ROBO SNACKS
SNACK DELIVERY ROBOTS

March 13, 2022

Dr. Michael Hegedus
School of Engineering
Simon Fraser University
8888 University Drive
British Columbia, V5A 1S6

RE: ENSC 405/440 Design Specification for Snack Bot O7

Dear Dr. Hegedus,

This design specifications document for SnackBot O7 was prepared by Robo Snacks Company 7 for our Capstone courses.

Our goal is to autonomously deliver snacks to conference rooms, networking events and presentations happening on university campuses and offices before an event begins. Snack Bot O7 is targeted towards catering services that deliver to such events and locations.

The attached Design Specifications document will describe the various subsystems of the Snack Bot O7. These include schematics, rough diagrams, block diagrams and background theory for the hardware and software for movement of the robot, its perception system, and the associated user interface. The document will also describe how these subsystems are integrated and interact with each other.

We would like to thank you in advance for taking your time to review this document. If you have any questions, please email us at skd24@sfu.ca.

Sincerely,

Sirpreet

Sirpreet Kaur Dhillon
CEO
Robo Snacks Company 7 (RSC-7)

Design Specifications

Snack Bot O7

Presented by:

Robo Snacks Company 7

ROBO SNACKS
SNACK DELIVERY ROBOTS

Company No. 7

Date: March 13th, 2022

Authors (Team Members)		Roles / Affiliations
Full Name	**Email**	
Favour Amah-Nnachi	famahnna@sfu.ca	Firmware Lead
Sirpreet Kaur Dhillon	skd24@sfu.ca	Automation Team, and Structure Team
Emmanuel Komolafe	ikomolaf@sfu.ca	Systems Lead, and Automation Team
Veronica Lund	vlund@sfu.ca	Automation Team
Robert Smyczynski	rsmyczyn@sfu.ca	Electronics, and Structure Lead
Eddie Zheng	eza7@sfu.ca	Automation Lead

Abstract

Presentation rooms, board rooms and Conference halls are pre-stocked with snacks during an event by the catering services. This process can be automated to save time and resources of several catering companies that operate in offices and university campuses. The Snack Bot O7 uses indoor localization and navigation to deliver snacks to conference and presentation rooms during off hours.

This document aims to outline all hardware and software design specifications and will serve as a reference for the design team and when assessing the project’s success and whether the project’s goals have been met.

Table of Contents

List of Figures

List of Tables

Introduction

This document highlights the design specifications and design decisions for the Snack Bot O7 robot. Snack Bot O7 is a self-driving robot that is pre-stocked with snacks and drinks for delivery to the requested locations by the customers, such as conference rooms or a study hall. The robot operates during off hours to mitigate the risk of running into people and having difficulty maneuvering through crowds. These situations can hinder the robot from navigating to the correct location or increasing the amount of time it takes to reach the customers.

The objective for this robot is to autonomously deliver snacks to a user in a meeting room in a safe and efficient way, thereby providing a convenient catering service for the customers. To achieve this goal, the Snack Bot robot would need to scan its environment for objects, then navigate through these objects in the environment to a predefined destination. The Snack Bot O7 would utilize the integration of LIDAR (Light Detection and Radiation) technology, ROS NAV and a Raspberry PI 4 which would be discussed in the following sections in greater detail.

Taking into consideration the feedback received, we have made the following changes to our initial project,

- Redefined targeted customers to catering service companies.
- Renamed the robot from RSC-07 A1 to Snack Bot O7 for readability.
- Chassis design structure changed from a 3 omni wheeled to a design with 2 castors and 2 driving wheels for better stability.

These changes have been incorporated into these design specifications.

Structure & Mechanics

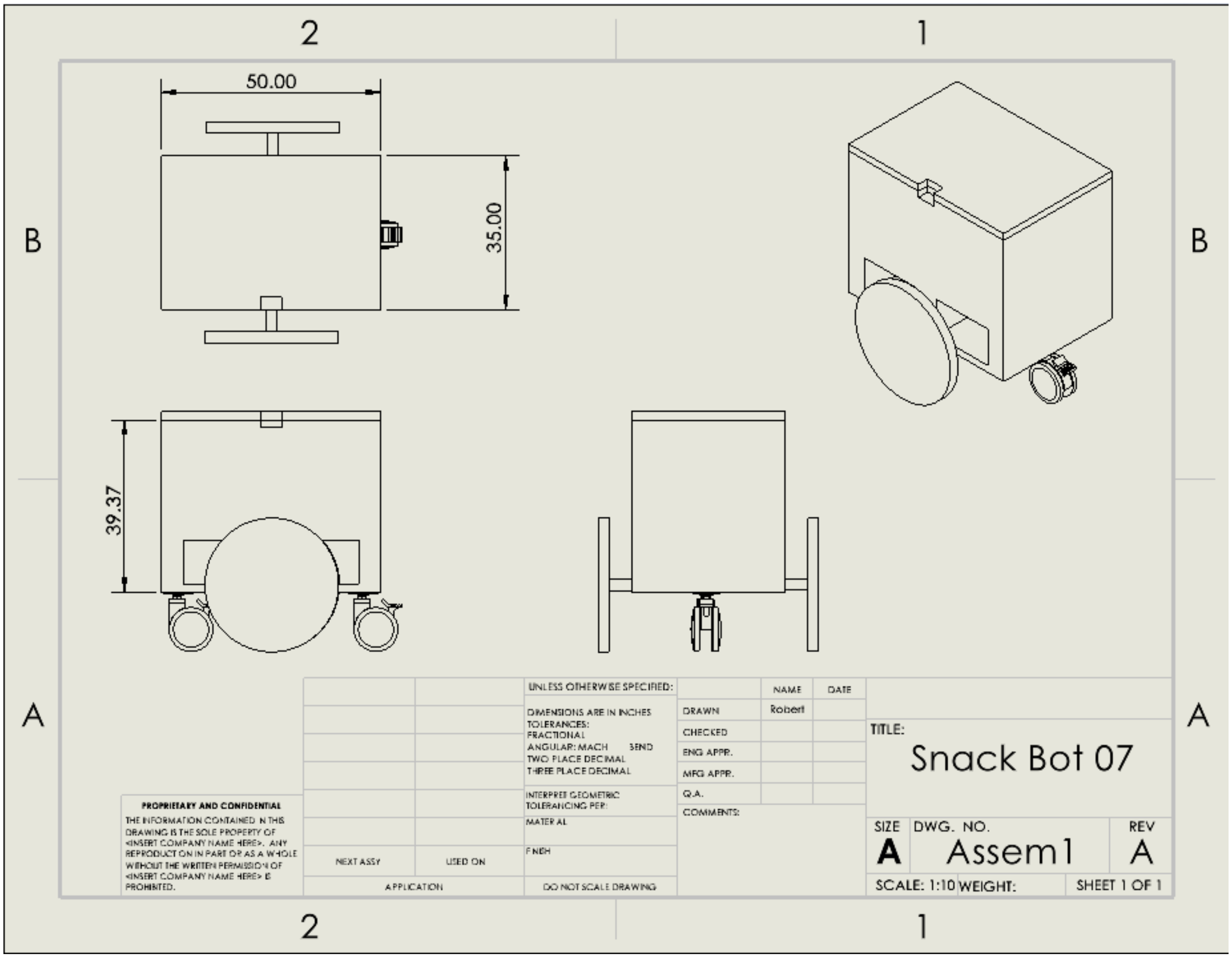


Figure 1: Schematic of Snack Bot O7’s structure

Structure is taken in careful consideration of our design, as it has to meet the requirements for safety and protection of the internal components while also maintaining full functionality. Snack Bot O7's objective is to provide a product that will be able to last years of wear and tear for moderate to heavy usage. Catering companies should be able to open up the product and use the robot while not having to worry about maintenance.

In the figure 1 above, the schematic of the overall structure is provided. The structure will be constructed from two main components: a chassis and a body. The chassis will be built from two rectangular plywood pieces of ¾” thickness that are 50 centimeters in length and 35 centimeters in width. They will be placed on top of each other using wooden beams as support, with about 3” of air gap. In between the two pieces, the essential components of the robot will be placed. This includes the electronics, microcontrollers, battery, motors and other smaller parts. In the provided drawings, this is seen as two rectangular holes in the bottom of the structure. During the initial phases, this will be left uncovered to provide easy access for troubleshooting the electrical components. However, in later stages, there will be a thin cover screwed onto the sides to protect both the robot from the outside environment and the user from any harm.

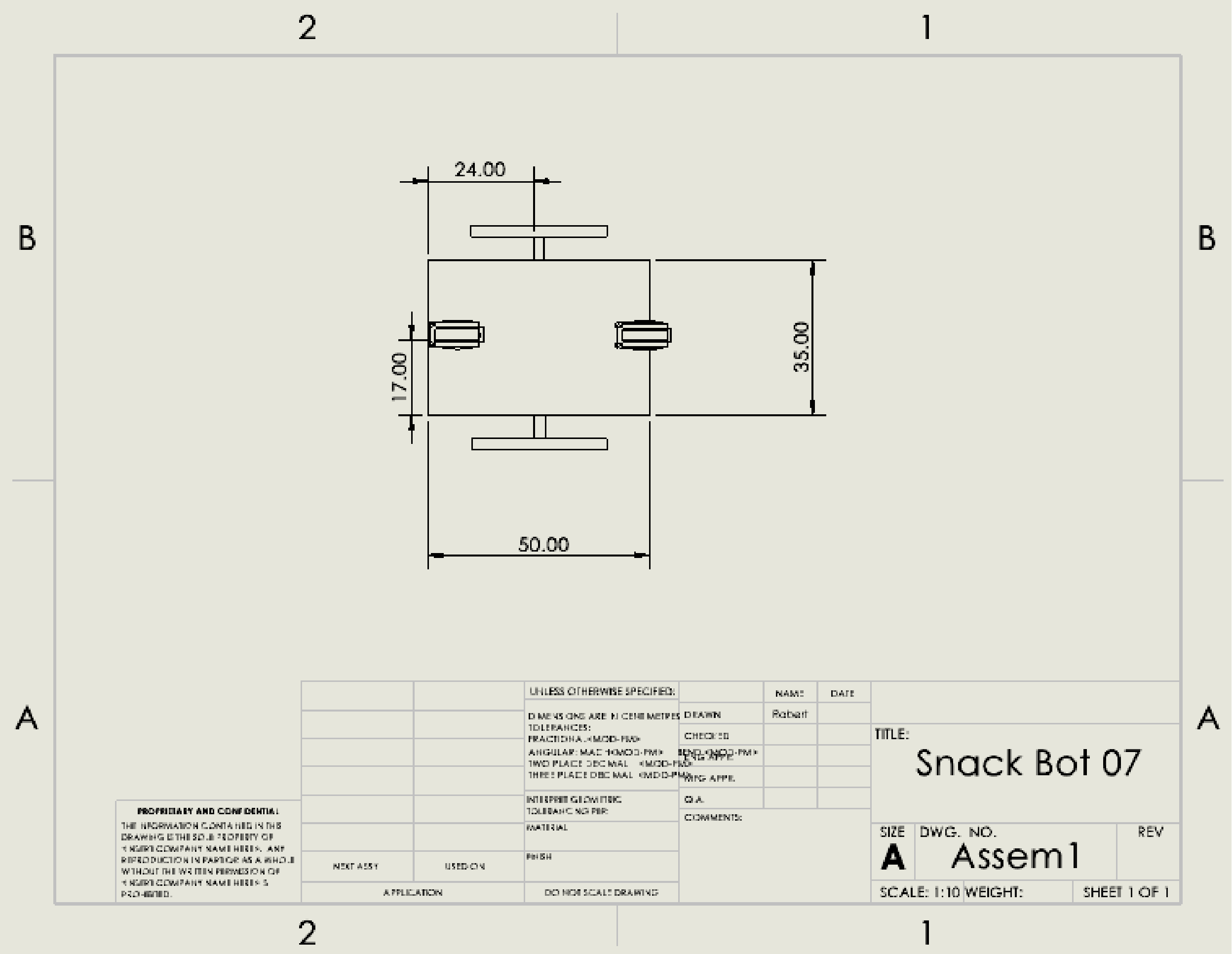

2
1
B
B
A
A
24.00
17.00
35.00
50.00
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN CENTIMETRES
TOLERANCES:
FRACTIONAL <MOD-PM>
ANGULAR: MAC <MOD-PM> BEND <MOD-PM>
TWO PLACE DECIMAL <MOD-PM>
THREE PLACE DECIMAL <MOD-PM>
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
Robert
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
Snack Bot 07
SIZE
A
DWG. NO.
Assem1
REV
SCALE: 1:10
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

Figure 2: Snack Bot O7’s underside to show wheel positioning

In the figure 2 above, the bottom side of the robot is shown to indicate positioning of the wheels. The configuration includes two driving wheels, placed on the top and bottom of the structure that will be powered by motors and provide the torque necessary to move the robot. On the left and right of the chassis, casters will be screwed underneath. The main purpose is to keep the structure stable during movement. A configuration of four casters were considered as well, for optimal balance and may be reconsidered after further testing. The hub motor wheels need to be attached to a fixed end onto the chassis, therefore it will be coupled onto another piece of wood. It may be viewed in the previous figures as the structural piece separating the storage for electrical components.

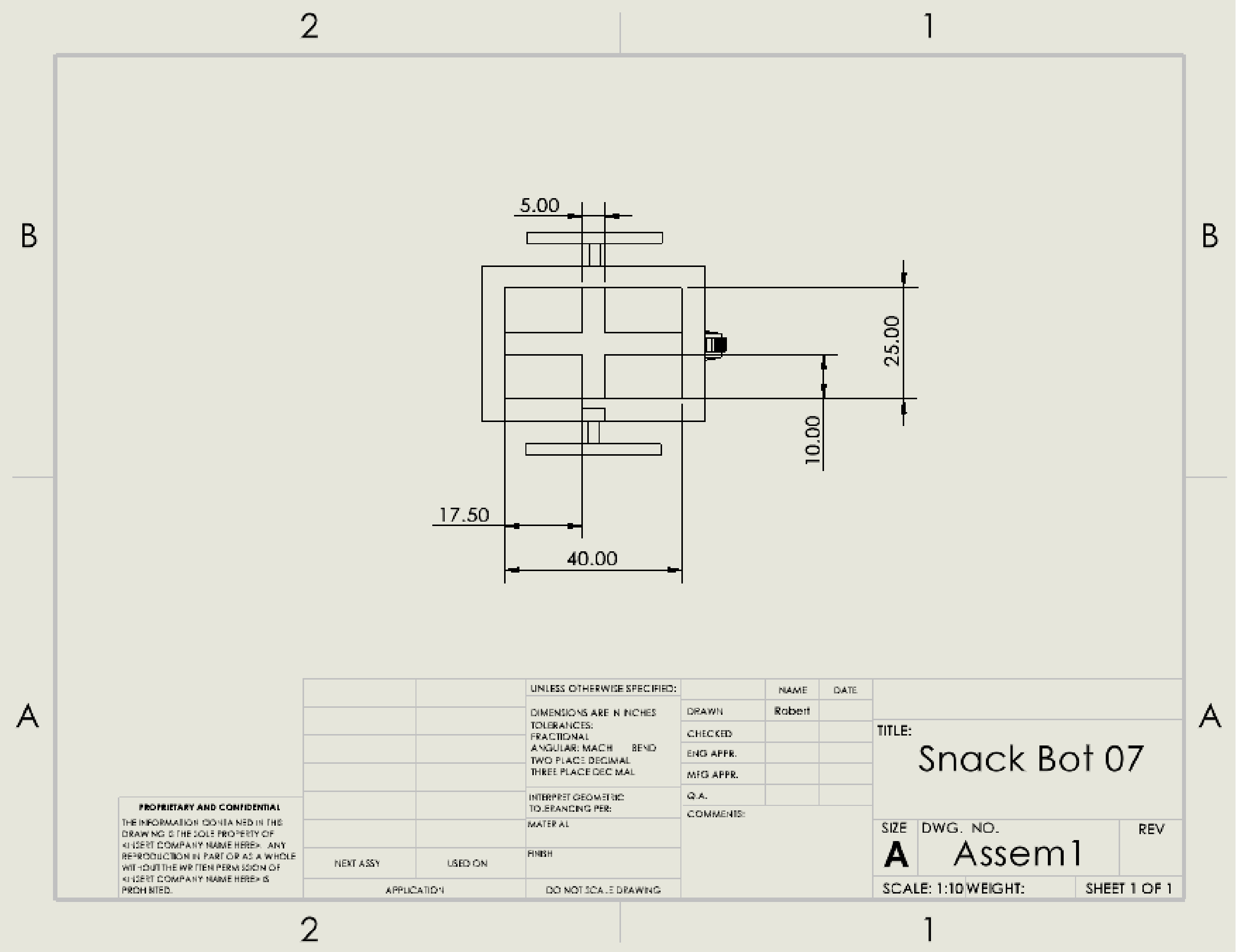
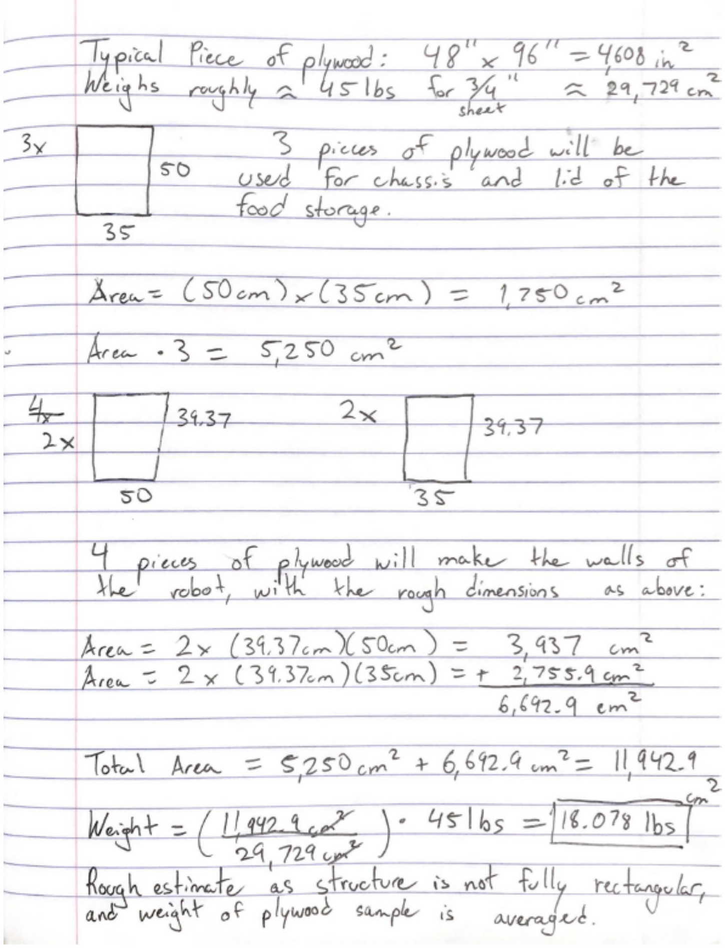


Figure 3: Snack Bot O7’s top view

The top portion of the structure (Figure 3) will hold the catering company’s food items. The shape will follow that of the chassis, however it will need to be enclosed with a lid to protect from outside elements. The lid will include an electromagnetic lock. As seen in the figure above, the inner hold is divided into four sections. From a practical standpoint, each section could hold a different variety of snacks/drinks and prevent the softer food items from being crushed by the heavier drinks.

Weight Design Requirements



Typical Piece of plywood: 48" × 96" = 4,608 in^2
Weighs roughly ≈ 45 lbs for 3/4" sheet ≈ 29,729 cm^2

3x
50
35

3 pieces of plywood will be used for chassis and lid of the food storage.

$Area = (50cm) \times (35cm) = 1,750\ cm^2$

$Area \cdot 3 = 5,250\ cm^2$

~~4x~~ 2x
39.37
50

2x
39.37
35

4 pieces of plywood will make the walls of the robot, with the rough dimensions as above:

$Area = 2\times(39.37cm)(50cm) = 3,937\ cm^2$
$Area = 2\times(39.37cm)(35cm) = +\ 2,755.9\ cm^2$
$6,692.9\ cm^2$

$Total\ Area = 5,250\ cm^2 + 6,692.9\ cm^2 = 11,942.9\ cm^2$

$Weight = \left(\frac{11,942.9\ cm^2}{29,729\ cm^2}\right) \cdot 45\ lbs = \boxed{18.078\ lbs}$

Rough estimate as structure is not fully rectangular, and weight of plywood sample is averaged.

Figure 4: Weight calculations

For Phase 1, the material used will be plywood as it provides a sturdy structure while still being able to make changes to prototype rapidly. However, for further phases of design, the material will be readjusted to a form of plastic. As seen from the calculations above (Figure 4), the rough estimate of the weight of the structure will be around 18 pounds.

Design ID	**Design Specification Requirements**	**Corresponding Requirements ID**
Des 1.1 A	Structure will be made of dense materials, such as wood or plastic, that will be	Req. 7.1: *Chassis must be able to support up to 50 lbs of weight*
Des 1.2 B	Wooden structure can withstand impact, further testing can be implemented	Req. 7.2: *Initial structure must be rigid enough to survive impact with hard objects*
Des 1.3 B, C	For future stages of development, the wooden structure will be converted to plastic.	Req. 7.4: *Final design must include a combination of plastic and metal for the outer structure*
Des 1.4 B, C	Structure was sized to weigh about 18 pounds. Motors and electrical components weight is minimal, should be less than 20 lbs.	Req. 7.5: *Device should not exceed 100 lbs in weight*
Des 1.5 C	The food storage structure will be separated from the chassis to avoid possible leaks/spillage.	Req. 7.10: *For waterproofing the final product, a box type of lockable structure will be secured onto the final chassis*

Table 1: Structure and Mechanical Design Specifications

System Design

Design Overview

The Snackbot System can be divided into four main system:

1. Perception System,
2. Robot Movement System,
3. Locking System (or mechanism), and
4. Power Management System

The below figure 5 shows the system block diagram of Snack Bot O7. All the different systems are labeled and the connections and feedback shown with arrows. These systems are discussed in detail in the following sections.

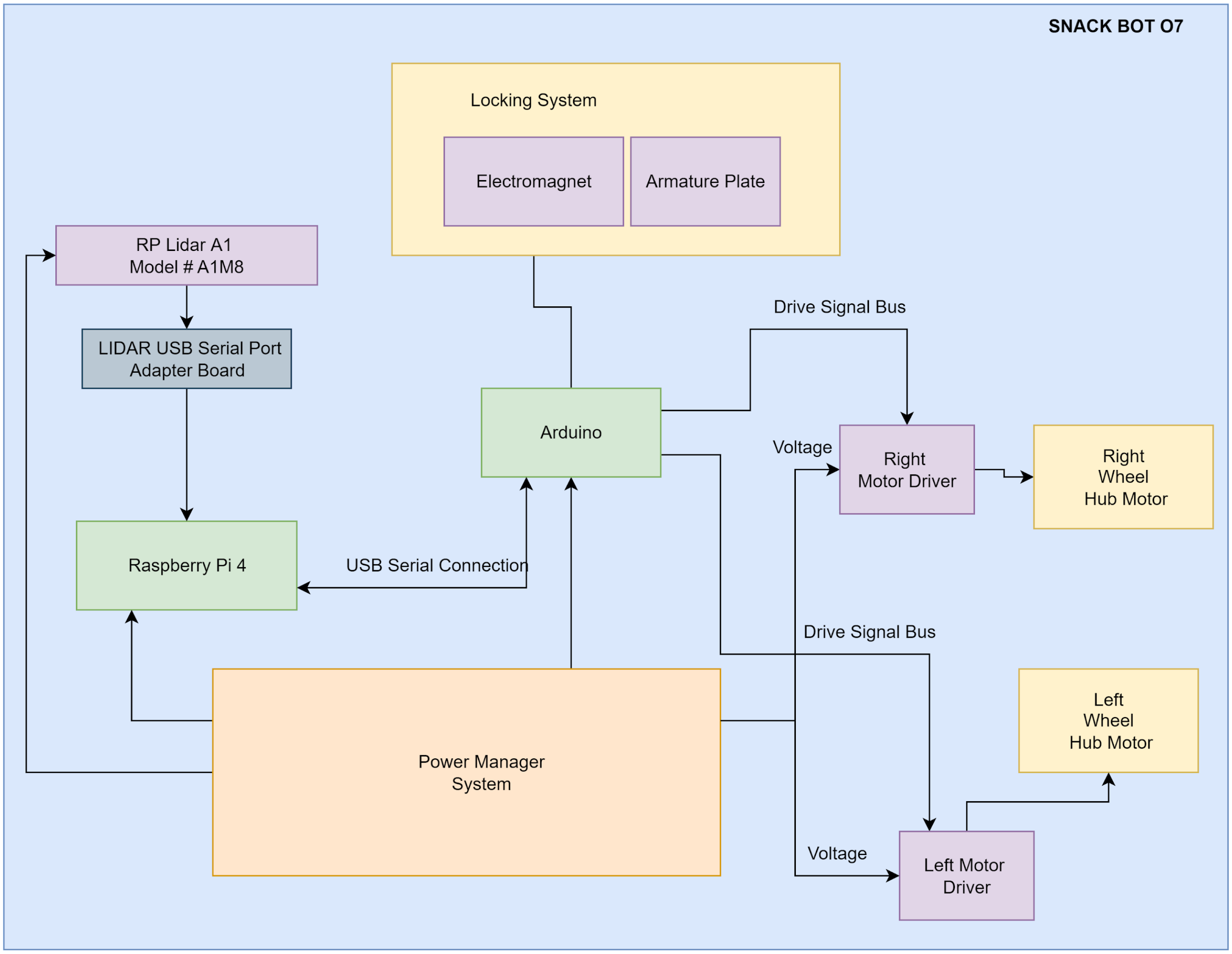


Figure 5: System Block Diagram for Snack Bot O7

The flowchart shown in figure 6 describes the different states the SnackBot O7 will be in and the signals responsible to change states. There is a basic error state defined in the below diagram. This state will be improved on and will be active in case the robot runs into any unusual situations.

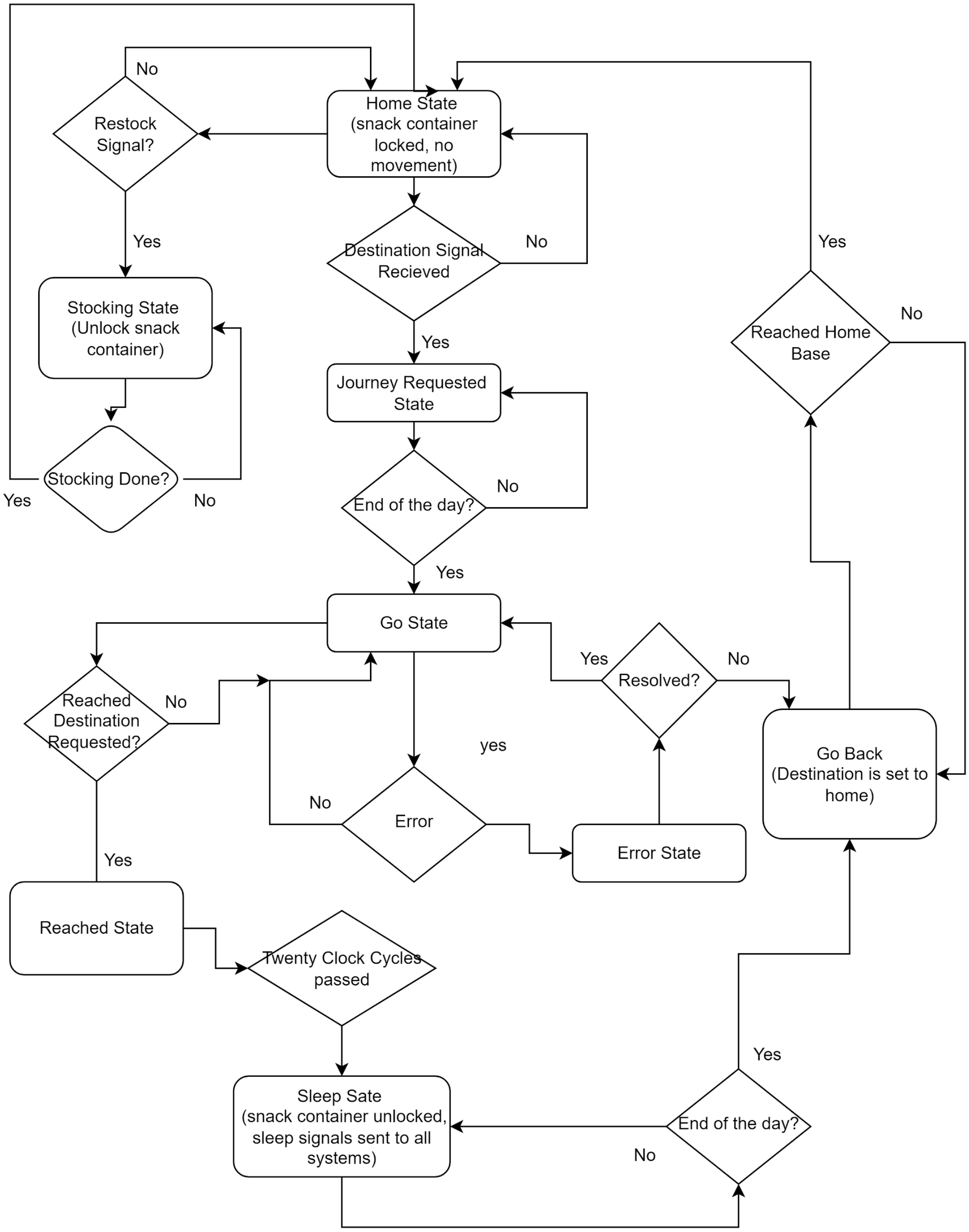

No
Restock Signal?
Home State (snack container locked, no movement)
Yes
Destination Signal Recieved
No
Yes
Reached Home Base
No
Stocking State (Unlock snack container)
Yes
Journey Requested State
Stocking Done?
Yes
No
End of the day?
No
Yes
Go State
Yes
Resolved?
No
Reached Destination Requested?
No
yes
Go Back (Destination is set to home)
No
Error
Error State
Yes
Reached State
Twenty Clock Cycles passed
Yes
Sleep Sate (snack container unlocked, sleep signals sent to all systems)
End of the day?
No

Figure 6: Flowchart

Communication between Systems

The SnackBot O7's entire system is controlled by two microcontrollers. The perception system which includes LIDAR and Uart-USB adapter boards is controlled by Raspberry Pi 4 while all other systems are controlled by the Arduino. Arduino controls the robot's movement system (motors and wheels), Locking mechanism and the power system.

It is essential that each system communicates with the other systems and maintains proper functionality.

Communication between the perception system's brain (Raspberry Pi 4) and Arduino UNO is set up using a USB wire connection as shown in Figure [1].

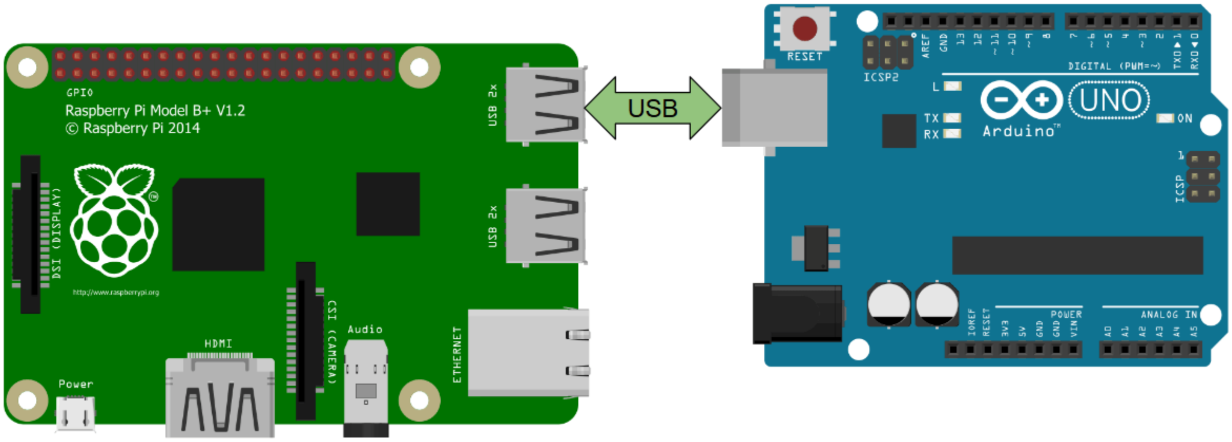


Figure 7: Serial Communication between [1]

Communication between Arduino UNO and the Power management system is set up through a relay shield. Relay is in a Normally Closed state until a sleep signal is sent to the relay which disconnects the power supply from the 36 V battery to the Motor Driver for the Brushless DC wheel hub motors.

The Arduino also sends signals to the Electromagnetic locking mechanism for the snacks container. It locks and unlocks based on the signal sent to the Arduino Pin.

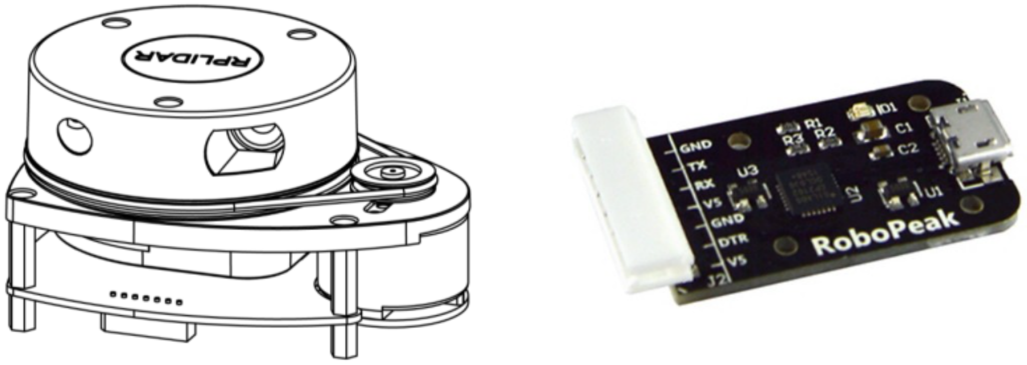
Design ID	**Design Specification Requirements**	**Corresponding Requirements ID**
Des 2.1 A	Serial Communication Set up between Raspberry Pi and Arduino through a wired USB connection	Req: 5.1 A *Raspberry Pi and Arduino should be able to send and receive signals between each other*

Table 2: Communication Design Specifications

Perception System

Once the robot goes into ‘Go State’, the Lidar and Raspberry Pi start mapping out the environment and making decisions to start moving to reach the destination. The software design will go over the process in more detail.

In this section, the electronic components used and the communication between these components will be described. The design specifications that pertain to the electronic setup will be defined here. For perception, Snackbot O7 uses a 360 degree 8m range LIDAR (Figure 8 (a)) with a USB serial port adapter board (Figure 8 (b)) taken from RP Lidar A1M8’s data sheet [3].



(a) (b)

Figure 8: (a) RP LIDAR M8A1 and, (b) LIDAR USB Serial Port Adapter Board [3]

The following image (Figure 9) shows the two parts connected together and the UART port connected to a Uart-USB wire. This wire is connected to one of the four USB ports on the Raspberry Pi [4]. The structural schematic of the LiDar is also included in Figure 10 [3] which will be referenced for the structure design.

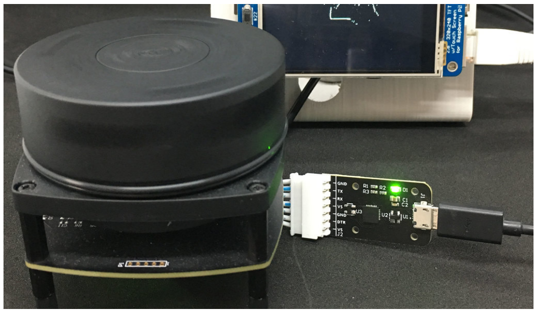


Figure 9: LiDar connected to adapter board [4]

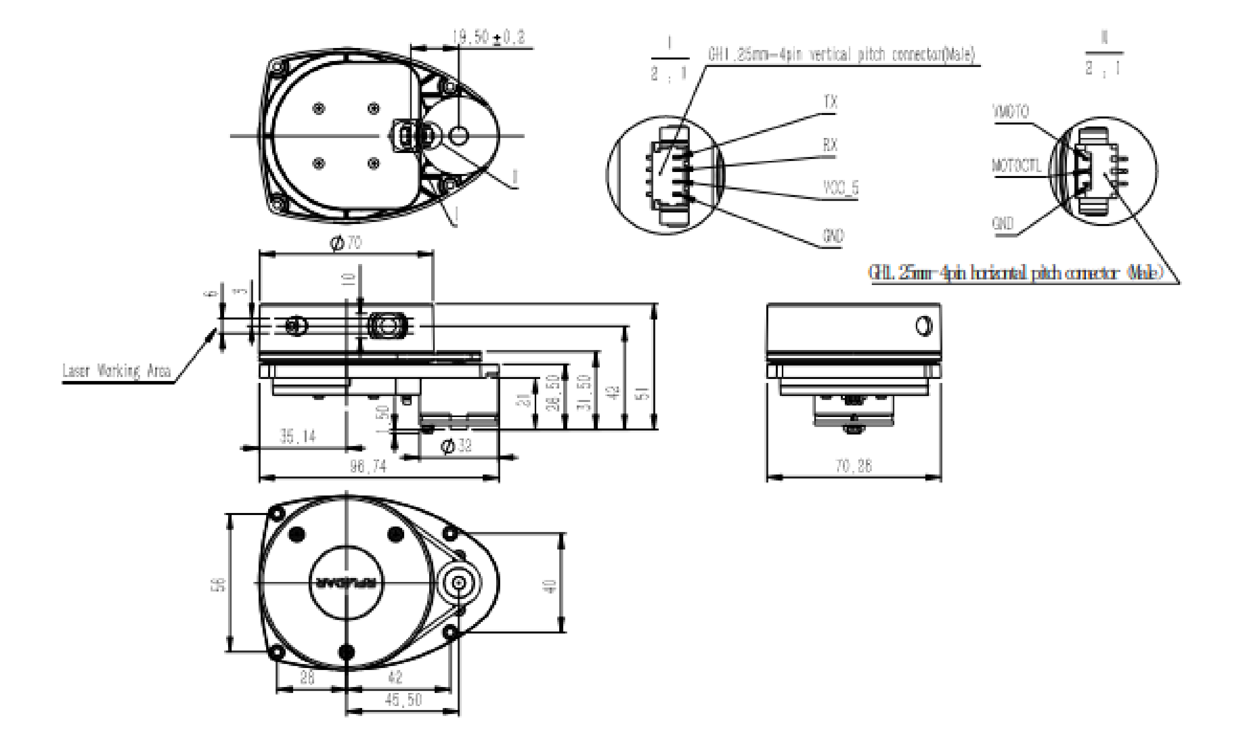


Figure 10: RP LiDar A1M8’s structural schematic [3]

Design ID	Design Specification Requirements	Corresponding Requirements ID
Des 3.1 B, C	RP Lidar A1 M8 is connected to the USB to Uart Board. The board is connected to Raspberry Pi’s USB Port.	Req 5.4 *LIDAR signals should be received by Raspberry pi and processed in real time*
Des 3.2 B, C	Signals are sent and received between Lidar and Raspberry Pi based on the algorithm designed by RSC-7 and Adafruit’s Rplidar library	Req 5.4 *LIDAR signals should be received by Raspberry pi and processed in real time*

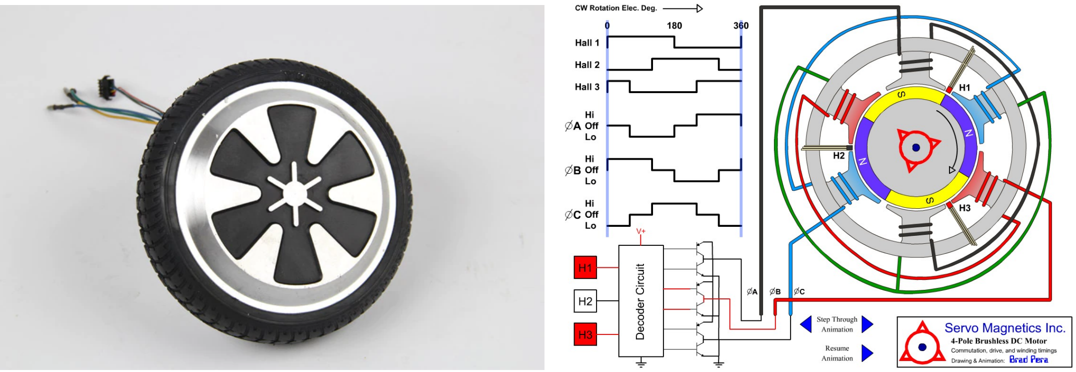
Table 3: Perception System Design Specifications

Robot Movement System

This system includes a microcontroller (Arduino Uno), two 3-phase brushless DC wheel hub motors (as seen in Figure 11) and a 3-phase motor controller board (Figure 13).

In ‘Go State’, the arduino will receive signals from Raspberry pi to move forward, turn left, turn right or stop. These signals will be translated into signals for the Motor Driver and the wheel hub motor will be controlled.

Figure 11 (b) shows the brushless DC in-wheel hub motor and how it is set up internally [5]. The next image (Figure 12) shows a diagram of the internal components of a brushless DC motor [5].



(a) (b)

Figure 11: (a) Brushless DC in-wheel hub motor (b) Internal setup showing the 3 out of phase inputs needed to run the motor

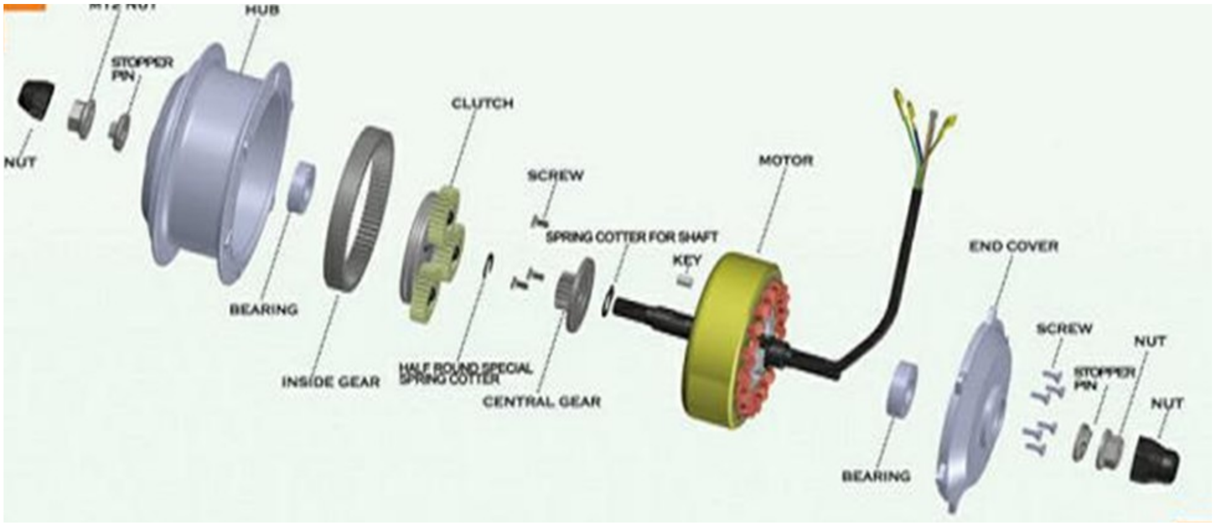


Figure 12: Exploded Diagram of the BRushless DC In-wheel hub motor

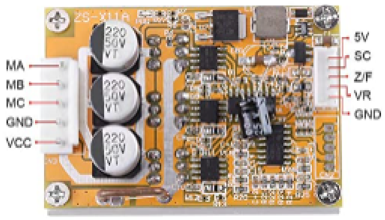


Figure 13: Brushless Hall Motor Controller Speed Controller 5V-36V 350W

Design ID	**Design Specification Requirements**	**Corresponding Requirements ID**
Des 4.1 A, B, C	Brushless DC motor connected to Brushless Hall Motor controller which is connected to arduino via wires	Req 3.1 *Robot should be able to move forward, backward, left, right and make turns when necessary* Req 7.3 *Must travel faster than a metre per minute*
Des 4.2 B, C	Arduino receives signals from Raspberry Pi and sends the corresponding signals to the motor driver to move at a certain speed forward or stopping (braking) and turning left or right.	Req 3.5 *Robot should be able to start its journey and navigate to the destination address effectively (avoid getting stuck or bumping into things and avoiding obstacles like stairs, pillars, etc)* Req 5.2 *Arduino should be able send signals and switch the motors on and off and pick the direction of rotation* Req 7.3 *Must travel faster than a meter per minute*

Des 4.3 C	A Real Time Clock is running on Arduino. After a set number of clock cycles, the arduino sends signals for Stand-by-mode. These signals include a stop to Raspberry Pi, a stop to the Power Management System to switch off the 36 V battery connected to motors and a signal to the lock based on the destination.	Req 3.8.2 *After reaching the delivery location, the robot will initiate stand-by-mode* Req 3.9 *Robots will have an automatic locking and unlocking system connected to the stand-by and power management system*

Table 4: Movement System Design Specifications

Power Management System

The Power Management System will contain a 36 V battery that will be used to power the two wheel hub motors used to move the robot and a 9 V battery connected to Arduino UNO, Raspberry Pi and the electromagnet of the locking mechanism.

The 36 V battery will be connected to a relay module in the Arduino Relay shield mounted onto the Arduino UNO. The relay will be in a normally closed state until a signal is sent by the arduino to open the circuit and disconnect it from the Motor Driver [6]. This is done to preserve and extend the battery life in sleep and home states.

The 9 Volt battery will be connected directly to the required components and the entire system will be connected to a power on/off button. Once the power button is pressed, the entire system will turn on or off.

Design ID	**Design Specification Requirements**	**Corresponding Requirements ID**
Des 5.1 A	Power On/Off button is included and attached to the power management system	Req 6.1 *Power supply must provide enough power for a full day of operation*
Des 5.2 B	36 V power supply for motors	Req 6.5 *Power supply must be over 12V and 6 Ah to provide sufficient voltage to the motors*
Des 5.3 A	9 V power supply for Raspberry Pi and Arduino	Req 6.2

		Raspberry Pi needs between 3.3V to 5V to operate Req 6.1 *Power supply must provide enough power for a full day of operation*
Des 5.4 B	Relay controls the 36 V DC voltage supply to the motor driver and the motors	Req 6.6 *Overvoltage and current limiters must be present in the final design for safety measures*
Des 5.5 B	The two batteries are completely separate	Req 6.6 *Overvoltage and current limiters must be present in the final design for safety measures*
Des 5.6 A, B, C	Appropriate connectors will used to connect the wires together	Req 6.1.1 *Power supply must be properly shielded from external conditions* Req 6.4 *Connector wires cannot come loose during operation*
Des 5.7 B,C	Structure will include separate enclosed housing for this entire system	Req 6.1.1 *Power supply must be properly shielded from external conditions* Req 6.3 *Circuitry must be separated from food storage area* Req 6.7 *Electrical system will be waterproofed by encasing in a plastic box with only the required parts exposed*

		Req 7.7 *Sensors on the external body of the robot will have proper protection in case of sudden impact* Req 7.9 *For waterproofing the prototype, the snacks will be put in a plastic basket which will be secured to the structure*

Table 5: Power Management System Design Specifications

Locking System

A simple locking mechanism which is controlled by an arduino is proposed for the Snacks container of the Snack Bot O7. When Snack Bot O7 is in Restocking and Reached states, refer to the flow chart in figure 6 for different states, the snacks container will get locked and stay locked.

The following image (Figure 14) is taken from the official Arduino website and will be used as a reference to develop the locking mechanism [7].

When the Robot changes states, a signal of High or Low will be sent through arduino to the Electromagnet subsystem and it will lock or unlock the snacks container.

The Locking mechanism makes it possible for the user to open the snacks container. It does not open the snacks container for the user. The users would need to lift the lid and take out the snack themselves.

The following images show the locking mechanism setup and the electromagnet used for this system (Figure 14 and 15 respectively) [7].

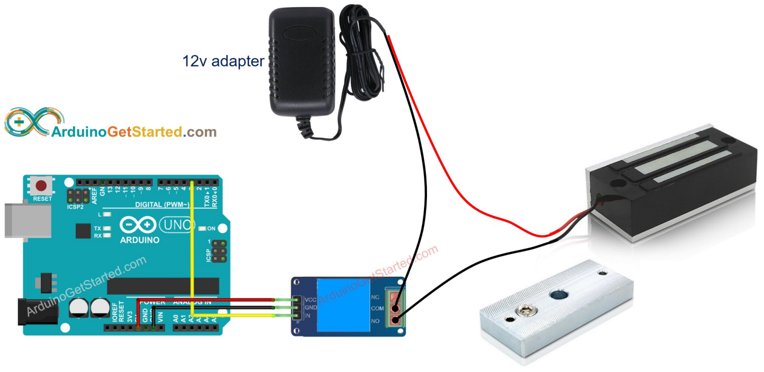


Figure: 14: Arduino Powered Electromagnetic Locking mechanism [7]

Figure: 15: Electromagnet and Armature plate used for the locking mechanism [7]

Design ID	Design Specification Requirements	Corresponding Requirements ID
Des 6.1 C	An electromagnet will be attached to the snack container that receives signals from arduino and locks or unlocks the container	Req 3.9 *Robots will have an automatic locking and unlocking system connected to the stand-by and power management system* Req 7.8 *Locking mechanism to ensure protection of cargo*
Des 6.2 C	A High signal sent to the pin connected to the electromagnet locks the snacks container	
Des 6.3 C	A Low signal sent to the pin connected to the electromagnet locks the snacks container	

Table 6: Locking Mechanism Design Specifications

User Interface Design

Note: For a full UI appendix please refer to the previously submitted User Interface and Appearance Design Appendix. Below is a brief summary and recap for completeness and for the convenience of the reader.

UI Summary

Due to the substantial work-load and associated timeline of the mechanical, electrical, and autonomous navigation software components of Snack Bot 07, two separate user interfaces have been designed. The first being a simple remote, using basic forward, left, right, and start/end trip controls, either wirelessly via IR remote or through a wired device or a keyboard. This interface will be primarily utilized in the proof-of-concept phase and through the final prototype phase. This rudimentary interface has been selected for the sake of simplicity and easier debugging during development.

The second interface is a smartphone application, the goal of which will be to offer a fully functional user interface to the university campus and catering staff that are the target market of Snack Bot 07, and who will be responsible for using the application to manage Snack Bots so they can serve their customers, the end users. This user interface is currently slated as a “future feature” that is planned to be implemented after the final physical prototype is finished at the end of ENSC 440. This places it outside of the scope of both ENSC 405 and 440, however, it is important to recognize and plan for the eventuality of this feature being implemented, as it would be a necessary prerequisite to Snack Bot 07 going to market.

Having planned for and undertaken the design and prototyping of a functional smartphone application user interface, certain functions of this feature can then be potentially implemented at an earlier phase, potentially during ENSC 440, as a “nice-to-have” feature in the event of the structural, electrical, and autonomous navigation portions of the project finishing ahead of schedule.

Graphical Presentation

Alpha UI

Figure 16 shows the layout of a typical IR remote that will provide the testing team with the ability to perform basic movements for the proof-of-concept demo, as well as basic start and cancel operations for later prototype demos when the Snack Bot 07 is expected to be able to perform autonomous navigation and user control is limited to a simple start or cancellation of delivery.

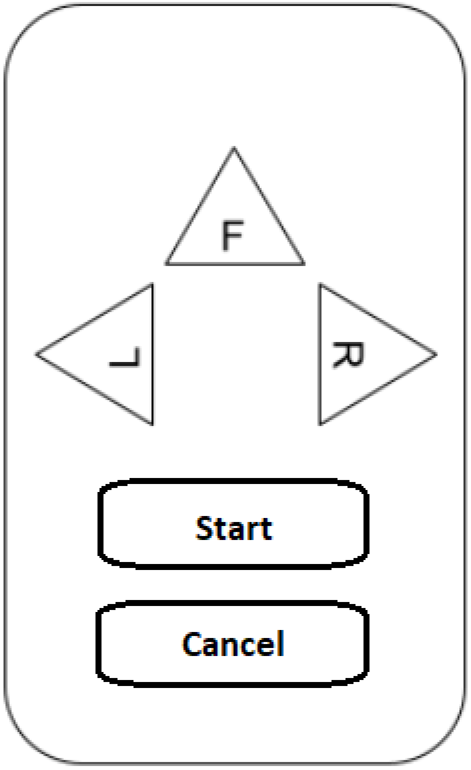


Figure 16: Layout for the remote controller

Beta Software UI

Figure 17 shows a prototype design illustrating a typical start of delivery procedure, from initial log in to creating a delivery, selecting a delivery bot, managing inventory, selecting a destination, and finishing the delivery creation, at which point the delivery begins. It should be noted that this is merely a prototype design, and even in the event that the Beta Software UI is implemented, most of the secondary logic such as inventory and robot management will be non-functioning stubs. The planned function of the beta software UI for potential ENSC 440 final prototype demonstration would be solely to initiate and cancel a delivery through a smartphone application in a manner similar to how the end product would function.

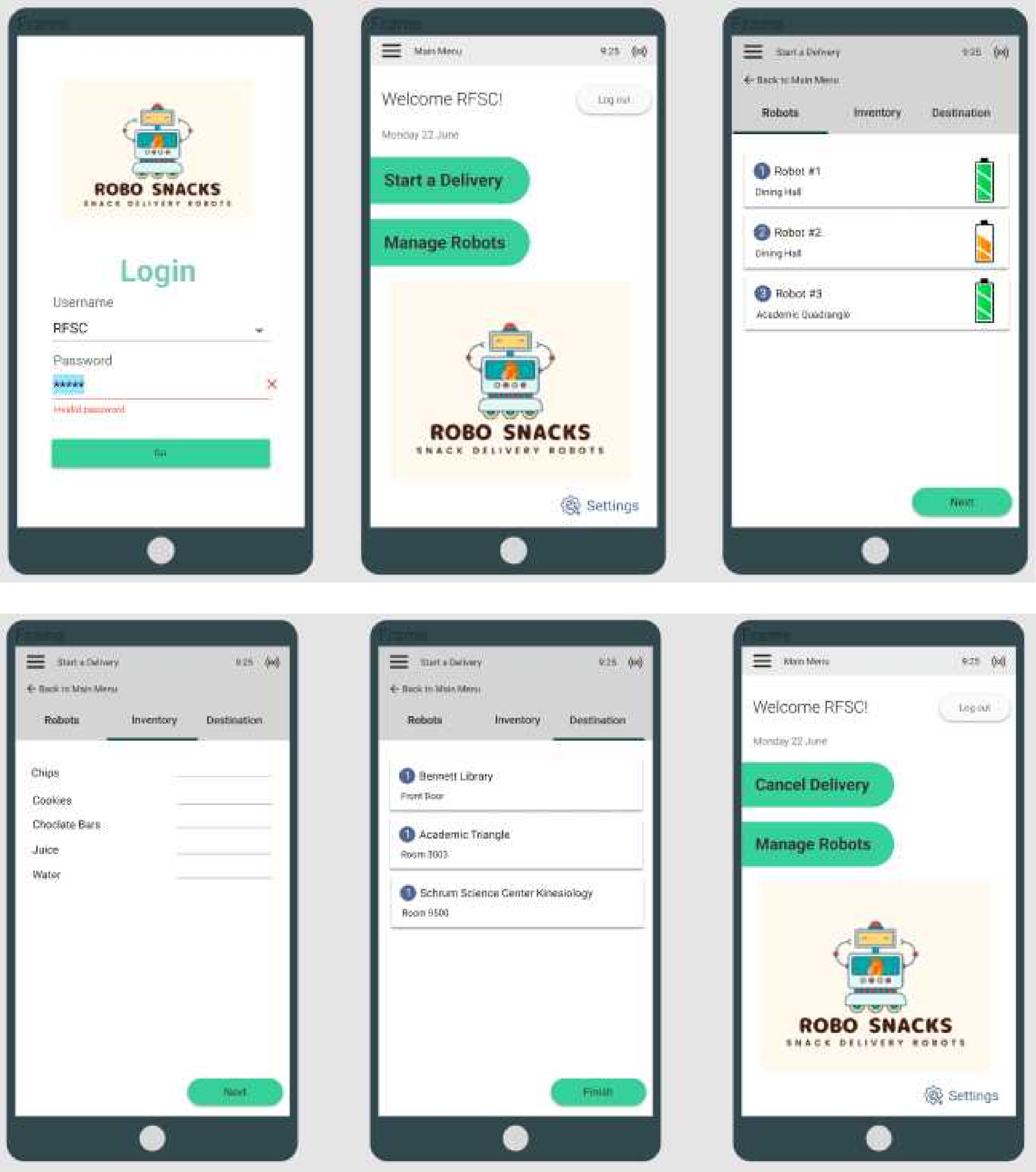


Figure 17: User Interface for the proposed Android App

Autonomous Movement Software Design

Due to the amount of online resources, Raspberry Pi 4 will be used as Snack Bot's microprocessor. In addition, ROS Moledic will be installed in an Ubuntu 18.04 environment. The decision for choosing ROS Moledic rather than ROS2 Moledic is due to the reason that it comes with long term support. In terms of language, Python 3 was chosen as it provides many predefined functions and libraries to be used.

Note, Snack Bot will be using ROS 1 instead of ROS 2 specified in requirement specification now due to ROS 1 having more learning resources.

For PoC, the software will primarily focus on controlling Snack Bot in different directions. It will be written in C++ on the arduino to control each wheel. In the script, each wheel's speed, direction whether it should stop can be configured via GPIO mappings to the controller.

For the final product, Snack Bot will utilize the ROS NavStack in order to navigate to the drop off point. The gmapping package will be used to map out the building floor plan from starting to end location. Then, using the NavFN global planner, a shortest cost path will be generated for the Snack Bot to take. The odometry of the robot will be calculated by using data from LiDar and wheel encoders that is processed by the arduino.[8]

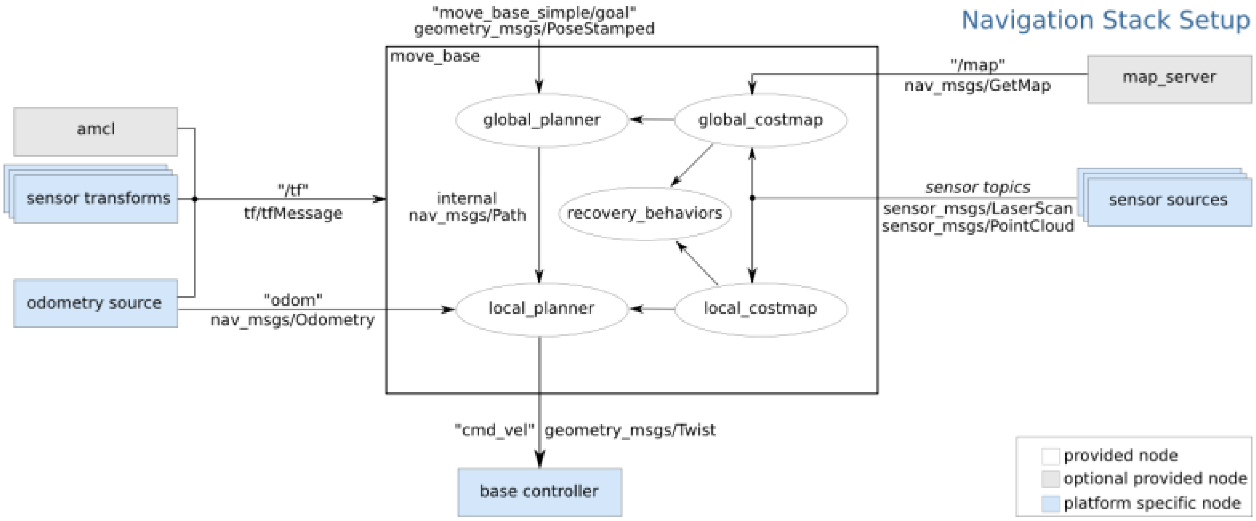


Figure 18: Diagram of data flow using ROS Navigation [2]

Figure 18 shows a map of how our system will work using the NavStack. Using GMapping, the robot will create a mapping for our robot. This map will be fed into our global_costmap node. From information coming from Snack Bot's sensor sources (LiDar), data will be fed into local_costmap which is an input for local_planner. In local_planner, Snackbot will send inputs to our base controller (arduino) to control the motors.

Design ID	**Design Specification Requirements**	**Corresponding Requirements ID**
Des 7.1 A	Python program will be able to move Robot in all directions	Req 3.1 *Robot should be able to move forward, backward, left, right and make turns when necessary* Req 4.1 *The robot must be able to run ROS2 NAV stack on raspberry pi and will be connected via Wi-Fi for programming* Req 4.2 *The robot must be able to move in all directions on the ROS2 NAV stack tested using keyboard keys to navigate left, right, forwards and backwards.*
Des 7.2 B	Raspberry Pi 4 will receive data from Arduino sent from encoders to calculate robots odometry	Req 4.4 *Using ROS2 NAV stack, we should be able to communicate to all parts wirelessly*
Des 7.3 B	Raspberry Pi 4 will receive data from LiDar to be read into local_costmap. Note: No wireless communication between any parts.	Req 4.4 *Using ROS2 NAV stack, we should be able to communicate to all parts wirelessly* Req 5.4 *LIDAR signals should be received by raspberry pi and processed in real time*
Des 7.4 B	Odometry is setup, Raspberry Pi 4 program can make robot move and knows where robot is moving	Req 4.4 *Using ROS2 NAV stack, we should be able to communicate to all parts wirelessly* Req 4.6 *Using ROS2 NAV stack the robot should be aware of the map*
Des 7.5 B	Localization is setup, and robot will be viewable on map	Req 3.4 *Robot should be able to approximate its general location*

		at any given time Req 4.6 *Using ROS2 NAV stack the robot should be aware of the map*
Des 7.6 B	Python program is able to detect objects in environment	Req 3.5 *Robot should be able to start its journey and navigate to the destination address effectively (avoid getting stuck or bumping into things and avoiding obstacles like stairs, pillars, etc)* Req 4.6 *Using ROS2 NAV stack the robot should be aware of the map*
Des 7.7 B	Python program can move robot through waypoints A to B on map using NAVFN	Req 3.3 *Robot should be able to select the best path from preloaded map* Req 4.5 T*he algorithm to navigate from A to B and back to A will be programmed*
Des 7.8 C	Raspberry Pi 4 will deactivates Robot	Req 3.8.1, *After reaching the delivery location, the robot will initiate stand-by-mode* Req 4.8.1 *The robot will go into stand-by mode once it reaches the destination location* Req 4.8.2 *The robot will come out of the stand-by mode (see requirement ID: 4.3) "at the end of the day" (after a set number of clock cycles)*

Table 7: Locking Mechanism Design Specifications

Conclusion

In summary, Snack Bot O7 is an indoor mobile autonomous robot that provides catering services to its customers by delivering non-perishable snacks and drinks to a predetermined location. Based on feedback from the progress review meeting, the team went ahead to make changes to the target customers, structure design and product name and these were outlined in the document.

The general system design specifications of Snack Bot were discussed with reference to the requirement specification document and reasoning behind the selected specification. Block diagrams as well as schematics were provided to help clarify how the system will be assembled. Then, a detailed test plan was provided to catalog the test strategy and necessary resources for performing tests on Snack Bot O7 to ensure that it is working properly.

At Robo Snacks Company, our intention is to provide our customers with an authentic, safe and eco-friendly product that is centered on innovative and efficient design requirements and features.

References (for everything above)

[1] "The Robotics Back-End - Raspberry Pi Arduino Serial Communication," [Online]. Available: https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/ [Accessed 11 March 2022].

[2] “Setup and Configuration of the Navigation Stack on a Robot”, [Online]. Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup [Accessed 11 March 2022].

[3] slamtec, “RPLIDAR A1 Low Cost 360 Degree Laser Range Scanner Introduction and Datasheet Model: A1M8,”, 2016-07-04 rev.1.0. Available: https://www.generationrobots.com/media/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf [Accessed 13 March 2022].

[4] D. Astels, "adafruit - Using the Slamtec RPLIDAR on a Raspberry Pi," 10 March 2022. [Online]. Available: https://learn.adafruit.com/slamtec-rplidar-on-pi?view=all. [Accessed 13 March 2022].

[5] "avdweb.nl - Permanent magnet DC electric motor tuning," [Online]. Available: https://www.avdweb.nl/solar-bike/hub-motor/permanent-magnet-dc-hub-motor-tuning. [Accessed 11 March 2022].

[6] "Control Global - Understanding switch configurations," [Online]. Available: https://www.controlglobal.com/articles/2019/understanding-switch-configurations/#:~:text=NC%20means%20normally%2Dclosed%20contact,open%2C%20and%20NO%20contact%20closes. . [Accessed 13 March 2022].

[7] "arduino.getstarted - Arduino Electromagnetic Lock," [Online]. Available: https://arduinogetstarted.com/tutorials/arduino-electromagnetic-lock . [Accessed 11 March 2022].

[8] “Set Up the Odometry for a Simulated Mobile Robot in ROS 2” [Online]. Available: https://automaticaddison.com/set-up-the-odometry-for-a-simulated-mobile-robot-in-ros-2/ . [Accessed 11 March 2022]

APPENDIX: Design Alternatives

Motor Alternatives

Two brushed 350 W DC motors connected to the two driving wheels could be used as the movement mechanism. The main concern with these motors was the maintenance in the long run [1]. RSC-7 wanted the robot to be a long functioning robot with minimum maintenance.
The second concern with this design was the total weight of the two motors, two wheels and any attachments connected to it would come out to be about 1 kg in total but would only be able to drive about 50 - 70 lbs. On the other hand the in-wheel hub motors come to about the same weight and can drive close to 240 lbs [2].

Microcontroller Alternatives

According to Robocademy, while there are several microcontroller alternatives for using ROS and Lidar navigation, Raspberry Pi and Arduino are highly recommended due to the amount of documentation and support that exist for these platforms[4]. Alternatives such as Nvidia Jetson and Intel NUC were investigated and are available as alternatives, but at time of writing no Jetson units in stock could be found, and the price of NUC is many times greater than what using a Pi and Arduino would cost.

Structure Alternatives

The alternative structure that was considered was an isometric triangle chassis that would be driven by omni wheels [3]. It would have 2 wheels driving and one wheel dragged along as it would go in one direction. However, the main feedback received was that our weight would exceed the frictional force required for dragging. The concept itself is used in many thesis papers but is not a common design in other applications. It would have taken more hours to prove the concept before integrating autonomy into the robot.

ROS Alternatives

ROS 2 was an seeked as an alternative which would operate similar to ROS 1 but there’s generally more well written documentation and tutorials around ROS 1 and NavStack online.

References

[1] "Association for Advanced Automation," [Online]. Available: https://www.automate.org/blogs/brushed-dc-motors-vs-brushless-dc-motors. [Accessed 3 March 2022].

[2] "aliexpress," [Online]. Available: https://www.aliexpress.com/item/32850540959.html. [Accessed 11 March 2022].

[3] "Research Gate," [Online]. Available: https://www.researchgate.net/figure/Three-wheel-Omnidirectional-robot_fig10_256089781. [Accessed 1 March 2022].

[4] L. Joseph, “How to choose a brain for your robot?,” Robocademy, 22-Aug-2020. [Online]. Available: https://robocademy.com/2020/04/18/how-to-choose-a-brain-for-your-robot/2/. [Accessed: 13-Mar-2022].

APPENDIX B: Test Plan

Introduction

This section highlights the details and steps involved in testing the robot Snack Bot O7.
It entails a comprehensive understanding of the workflow and capabilities of the robot by testing each sub-system and ensuring that the requirements listed in the requirements documents are met. It is important that a test plan is written and implemented to find out if a product meets quality standards before it is released.

Tests are conducted to verify that the system works in accordance with its design, to know the limitations, restrictions and restraints of the system and to detect glitches or bugs in general and fix them before the official release of the product.

Purpose and Scope

The goal of this document is to explain in detail the process involved in testing the Snack Bot O7 robot by providing all the necessary information about each component that makes up the entire system and the steps needed to verify and validate their functionality. The tests include:

- Software testing
- Electrical testing
- System testing
- Structural testing

Software Testing

Step	Requirement Description	Test Step	Expected Result	Pass/Fail
1	The robot must be able to run ROS1 NAV stack on raspberry pi and will be connected via Wi-Fi for programming	Connect the raspberry pi to a monitor and verify that ROS1 is installed and able to run on it	The raspberry pi is able to run ROS1 NAV stack on it.	
2	The robot must be able to move in all directions on the ROS1 NAV stack tested using keyboard keys to navigate left, right, forwards and backwards	Depending on the keyboard key pressed, the robot should respond accordingly. For example if the left key is pressed, robot should turn left	The robot is able to move in all directions on the ROS1 NAV stack tested using keyboard keys to navigate left, right, forwards and backwards	
3	The robot must go into	Use a voltmeter	The robot goes	

	stand-by mode once it reaches the destination location	to verify that the input supplied to the motor drivers is 0	into stand-by mode once it reaches the destination location	
4	Using ROS1 NAV stack the robot should be aware of the map	Manually verify the map has been loaded to memory and can be accessed	Map is there	
5	Using ROS1 NAV stack, the robot should have a fail and fall back plan to return to base when dealing with unfamiliar situation	Trap the robot in a cardboard box and wait for the destination to change to home base after 5 minutes of being stuck	Robot changes destination and signals state change	
6	The robot will come out of the stand-by mode (see requirement ID: 4.3) “at the end of the day” (after a set number of clock cycles)	Check power restored to Motor driver using Voltmeter	Voltage is observed on the voltmeter	

System Testing

Step	Requirement Description	Test Step	Expected Result	Pass/Fail
1	Raspberry Pi and Arduino should be able to send and receive signals between each other	Connect the Raspberry Pi and Arduino with a usb and verify that they are able to send and receive signals to each other	Raspberry Pi and Arduino are able to send signals between themselves	
2	Arduino should be able to send signals and switch the motors on and off and pick the direction of rotation	Connect the Arduino to the motor drivers and motor drivers to the motors, power each of them	Arduino is able to send signals, switch motors on and off and pick the direction of rotation	

		and verify that the arduino sent signals to the motors		
3	The motors connected to the wheels should function simultaneously	Check to see that the two motors are supplied with the same voltage. Check the arduino code for any time lag between the motors	Motors connected to the wheels function at the same time	
4	LIDAR signals should be received by the raspberry pi and processed in real time	Connect the LIDAR to raspberry pi with a usb and verify that the pi received and processed the signals	LIDAR signals are received by the raspberry pi and processed in real time	

Electrical Testing

Step	Requirement Description	Test Step	Expected Result	Pass/Fail
1	Power supply must be properly shielded from external conditions	By visually inspecting, verify that the power supply has an enclosure for protection	Power supply has an enclosure to shield it from external conditions	
2	Raspberry Pi needs between 3.3V to 5V to operate	Check to see if the red and green LEDs on the raspberry pi blinks. If they do, 3.3 to 5.5 volts was supplied to it	The voltage supplied to the raspberry pi is between 3.3 to 5 volts	

3	Circuitry must be separated from food storage area	By visually inspecting, verify that the circuitry is in Chassis and is a different component from the the component from the food storage component	The Circuitry and food storage area are different compartments	
4	Power supply must be over 12V and 6 Ah to provide sufficient voltage to the motors	By visually inspecting, verify that the power supply/ batteries used have a voltage rating above 12V and current rating above 6Ah	Power supply of the device has a voltage and current rating of 12V and 6Ah respectively	
5	Overvoltage and current limiters must be present in the final design for safety measures	By visually inspecting, verify that overvoltage and current limiters are present in the circuit	Overvoltage and current limiters are present in the final design of the robot	
6	Electrical system will be waterproofed by encasing in a plastic box with only the required parts exposed	By visually inspecting, verify that the electrical system is encased in a plastic box with only required parts exposed.	Electrical system is encased in a plastic box with only the required parts exposed	

Structural Testing

Step	Requirement Description	Test Step	Expected result	Pass/ Fail
1	Chassis must be able to support up to 50 lbs of	Weigh snacks of up to 50 lbs on a scale,	Chassis supports up to 50 lbs of weight	

	weight	put inside the robot and see that it is able to move and perform all its basic		
2	Structure must be rigid enough to survive impact with hard objects	At the maximum speed of 1 meter per second, the bare chassis will be run into a wall to check for structural integrity. Electronic components will be protected with extra insulation to absorb the impact.	Structure remains intact and motors stay put after applying a force	
3	Final design must include a combination of plastic and metal for the outer structure	By visually inspecting, verify that the outer structure of the final prototype is a combination of metal and plastic	The outer structure of the final design is a combination of plastic and metal for the outer structure	
4	Device structure should not exceed 100 lbs in weight	Weigh the structure on a scale and check to see that its weight is less than 100 lbs	The weight of the structure of the device is less than 100lbs	
5	Sensors on the external body of the robot will have proper protection in case of sudden impact	By visually inspecting, verify that the sensors on the structure have a shield to protect them from damage	Sensors on the external body of the robot have proper protection to protect them from sudden impact	
6	Locking mechanism to ensure protection of cargo	By visually inspecting, verify that the robot has a locking mechanism	Robot has a locking mechanism	

References

[1] "Techopedia - Test Plan," [Online]. Available: https://www.techopedia.com/definition/30546/test-plan. [Accessed 12 March 2022].