

The University of Akron

IdeaExchange@UAKron

Williams Honors College, Honors Research
Projects

The Dr. Gary B. and Pamela S. Williams Honors
College

Spring 2023

BotSitter

Maria Hatzis
mjh210@uakron.edu

Gregory Blondheim Jr.
gdb22@uakron.edu

Marian Bonto
mb321@uakron.edu

Brett Jacobsen
bj57@uakron.edu

Follow this and additional works at: https://ideaexchange.uakron.edu/honors_research_projects



Part of the [Electrical and Electronics Commons](#), and the [Signal Processing Commons](#)

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

Recommended Citation

Hatzis, Maria; Blondheim, Gregory Jr.; Bonto, Marian; and Jacobsen, Brett, "BotSitter" (2023).
Williams Honors College, Honors Research Projects. 1666.

https://ideaexchange.uakron.edu/honors_research_projects/1666

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAKron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAKron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

BotSitter

Gregory Blondheim Jr., Marian Bonto, Maria Hatzis, Brett
Jacobsen

Department of Electrical and Computer Engineering

Honors Research Project

Submitted to

The Williams Honors College
The University of Akron

Approved:

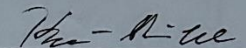
Accepted:



Honors Project Sponsor (signed)

Gregory A. Lewis 04/25/23

Honors Project Sponsor (printed) Date



Honors Faculty Advisor (signed)

Kye-shin Lee 04/24/23

Honors Faculty Advisor (printed) Date



Honors Project Reader (signed)

HAMID BAHRANI 04/24/23

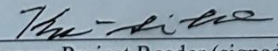
Honors Project Reader (printed) Date



Department Chair (signed)

J.A. De Abreu-Garcia 25 April 2023

Department Chair (printed) Date



Honors Project Reader (signed)

Kye-shin Lee 04/24/23

Honors Project Reader (printed) Date

As an Electrical Engineer, Maria was mainly responsible for assisting in the hardware design of the BotSitter project. She assisted in making the block diagrams of the hardware system. Within the hardware subsystem, she assisted in selecting the ultrasonic sensors used to detect objects in the vicinity of the robot. Using the Explorer 16-32 board and the PIC24FJ128GA010 processor, she provided the correct power and code used to generate the distance measurements from four ultrasonic sensors. She also helped Marian in general tasks needed to implement the power system and motor drivers. She also assisted in the general hardware packaging in the BotSitter prototype by braiding wires, heat shrinking said wires, and attaching hardware components to the robot chassis.

Marian was in charge of the power supply, the hardware portion of the motor subsystem, and all the hardware interconnections in the assembly of the BotSitter. She designed and built the entirety of the charging circuit, power connections to the devices, and the interconnections between the motors and motor controller. She also soldered and assembled majority of the devices and interconnections together in the current BotSitter prototype.

Greg is the project leader in charge of team management. He worked on motor controller design and calculations. He also worked on controlling the budget, parts list, and parts requests. Greg also supported all hardware and software debugging in finalization of the project. He also worked on the Bluetooth location code and hardware module design. With documentation Greg worked on drawing of schematics in Eagle CAD and research of Bluetooth communications.

Senior Design Project Bot Sitter

Final Design Report

Design Team 03

Gregory Blondheim Jr.

Marian Bonto

Maria Hatzis

Brett Jacobsen

Dr. Hamid Bahrami

24 Apr 2023

Contents

1	Problem Statement	6
1.1	Need GB, MH, BJ	6
1.2	Objective GB, MH, BJ	7
1.3	Background GB, MB, MH, BJ	7
1.4	Marketing Requirements GB, MH, BJ, MB	11
2	Engineering Analysis	13
2.1	Circuits – MB, GB	13
2.1.1	Power Sources MB	13
2.1.2	Charging Methods MB	14
2.2	Electronics – MH, MB	16
2.2.1	Audio Signal Enable MH	16
2.2.2	Overvoltage and Overcurrent Protection MB	17
2.3	Signal Processing – MH	18
2.3.1	Sensor Selection MH, MB	19
2.4	Communications – BJ	20
2.5	Electromechanics – GB, MB	21
2.5.1	Motor Controller Requirements GB	21
2.5.2	Motor Calculations GB	22
2.5.3	DC Brushless vs. DC Brushed GB	23

2.6	Computer Networks – GB.....	23
2.7	Embedded Systems – BJ, MH.....	24
2.7.1	Ultrasonic Sensor MH.....	24
2.7.2	Trilateration Calculations.....	26
2.7.3	Motor Control:	29
3	Engineering Requirements Specification – GB, MB, MH, BJ	32
4	Engineering Standards Specification	34
4.1	Safety – MB	34
4.2	Communication – GB.....	34
4.3	Design Methods – MH.....	35
4.4	Programming Languages - BJ	35
5	Accepted Technical Design	36
5.1	Hardware Design – MH, MB, BJ.....	36
5.1.1	Circuit Schematics GB, MB	40
5.2	Software Design – BJ, GB	47
6	Mechanical Sketch – MB.....	57
7	Team Information – GB, MB, MH, BJ.....	61
8	Parts List - GB	62
9	Project Schedule – GB, MB, MH, BJ.....	65
10	Conclusions and Recommendations GB BJ	72

11	References – GB, MB, MH, BJ	74
----	-----------------------------------	----

List of Figures – GB, MH

Figure 1 - Pinout Diagram from PIC24FJ128GA010 Datasheet Where Gray Shading Indicates 5.5V.....	17
Figure 2 - Timing Diagram Taken from HC-SR04 datasheet.....	20
Figure 3 - Embedded Code to Initialize Onboard LCD Screen for HC-SR04 Sensor Data	25
Figure 4 - Embedded Code to Generate Trigger Signal for HC-SR04 Sensor and Read Echo Pin to Calculate Distance	26
Figure 5 - C code to calculate the coordinates of a moving Bluetooth module within the system	28
Figure 6 - C code to control motor movement, including initialization and forward and backward movement.....	30
Figure 7- MPLAB functions for Turning.....	31
Figure 8-Bluetooth Protocol Stack.....	34
Figure 9-Level 0 Hardware Block Diagram	36
Figure 10-Level 1 Hardware Block Diagram	37
Figure 11-Level 2 Hardware Block Diagram	38
Figure 12 - Level 3 Hardware Block Diagram	39
Figure 13 - Ping Sensor Schematic.....	40
Figure 14 - Buzzer Schematic.....	41
Figure 15 - Motor Controller Schematic.....	41
Figure 16 - Bluetooth Corner Module 1 Schematic.....	42
Figure 17 - Bluetooth Corner Module 2 Schematic.....	43

Figure 18 - Bluetooth Corner Module 3 Schematic.....	44
Figure 19 - Bluetooth Corner Module 4 Schematic.....	45
Figure 20 - Bluetooth Robot Schematic.....	45
Figure 21 - Bluetooth Target Schematic.....	46
Figure 22 - Explore 16/32 Schematic.....	46
Figure 23 - Charging Circuit Schematic.....	47
Figure 24-Level 0 Software Block Diagram.....	48
Figure 25-Level 1 Software Block Diagram.....	48
Figure 26-Level 2 Software Block Diagram.....	50
Figure 27-Level 1 Navigation Flowchart for seeking and avoiding obstacles, contained in Movement Decision block.....	53
Figure 28-Level 2 Navigation Flowchart.....	54
Figure 29-Level 3 Navigation Flowchart.....	55
Figure 30 - Sequence Diagram representing software design and flow of control.....	56
Figure 31-Level 1 Sketch of Environment.....	57
Figure 32 - Trilateration of Room using bottom Anchor Points.....	58
Figure 33 - Iso View of Robot.....	59
Figure 34 - Side View of Robot.....	59
Figure 35 - Top View of Robot.....	60
Figure 36 - Original Design Parts List.....	62
Figure 37 - Final Design Parts List.....	62
Figure 38 - Original Design Budget List.....	63
Figure 39 - Final Design Budget List.....	64

Figure 40 - Midterm Project Schedule.....	65
Figure 41 - Final Project Schedule Part 1	66
Figure 42 - Final Project Schedule Part 2	67
Figure 43 - Complete Charging Circuit	68
Figure 44- Internal Hardware Integration Inside Robot	69
Figure 45 - Front View of Robot with Front, Left, and Right Object Detection Sensors.....	70
Figure 46 - Rear View of Robot Including Motor System and Rear Directional Sensor	71

List of Tables – GB

Table 1-Marketing Requirements	11
Table 2-Battery Options.....	13
Table 3-Engineering Requirements	32
Table 4-Level 0 Functional Requirements Table.....	36
Table 5-Level 1 Functional Requirement Table	37
Table 6-Level 2 Functional Requirements Table.....	38
Table 7 - Level 0 Functional Requirements.....	48
Table 8 - Locate Target and Surroundings	49
Table 9 - Determine Movement.....	49
Table 10-Read Beacons	49
Table 11-Locating Target and Surroundings	50
Table 12-Read RSSI from Modules.....	50
Table 13-Calculate position of other device	51
Table 14-Determine Movement.....	51

Table 15-Signal Motors	51
------------------------------	----

Abstract MH, GB

As society progresses into an era where both parents work, whether it is online or in person, children in the home may be put in dangerous situations if they are not given the attention they need. The BotSitter is an automated system that follows the child around and makes an audio alarm to alert both the child and the nearby guardian. Using RSSI trilateration, predetermined danger areas, and embedded controls, the BotSitter will be able to follow the child. The device can manage to keep track of the child for the guardian while being almost completely automated outside of setup.

Key features of the product include:

- Charging compatible with standard U.S. wall outlets
- Autonomous Navigation
- Audio Warnings
- Frequent Status Updates

1 Problem Statement

1.1 Need GB, MH, BJ

Unattended children in the household can cause many unnecessary accidents. From falling appliances, to sharp objects, these items can cause harm when children are unsupervised. Guardians may not be able to keep track of their children at all times, especially in single parent homes, which is why there needs to be a way to prevent these incidents or alert the guardians of imminent danger while they are busy. Currently, there are some devices in place like child proof locks, but these safety measures can be improved upon to help further ensure child safety in the home.

1.2 Objective GB, MH, BJ

The objective of this project is to design and prototype a robot that will detect when a child gets too close to a hazardous object or a dangerous situation. The child will wear some device that will allow the robot to follow the child around. The robot will have a series of sensors that will relay a warning via an audio alarm to notify the child and their guardian of the imminent danger.

1.3 Background GB, MB, MH, BJ

The concept is to build an autonomous robot to help monitor your children and household while the user is busy. The robot will use sensors like a Roomba for navigation but will add features such as odor detection and use of verbal warnings for the children. The child will wear a device that connects to the robot so the robot can track the child more directly.

The basic idea behind the project is to build a small robot to assist parents with keeping their child safe in the home. Typically, adults are doing chores or even working at home nowadays, so they are not able to be fully attentive to their small children. This is very relevant for homes where there is a single parent or where both parents are working individuals. This small robot would have a network of sensors that could notify the parents via an audio alarm when it detects an issue. The robot would also give a verbal warning to the child to try to prevent them from getting into anything dangerous before the parent arrives. Some examples of these sensors could be when a child has left the confines of the home or even something more complicated as when they need their diaper changed. Some other functions of the robot would include detecting if the child has left their bed in the middle of the night or if they are approaching a dangerous area of the house that they are not supposed to be in, such as the kitchen or an area with heavy objects such as an at-home gym. The robot would essentially follow the child around at a safe distance

and keep an “eye” on them using a wearable device on the child. The robot would also have some sensors on it to ensure the child could not damage the device.

To implement the verbal warnings, a speaker could be added to the robot. This speaker would have to be an appropriate size depending on the overall dimensions of the robot. When writing the software programs that control the robot, certain values read from the sensor would trigger an audio file to be played. For example, if a child is approaching a dangerous area, the software could use the tracked location of the child and warn them to stop moving or turn around. There also could be capacitive sensors on the robot to tell the child, or whoever touches the robot, to be careful with it or to put it down. The basic coding architecture of verbal warnings could be modeled after an arduino project that made a “talking robot” that exists on Arduino’s project hub forum [12].

An issue with this design would be keeping the size of the robot at a manageable size to be able to maneuver around a home with ease. In the journal titled “Intelligence for Miniature Robots”, Flynn, Brooks, Wells, and Barrett describe how they build a robot that is about one square inch in size. They also detail the mechanical structure of how the robot moves: “One rear wheel is driven directly by the rear axle, while the other slips through a unidirectional clutch. The front is supported by a simple castor”. Along with making sure this tiny robot could move, they also built the robot with off-the-shelf products [3].

As mentioned previously, the robot will use sensors for navigation and other features. For navigation, the use of “exteroceptive sensors [that] monitor the robot’s environment” [7] are important for the robot’s autonomous feature. Regtien mentioned some sensors like scalar sensors and image sensors that would be useful for the robot’s autonomous feature. Scalar sensors, which are used for acquiring “metric information with respect to quantities related to

distance” [7], is a necessary addition to allow for the robot to track the child and navigate its environment. A capacitive proximity sensor (CPS) is a certain type of scalar sensor that would most likely be used for the robot as it has desirable properties such as “low cost, low energy consumption, and flexible and variable structure design” [9]. A CPS is capable of many applications of displacement sensing using electric fields such as being able to detect objects at a large distance as one of the main features of the autonomous robot is to track the child [9]. It is also mentioned that CPSs are used in building “smart environments” with embedded smart devices wherein physical state and activities can be perceived [9]. For an autonomous robot tracking a child and perceiving its environment, CPSs are the preferred scalar sensors. Other than scalar sensors, image sensors would also be useful to make the robot’s functionality run effectively. Image sensors acquire “information related to structures and shapes” [7] which will be useful in making the robot recognize obstacles on its path. Other sensors would also most likely be used in the robot when applicable in the design process.

The main basis of the control system portion of the robot are sensors, but they also have their limitations. Regtien has mentioned a lot of the limitations of different sensors like the scalar sensors and image sensors mentioned previously. Scalar sensors, due to their typical applications demanding specifications like the accurate, contactless measurement of distances, are limited in that only some of the scalar sensors available may be able to meet the necessary specifications needed [7]. An example would be if a project’s specification needs a certain range for distance sensing and only some (and not all) scalar sensors are able to meet the specifications. Image sensors, with the information content of their output being much greater due to the sensor data referring to one-, two- or three-dimensional images, is limited in that the data acquisition and processing are more complex and time consuming compared to scalar sensors [7].

Currently there are relevant designs done using UHF RFID [4]. The design is based on having a central control tower in an area that uses all the phones in the area to transfer location of one user to another. This is relevant because a central device could be used in the home to provide location throughout if a child is lost or in an area, they shouldn't be in. According to Pang UHF RFID is lower cost and more energy efficient than other device tracking available today. Other types of tracking include GPS and BLE (Bluetooth low energy). GPS is popular but consumes a lot of energy when running continuously, which can lead to a very inefficient system. BLE could be used in the home, but due to its lack of range can lead to problems, if the child leaves the range of the device. A smaller home could benefit from using BLE, as you don't need a central tower device to connect transmissions and can directly connect parent to child devices. If a robot is used as the central tower, you can count out GPS as the tracking system and choose between BLE or UHF RFID as your tracking and communication system. The wristband device the child wears will be based on the child tracking device [11] that has already been created, with some modifications. The main modification will be removing the LCD screen and making a lock to keep the child from removing the device. This specific device also just connects directly to a smartphone instead of connecting to a mobile robot that will be used to track the child.

One of the features on the robot could be detecting if the child needs their diaper changed. According to the journal by Ema, Yokoyama, Nakamoto, and Moriizumi, a sensor to detect smell is very difficult, maybe even impossible to build as detailed in "Odour-Sensing System Using a Quartz-Resonator Sensor Array and Neural-Network Pattern Recognition". Their study includes several quartz resonators and CMOS oscillator circuits, so this would be very difficult to recreate within a small robot with a limited budget. Although this study was done in the 1980's, this type

of sensor would be difficult to implement in a university project with limited time and money.

[2]

With the project being an autonomous robot, other relevant technologies would be household robots such as the Roomba. Household environmental control robots are very common, however, there are few, if any, that exist with the purpose of providing child safety or autonomous care. Many robots exist with different features such as internal mapping systems, air filtration or purification [10], improved vacuums and brushes, and many more. With how widespread these inventions are, it is safe to say that the technology needed to create the artificial intelligence and mobility of the device exists and is accessible. One known method is to use mapping technology to allow the robot to improve its ability to navigate through the living space over time. This can be done using vSLAM navigation (visual based simultaneous localization and mapping), which allows the robot to create a map of the living space that improves over time [1]. With enough budget, this can even be used to create a UI-based image of the map on an LCD screen to allow users to modify the robot's behavior or patrolling area. This could make it possible for users to cut off portions of the map and even modify or redraw the map manually. This design would also include a processor connected to a memory and wireless network circuit [1]. This technology would allow for the memory to store different routines that generate commands that are received by the wireless network circuit. With this, different routines and commands can be programmed completely independent of the original invention's purpose, creating an entirely different usage.

1.4 Marketing Requirements GB, MH, BJ, MB

Table 1-Marketing Requirements

<ul style="list-style-type: none"> • The system should be compact and lightweight.
<ul style="list-style-type: none"> • The system should be able to safely navigate around a single room or open floor of the house.

<ul style="list-style-type: none">• The system should have a long-lasting power source.•
<ul style="list-style-type: none">• The system should be able to locate the child within a reasonable distance.•
<ul style="list-style-type: none">• The system should be able to communicate efficiently with the parent or guardian.
<ul style="list-style-type: none">• The system should be able to quickly respond to obstacles.

2 Engineering Analysis

2.1 Circuits – MB, GB

2.1.1 Power Sources MB

To obtain a general idea of what batteries could be used to have a functional robot, power is needed to obtain expected battery capacity. To have an estimate on how much power the Bot Sitter will need overall to operate, the power calculated for the motors in a later section, section 2.5.2, is used as the basis, as motors would presumably be the most power-costly out of all the devices integrated in the bot design. From the motor calculations, the power delivered throughout the bot could be estimated as higher than 65.064 Watts. In taking an overestimation of 80 Watts to consider the sensors, controllers, and more devices implemented in the bot design, Watt-hours (Wh) = 80Watts * 5hours = 400Wh. Using the calculated Watt-hours and a range of nominal battery voltages the bot could use, Table 2 shows resulting Amp-hours for common rechargeable battery configurations.

Table 2-Battery Options

Battery (nominal voltage)	Battery Type	Equation	Amp-hours
3.6 V	NiMh	$Wh/V = 400/3.6$	111.11 Ah
3.7 V	LiPo	$Wh/V = 400/3.7$	108.108 Ah
4.8 V	NiMh	$Wh/V = 400/4.8$	83.33 Ah
6 V	NiMh	$Wh/V = 400/6$	66.67 Ah
7.2 V	NiMh	$Wh/V = 400/7.2$	55.56 Ah
7.4 V	LiPo	$Wh/V = 400/7.4$	54.05 Ah
9.6 V	NiMh	$Wh/V = 400/9.6$	41.67 Ah

11.1 V	LiPo	Wh/V = 400/11.1	36.036 Ah
--------	------	-----------------	-----------

From the table, a reasonable choice of battery configuration would be 7.2 V NiMh. Keeping in mind that only rechargeable battery configurations were considered and that the calculations came from justifiable overestimates and engineering requirements, the tentative battery choice could still be changed for the final design.

The chosen battery for the Bot Sitter is a rechargeable 2S1P 7.2V lithium-ion battery with a capacity of 3.25Ah. In layman's terms, the battery basically contains two lithium-ion cells in series which has an overall nominal voltage of 7.2V and capacity of 3.25Ah. This battery is chosen for its light weight (being a lithium-ion battery) and its decent capacity of 3.25Ah. Since most batteries in the market do not have large capacities unless the battery configuration involves cells being in parallel to increase Amp-hours or have higher nominal voltages for higher capacities which all usually mean heavier and bigger batteries, this battery covers what the bot needs to be powered especially if the weight of the bot is being kept in mind.

Another power source to keep in mind is for the Bluetooth modules used for tracking and navigating the robot. For these Bluetooth beacons, lithium coin cell batteries can be used to power them since these modules require low power to function. In this case, 3V coin cell batteries are chosen.

2.1.2 Charging Methods MB

To have an efficient and functional Bot Sitter doing its job as needed, recharging is a must. Three ways to charge batteries were researched. The original idea was to use resonant wireless charging to recharge the batteries to be used in the bot as that charging method allowed for more

space in between the transmitting end and the receiver end of the charger and that method did not necessitate strict alignment of the receiver and transmitter of the charger compared to the inductive wireless charging where the gap between receiver and transmitter is basically nonexistent and the misalignment of the receiver and transmitter does not allow for the charging to work [6]. In comparing inductive wireless charging and resonant wireless charging, resonant wireless charging is less efficient, thus not preferable for this project [6]. As the bot would be able to autonomously move around, there is no guarantee to the accuracy of the bot's finer movements, so to be efficient in charging the bot, inductive wireless charging, which requires no misalignment of the receiver and transmitter of the charger, cannot be used. In the case that the bot would not be able to be accurate enough to use inductive wireless charging and in the case that resonant wireless charging is not efficient enough with its power transfer, the best charging method for the bot would be wired charging. Wired charging is a reasonable charging method as existing examples of them already exist like the charging docks for Roombas. This charging method would need an AC/DC converter, which would change the outlet 120 VAC into a DC voltage.

Since a lithium-ion battery is chosen, to implement a working charging configuration, a compatible lithium-ion charger MCP73844 IC chip along with the LS10-13B12R3 AC/DC converter are chosen. From the 7.2V lithium-ion battery data sheet [14], the battery's charge voltage is 8.4V in which the MCP73844 IC chip has a preset voltage regulation option for [15]. The LS10-13B12R3 AC/DC converter changes the outlet 120 VAC into a 12 VDC in which the MCP73844 IC chip accepts as an input for the 8.4V preset voltage regulation option [15]. The connection between the AC/DC converter, IC charging chip, and lithium-ion battery is shown clearly in Figure 23.

2.2 Electronics – MH, MB

2.2.1 Audio Signal Enable MH

The selected device to emit the audio alarm is the CMI-1295-0585T magnetic buzzer that has an operating voltage of 5V. According to the PIC24FJ128GA010 family data sheet [13], some GPIO (General Purpose Input/Output) pins can handle a maximum of 5.5V. According to the datasheet of the buzzer [16], the operating current is 30 mA. Due to these operating values and the supply values of the Explorer 16-32 board, no voltage or current amplifier are needed for the audio alarm. One of the GPIO pins on the PIC24 processor can be hooked up to the anode of the buzzer and can be enabled by software. When a minimum distance, that is defined in the software, is exceeded, the pin will go HIGH and the buzzer will sound

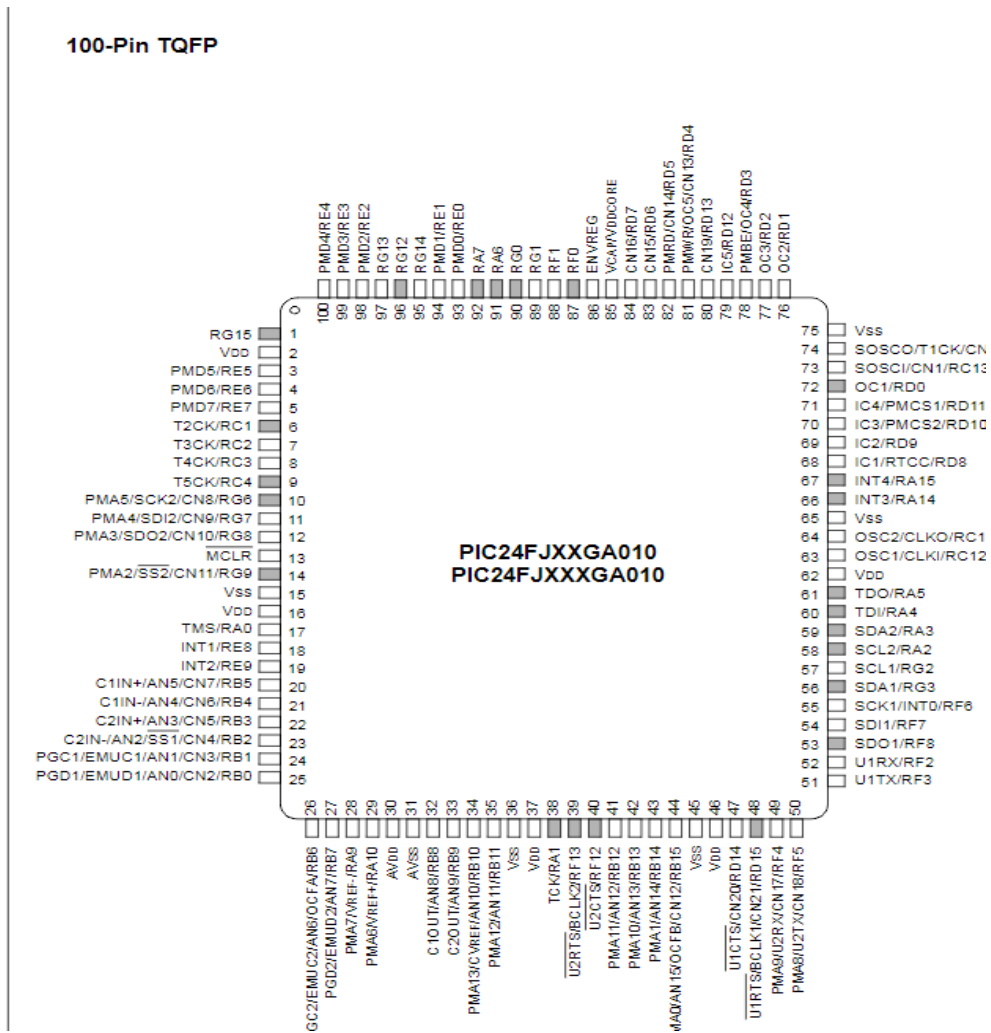


Figure 1 - Pinout Diagram from PIC24FJ128GA010 Datasheet Where Gray Shading Indicates 5.5V

2.2.2 Overvoltage and Overcurrent Protection MB

Generally, electronic devices have a limit to how much voltage and current could run through those devices. To prevent the degradation of the electronic devices in the bot, overvoltage and overcurrent protection need to be put in place. In Electronic Design class, many configurations of common electronic devices were taught, such as diodes. Configurations of these devices will aid in the overvoltage and overcurrent protection. One configuration for overvoltage protection could be using Zener diodes. A Zener diode is a special type of diode which operates in the

breakdown region which allows the current to flow in both the forward and reverse direction, whereas a normal diode only operates in the forward bias region in which the current only flows in the forward direction. A Zener diode can be used as a voltage regulator where the output voltage can be limited to what is the desired range of voltages, therefore, a working overvoltage protection configuration. One configuration for overcurrent protection could be using diodes. As mentioned previously, diodes operate in the forward bias region and only allows the current to flow in the forward direction, so in the case of overcurrent occurring, a circuit configuration which redirects the overcurrent through a path where a diode is flipped, which stops the current flow, will work as the overcurrent protection.

For the power connection to the PIC24FJ128GA010 board and connected devices, overvoltage protection comes in the form of the linear voltage regulator, LT1086IT-5. Since the battery chosen has a nominal voltage of 7.2V, the voltage sent to the PIC24FJ128GA010 board and other devices connected to it needs to be stepped down to roughly 5V as none of the devices need more voltage than that to be operational. The use of the LT1086IT-5 regulator dropped down the battery-supplied voltage to roughly 5.2V, which is in range of the operational voltage for the PIC24FJ128GA010 board and other connected devices.

2.3 Signal Processing – MH

Throughout the entire BotSitter system, there will be several signals for the microprocessor to handle. On the actual bot itself, there will be a Bluetooth module and a sensor that acts as a transceiver, whose details will be discussed later. There will also be Bluetooth module transceivers and sensors in several fixed areas around the room or open floor plan of the home. On the child, there will be the receiver on the microprocessor to collect information from the scattered Bluetooth transceivers on fixed areas throughout the room or open floor plan. The

microprocessor will take this locational data from the sensors and Bluetooth modules in the environment at a rate determined by the hardware clock on the microcontroller board. Therefore, the speed of calculations for moving the robot and processing speed of data is dictated by the chosen microcontroller. The nature of the signals emitted by the sensors will be a digital pulse read in by the microprocessor, and the characteristics of the Bluetooth communication can be detailed in the section regarding the Bluetooth 4.2 standard.

2.3.1 Sensor Selection MH, MB

When selecting the sensors to output the distance from the child to the robot or the child to a stationary danger area, predetermined by location such as a stove or a knife drawer, three main types of sensors came to mind. The first type of sensors researched was RFID; however, this technology was far too expensive and difficult to reverse engineer to apply to the scope of this project. The second sensor classification is infrared sensors. These are sensors that can detect distance by sending out infrared radiation. However, since these sensors are light based, the main drawback is that the signal is highly disrupted if there is an obstacle in the way [8]. An alternative sensor technology that also solves the issue of the sensitivity to obstacles is ultrasonic. Ultrasonic sensors send a pulse of high frequency sound that returns location data. Ultrasonic sound-based sensors deal with obstacles better since sound can bend easier around objects than light [5]. Therefore, the sensors in the BotSitter system will be ultrasonic technology.

The chosen ultrasonic sensors are the HC-SR04 distance sensors. These sensors have four pins- V_{cc} , Trigger, Echo, and Ground. The sensor operates using 5V DC and is triggered to initiate a reading by a 50kHz square wave. After this triggering, the sensor sends out 8 40kHz

pulses, then the Echo pin will receive another square wave that will be proportional to the distance detected by the sensor, as seen in Figure 2:

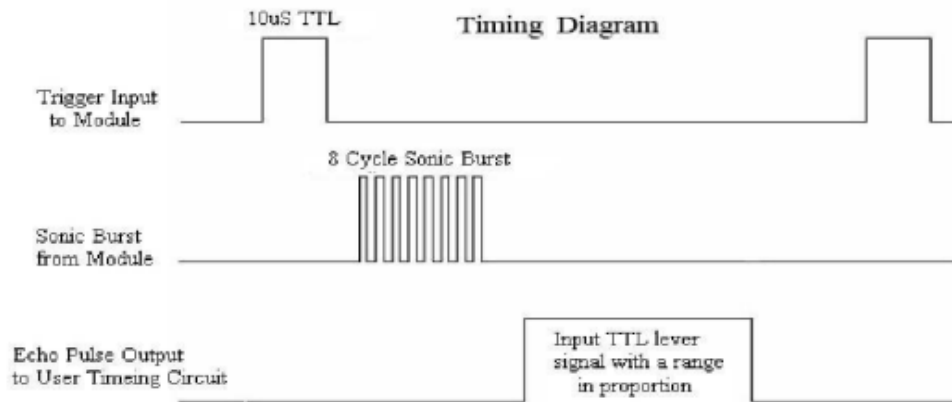


Figure 2 - Timing Diagram Taken from HC-SR04 datasheet

According to the datasheet previously cited, the range of this sensor is 2cm-400cm with a 15° measuring angle. The maximum and minimum width of the beam of measurement can be calculated using the boundary values of the sensor's range and the following equations based off a generic arc length formula:

$$\text{Angle Ratio (A.R.)} = 360^\circ / 15^\circ = 24$$

$$\text{Beam Width} = \frac{2\pi r}{A.R.}$$

The maximum beam width corresponds to the maximum range of the sensor and is approximately 105cm. Similarly, the minimum beam width corresponds to the minimum range of the sensor and is approximately 0.5cm.

2.4 Communications – BJ

Communication with any sensors or transceivers located on the device will require a communication protocol of choice. Since the robot will require high-speed calculations and a fast

response time, speed will be prioritized above all else in communication. For that purpose, the SPI protocol would be optimal for most use cases as it is the fastest available protocol. However, another thing to consider is the fact that the device will most likely only need to read data from peripherals rather than write, meaning bi-directional communication is of little importance.

Because of this, the I²C protocol can be considered because it would lose a key disadvantage in its lack of duplex communication. I²C also benefits from the ability to support multiple master devices with only 2 wires, unlike SPI. This will make I²C an optimal choice if the device needs to read data from more than 3 peripherals.

2.5 Electromechanics – GB, MB

The power storage in the robot will be rechargeable from a charging dock. The internal power system will be providing power to the motor subsystem, directional sensors, Bluetooth module, and audio device. The motor subsystem will consist of one motor controller for every two motors being used in the system. Therefore, the subsystem will consist of 2 motor controllers and 4 motors. In this section, the motor subsystem will be analyzed to show minimum values to determine what motors are viable in the system.

2.5.1 Motor Controller Requirements GB

To determine which motor and motor controllers are viable in the system, the factors to investigate will need to be the power needed for each motor, the radial velocity (rotations per minute), current draw from the motors, and the torque of the motors. Given the marketing requirements the options for the motor can be cut down and selected from a smaller group. To ensure the motors work correctly some results have been given a worst-case scenario value to overestimate what is needed in case of error during operation. These values include the friction

coefficient, weight of the robot, maximum speed, size of wheels, and the amount of time the robot will run without charge.

2.5.2 Motor Calculations GB

The Bot Sitter must be able to move through its environment sufficiently. To do this, calculations have been done to determine motor requirements for the robot.

$$\text{Gravity } (g) = 9.81 \frac{m}{s^2}$$

$$\text{Average friction coefficient of carpet } (\mu) = 0.7$$

$$\text{Weight of Robot } (m) = 12\text{lbs} = 5.44 \text{ Kg}$$

$$\text{Max Speed } (v) = 1 \frac{ft}{s} = 0.3048 \frac{m}{s}$$

$$\text{Size of Wheel } (r) = 20 \text{ mm}$$

$$\text{Time of Robot running } (t) = 5 \text{ hours} = 18,000 \text{ s}$$

$$\text{Normal Force } (F_N) = 9.81 \frac{m}{s^2} * 5.44 \text{ Kg} = 53.366 \text{ N}$$

$$\text{Force of Friction } (F_f) = 0.7 * 53.366 \text{ N} = 37.357 \text{ N}$$

$$\text{Rotational Velocity} = \frac{0.3048 \frac{m}{s}}{20 \text{ mm}} = 15.24 \frac{rad}{s} = 2.4255 \frac{rev}{s} = 145.53 \text{ rpm}$$

$$\text{Motor Torque } (T) = 53.366 \text{ N} * 20 \text{ mm} = 1.06732 \text{ N} * m$$

$$\text{Motor Power } (P) = 15.24 \frac{rad}{s} * 1.06732 \text{ N} * m = 16.266 \text{ Watts}$$

$$\text{Motor Energy } (E) = 16.266 \text{ Watts} * 18,000 \text{ s} = 292.787 \text{ KJ}$$

$$\text{Motor Current}(I) = \frac{16.266 \text{ Watts}}{12 \text{ V}} = 1.355 \text{ A}$$

These calculations are for just one motor so the total power would be 65.064 Watts with 1.4 A from each motor. With these calculations made, a viable motor can be easily chosen for the Bot Sitter. Given these calculations a motor controller board selection can be narrowed down using the current and assumed voltage of the motors.

2.5.3 DC Brushless vs. DC Brushed GB

When choosing a DC motor to use there is the choice between brushed and brushless. While both could work, a brushed motor will be better for the Bot Sitter. Just from availability and simplicity alone, a brushed motor is much more useful to the motor subsystem. Its only negatives are the efficiency and maintenance of the brushes as they wear down over time

2.6 Computer Networks – GB

Bluetooth Low Energy or BLE is a subset of the wireless protocol Bluetooth developed under IEEE 802.15.1 to provide communication through radio signals. BLE is great for low power consumption and decent data transfer rates.

Another wireless protocol is Zigbee, which is like Bluetooth. The main difference is that Zigbee has better range but slower transfer rate. Bluetooth is the best candidate for the Bot Sitter in helping with trilateration of the child in a singular room or space, while keeping data transfer to a maximum for constant updates to the robot for navigation and alarms.

Most of the trilateration calculations can be seen below in section 2.7.2 below. But the generalized form of it is to take RSSI values from preset modules in the room to convert to distances and confirm the distance and position of the child and robot in the room. Once that is calculated the distance between the two is calculated and the robot moves.

2.7 Embedded Systems – BJ, MH

The microcontroller will be the brain of the BotSitter system. It will read in information from the Bluetooth transceivers and ultrasonic ping sensors and then dictate how far the robot needs to move to keep close to the child. The microcontroller of choice will likely be the PIC24 provided in the Explorer 16/32 Development board. This controller will fit the requirements due to it providing multiple modules for SPI, I²C, and UART to fit whatever protocol is needed along with sufficient I/O ports. Communication protocols will be used to communicate with the transceivers and navigation sensors while general-purpose I/O will be used for motor control.

2.7.1 Ultrasonic Sensor MH

One of the main tasks of the embedded controller is to read the value of the echo pin on the ultrasonic sensor and calculate the distance. As seen in the following two figures, the embedded controller generates the trigger function, reads the time that the echo pin is high, and calculates the distance in centimeters. Since the echo pin transmits the distance of the object as a time high in microseconds, the code below multiplies the time by the speed of the ultrasonic waves in centimeters per microsecond to yield the distance in centimeters. Then there is a 4-millisecond delay, so the sensor does not get bogged down by getting triggered too frequently. This calculated distance is then displayed on the LCD display on the Explorer 16-32 board by using modulus and division to isolate each digit and converting it to hexadecimal. The LCD on the board then displays each hexadecimal value in XX.X cm format. To ensure the timing of these calculations and to make sure the microcontroller can focus on other tasks, these commands are done in an interrupt service routine. The separate cases of the switch statement in the code shown below are the generation of trigger for each directional sensor with their

corresponding echo pins. The accuracy of each measurement was checked by using a centimeter ruler and was accurate up to 3mm.

```
void init_ping()
{
    TRISB = 0b010101;
    TRISA = 0x04;
    AD1PCFG = 0xFF;
    T1CON = 0x8000;
    _T1IP = 4; //standard interrupt priority
    PR1 = 180000;
    _T1IF = 0;
    _T1IE = 1;
}

int pingDistance(int side)
{
    int distance = 0;
    float duration = 0;
    int i = 0;

    switch(side)
    {
        case 0: // Front Ping
            PORTBbits.RB1 = 0;
            for (i=0; i<1040; i++)
            {
                PORTBbits.RB1 = 1;
            }
            PORTBbits.RB1 = 0;
            TMR1 = 0; while (PORTBbits.RB0 == 0);
            TMR1 = 0; while (PORTBbits.RB0 == 1);
            duration = TMR1;
            distance = ((474)*(long)duration)/5000;
            distance = distance/100;
            break;
    }
}
```

Figure 3 - Embedded Code to Initialize Onboard LCD Screen for HC-SR04 Sensor Data

```

47 #define putLCD(d) writeLCD(1,(d))
48 #define cmdLCD(c) writeLCD(0,(c))
49 #define homeLCD() writeLCD(0,2)
50 #define clrLCD() writeLCD(0,1)
51 int main(void)
52 {
53     // initialize the device
54     SYSTEM_Initialize();
55     initLCD();
56     TRISB=0x01;
57     AD1PCFG=0x03;
58
59     while (1)
60     {
61         duration=0;
62         PORTBbits.RB1=0;
63
64         for (int i=0; i<=160; i++)
65         {
66             PORTBbits.RB1=1;
67             if (PORTBbits.RB0==1)
68             {
69                 duration++;
70             }
71             distance=(34*duration)/2000; //convert pulse to cm
72             distance=distance/1000; //"MPLab HATES floating point arithmetic"- Dr. Adams
73         }
74         d3=distance%10; //isolate digit 3
75         distance=distance/10; //divide by 10 for next digit
76         d2=distance%10; //isolate digit 2
77         distance=distance/10; //divide by 10 for next digit
78         d1=distance;
79         putLCD(d1+0x30); //print digits and convert to hex so LCD happy
80         putLCD(d2+0x30);
81         putLCD(d3+0x30);
82         putLCD(' ');
83         putLCD('c');
84         putLCD('m');
85         cmdLCD(0x80); //clears LCD
86
87         for (int i=0; i<65500; i++); //delay roughly 4 milliseconds
88     }
89
90     return 1;
91 }

```

Figure 4 - Embedded Code to Generate Trigger Signal for HC-SR04 Sensor and Read Echo Pin to Calculate Distance

2.7.2 Trilateration Calculations

One of the most costly tasks for the controller to perform will be to calculate the positional data of the two mobile Bluetooth modules. Luckily, this can all be found using signal strength through a value known as the “received signal strength indicator,” or RSSI. The RSSI between two Bluetooth modules can be related to their distance using the following equation:

$$RSSI = -(n \cdot 10 \log_{10}(d) + A)$$

Where d is equal to distance and A and n are constants. Using this, the following equation can be derived to calculate distance:

$$d = 10^{\frac{-A-RSSI}{10n}}$$

With the ability to calculate the distance between two Bluetooth Modules, the distance between two mobile devices in the system can be consistently known. However, to follow the target, the device must also know a direction to move to approach it. To do this, four “corner modules” can be placed in a rectangle around the area, each with their own Bluetooth module. If these corner modules have known distances from each other along with calculated distances between the two mobile modules, trilateration can be used to create a coordinate system inside the area, find the direction a signal is coming from, and an alternative way to find distance between the two mobile modules. The following system of equations can be used to find the coordinates of a Bluetooth module in the system:

$$(x - x_3)^2 + (y - y_3)^2 = d_3$$

$$(x - x_4)^2 + (y - y_4)^2 = d_4$$

Where x_3 , x_4 , y_3 , are y_4 are the coordinates of the bottom left and right corner modules and d_3 and d_4 are the distance between those modules and the moving module in the system. 2x2 systems such as these can be solved algorithmically using a Newton algorithm, however, since there will be no variation to this system, a solution can be algebraically found and applied for a much faster, less general, algorithm.

```

44 void findCoordinates(float *x, float *y, int x1, int y1, int x2, int y2, float d1, float d2)
45 {
46     float u;
47     float v;
48     float w;
49
50     float a;
51     float b;
52     float c;
53
54     u = pow(x2, 2) - pow(x1, 2) + pow(d1, 2) + pow(y2, 2) - pow(y1, 2) - pow(d2, 2);
55     v = 2 * (x2 - x1);
56     w = y2 - y1;
57
58     a = 1 + ((4*pow(w, 2))/(2*pow(v, 2)));
59     b = ((4*w*(x1-u))/v) - 2*y1;
60     c = (pow(u, 2)/pow(v, 2)) - ((2*x1*u)/v) + y1 - pow(d1, 2);
61
62     *y = (-b + sqrt(pow(b, 2) - 4*a*c))/2*a;
63     if (*y < 0)
64     {
65         *y = abs(*y);
66     }
67     *x = sqrt(pow(d1, 2) - pow(*y, 2));
68 }
69

```

Figure 5 - C code to calculate the coordinates of a moving Bluetooth module within the system

With coordinates assigned to each corner module and hard coded along with newly calculated coordinates for the mobile modules, an angle between the two points can be calculated with many different trigonometric functions, with the following being used:

$$\angle A = \sin^{-1}\left(\frac{y_2 - y_1}{d}\right)$$

However, this angle is not the direction needed, but instead an angle between the two points. To find a true direction, a “zero direction” must be established, as in, a direction equal to zero degrees or radians. In the case where the negative x direction is considered zero, the following equations can be used to find a direction:

If $x_2 > x_1$ and $y_2 > y_1$: $direction = 180 - \angle A$ or $direction = \pi - \angle A$

If $x_2 < x_1$ and $y_2 > y_1$: $direction = \angle A$

If $x_2 < x_1$ and $y_2 < y_1$: $direction = -\angle A$

If $x_2 > x_1$ and $y_2 < y_1$: $direction = 180 + \angle A$ or $direction = \pi + \angle A$

As long as the robot is able to keep track of its current direction through its motor movements, it should be able to find the correct direction to move in once all positional data is determined.

When simulating these calculations in MATLAB, an average running time of around 55ms was found, an amount of time adequate to allow the robot to check for positional data at least every second, even in the case of poor optimization.

In the c-code we can access the RSSI data by opening up the command window of the Bluetooth module. This is done through multiple commands as seen in the figure below. Once it is accessed you can type M to get the RSSI value output and then read that back into the explorer board using the get command. The code for this is in the appendix.

2.7.3 Motor Control:

Motor control can be achieved with simple input/output from the PIC24 controller. The motor controller allows the direction of two motors to be decided with two bits of logic each. This can easily be done in a C script through MPLAB using the TRISx and LATx registers. The TRISx registers are used to decide which pins on port x are inputs or outputs while the LATx registers can be used to configure the logic of each port x pin to high or low. Port E of the PIC controller was chosen, with pins 0-4 being used for the AI1, AI2, BI1, BI2, and STBY inputs on the motor controller. Each different logic configuration can then be assigned to a function for each possible movement mode. However, this does not consider speed control. The motor

controller also allows for speed control through the PWMA and PWMB input pins. This allows the speed of the motor to be higher or lower depending on the period of the oscillation period of the two PWM inputs. Without speed control, the device would only be capable of rotating in place, but speed control allows it to turn while moving by having one motor rotate faster than the other.

```
15 void init_motors()
16 {
17     TRISE = 0xE0;
18     LATE = 0x10;
19
20     T2CON = 0x8030;
21     T3CON = 0x8030;
22     OC1CON = 0x000E;
23     OC2CON = 0x0006;
24
25     OC1R = OC1RS = period/4;
26     OC2R = OC2RS = period/4;
27
28     _T2IF = 0;
29     _T3IF = 0;
30
31     _T2IE = 1;
32     _T3IE = 1;
33
34     PR3 = period;
35     PR2 = period;
36 }
37
38 void forward()
39 {
40     LATE = 0b11001;
41     OC1RS = period/2;
42     OC2RS = period/2;
43 }
44
45 void backward()
46 {
47     LATE = 0b10110;
48     OC1RS = period/2;
49     OC2RS = period/2;
50 }
51
```

Figure 6 - C code to control motor movement, including initialization and forward and backward movement

```
52 void rotateRight()  
53 {  
54     LATE = 0b10101;  
55     OC1RS = period/2;  
56     OC2RS = period/2;  
57 }  
58  
59 void rotateLeft()  
60 {  
61     LATE = 0b11010;  
62     OC1RS = period/2;  
63     OC2RS = period/2;  
64 }  
65  
66 void turnRight()  
67 {  
68     LATE = 0b11001;  
69     OC1RS = period/2;  
70     OC2RS = period/8;  
71 }  
72  
73 void turnLeft()  
74 {  
75     LATE = 0b11001;  
76     OC1RS = period/8;  
77     OC2RS = period/2;  
78 }  
79  
80 void stop()  
81 {  
82     LATE = 0b1000011;  
83     OC1RS = period/4;  
84     OC2RS = period/4;  
85 }  
86
```

Figure 7- MPLAB functions for Turning

3 Engineering Requirements Specification – GB, MB, MH, BJ

Table 3-Engineering Requirements

Marketing Requirements	Engineering Requirement	Justification
1	1. The weight of the robot should not exceed 12 lbs.	This is a reasonable weight to keep everything lightweight but not too light that a child will easily pick up the robot.
1, 6	2. The robot should maintain a speed of 1 ft/s with a maximum speed of 2 ft/s	This is a reasonable max speed as we don't want to go too fast and run into obstacles or the child.
3	3. The robot will be able to go without charge for a minimum of 2.5 hours	This is a reasonable time to run the robot, without neglecting the child for too long. This also allows for a smaller battery that keeps the robot lightweight
3	4. The robot will be able to charge itself in 5 hours	This is reasonable for the robot to be used to keep track of the child multiple times a day
4	5. The robot will stay within 1-4ft of the child	This is reasonable as the child won't move terribly fast but also keeps the robot from getting too close to the child and preventing harm.
2, 6	6. The robot will recognize an obstacle within 1 ft, 360 degrees of robot	This is a reasonable distance for a sensor to detect an obstacle from as the sensors can pick stuff up from 2cm to 400cm.
2, 6	7. The robot will start correcting its direction from an obstacle within 0.5 seconds of detection of the obstacle.	This is more than reasonable for communication from the sensor through the controller to the motor controller. Most communications take 0.1 seconds so allowing for error and multiple communications we can assume 0.5 seconds.
4	8. The robot will check on the child's location every 5 seconds minimum.	This is a reasonable time to check in on the child periodically. This is because we want the robot to "ping" the child periodically to assure safety of the child. If the robot doesn't have any obstacles or other things to worry about the child's location will be checked faster.
5	9. The robot will alert within 1 second of detecting child near the perimeter of the "play-zone".	This is reasonable for the detection to alert the audio cue from the controller. We want to hold off on the alarm right away in case the child is just passing by so waiting the extra time for communication to come through at about 0.1sec per comm allows us to wait a second in total before checking.

4	10. The trilateration calculations of the child's exact location will be communicated and completed by the robot within 3 seconds	This is reasonable as it needs to be faster than requirement 8 and still gives time for calculations to be completed.
Marketing Requirements Above in 1.4		

4 Engineering Standards Specification

4.1 Safety – MB

See section 2.2.2 above.

4.2 Communication – GB

Bluetooth is the communication protocol being used to communicate between the beacons, robot, and child. Bluetooth 4.2 is low energy and can be used consistently inside of a single room. Its theoretical data speed is around 1 Mbps which is more than enough for our purposes. The protocol stack in Bluetooth being used is shown in Figure 8.

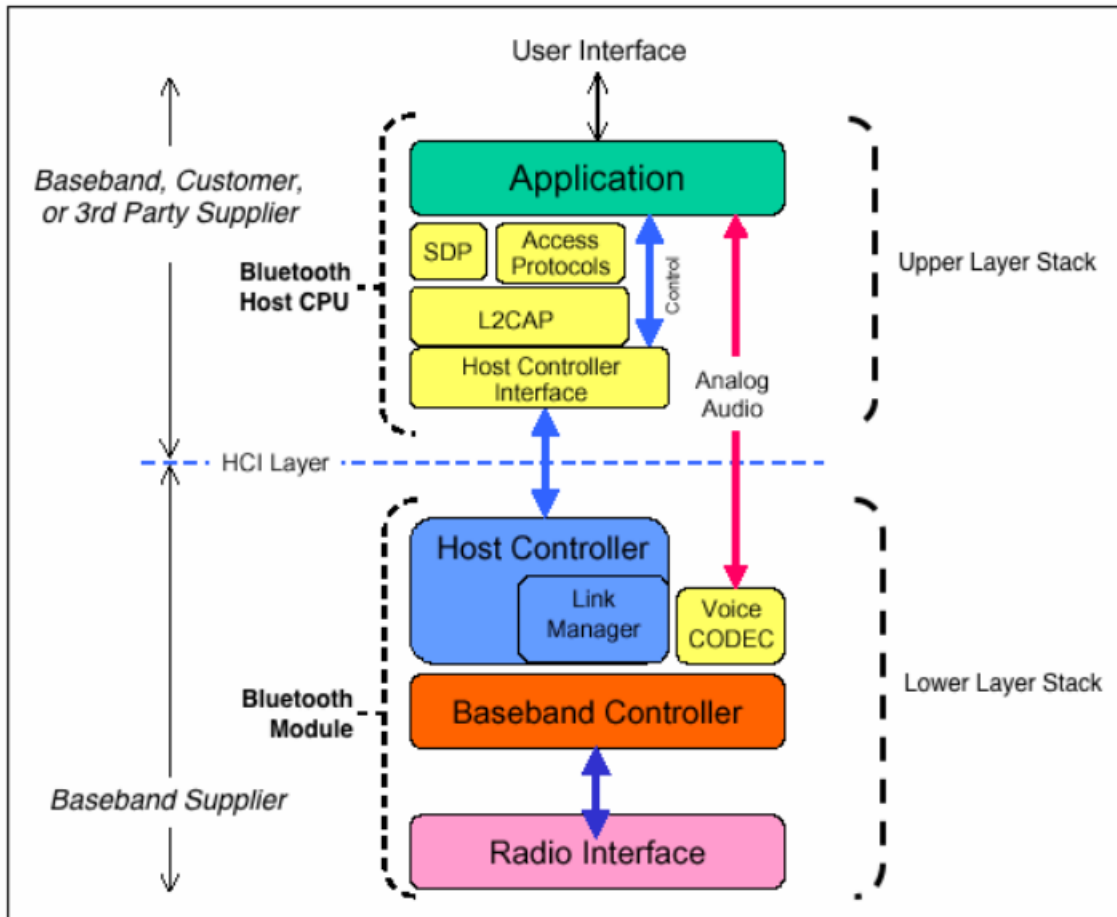


Figure 8-Bluetooth Protocol Stack

4.3 Design Methods – MH

To develop the scope and theory behind this system took several steps and revisions. First, the problem of a child getting into something while their guardian is not paying attention was defined. As seen in the first section of the report, the next step was to figure out a system that would help solve this problem. One solution to this problem would be the objective of the BotSitter system. After defining the objective, the next step was to research ways in which the child could be tracked. The overall topics that were researched were passive and active RFID, Bluetooth trilateration, Zigbee protocol, and the types of sensors in section 2.3.1. Once the general idea of devices used for tracking were selected, a system that would process all the signals needed to be selected. These previous steps could fall under the umbrella term of ‘Engineering Analysis’ like all of the other topics mentioned in that section. The existence of the Gantt chart that was completed earlier in the semester helps keep track of when tasks need to be completed and which specific team member is the leader of said tasks.

4.4 Programming Languages - BJ

In embedded programming, priorities tend to stray towards speed, efficiency, and lower-level access to machine hardware. Because of this, the C language tends to be favored for use cases such as this due to high efficiency and lack of high-level computations being needed. On the other hand, a language such as Python runs far slower and is more suited for automation of repetitive tasks. Use of a slow or inefficient language could force the robot to make occasional pauses for calculations or the retrieval of data. C will likely be the sole language used, but in the unlikely scenario where computations are too high-level, C++ code could be used as a supplement.

5 Accepted Technical Design

5.1 Hardware Design – MH, MB, BJ

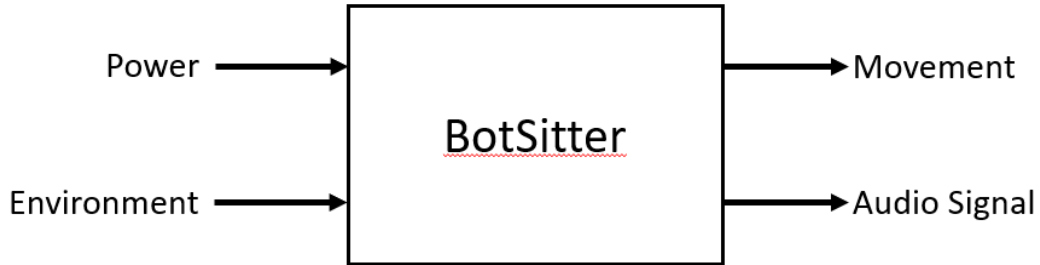


Figure 9-Level 0 Hardware Block Diagram

Table 4-Level 0 Functional Requirements Table

Module	BotSitter
Designers	Gregory Blondheim Jr., Marian Bonto, Maria Hatzis, Brett Jacobsen
Inputs	Power: ?V DC Battery Child and Environment Detection Input (Capacitive Pressure Sensor Gyroscope Sensor)
Outputs	Audio Output: ?V peak value, Movement
Description	Set applicable distance allowed for a child to wander around household areas. If the child approaches an unsafe area or escapes a confined area, such as the bed in the middle of the night, an audio alarm rings.

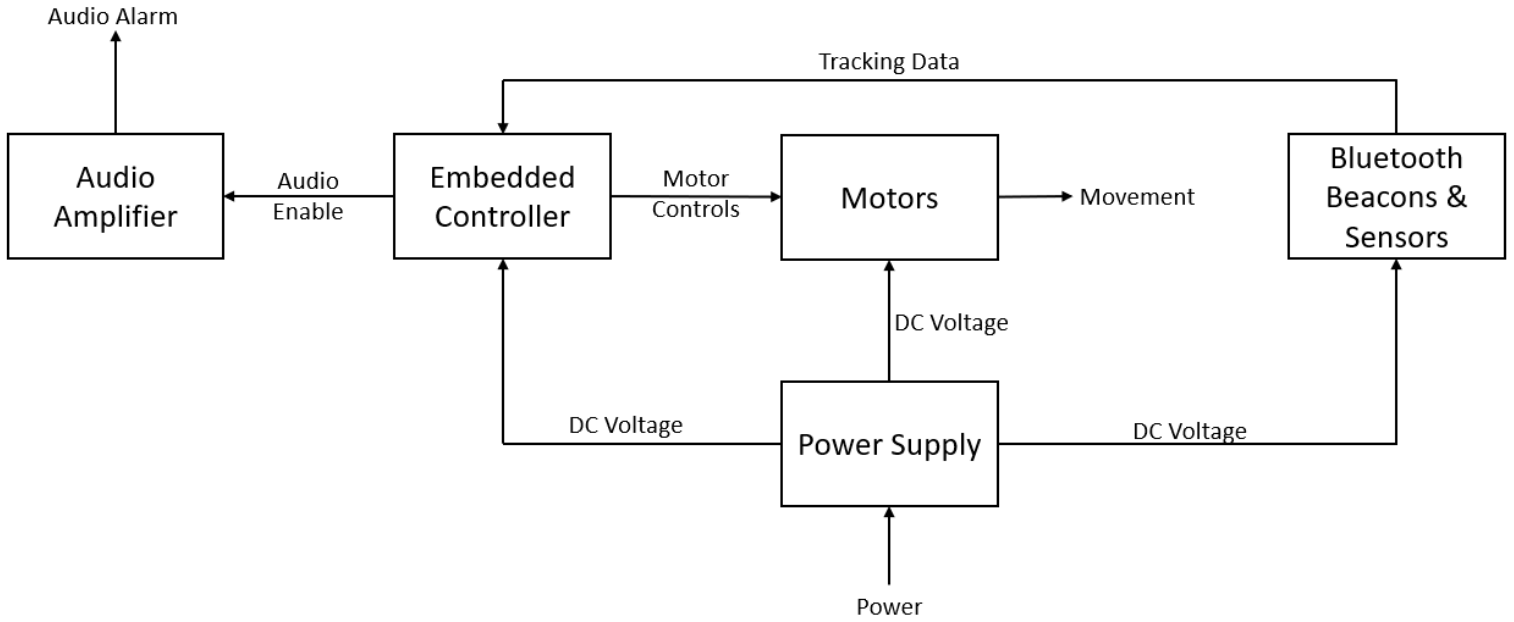


Figure 10-Level 1 Hardware Block Diagram

Table 5-Level 1 Functional Requirement Table

Module	BotSitter
Inputs	?V DC *See Table 2*, Embedded Controller Code, Motor Controls, Sensor Outputs
Outputs	Audio Alarm, Movement
Functionality	Use embedded controller to collect information from the stationary Bluetooth beacons and ping ultrasonic sensors for tracking the child on the floor of the home, moving the robot towards the child, and giving the signal to sound the audio alarm when they are in a dangerous area.

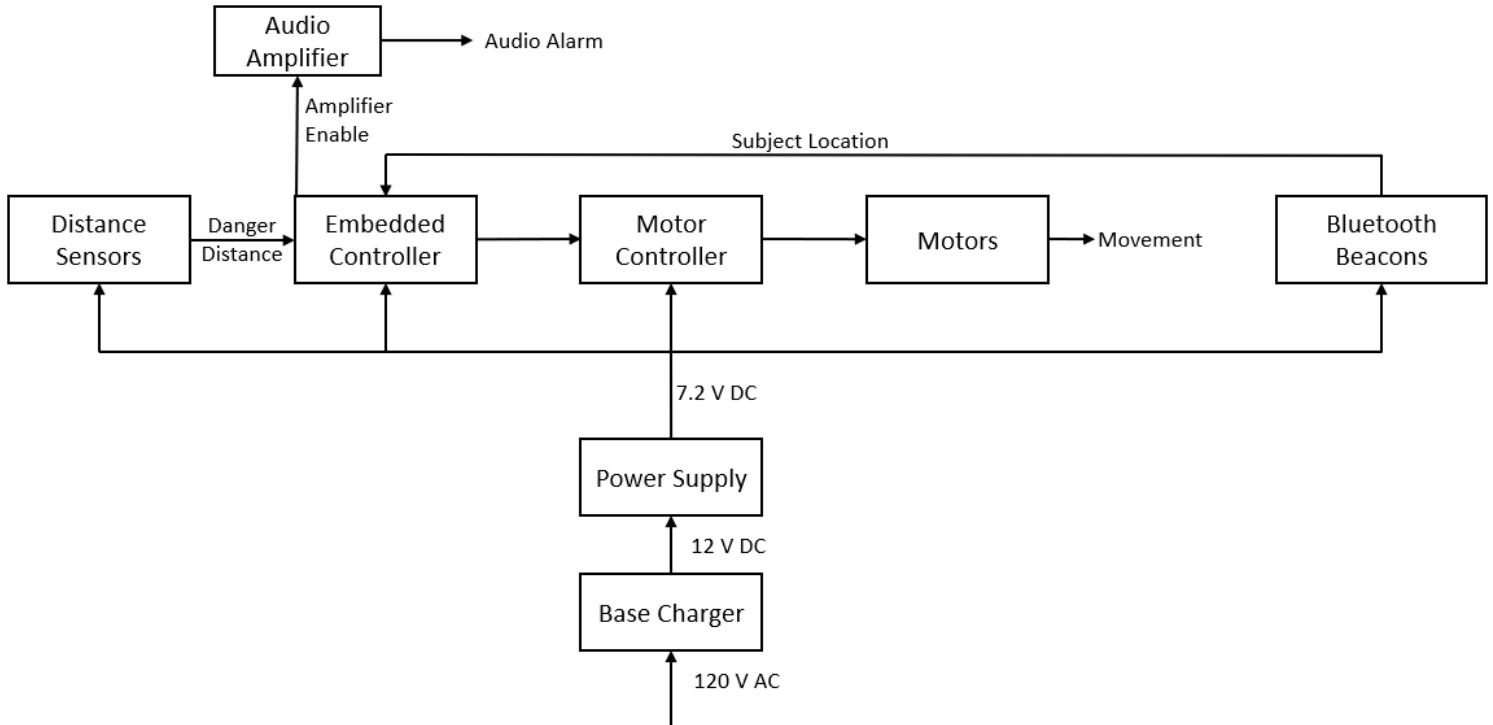


Figure 11-Level 2 Hardware Block Diagram

Table 6-Level 2 Functional Requirements Table

Module	BotSitter
Inputs	120VAC rms, ?VDC *See Table in Section 2.1.2*, Motor Controls, Sensor Outputs (Subject Location, Danger Distance)
Outputs	Audio Alarm, Movement
Functionality	Use embedded controller to collect information for tracking the child, collecting data from the danger distance sensors, and giving an audio alert if the child enters a dangerous area. The motor controller controls the mobility of the robot from information given by the embedded controller.

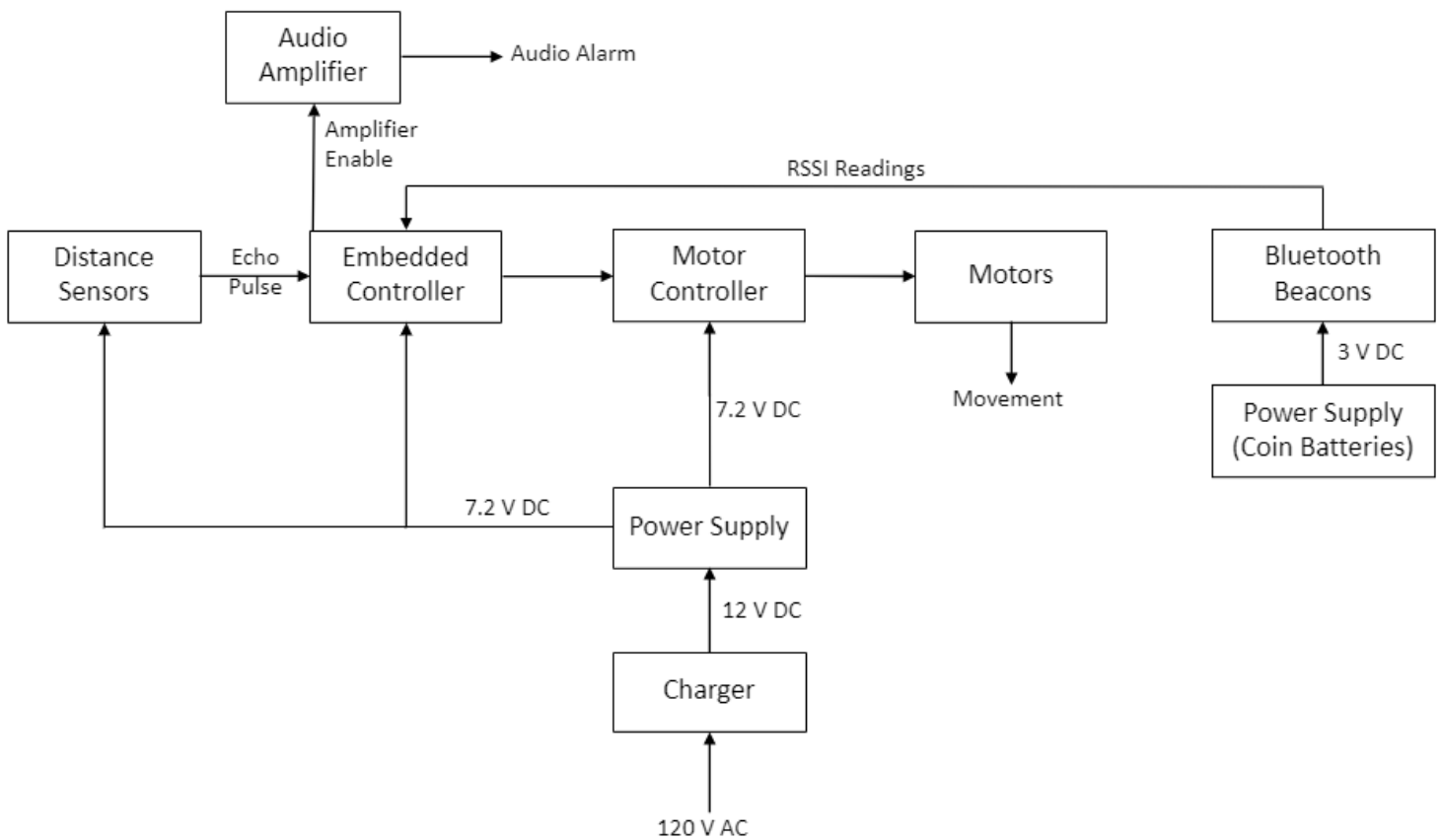


Figure 12 - Level 3 Hardware Block Diagram

Table 7-Level 3 Functional Requirements Table

Module	BotSitter Hardware System
Designers	Marian Bonto, Maria Hatzis
Inputs	120VAC rms, 3VDC, 12VDC, 7.2VDC, Motor Controls, Sensor Outputs (Subject Location, Danger Distance)
Outputs	12VDC, 7.2VDC, Audio Alarm
Description	Use embedded controller (PIC24FJ128GA010) to collect information for tracking the child, collecting data from the danger distance sensors (HC-SR04), and giving an audio alert (CMI-1295-0585T) if the child enters a dangerous area. The motor controller (Dual TB6612FNG-1A) controls the mobility (using 20.4:1 Metal Gearmotor 25Dx50L mm LP 6V) of the robot from

	information given by the embedded controller.
--	---

5.1.1 Circuit Schematics GB, MB

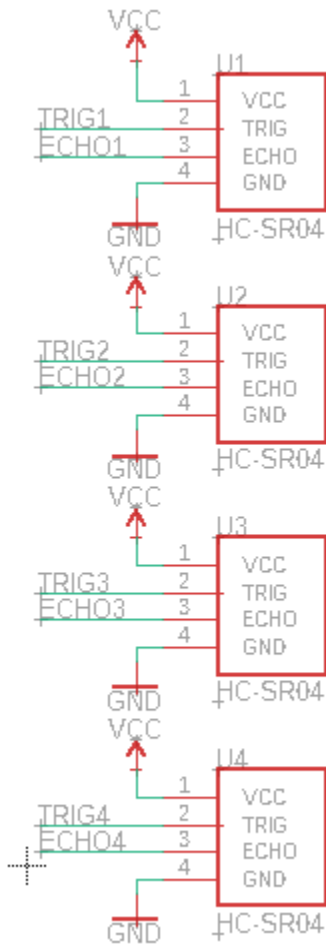


Figure 13 - Ping Sensor Schematic

Ping sensors are placed in the front, back, and sides of the bot which are then used for object detection and avoidance.

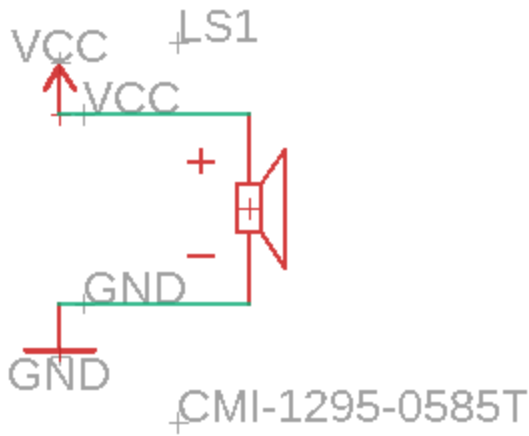


Figure 14 - Buzzer Schematic

Buzzer is used as the audio alarm when the bot registers the child to be in “danger”.

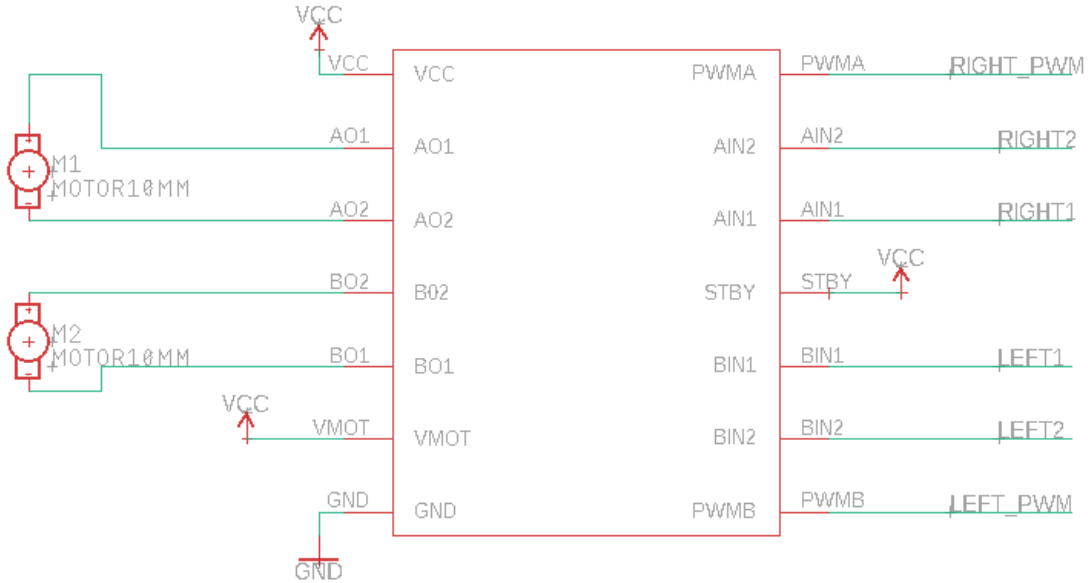


Figure 15 - Motor Controller Schematic

Motor controller is part of the motor subsystem which helps control the navigation of the bot.

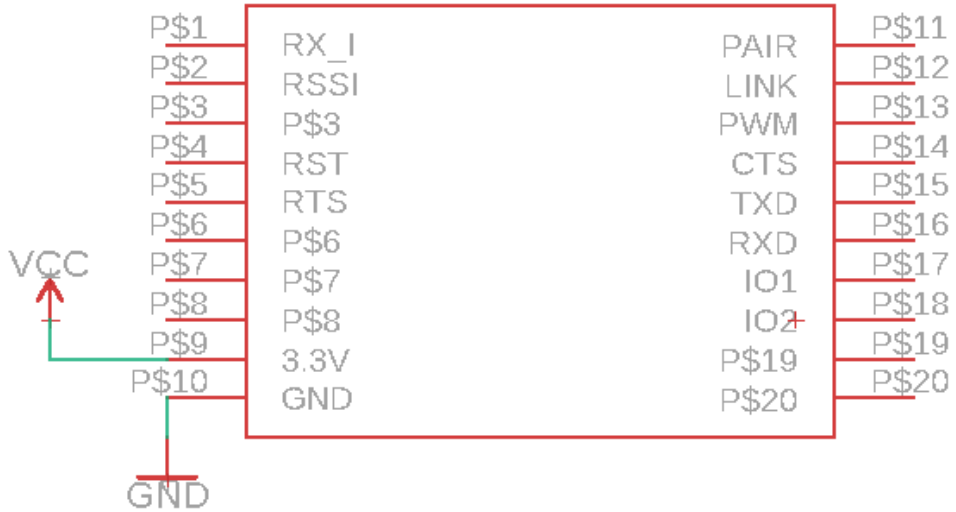


Figure 16 - Bluetooth Corner Module 1 Schematic

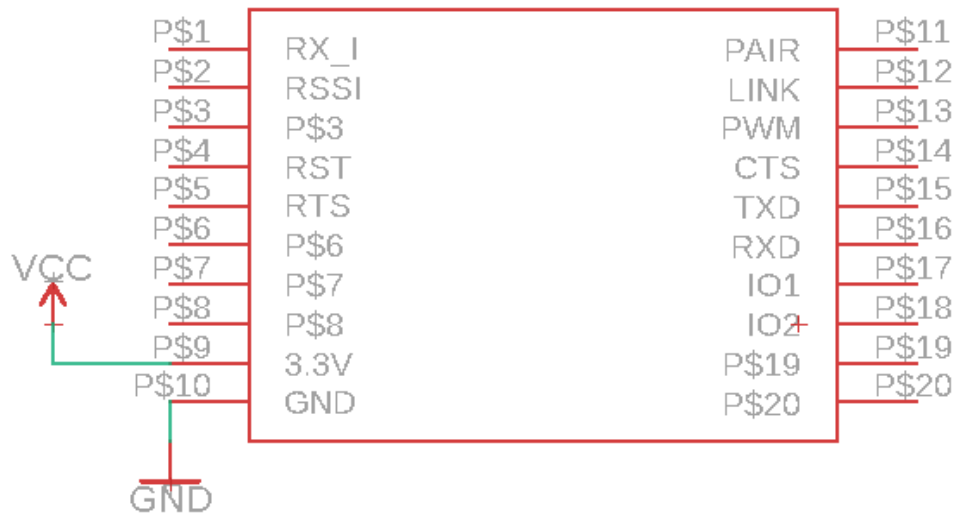


Figure 17 - Bluetooth Corner Module 2 Schematic

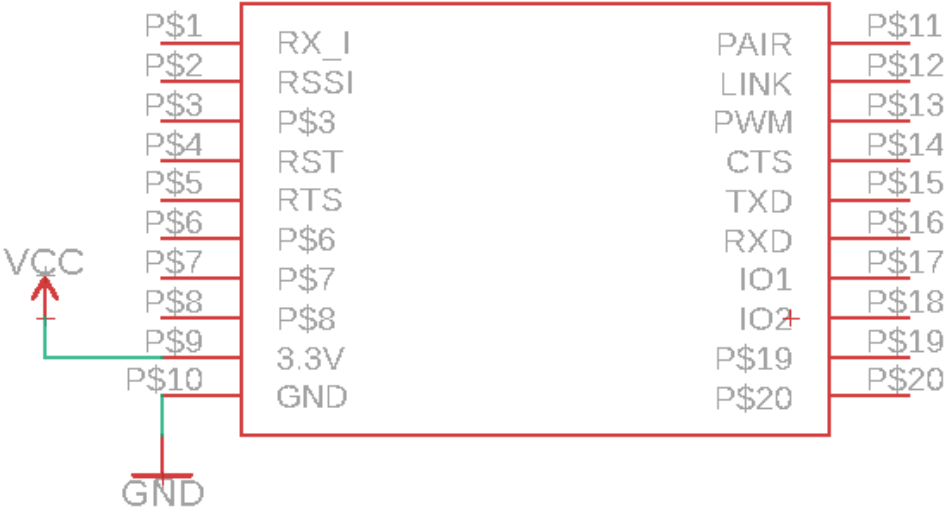


Figure 18 - Bluetooth Corner Module 3 Schematic

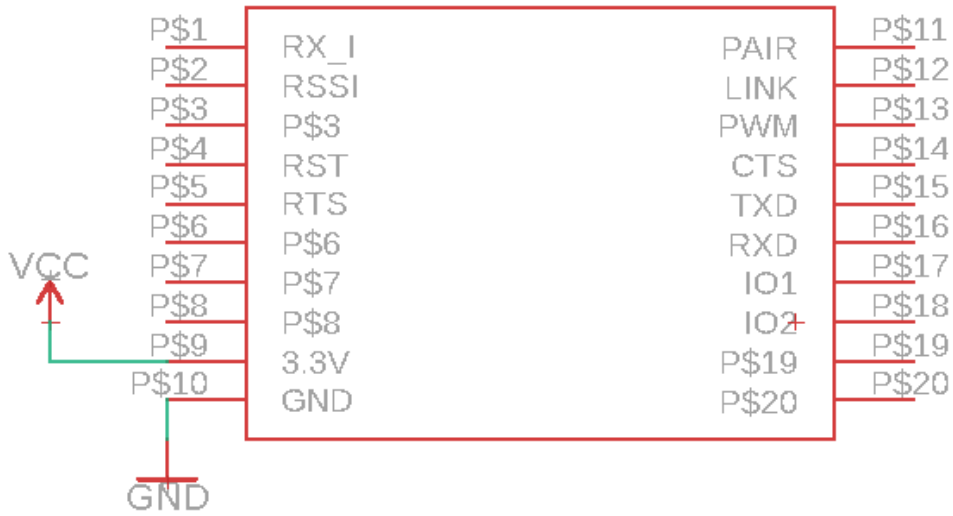


Figure 19 - Bluetooth Corner Module 4 Schematic

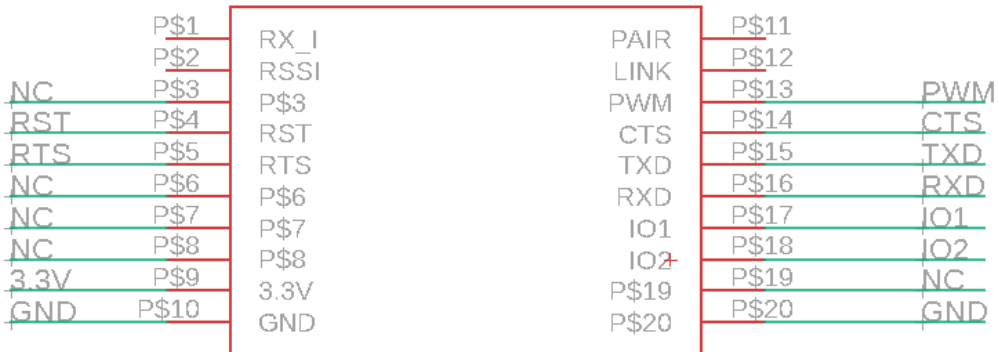


Figure 20 - Bluetooth Robot Schematic

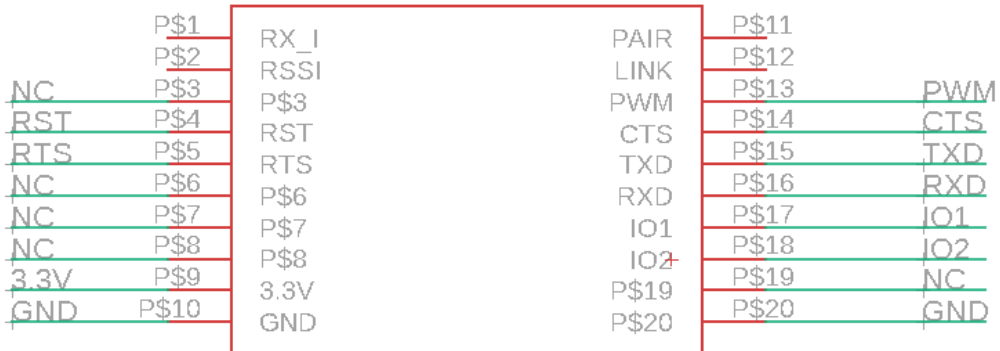


Figure 21 - Bluetooth Target Schematic

The Bluetooth modules (corner, robot, and target) are used by the bot to track and navigate.

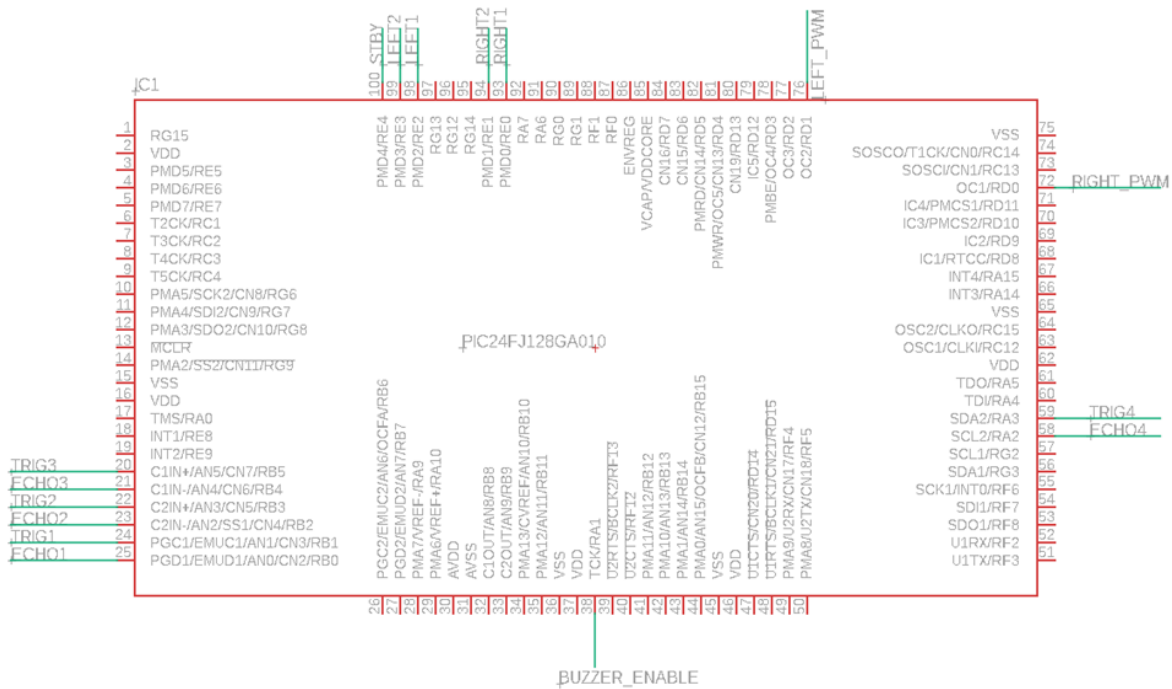


Figure 22 - Explore 16/32 Schematic

The Explore 16/32 board is the brain of the bot where it is used as the processor that allows the tracking, object detection, motor, and audio alarm to cohesively work together.

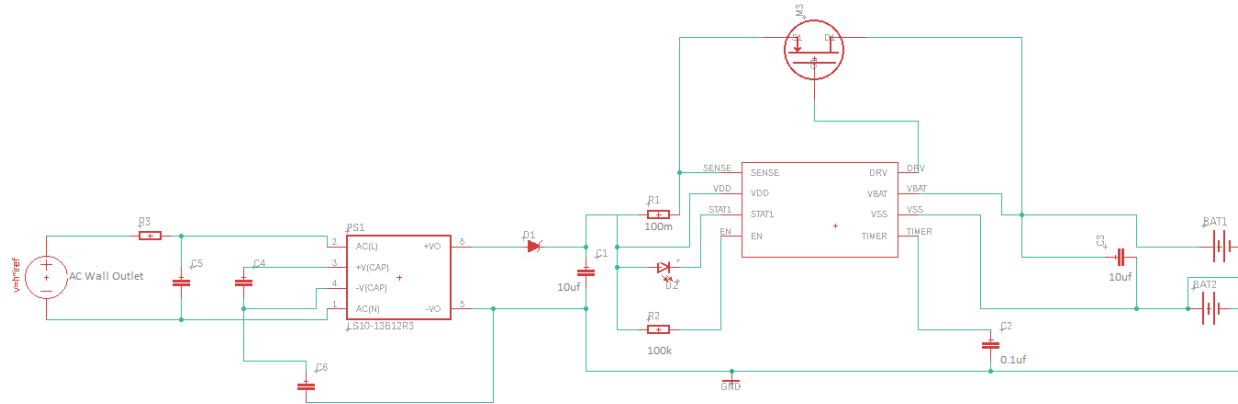


Figure 23 - Charging Circuit Schematic

The charging circuit is used to recharge the main power supply of the bot, the 7.2V lithium-ion battery.

5.2 Software Design – BJ, GB

All software is contained in an MPLAB IDE project written in the C language. The software was designed to separate all major software concerns into different source implementation files and header files. These concerns include localization, Bluetooth-reading, ping sensor-reading, motor control, and deciding movement, with movement decision being contained in the main source file. Other files were also included in the project to simplify common behavior of the Explorer Board, including I²C communication, delay times, and LCD writing for testing and debugging. With each concern separated into a separate file with their own sets of functions, different actions of the device can be easily programmed into the main source file with a simple function call.

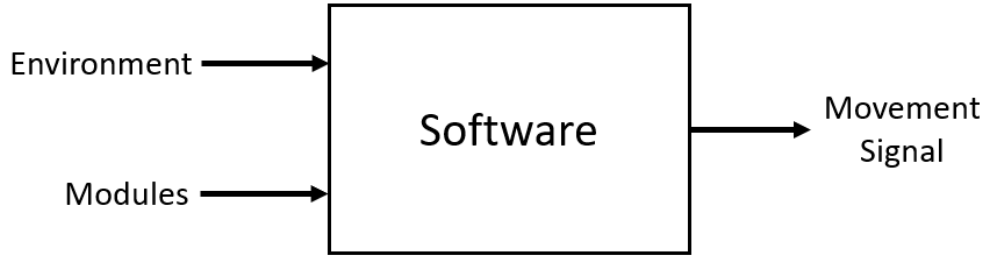


Figure 24-Level 0 Software Block Diagram

Table 7 - Level 0 Functional Requirements

Module	Software System
Designers	Brett Jacobsen, Greg Blondheim
Inputs	Sensors
Outputs	Movement and Alerts
Description	The software system must collect data from sensors on the robot to determine what the robot will do next. The software will determine its direction of movement and whether to alert.

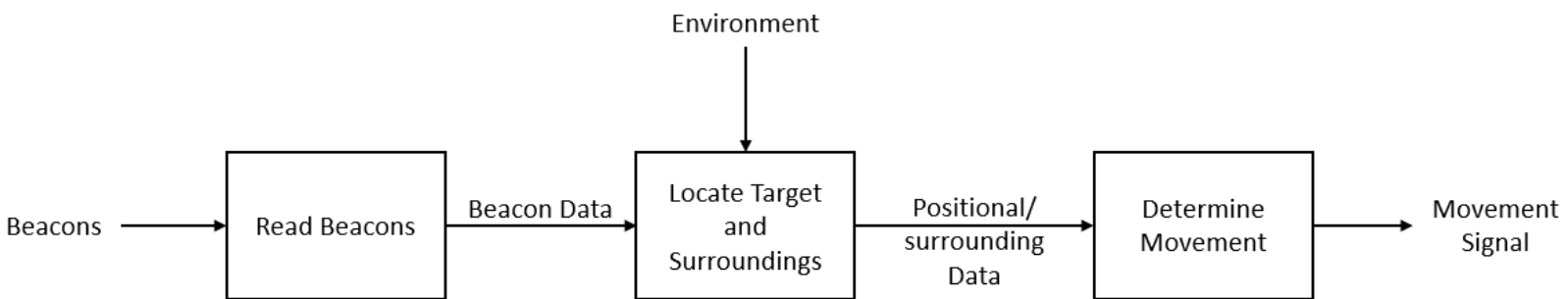


Figure 25-Level 1 Software Block Diagram

Table 8 - Locate Target and Surroundings

Module	Locate Target and Surroundings
Designers	Brett Jacobsen, Greg Blondheim
Inputs	Environment, Beacon Data
Outputs	Positional and Surrounding Data
Description	The software will take the sensor data and output the location of the child, hazards, and any obstacles in front of the robot

Table 9 - Determine Movement

Module	Determine Movement
Designers	Brett Jacobsen, Greg Blondheim
Inputs	Child/Obstacle Locations
Outputs	Movement Signal
Description	If there is an obstacle the robot will correctly maneuver around while continuously moving towards the child. If there is no obstacle, then the robot will just continue following the child.

Table 10-Read Beacons

Module	Read Beacons
Designers	Brett Jacobsen, Greg Blondheim
Inputs	Beacons
Outputs	Beacon Data
Description	The software interprets the data received from the beacon modules to obtain distance and other necessary information.

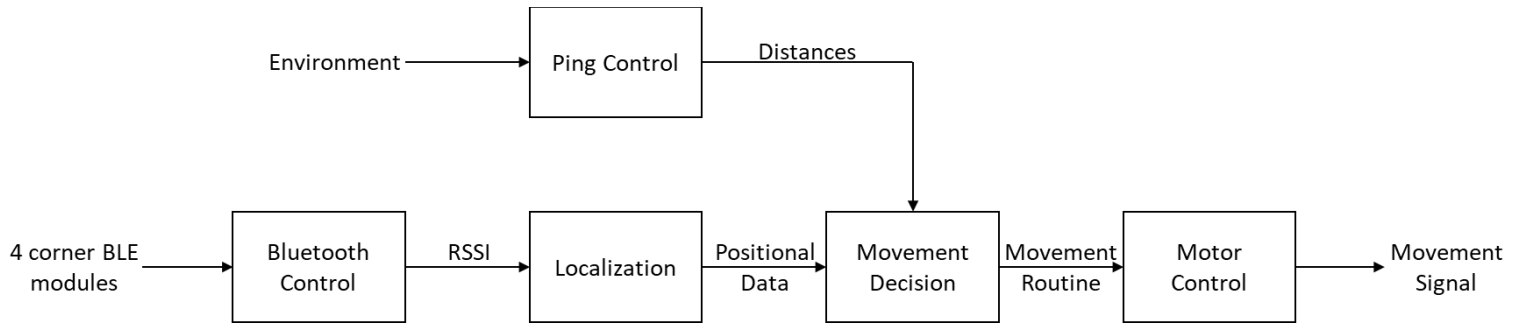


Figure 26-Level 2 Software Block Diagram

Table 11-Locating Target and Surroundings

Module	Ping Control
Designers	Maria Hatzis, Brett Jacobsen
Inputs	Environment
Outputs	Distances
Description	The data obtained from ping sensors is used to obtain rough distances from nearby obstacles

Table 12-Read RSSI from Modules

Module	Bluetooth Control
Designers	Greg Blondheim, Brett Jacobsen
Inputs	Four Corner BLE Modules
Outputs	RSSI

Description	Embedded software obtains RSSI values between the device and corner BLE modules to be written into memory.
-------------	--

Table 13-Calculate position of other device

Module	Localization
Designers	Brett Jacobsen, Greg Blondheim
Inputs	RSSI
Outputs	Positional Data
Description	Calculate the rough coordinates of the other device in the system along with an approximate direction to move in.

Table 14-Determine Movement

Module	Movement Decision
Designers	Brett Jacobsen
Inputs	Positional Data, Distances
Outputs	Movement Routine
Description	Determine the motor routine needed at the moment based on positional data and the existence of nearby obstacles.

Table 15-Signal Motors

Module	Motor Control
Designers	Brett Jacobsen
Inputs	Movement Routine
Outputs	Movement Signal

Description	Embedded software sends signals to the motor controller to enact the determined movement.
-------------	---

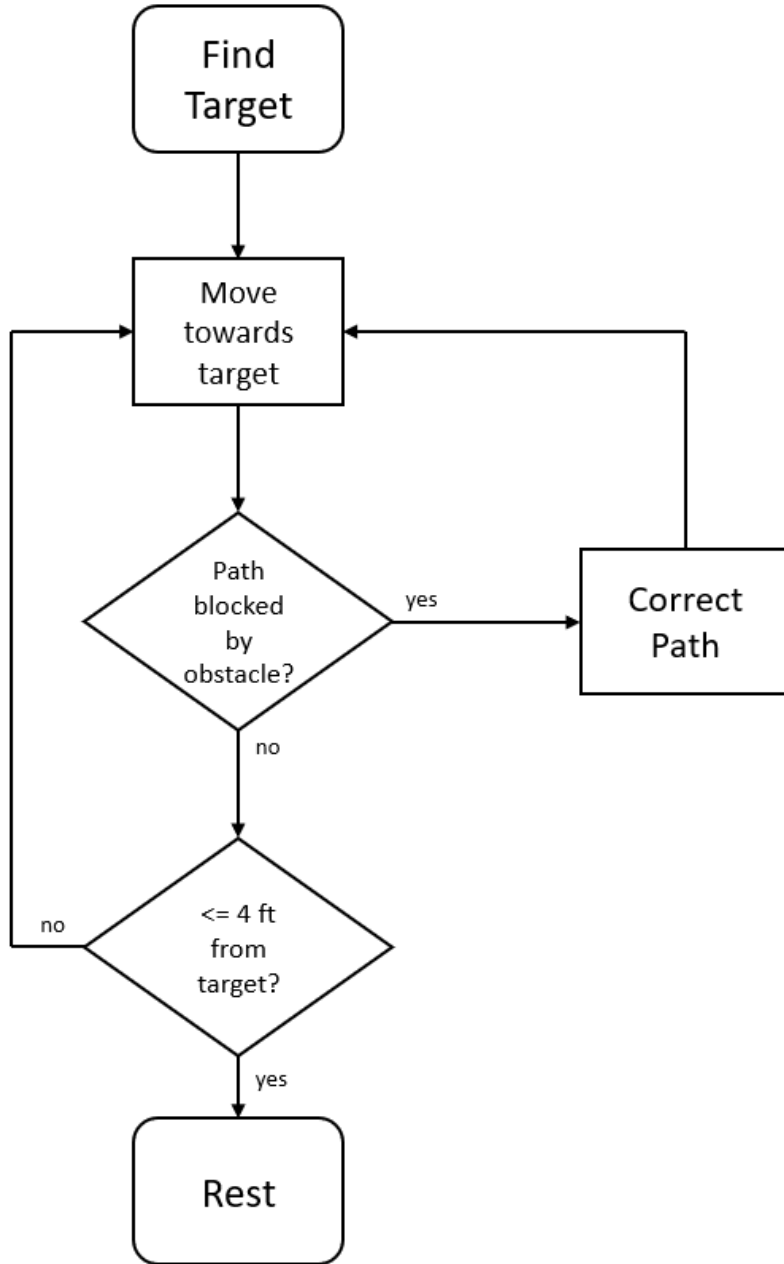


Figure 27-Level 1 Navigation Flowchart for seeking and avoiding obstacles, contained in Movement Decision block

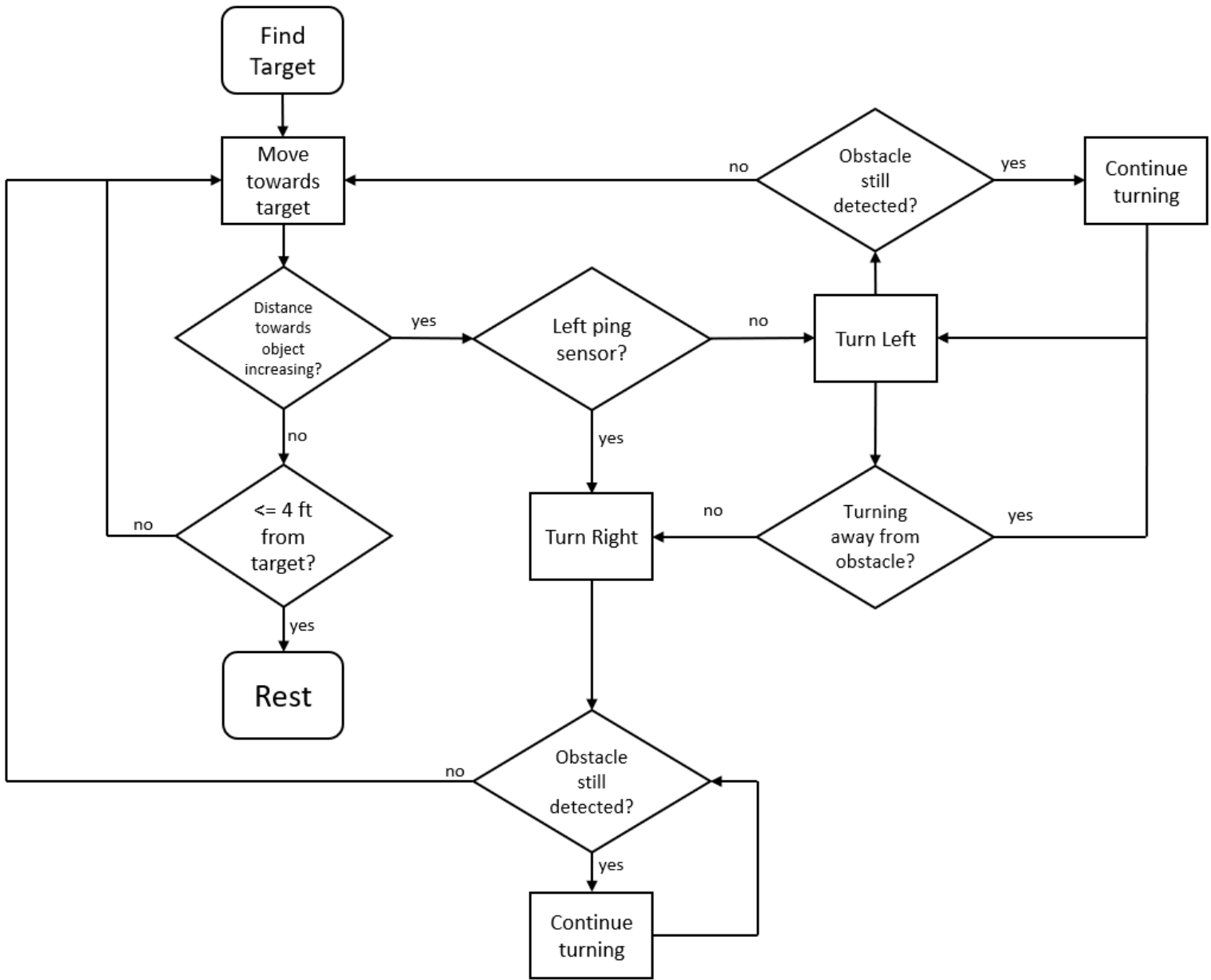


Figure 28-Level 2 Navigation Flowchart

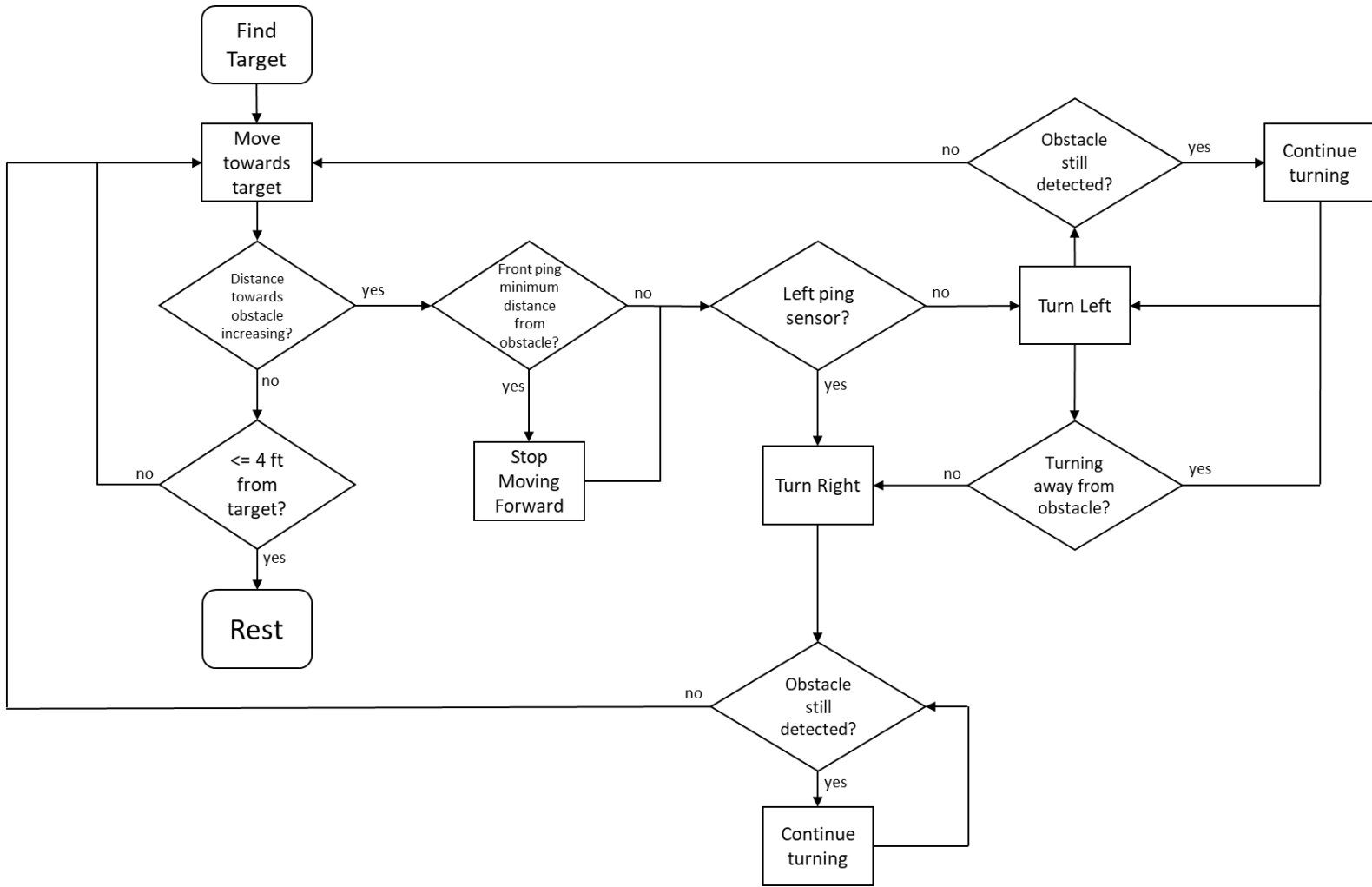


Figure 29-Level 3 Navigation Flowchart

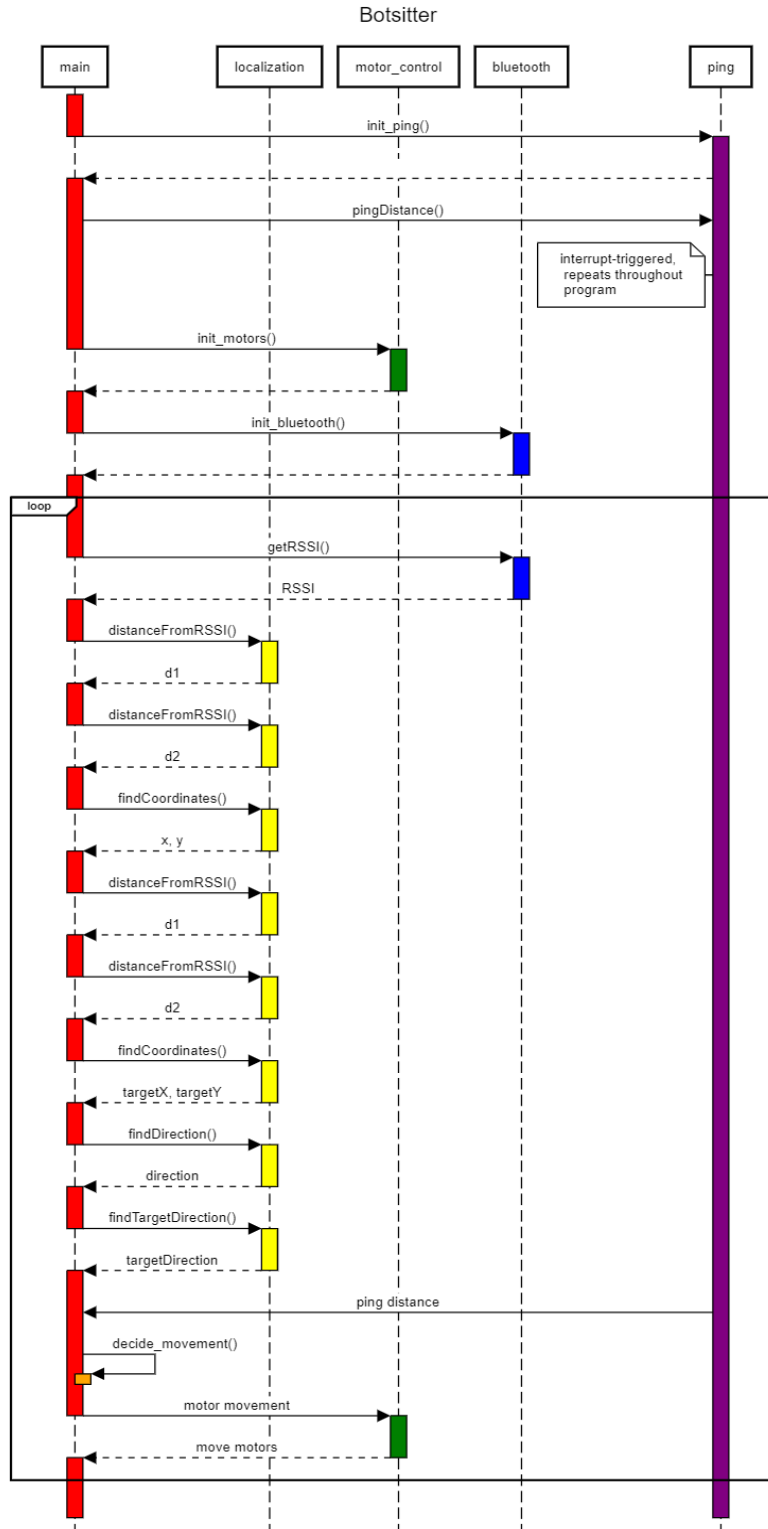


Figure 30 - Sequence Diagram representing software design and flow of control

6 Mechanical Sketch – MB

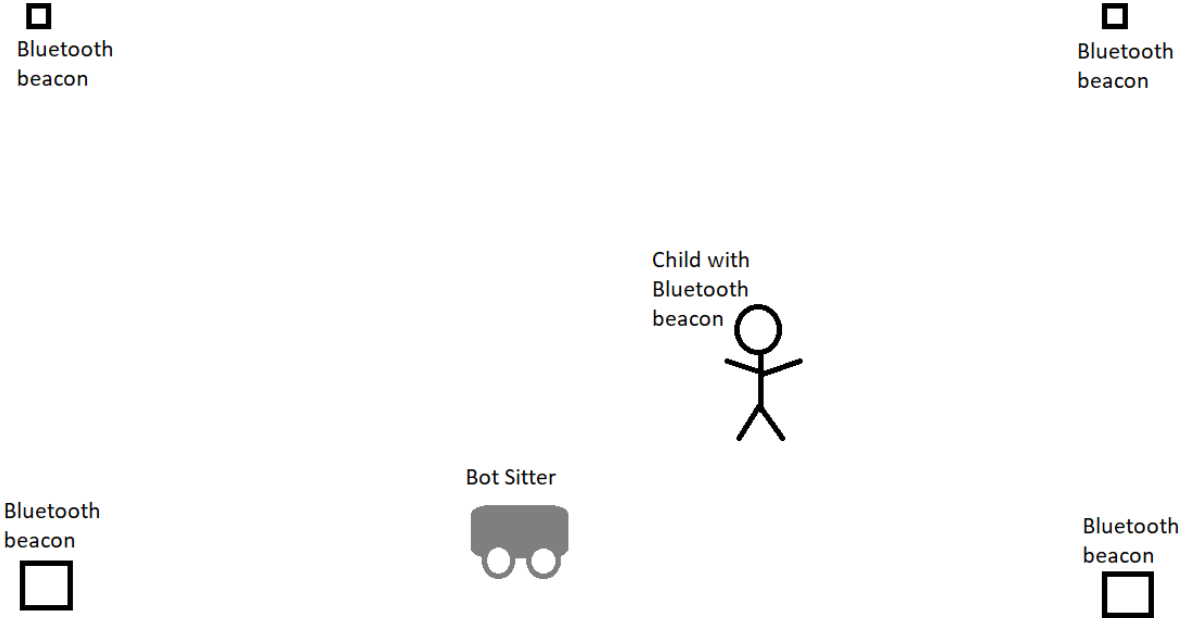


Figure 31-Level 1 Sketch of Environment

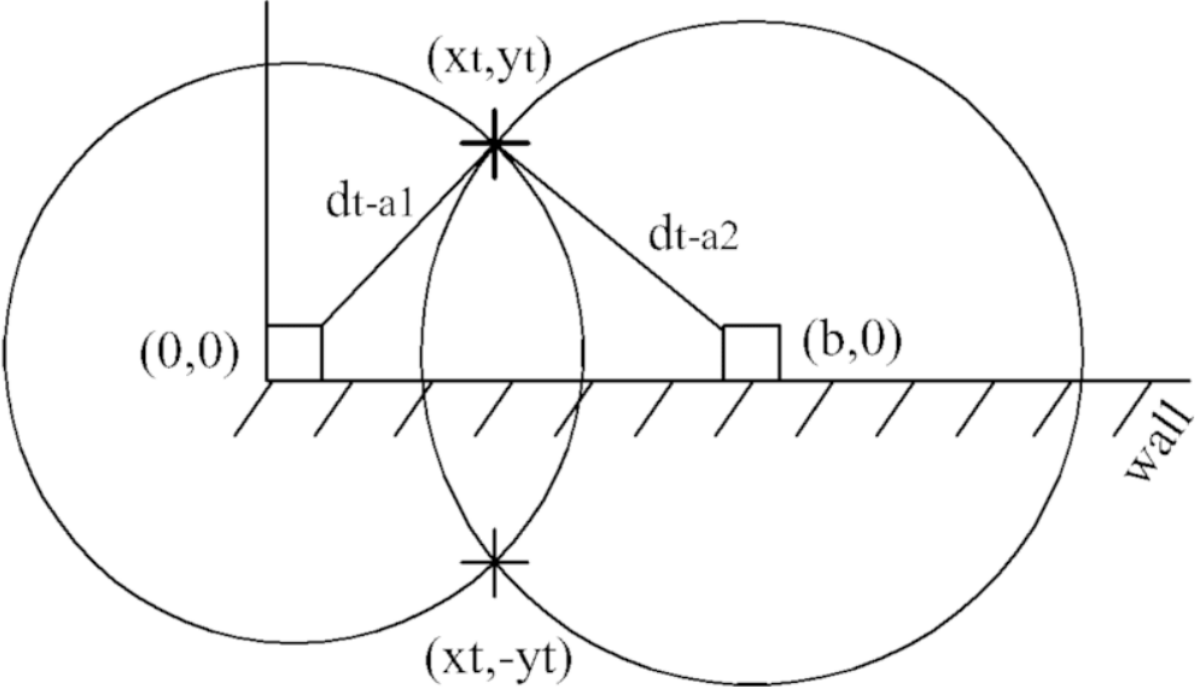


Figure 32 - Trilateration of Room using bottom Anchor Points

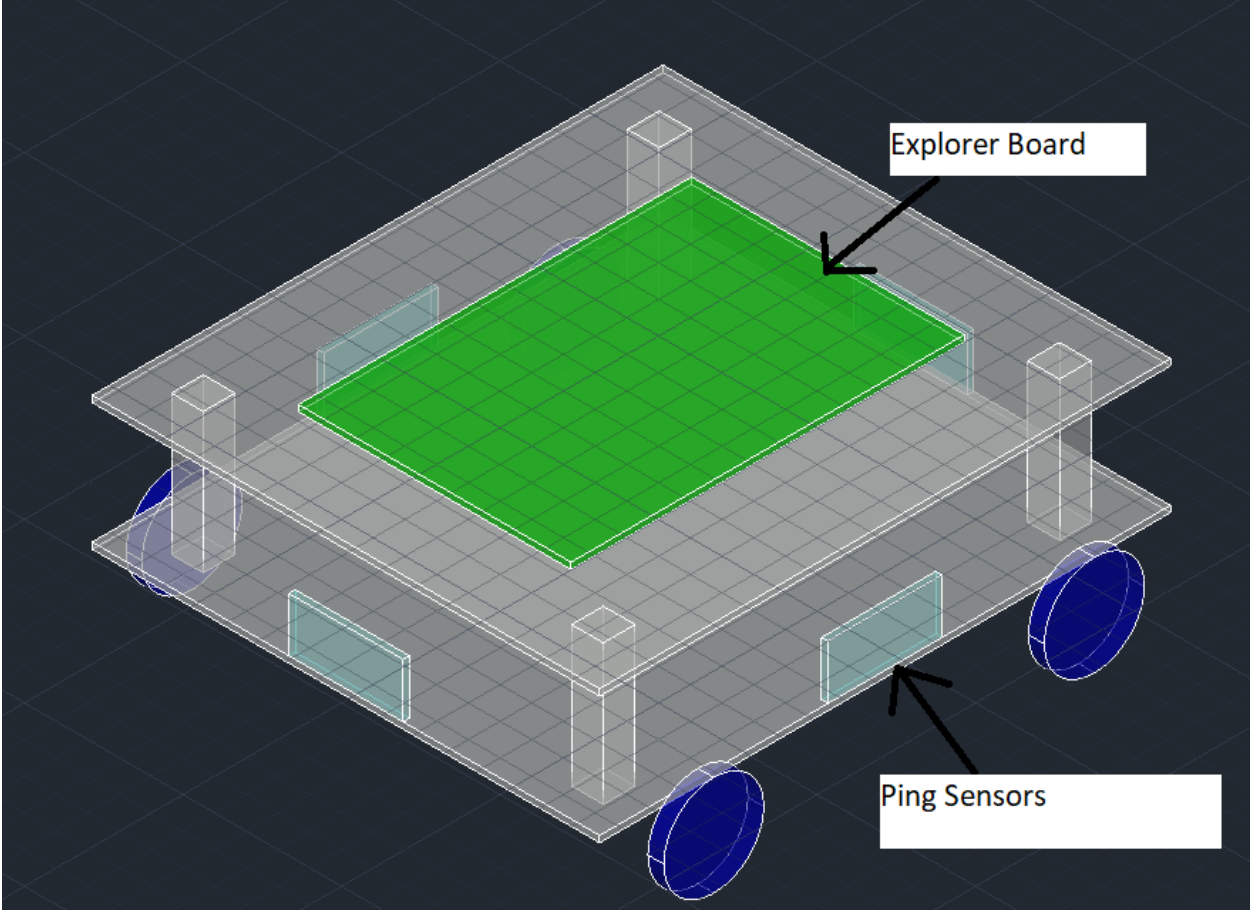


Figure 33 - Iso View of Robot

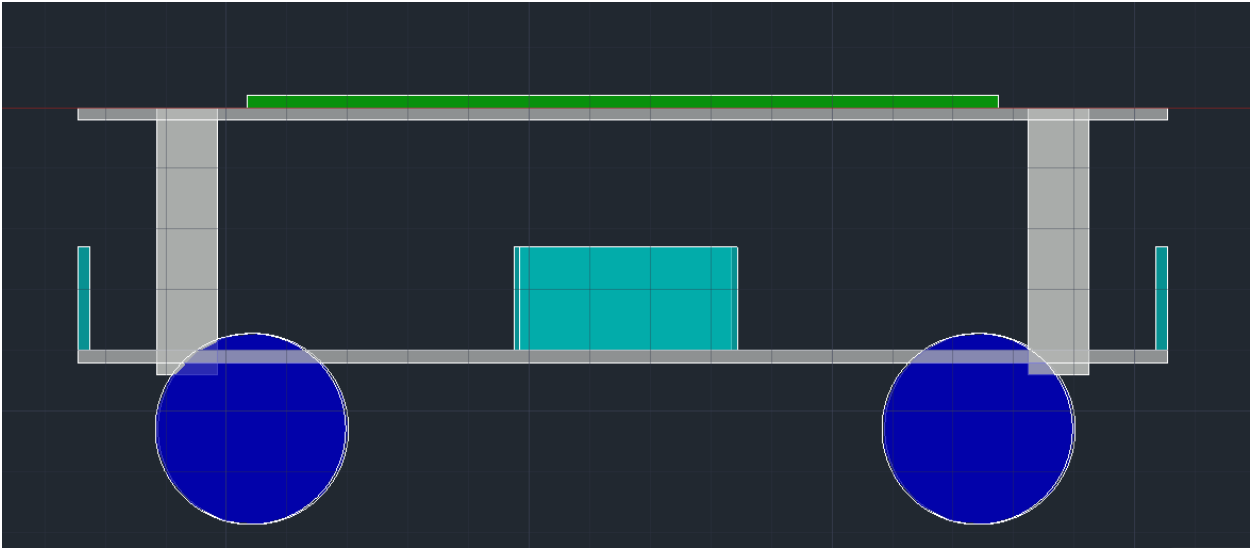


Figure 34 - Side View of Robot

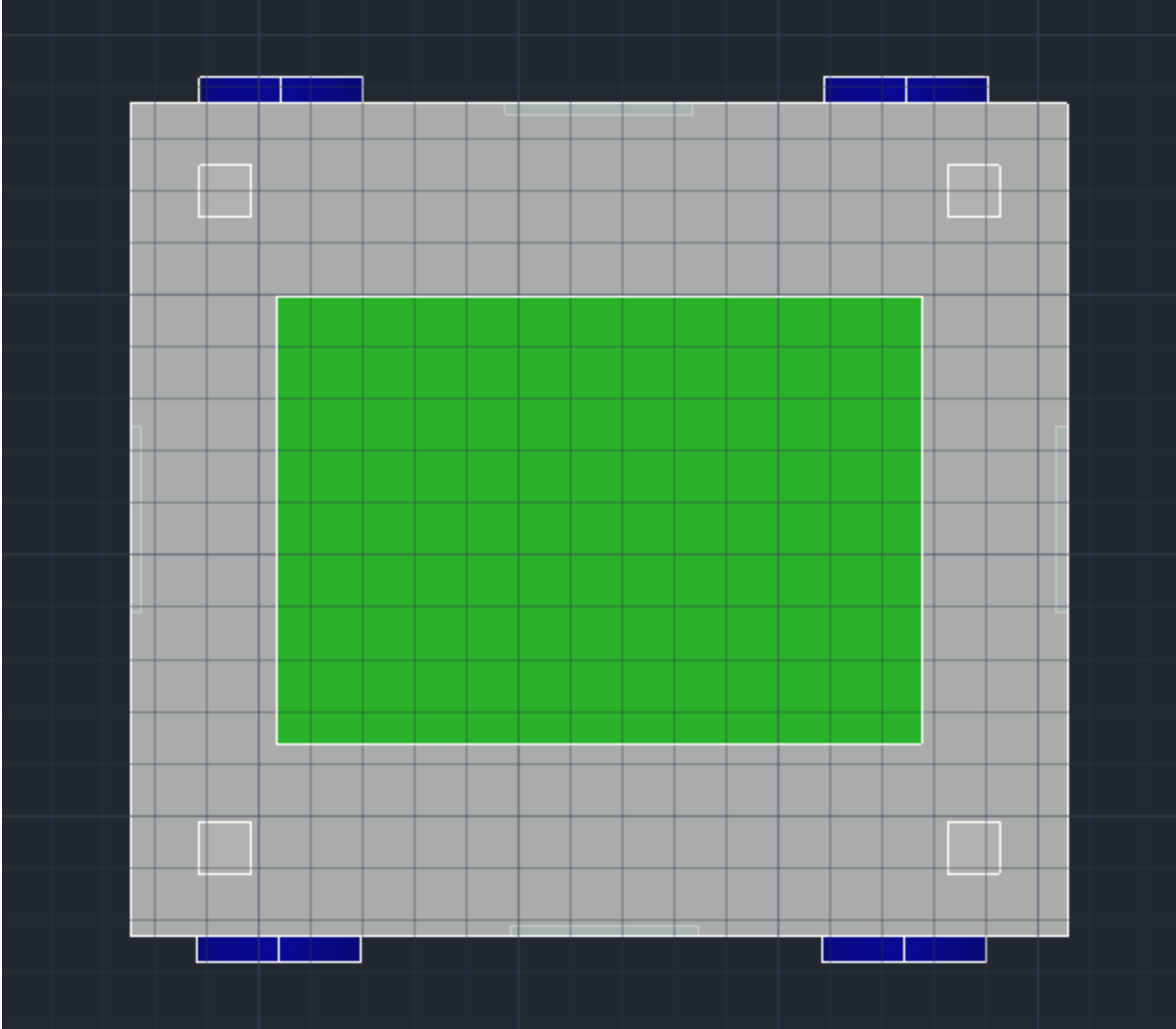


Figure 35 - Top View of Robot

7 Team Information – GB, MB, MH, BJ

Greg Blondheim: Computer Engineer, Project Leader

Marian Bonto: Electrical Engineer, Hardware Manager

Maria Hatzis: Electrical Engineer, Engineering Documentation Leader

Brett Jacobsen: Computer Engineer, Software Manager

8 Parts List - GB

This is the Final Design Parts List and budget information from our design.

Qty.	Refdes	Part Num.	Description
4	U1-U4	28015	ULTRASONIC SENSOR PING 40 KHZ
1	U5	AT-1127-ST-2-R	BUZZER MAGNETIC 3V 9MM TH
1	LS1	CMI-1295-0585T	BUZZER MAGNETIC 5V 12MM TH
1	U\$1	ROB-14450	TB6612FNG MOTOR DRIVER BOARD
6	U5-U10	MIKROE-2543	RN4870 Click Board
2	IC1, IC2	DM240001-3	EXPLORER 16/32 DSPIC/PIC24/PIC32
2	M1,M2	1583	GEARMOTOR 290 RPM 6V METAL
6	B1-B6	CR2032	BATTERY LITHIUM 3V COIN 20MM
1	C1	3628	ADAPT USB-A PLG TO TERM BLK 5POS
1	U11	LS10-13B12R3	AC/DC CONVERTER 12V 10W
1	U12	MCP73844-840/M	IC BATT CNTL LI-ION 1-2CEL 8MSOP

Figure 36 - Original Design Parts List

Qty.	Refdes	Part Num.	Description
4	U1-U4	28015	ULTRASONIC SENSOR PING 40 KHZ
1	U5	AT-1127-ST-2-R	BUZZER MAGNETIC 3V 9MM TH
2	LS1	CMI-1295-0585T	BUZZER MAGNETIC 5V 12MM TH
1	U1	ROB-14450	TB6612FNG MOTOR DRIVER BOARD
6	U5-U10	MIKROE-2543	RN4870 Click Board
2	IC1, IC2	DM240001-3	EXPLORER 16/32 DSPIC/PIC24/PIC32
2	M1,M2	1583	GEARMOTOR 290 RPM 6V METAL
6	B1-B6	CR2032	BATTERY LITHIUM 3V COIN 20MM
1	C1	3628	ADAPT USB-A PLG TO TERM BLK 5POS
1	U11	LS10-13B12R3	AC/DC CONVERTER 12V 10W
1	U12	MCP73844-840/M	IC BATT CNTL LI-HON 1-2CEL 8MSOP
2	IC4	PA0026	MSOP-8 to DIP-8 SMT Adapter
1	U13	MCP73844-840/M	IC BATT CNTL LI-HON 1-2CEL 8MSOP
1	IC4	PA0001	SOIC-8 to DIP-8 SMT Adapter
2	U14-U15	SK33-TP	
2	U16-U17	DMP2038USS-1	MOSFET P-CH 20V 6.5A 8SO
1	B7	ALITOVE DC 12V	ALITOVE DC 12V 5A Power Supply Adapter Converter Transformer AC 100-240V Input with 5.5x2.5mm DC Output Jack for 5050 3528 LED Strip Module Light
3	B8	LI18650JP2S1P	BATT LITH ION 7.2V 3.25AH
2	B9-B10	TEK012	BATT CHARGER COIN CELL
2	B11-B18	LIR2032	10PCS EEMB LIR2032 Rechargeable Battery 3.7V Lithium-ion Coin Button Cell Batteries 45mAh 2032 Rechargeable Battery
1	U18	ICM-20948	IMUs - Inertial Measurement Units World's Lowest Power 9-Axis MEMS MotionTracking Device
4	U1-U4	SEN-15569	Distance Sensor Development Tool HC-SR04 Ultrasonic Sonar Distance Sensor + 2 x 10K resistors
2	U19	BC337	T, NPN, GP, 45V, 800mA, TO-92
4	IC9	OP177	IC, OP-AMP, Ultra Precision
2	IC5-IC6	DIP300T600P08	8 pin 0.3 inch DIP to 0.6 inch DIP adapter.

Figure 37 - Final Design Parts List

	A	B	C	D	E
1				Unit	Total
2	Qty.	Part Num.	Description	Cost	Cost
3	4	28015	ULTRASONIC SENSOR PING 40 KHZ		
4	1	AT-1127-ST-2-R	BUZZER MAGNETIC 3V 9MM TH	0.80	0.80
5	2	CMI-1295-0585T	BUZZER MAGNETIC 5V 12MM TH	1.18	2.36
6	1	ROB-14450	TB6612FNG MOTOR DRIVER BOARD	5.95	5.95
7	6	MIKROE-2543	RN4870 Click Board		
8	2	DM240001-3	EXPLORER 16/32 DSPIC/PIC24/PIC32		
9	2	1583	GEARMOTOR 290 RPM 6V METAL	26.95	53.90
10	6	CR2032	BATTERY LITHIUM 3V COIN 20MM		
11	1	3628	ADAPT USB-A PLG TO TERM BLK 5POS	4.95	4.95
12	1	LS10-13B12R3	AC/DC CONVERTER 12V 10W	3.89	3.89
13	1	MCP73844-840	IC BATT CNTL LI-ION 1-2CEL 8MSOP	1.77	1.77
14	2	PA0026	MSOP-8 to DIP-8 SMT Adapter	3.49	6.98
15	1	MCP73844-840/M	IC BATT CNTL LI-ION 1-2CEL 8MSOP	1.77	1.77
16	1	PA0001	SOIC-8 to DIP-8 SMT Adapter	3.49	3.49
17	2	SK33-TP		0.37	0.74
18	2	DMP2038USS-13	MOSFET P-CH 20V 6.5A 8SO	0.48	0.96
19	1	ALITOVE DC 12V	ALITOVE DC 12V 5A Power Supply Adapter Converter Trans	11.99	11.99
20	3	LI18650JP2S1P+P	BATT LITH ION 7.2V 3.25AH	27.01	81.03
21	2	TEK012	BATT CHARGER COIN CELL	10.40	20.80
22	2	LIR2032	10PCS EEMB LIR2032 Rechargeable Battery 3.7V Lithium-ico	14.99	29.98
23	1	ICM-20948	IMUs - Inertial Measurement Units World's Lowest Power 9-Ax	11.21	11.21
24	4	SEN-15569	Distance Sensor Development Tool HC-SR04 Ultrasonic Sonar Distance Sensor + 2 x 10K resistors	\$3.95	15.80
25	2	BC337	T, NPN, GP, 45V, 800mA, TO-92		
26	4	OP177	IC, OP-AMP, Ultra Precision		
27	2	DIP300T600P08	8 pin 0.3 inch DIP to 0.6 inch DIP adapter.	1.69	3.38
28					
29					
30					
31				Total	\$261.75

Figure 39 - Final Design Budget List

9 Project Schedule – GB, MB, MH, BJ

Project Design	95 days	Mon 9/12/22	Fri 12/16/22		
Midterm Report	28.38 days	Mon 9/12/22	Mon 10/10/22		All
Cover page	25.38 days	Mon 9/12/22	Fri 10/7/22		Maria
T of C, L of T, L of F	25.38 days	Mon 9/12/22	Fri 10/7/22		Greg
Problem Statement	25.38 days	Mon 9/12/22	Fri 10/7/22		All
Need	25.38 days	Mon 9/12/22	Fri 10/7/22		Brett
Objective	25.38 days	Mon 9/12/22	Fri 10/7/22		Marian
Background	25.38 days	Mon 9/12/22	Fri 10/7/22		Greg
Marketing Requirements	25.38 days	Mon 9/12/22	Fri 10/7/22		Maria/Marian
Engineering Requirements Specification	25.38 days	Mon 9/12/22	Fri 10/7/22		Maria
Engineering Analysis	21.38 days	Mon 9/12/22	Mon 10/3/22		All
Circuits (DC, AC, Power, ...)	21.38 days	Mon 9/12/22	Mon 10/3/22		Greg/Marian
Electronics (analog and digital)	21.38 days	Mon 9/12/22	Mon 10/3/22		Greg/Maria
Signal Processing	21.38 days	Mon 9/12/22	Mon 10/3/22		Maria/Marian
Communications (analog and digital)	21.38 days	Mon 9/12/22	Mon 10/3/22		Marian/Maria
Electromechanics	21.38 days	Mon 9/12/22	Mon 10/3/22		Maria/Marian/Greg
Computer Networks	21.38 days	Mon 9/12/22	Mon 10/3/22		Brett/Greg
Embedded Systems	21.38 days	Mon 9/12/22	Mon 10/3/22		Brett/Maria/Greg
Accepted Technical Design	21.38 days	Mon 9/12/22	Mon 10/3/22		All
Hardware Design: Phase 1	21.38 days	Mon 9/12/22	Mon 10/3/22		All
Hardware Block Diagrams Levels 0 thru N (w/ FR tables)	21.38 days	Mon 9/12/22	Mon 10/3/22		All
Software Design: Phase 1	21.38 days	Mon 9/12/22	Mon 10/3/22		All
Software Behavior Models Levels 0 thru N (w/FR tables)	21.38 days	Mon 9/12/22	Mon 10/3/22		All
Mechanical Sketch	46 days	Mon 9/12/22	Fri 10/28/22		Marian/Greg
Team information	11.38 days	Wed 8/24/22	Sun 9/4/22		All
Project Schedules	19.38 days	Wed 8/24/22	Mon 9/12/22		All

Figure 40 - Midterm Project Schedule

ID	Task Name	Duration	Start	Finish
1	SDP2 Implementation 2023	106 days	Mon 1/9/23	Mon 4/24/23
2	Revise Gantt Chart	14 days	Mon 1/9/23	Sun 1/22/23
3	Implement Project Design	92 days	Mon 1/9/23	Mon 4/10/23
4	Hardware Implementation	43 days	Mon 1/9/23	Tue 2/21/23
5	Breadboard Components	29 days	Mon 1/9/23	Mon 2/6/23
6	Power System	14 days	Tue 1/24/23	Mon 2/6/23
7	Motors and Controller	14 days	Mon 1/9/23	Sun 1/22/23
8	Distance Sensors	14 days	Mon 1/9/23	Sun 1/22/23
9	Alarm System	14 days	Mon 1/9/23	Sun 1/22/23
10	Bluetooth Beacons	14 days	Mon 1/9/23	Sun 1/22/23
11	Robot Embedded Controller Connections	14 days	Tue 1/24/23	Mon 2/6/23
12	Layout and Generate PCB(s)	29 days	Mon 1/9/23	Mon 2/6/23
13	Power System	14 days	Tue 1/24/23	Mon 2/6/23
14	Motors and Controller	14 days	Mon 1/9/23	Sun 1/22/23
15	Distance Sensors	14 days	Mon 1/9/23	Sun 1/22/23
16	Alarm System	14 days	Mon 1/9/23	Sun 1/22/23
17	Bluetooth Beacons	14 days	Mon 1/9/23	Sun 1/22/23
18	Robot Embedded Controller Connections	14 days	Tue 1/24/23	Mon 2/6/23
19	Assemble Hardware	22 days	Mon 1/23/23	Mon 2/13/23
20	Power System	7 days	Tue 2/7/23	Mon 2/13/23
21	Motors and Controller	14 days	Mon 1/23/23	Sun 2/5/23
22	Distance Sensors	14 days	Mon 1/23/23	Sun 2/5/23
23	Alarm System	14 days	Mon 1/23/23	Sun 2/5/23
24	Bluetooth Beacons	14 days	Mon 1/23/23	Sun 2/5/23
25	Robot Embedded Controller Connections	7 days	Tue 2/7/23	Mon 2/13/23
26	Test Hardware	16 days	Mon 2/6/23	Tue 2/21/23
27	Power System	7 days	Mon 2/13/23	Sun 2/19/23
28	Motors and Controller	7 days	Mon 2/6/23	Sun 2/12/23
29	Distance Sensors	7 days	Mon 2/6/23	Sun 2/12/23
30	Alarm System	7 days	Mon 2/6/23	Sun 2/12/23
31	Bluetooth Beacons	7 days	Mon 2/6/23	Sun 2/12/23
32	Robot Embedded Controller Connections	7 days	Mon 2/13/23	Sun 2/19/23
33	Revise Hardware	9 days	Mon 2/13/23	Tue 2/21/23
34	Power System	1 day	Tue 2/21/23	Tue 2/21/23
35	Motors and Controller	9 days	Mon 2/13/23	Tue 2/21/23
36	Distance Sensors	9 days	Mon 2/13/23	Tue 2/21/23
37	Alarm System	9 days	Mon 2/13/23	Tue 2/21/23
38	Bluetooth Beacons	9 days	Mon 2/13/23	Tue 2/21/23
39	Robot Embedded Controller Connections	1 day	Tue 2/21/23	Tue 2/21/23

Figure 41 - Final Project Schedule Part 1

ID	Task Name	Duration	Start	Finish
40	MIDTERM: Demonstrate Hardware Subsystem	0 days	Tue 2/21/23	Tue 2/21/23
41	Software Implementation	43 days	Mon 1/9/23	Tue 2/21/23
42	Develop Software	35 days	Mon 1/9/23	Sun 2/12/23
43	Reading RSSI	28 days	Mon 1/16/23	Sun 2/12/23
44	Bluetooth Distance measurements	28 days	Mon 1/16/23	Sun 2/12/23
45	Check Surroundings for Obstacles	28 days	Mon 1/16/23	Sun 2/12/23
46	Determine Movement	28 days	Mon 1/16/23	Sun 2/12/23
47	Motor Control	28 days	Mon 1/16/23	Sun 2/12/23
48	Test Software	35 days	Mon 1/9/23	Sun 2/12/23
49	Reading RSSI	28 days	Mon 1/16/23	Sun 2/12/23
50	Bluetooth Distance measurements	28 days	Mon 1/16/23	Sun 2/12/23
51	Check Surroundings for Obstacles	28 days	Mon 1/16/23	Sun 2/12/23
52	Determine Movement	28 days	Mon 1/16/23	Sun 2/12/23
53	Motor Control	28 days	Mon 1/16/23	Sun 2/12/23
54	Revise Software	7 days	Mon 2/13/23	Sun 2/19/23
55	Reading RSSI	7 days	Mon 2/13/23	Sun 2/19/23
56	Bluetooth Distance measurements	7 days	Mon 2/13/23	Sun 2/19/23
57	Check Surroundings for Obstacles	7 days	Mon 2/13/23	Sun 2/19/23
58	Determine Movement	7 days	Mon 2/13/23	Sun 2/19/23
59	Motor Control	7 days	Mon 2/13/23	Sun 2/19/23
60	MIDTERM: Demonstrate Software Subsystem	0 days	Tue 2/21/23	Tue 2/21/23
61	System Integration	49 days	Tue 2/21/23	Mon 4/10/23
62	Assemble Complete System Integration	14 days	Tue 2/21/23	Mon 3/6/23
63	Navigation(Motor & Ping Sensors)	14 days	Tue 2/21/23	Mon 3/6/23
64	Bluetooth Beacons	14 days	Tue 2/21/23	Mon 3/6/23
65	Robot Chassis	14 days	Tue 2/21/23	Mon 3/6/23
66	Alarm System	14 days	Tue 2/21/23	Mon 3/6/23
67	Test Complete System Integration	7 days	Tue 3/7/23	Mon 3/13/23
68	Navigation(Motor & Ping Sensors)	7 days	Tue 3/7/23	Mon 3/13/23
69	Bluetooth Beacons	7 days	Tue 3/7/23	Mon 3/13/23
70	Robot Chassis	7 days	Tue 3/7/23	Mon 3/13/23
71	Alarm System	7 days	Tue 3/7/23	Mon 3/13/23
72	Revise Complete System Integration	24 days	Tue 3/14/23	Thu 4/6/23
73	Navigation(Motor & Ping Sensors)	24 days	Tue 3/14/23	Thu 4/6/23
74	Bluetooth Beacons	24 days	Tue 3/14/23	Thu 4/6/23
75	Robot Chassis	24 days	Tue 3/14/23	Thu 4/6/23
76	Alarm System	24 days	Tue 3/14/23	Thu 4/6/23
77	Preliminary Demonstration of Complete System	4 days	Fri 4/7/23	Mon 4/10/23
78	Develop Final Report	106 days	Mon 1/9/23	Mon 4/24/23
79	Write Final Report	106 days	Mon 1/9/23	Mon 4/24/23
80	Submit Final Report	0 days	Tue 4/25/23	Tue 4/25/23

Figure 42 - Final Project Schedule Part 2

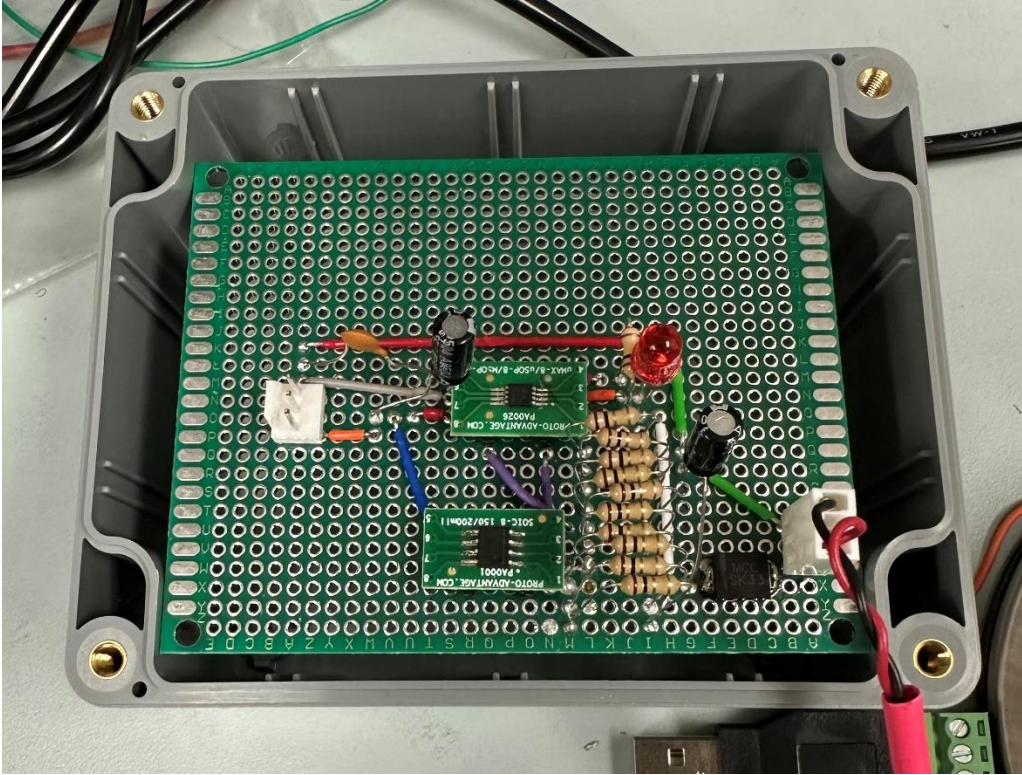


Figure 43 - Complete Charging Circuit

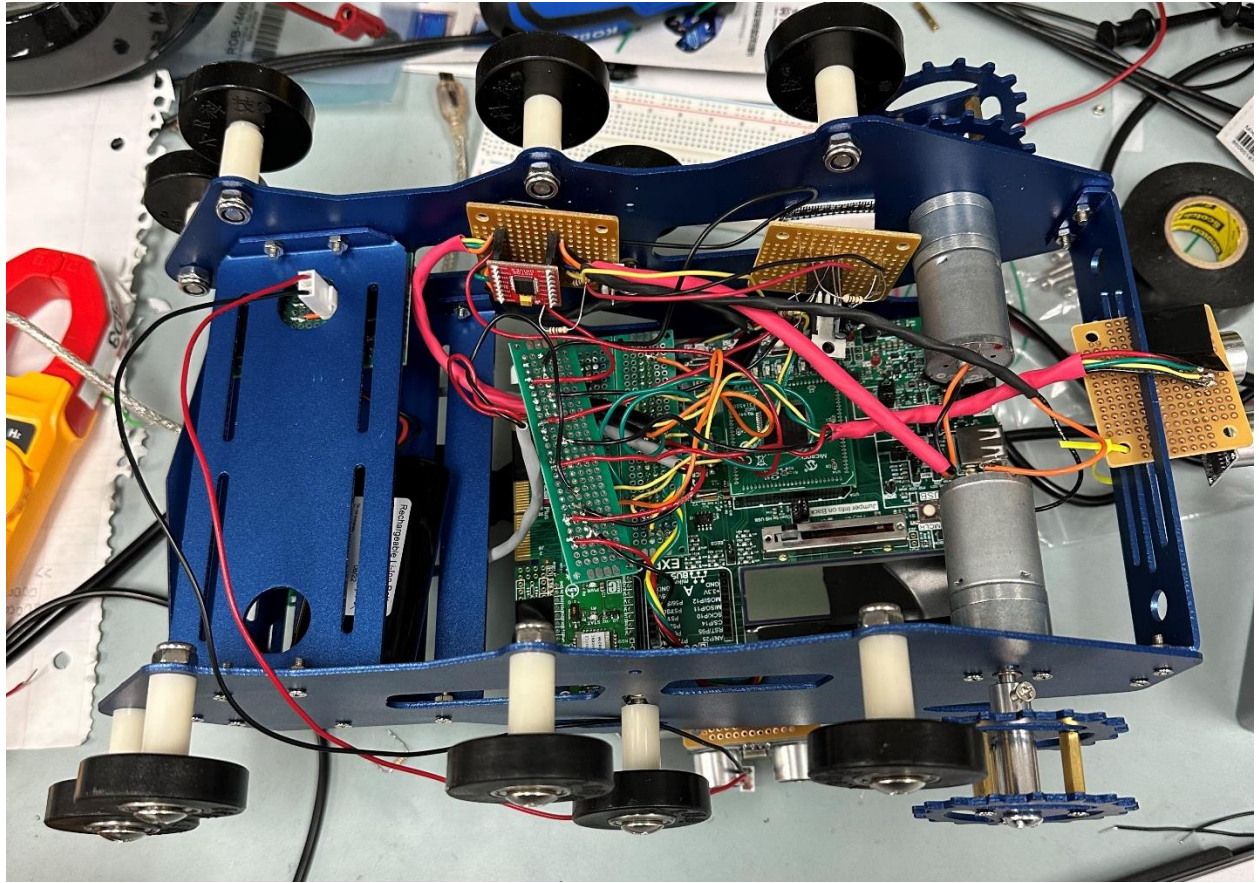


Figure 44- Internal Hardware Integration Inside Robot

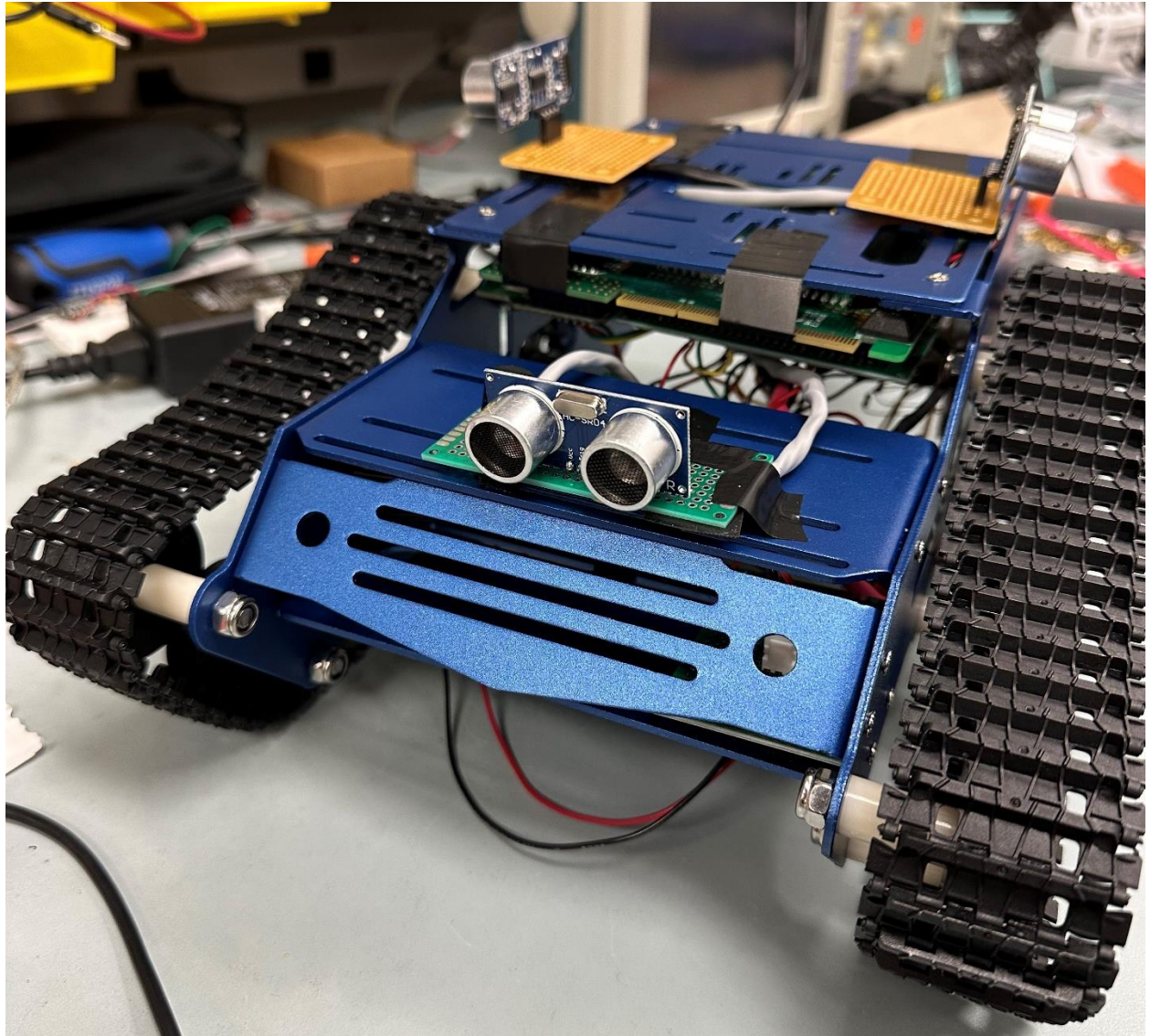


Figure 45 - Front View of Robot with Front, Left, and Right Object Detection Sensors

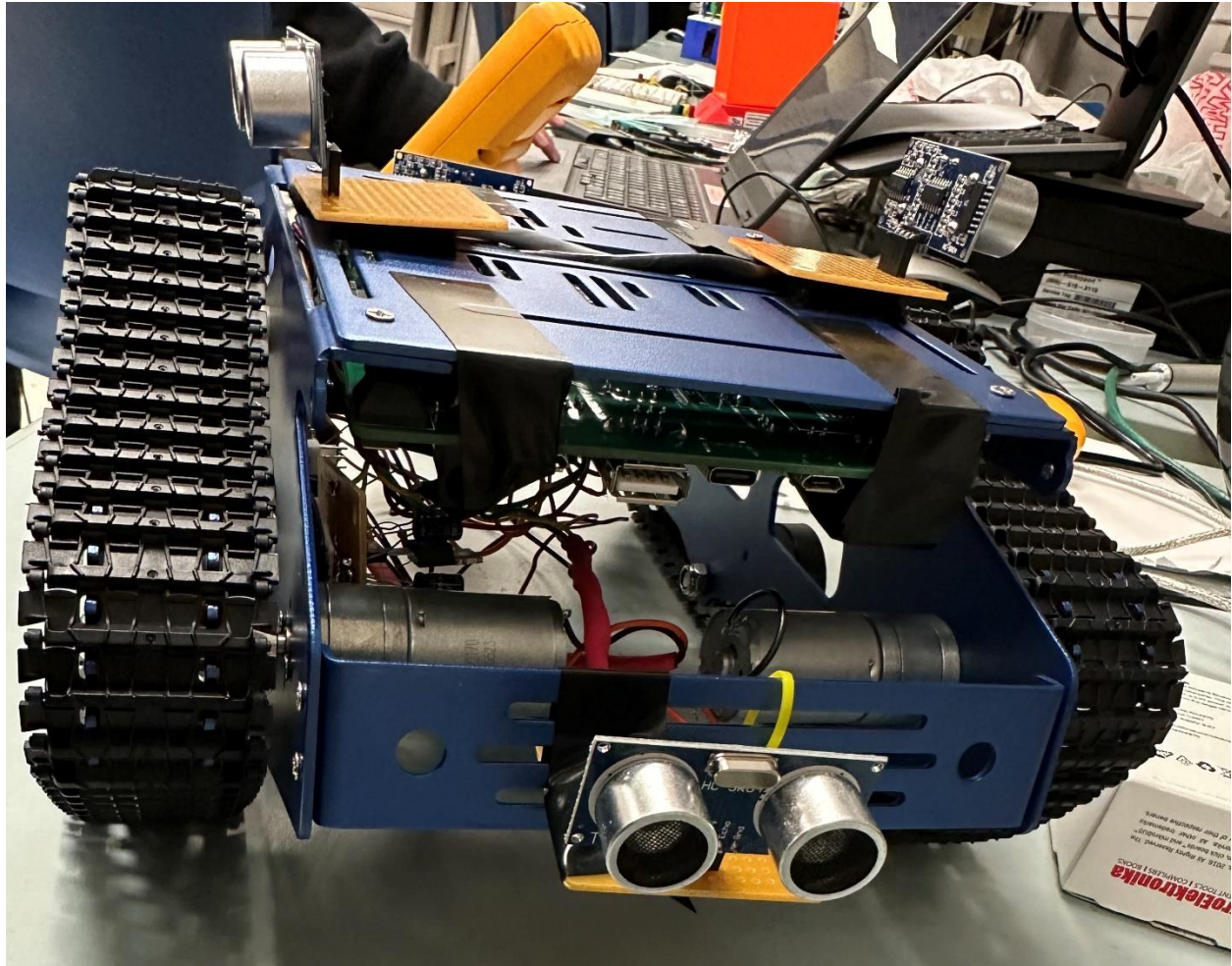


Figure 46 - Rear View of Robot Including Motor System and Rear Directional Sensor

10 Conclusions and Recommendations GB BJ

A key problem that remains with the project is its lack of direction-finding. While the robot is capable of calculating an angle between itself and its target, it is currently unable to accurately find the direction it's currently facing. In its current state, this could be solved in two different ways. One method would be a form of "direction memory". This could be accomplished by knowing the approximate rotational movement with each iteration of a motor control function and update an angle variable accordingly with each iteration of the function, essentially allowing the robot to "document" its current direction each time it decides which movement procedure to act on, having a rough idea of which direction it's facing at all times. However, a key issue with this solution would be the introduction of angular error. Differences in terrain could cause the robot to have slightly different angular movement in different environments, causing its assumed direction to be inaccurate. While this would not be too problematic over a short period of time, the angular error could eventually accumulate, resulting in the robot moving in a different direction than the target is actually while assuming it is moving in the correct direction, effectively moving off in the wrong direction indefinitely without ever approaching its target. With more resources, this could be prevented with a better understanding of movement AI and allowing the robot to automatically calibrate its direction if it finds it is not approaching its target. A much simpler, albeit rougher, solution to the lack of direction-finding could be 90-degree movement. With this implementation, the robot would simply move in 90-degree increments, only being able to move in directions it designates as north, south, east, and west. While the movement would be far rougher, it would be far simpler to implement as the only thing designers would need to calculate would be the amount of time the motors take to rotate 90 degrees. Angular

error would remain an issue but, unlike with angular movement and direction memory, it would not accumulate as much due to an inaccurate assumption of 90 degrees working both ways. For example: if the robot rotates 88 degrees and assumes it has rotated 90 degrees, turning back the same amount would result in the angle remaining at zero, meaning the robot is still assuming the correct direction.

If new hardware is an option, true direction-finding could be implemented through a dedicated part, such as a gyroscope. One such part that was found during research and obtained was the ICM-20948 Low Power 9-Axis MEMS MotionTracking Device. This part contains a built-in gyroscope with 3-axis data being written to internal registers that can be read by an external device through the I²C or SPI protocols. With the angular rate data received from this device, much more accurate and less error-prone direction-finding could be implemented, causing designers to no longer need to worry about things such as directional memory, 90-degree movement, or manually calculating the angular rates of configurable motor movement.

Another recommendation is to use a better microcontroller and Bluetooth module that is more user friendly and documented than what was provided and used for our prototype. There are many issues when using microchip products that have been found during this year of design. With better Bluetooth modules the beacons could connect and not have difficulty reading simple values.

In conclusion, this project could be more readily possible with a larger group of engineers and computer scientists and access to better technology. The process could be streamlined and more ideas could be added such as use around the whole house at once and use outside in areas such as a pool or backyard.

11 References – GB, MB, MH, BJ

- [1] Irobot Corp, "Mobile robot providing environmental mapping for household environmental control", US10391638B2, 2019.
- [2] K. Ema, M. Yokoyama, T. Nakamoto and T. Moriizumi, "Odour-sensing system using a quartz-resonator sensor array and neural-network pattern recognition", *Sensors and Actuators*, vol. 18, no. 3-4, pp. 291-296, 1989.
- [3] A. Flynn, R. Brooks, W. Wells and D. Barrett, "Intelligence for Miniature Robots". *Sensors and Actuators*, vol. 20, pp. 187-196, 1989.
- [4] Y. Pang, H. Ding, J. Liu, Y. Fang and S. Chen, "A UHF RFID-Based System for Children Tracking", *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5055-5064, 2018.
- [5] "Ultrasonic Ranging Module HC-SR04", *ElecFreaks*, [Online]. Available: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf> [Accessed 8 November 2022].
- [6] Rehman, Masood, Baharudin, Zuhairi, Nallagownden, Perumai, Islam, Badar, « Design and Analysis of Resonant Wireless Power Transfer System », *Research Gate*, DOI : [10.1051/mateconf/201822502020](https://doi.org/10.1051/mateconf/201822502020)
- [7] P. Regtien, "Sensor Systems for Robot Control", *Sensors and Actuators*, vol. 17, pp. 91-101, 1989.

- [8] Shetty, Anitha, "Infrared Sensor-How it Works, Types, Applications, Advantage & Disadvantage", *Electrical & Electronics Encyclopedia* [Online]. Available: <https://electricalfundablog.com/infrared-sensor/#:~:text=What%20is%20Infrared%20Sensor%20IR%20sensor%20is%20a,of%20its%20features%20are%20heat%20and%20motion%20sensing> [Accessed 8 October 2022].
- [9]Y. Ye, C. Zhang, C. He, X. Wang, J. Huang and J. Deng, "A Review on Applications of Capacitive Displacement Sensing for Capacitive Proximity Sensor", *IEEE Access*, vol. 8, pp. 45325-45342, 2020.
- [10]Beijing Wanyu Youtu Environmental Protection Technology Co., Ltd, "Household smart Roomba with air purification function", CN104921658B, 2017.
- [11]A. Bartlett, "Child tracking device", US9747770B1, 2017.
- [12]"Arduino Text to Speech", *Arduino Project Hub*, 2019. [Online]. Available: <https://create.arduino.cc/projecthub/helloanimesh390/talking-arduino-arduino-text-to-speech-c31722>. [Accessed: 15- Mar- 2022].
- [13] "PIC24FJ128GA010 FAMILY", *Microchip*, 2012. [Online]. Available: <https://www.microchip.com/en-us/product/PIC24FJ128GA010> [Accessed: 1-Sep-2022]
- [14] "Jauch Battery Solutions." [Online]. Available: <https://www.jauch.com/downloadfile/61f114f59a2ba243e32fa0846647a765b/3350%20mAh,%20LI18650JLS%20HB%20s1p,%20Mat.%20250481.pdf>. [Accessed: 7-Nov-2022].

[15] "MCP73841/2/3/4 - Microchip Technology." [Online]. Available:

<https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21823D.pdf>. [Accessed: 14-Nov-2022].

[16] "CMI-1295-0585T- DigiKey" Available:

<https://www.cuidevices.com/product/resource/cmi-1295-0585t.pdf> [Accessed: 23-April-2023]