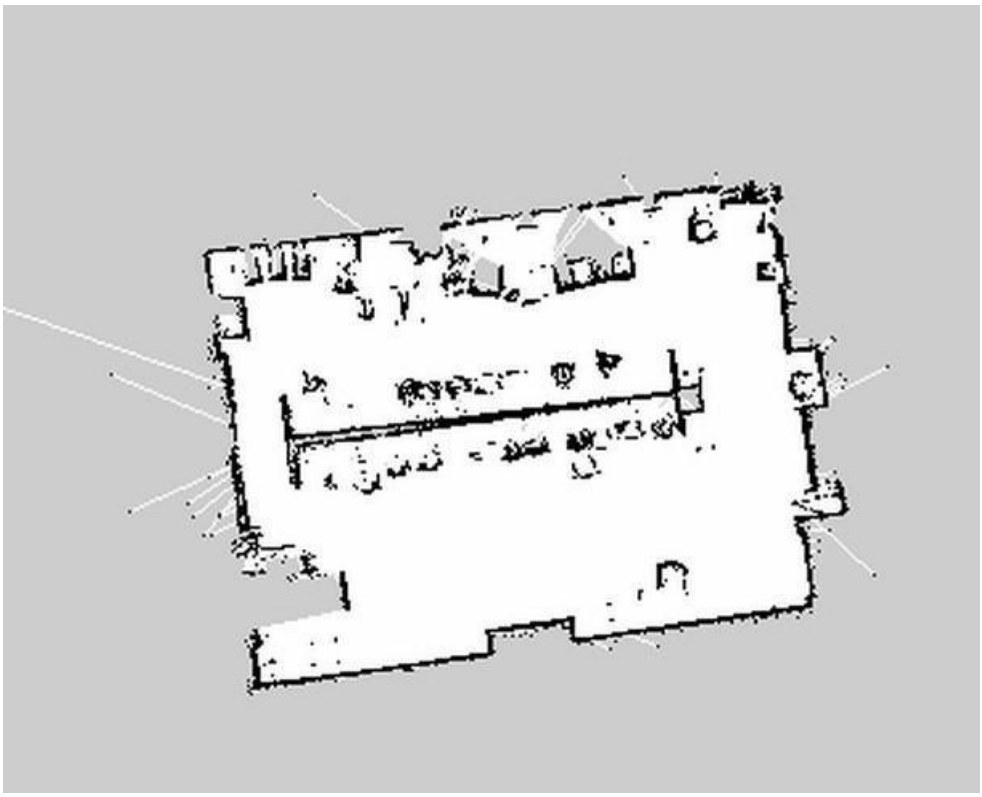

Figure 16: Map of EE96, Room 209

ROS Navigation Stack (Alejandro Moracen and Joseph Burton)

ROS navigation stack [12] provides a set of algorithms that take information from sensors, odometry, and a goal position to produce velocity commands to move the Turtlebot2. First, the navigation stack needs a map to perform navigation and path planning. Then, it can perform Localization by computing the relative position of the robot on the map. AMCL package was used to perform the localization [13]. AMCL is a probabilistic localization system that is responsible for the robot movement in 2D. The Adaptive Monte Carlo Localization approach helps the robot to detect its position in a known map. Path planning takes as input the current location of the robot and the position where the robot wants to go.

The move_base package was used to send goals to the Navigation Stack. This was done by specifying the final position and orientation, where the robot should go. The move_base packages link both the local planner and the global planner to complete the navigation.

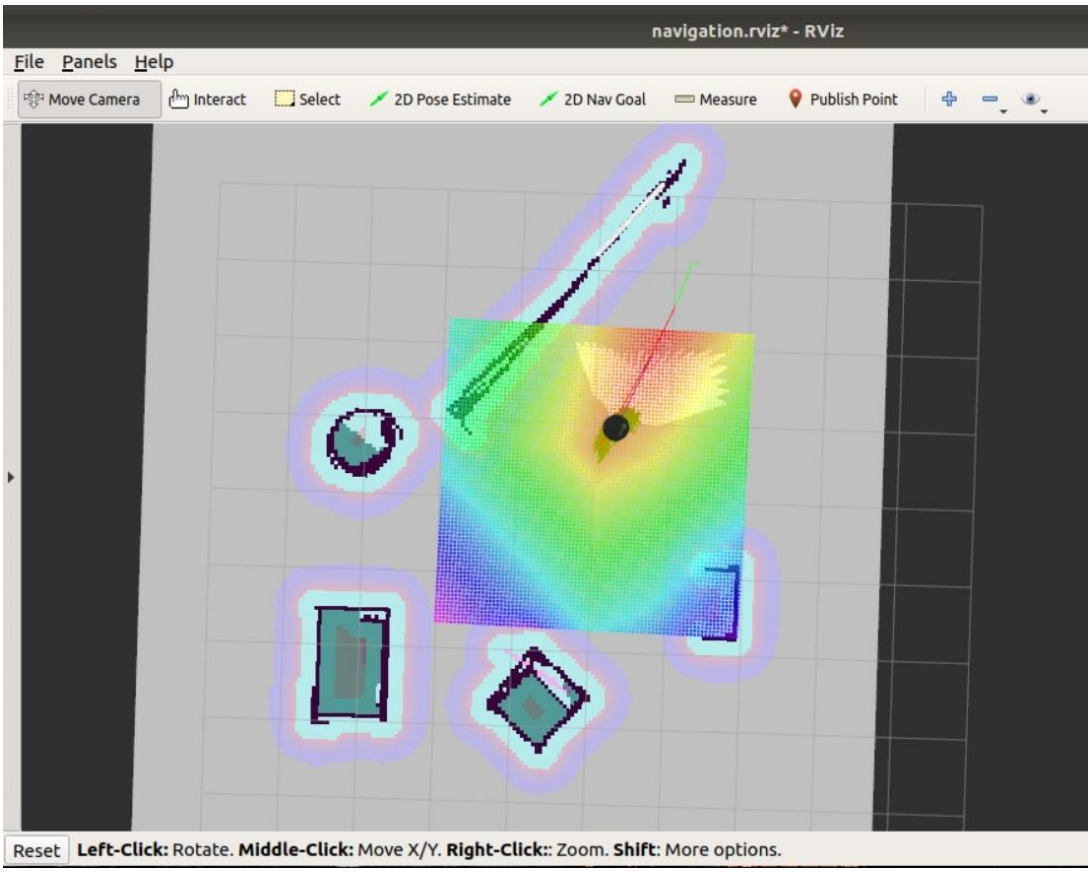

Figure 17: Navigation in RVIZ

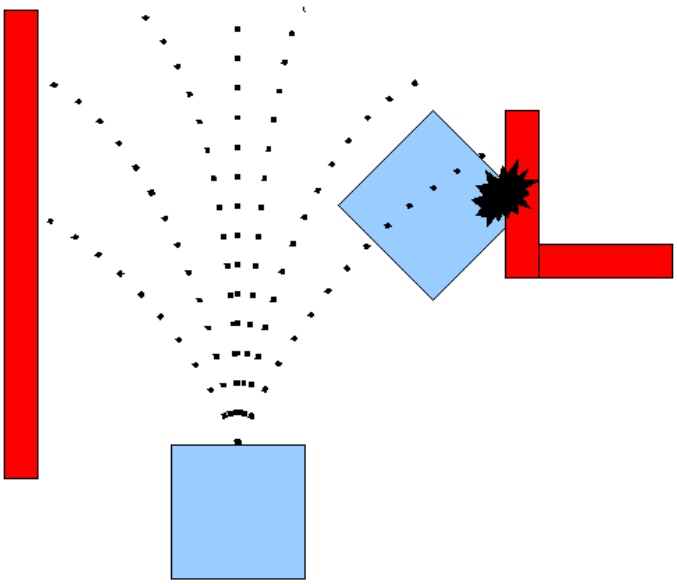

Figure 18: Trajectory planning Local Planner [14]

ROS Automatic Docking (Alejandro Moracen)

Automatic docking allows users to send a command from the app to the Jetson Nano; then, the Jetson Nano executes the code that starts the algorithm. To complete automatic docking, the docking station needs to be placed in an open space of at least 2 meters wide and 5 meters long and held by a wall to prevent the robot from pushing it. Kobuki's docking station has 3 IR emitters. The emitted IR lights cover three regions in front of the docking station. When the robot is placed within the field of the docking station, if the robot is placed on the central region, the robot can dock by following the center's region signal. If the robot is placed in the left region, it will turn counter clockwise until detecting the left region signal on his right sensor. At this point, the robot is facing the central region perpendicularly, so it just needs to move forward to reach the central region. Now the robot should see the central region's signal, so reaching the docking station gets trivial. Starting in the right region is similar, but directions are inverted.

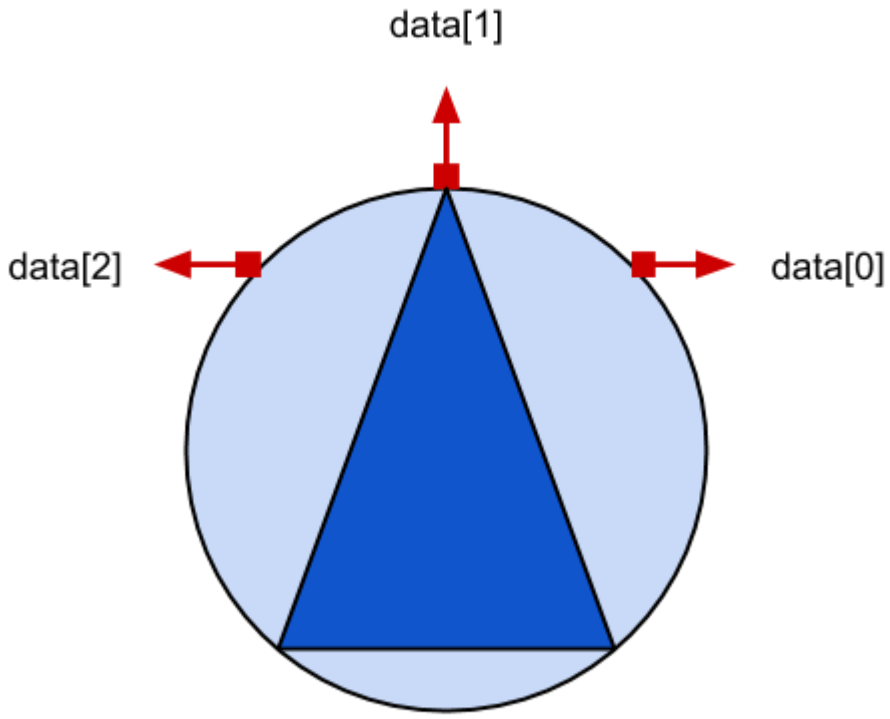

Figure 19: Robot Docking Data [15]

Bluetooth Server (Alejandro Moracen)

PyBluez was used to create the Bluetooth server and allow communication between the Jetson Nano and the App. PyBluez is a Python Bluetooth library for Windows and GNU/Linux operating systems. An RFCOMM Bluetooth Socket was used to accept incoming connections by attaching to the operating system resources with the bind method. Bind takes in a tuple specifying the address of the local Bluetooth adapter to use and a port number to listen on. Usually, there is only one local Bluetooth adapter. Once a socket is bound, a call to listen puts the socket into listening mode, and it is then ready to accept incoming connections.

Arduino (Nelson Ordonez)

A standard Arduino uno was used in order to control the feedback subsystem. This was done due to the relative ease in writing Arduino code and the capability Arduino has with running DC motors. Arduino also comes with a free development environment, where code can be compiled and downloaded. The Arduino was used to read input from an HCSR04 ultrasonic sensor and computed the distance between the sensor and the ground in front of the robot. Once the distance exceeds the established distance in the code, the Arduino will drive an alarming buzzer and vibration motors in the robot handle.

AI (Joseph Burton)

An image Listener node (named imageListener.py) was made for the purpose of listening to the node ros_deep_learning to detect objects in frame. The publisher listens to the output provided vs the topic detectnet/detections Initially it was found the Kinect on the Turtlebot was outputting a different signal than the initial ROS package required. This was indeed fixed. however, further testing did not produce any response from the node even though a subject (in this case a chair) was provided for viewing. Possible fixes would have included further testing with the connect and different objects as well as different cameras for the device. A launch file was created for the purpose of running the image detection node (Detectnet). For the autonomous navigation, two launch files were created depending on the use. Hectorbot is used for basic mapping functions while Hectorbotnav was used for the running of various different packages associated with the Turtlebot. The base AI navigation was implemented already through the combination of Hector SLAM for mapping and AMCL with the Turtlebot specific move base node for movement.

Testing and Performance Evaluation

Battery

The first battery was built using 3 18650's that were provided by the school. These batteries claimed to be 4500mAh. After completion and testing of these batteries they gave the correct output voltage that was being controlled by the (BMS) battery management system. However the capacity of the batteries was nowhere near the capacity stated by the batteries. Upton testing the robot died in under an hour. This problem led us to use other batteries, instead of using the chinese no brand batteries we opted to using Samsung 25R 16850's, with a capacity of 2500mAh each. We connected 4 rows in parallel with 3 in series each. This brought our total rating to 10000mAh. The output of the battery made was 12V and governed at 10A by the BMS. The testing of the second battery pack was successful; we were able to run the robot for a total of 10 hours straight without recharging it. Further modifications to the battery were also added.

Voltage Regulator

This was an expansion added to the battery pack that allowed 4 outputs of 5V 5A. This was needed because the Jetson nano has the ability to be plugged into a 4A source and run at its full potential. Using this setup we had to test that the specs given by the data sheet were true. This was done in the lab by connecting USB cable with exposed positive and negative cables. These terminals were connected to the Ammeter in Series to test the current and then we had a series of small power resistors. These power resistors were not made to handle the 25 Watts being produced by the 5V 5A, so they got hot rather quickly. We took off some resistors in the series and the current started to go up. We successfully got it to discharge at an amperage over 4A, so we stopped testing there since we only need 4A for the Jetson Nano.

Trench Detector

When the trench detector was first built, we tested it by just putting something in its way to make sure the buzzer did not sound, when the box moved out of the way the buzzer would sound. This first test was successful. Next, we had to mount it on the robot. We used a piece of metal and mounted the ultrasonic sensor on it. After our first attempt we realized that if the robot hit something the ultrasonic sensor would take the hit first and get damaged. We then repositioned it under the metal bracket so it would be protected. With it mounted we could test the trench detector function. We first tested it on the table, and it needed some small tweaking of the height in the code, but it worked. Final testing was done using some stairs. The robot was told to advance forward using the code and the interrupt from the trench detector was successful the robot stopped right away. Once this was achieved, we tested the vibration motors and the siren they both worked so then we implemented them. The siren was mounted near the Arduino and the ultrasonic sensor, the vibration motors were mounted all the way at the handle. The vibration motors were first soldered to a long wire to allow us to run it to the handle. Once we tested the vibration motors at the end of the wire, they were good to go, so we put them inside the handle and sealed them in there with some hot glue. The final result of the trench detector was successful and achieved everything it needed to do.

Mapping

To create a map of a room we run the following ROS packages in multiple terminals.

- roslaunch turtlebot_bringup minimal.launch
- roslaunch turtlebot_bringup 3dsensor.launch → **(kinect)**
- roslaunch rplidar_ros rplidar.launch → **(rplidar)**
- roslaunch turtlebot_navigation gmapping_demo.launch
- roslaunch turtlebot_rviz_launchers view_navigation.launch
- roslaunch turtlebot_teleop keyboard_teleop.launch
- rosrun map_server map_saver -f my_map_warehouse (Save Map)

Performance for Mapping with the robot was slow. We did not need to calibrate the Node. However, at times either the robot would suffer drift, or the nano would be overwhelmed. Both problems resulted in the map being partially copied in its space and failing the mapping attempt. The successful maps were chosen for their lack of noise in free (or accessible space)

Navigation

After creating the map we tested navigation using the following ROS packages.

- roslaunch turtlebot_bringup minimal.launch
- roslaunch rplidar_ros rplidar.launch
- roslaunch turtlebot_navigation amcl_demo.launch map_file:=/home/userName/Maps/my_map.yaml
- roslaunch turtlebot_rviz_launchers view_navigation.launch

RVIZ was used to send navigation goals and test that the navigation was working.

- 2D Pose Estimate was used for estimating the physical robot's position on the map
- 2D Nav Goal was used for setting a goal to a robot, so the robot will navigate towards the specified position.
- Publish Point was used for finding an exact position (x,y) and orientation on the map.

We used measurements of the robot and repeated tests of a set series of actions representing generalized use in order to calibrate the Navigation stack. A python script was created for random movement around a map to test navigation, but we found the former of the two more useful rather quickly.

AMCL

To test AMCL. We ran the Turtlebot to known or distinct point on the map via the commands

- roslaunch turtlebot_bringup minimal.launch
- roslaunch turtlebot_bringup 3dsensor.launch
- roslaunch rplidar_ros rplidar.launch
- roslaunch turtlebot_navigation amcl_demo.launch map_file:=/home/userName/Maps/my_map.yaml
- roslaunch turtlebot_rviz_launchers view_navigation.launch
- Roslaunch turtlebot_teleop keyboard_teleop.launch
- Told the robot via RVIZ an estimation of its point on the map.
- Then either rotated the robot in place or moved it a couple of inches.
- Calibration was done via a rqt gui which updated the navigation parameters temporarily as the robot was tested.
- Occasionally parameters we retested and refreshed
- A good set of parameters is one that creates a small area of generally agreeing points and generalizes well. In our case, the radius of the localization converged to close to the width of our robot in motion and was fairly accurate standing still.

Budget

Name	Price ($)	Quantity	Total ($)
RPLIDAR A2 2D 360 Degree	312.99	1	312.99
IMX219 Camera Module	30	1	30.00
Power Supply for Jetson Nano	9	1	9.00
Intel Dual Band Wireless	21	1	21.00
Samsung 25R 18650 batteries	3.69	12	44.29
NVIDIA Jetson Nano	99.00	1	99.00
Samsung 128GB Memory Card	19.49	1	19.49
LED Siren	5.99	1	5.99
Arduino Uno	20.90	1	20.90
Ultrasonic Sensor	5.0	1	5.0
Voltage Regulator	12.60	1	12.6
Turtlebot2	0	1	0
Miscellaneous	--	-	50.0
		Total:	$630.26

References

[1] Intelligent Guide Dog, by 姜丽红. (2014, Dec,11). CN104644400A. Accessed on Dec. 8,2019. [Online]. Available https://patents.google.com/patent/CN104644400A/en?oq=CN201410756510.

[2] Guide robot device and guide system, by Ichinose Ryoko and Junichi Tamamoto. (2007, Dec,17). US8195353B2 Accessed on Dec. 8,2019. [Online]. Available https://patents.google.com/patent/US8195353B2/en?oq=US8195353B2

[3] Guidance robot, by Yang Suonho and Choi Heaimin. (2017, Dec,22). US20180178390A1. Accessed on Dec. 8,2019. [Online]. Available https://patents.google.com/patent/US20180178390A1/en?oq=US20180178390A1

[4] Blind Guide Dog, by 于占泉. (2016, June,3). CN105796289A. Accessed on Dec. 8,2019. [Online]. Available https://patents.google.com/patent/CN105796289A/en?oq=CN105796289A

[5] Dog Guide Robot for Aid the Locomotion of Persons with Visual Deficiency or Reduced Mobility, by De Araujo Sellin Neidinalva. (2015, Feb,23). BR102015003790A2. Accessed on Dec. 8,2019. [Online]. Available https://patents.google.com/patent/BR102015003790A2/en?oq=BR102015003790A2

[6] Android Developers. (2019). *Make apps more accessible | Android Developers*. [online] Available at: https://developer.android.com/guide/topics/ui/accessibility/apps.html [Accessed 27 Jan. 2020].

[7] Dattalo, A. (2018). *ROS/Introduction - ROS Wiki*. [online] Wiki.ros.org. Available at: http://wiki.ros.org/ROS/Introduction [Accessed 5 Feb. 2020].

[8] Huang, G. (2019). *gaunthan/Turtlebot2-On-Melodic*. [online] GitHub. Available at: https://github.com/gaunthan/Turtlebot2-On-Melodic [Accessed 10 Feb. 2020].

[9] "Navigating the ROS Filesystem," *ros.org*. [Online]. Available: http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem. [Accessed: 05-Feb-2020].

[10] "Packages," *ros.org*. [Online]. Available: http://wiki.ros.org/Packages. [Accessed: 05-Feb-2020].

[11] S. Kohlbrecher, "Wiki," *ros.org*, 31-Aug-2012. [Online]. Available: http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot [Accessed: 24-Feb-2020].

[12] "Navigation," *ros.org*. [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 05-Mar-2020].

[13] "AMCL," *ros.org*. [Online]. Available: http://wiki.ros.org/amcl. [Accessed: 20-Mar-2020].

[14] “Base Local Planner,” *ros.org*. [Online]. Available: http://wiki.ros.org/base_local_planner [Accessed: 06-Apr-2020].

[15] “Automatic Docking,” *ros.org*. [Online]. Available: http://wiki.ros.org/kobuki/Tutorials/Automatic%20Docking [Accessed: 10-Apr-2020].

[16] Jetson Nano Developer Kit [Online] Available:

https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[17] http://www.slamtec.com/en/Lidar/A2

[18] Bogdon, C. (2020). Robots/TurtleBot - ROS Wiki. [online] Wiki.ros.org. Available at: http://wiki.ros.org/Robots/TurtleBot [Accessed 5 Feb. 2020].

[19] http://kobuki.yujinrobot.com/about2/