UNIVERSITY OF THE PHILIPPINES
OPEN UNIVERSITY

Master of Information Systems

GARRIVIC DE GUZMAN GLORIA

DEVELOPMENT OF DELIVERY ROBOT (DELIBOT) USING MICROCONTROLLER AND LINE TRACKER

Thesis/Dissertation Adviser:
Assoc. Prof. Concepcion L. Khan

Faculty of Information and Communication Studies

Date of Submission
January 21, 2022

Permission is given for the following people to have access to this thesis/dissertation:

Available to the general public	Yes
Available only after consultation with author/thesis/dissertation adviser	No
Available only to those bound by a confidentiality agreement	No

Student's Signature:
Signature of Thesis/Dissertation/Adviser:

University Permission Page

"I hereby grant the University of the Philippines a non-exclusive, worldwide, royalty-free license to reproduce, publish and publicly distribute copies of this thesis or dissertation in whatever form subject to the provisions of applicable laws, the provisions of the UP IRR policy and any contractual obligations, as well as more specific permission marking on the Title Page."

"Specifically, I grant the following rights to the University:

a) *To upload a copy of the work in the theses database of the college/school/institute/ department and any other databases available on the public internet;*

b) *To publish the work in the college/school/institute /department journal, both in print and electronic or digital format and online; and*

c) *To give open access to above-mentioned work, thus allowing "fair use" of the work in accordance with the provisions of the Intellectual Property Code of the Philippines (Republic Act No. 8293), especially for teaching, scholarly and research purposes."*

GARRIVIC DEGUZMAN GLORIA

09/21/2022

_____

Student Name over Signature and Date

ACCEPTANCE PAGE FOR MASTER'S THESIS

This thesis titled "Development of Delivery Robot (Delibot) Using Microcontroller and Line Tracker" is hereby accepted by the Faculty of Information and Communication Studies, UP Open University, Los Baños, Laguna in partial fulfillment of the requirements for the degree Master of Information Systems

ASSOC. PROF. CONCEPCION L. KHAN
Program Chairperson

_____
Date Signed

DR. ALEXANDER FLOR
Dean, Faculty of Information and
Computer Studies

_____
Date Signed

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1	**Scrum Methodology**	10
Figure 2	**Improved Food Ordering and Serving Flow using Delibot**	11
Figure 3	**Line following navigation**	13
Figure 4	**Tag following navigation**	13
Figure 5	**Line and Tags Track**	14
Figure 6	**Robot Steering Designs**	15
Figure 7	**Types of Navigation Systems**	16
Figure 8	**Track detection using proximity sensor**	17
Figure 9	**Parts of the Robot System Design**	19
Figure 10	**Mechanical design structure of Delibot**	21
Figure 11	**Operation Flow of Delibot**	22
Figure 12	**Delibot track layout for testing**	25

LIST OF TABLES

TABLE	TITLE	PAGE
Table 1	Robot Movement Control Using Wheel Directions	23
Table 2	Track Pattern Designs and Robot Behavior	24
Table 3	User Test Cases and Results	27

Abstract

The project was developed to transport items using an Automated Guided Vehicle (AGV) delivery robot named “Delibot”. For this project, the robot was used in a restaurant environment to transport food and dishes. The robot was able to move inside the workplace by following the black track that guides the robot to reach different locations from the kitchen to customers’ tables and return to the kitchen. The robot movements were controlled and guided by a microcontroller (ATMega2560) and proximity sensors. Four proximity sensors were installed for detecting the tracks and junction nodes. The robot can be programmed by the user to move and follow directions from origin to its destination. The robot was driven by 4 DC motors that allow it to move straight, turn left or right and rotate from its current position. To make the robot user-friendly, it was equipped with an audio module player that allows speaking like humans to communicate to the user.

The robot was tested using a small-scale prototype and successfully achieved the objective of transporting item and its capability to be reprogrammed according to the robot’s track path

I. The Problem Domain

A. Statement of the Problem

AGV robots were designed as material handling systems that can travel autonomously throughout workplaces. They are widely used in manufacturing, healthcare, agriculture, entertainment, food processing, and restaurants.

In this project, Delibot was created specifically to address issues in a restaurant environment when delivering food from the kitchen to the customer's table.

B. Background and Objectives of the Project

General Objective

- To develop a solution to resolve issues of manpower shortages, operating costs and productivity related to logistics using robot automation

Specific Objectives

- To develop a robot that could transport items from one location to desired location by following a black line track for guiding its movement and without human intervention.
- The robot can be programmed by the user with directions to follow when the robot reached junctions.
- The robot load capacity was limited to 10kg using the large scale prototype. For demonstration purposes, the robot will use a small prototype to simulate the functionalities described in this project are all working and the objectives were satisfied
- The robot will be used in a restaurant environment for transporting food and dishes to the customer

C. Significance and Scope of the Project

The recent Covid-19 outbreak has harmed businesses all around the world, particularly in the food and beverage industry (F&B). Many restaurants have closed, and some workers have lost their jobs as a result. Human-to-human contact and transmission were responsible for the virus's spread. To stop the virus from spreading, people started wearing face masks, face shields, and social isolation. Human touch and virus exposure could be reduced by using robots for meal delivery and food handling.

Some companies were looking into automating their processes to deal with the labor scarcity in some places (like Singapore), where labor costs are high and there is a dearth of staff in the food industry. Business owners were

investing in automation equipment to increase productivity and save operating expenses to keep their business afloat.

With the application of robots in the system, some steps can be modified to delegate tasks and improve productivity.

Here are some of the issues that could possibly resolved and benefits of using the robot automation:

- Lack of manpower – to overcome manpower shortage especially during peak hours, some repetitive tasks can be assigned to a robot (like food transporting). The human waiter can focus on providing quality service to the customer and improving productivity.
- High labor cost – Delibot can work multiple shifts, sending food to the customers' table accurately and safely thus reducing the cost of hiring more staff.
- Longer waiting time which affects productivity - Delibot can assist food delivery when human waiters are tied up and serving customers. They can reduce the overall waiting time of the customers. Besides, Delibot can also carry more dishes than a human waiter depending on the design of the robot.
- Possible spread of the virus through human contact - During this pandemic, safety is the most important thing for both the employee and customer. We should do things with caution to prevent getting infected and transferring the virus to others. Delibot can directly send the food straight to the customer, minimizing human contact and reducing the spreading of infectious viruses. Delibot can also be disinfected easily and don’t get sick.

II. Review of Existing Alternatives

F&B enterprises were doing well before the Covid-19 epidemic. The business owners can afford to recruit more staff to manage the business and

overcome manpower shortages. The F&B business was labor-intensive and pricey in the sense that there was a requirement for enough people - from floor manager to waiter - during peak hours but overstaffed during less busy periods

Then there was the Covid-19 pandemic. For the first time, restaurants throughout the world are facing difficult problems to stay in business. The Philippine government has imposed measures (Covid 19 Policies) such as a no dine-in policy or limiting the number of people who may dine in and maintaining 1-meter social distancing. The strict implementation of such policies drastically affects the F&B industry's profit and income and many food establishments were closed or forced to retrench their employees to cut their losses. These regulations might alter since current pandemic situations are still being monitored. The pandemic will not go away immediately. The challenge with the food and beverage industry is how to continue to operate and yet must follow the Covid-19 policies imposed by the government. The pandemic is proving to be a wake-up call for the global food industry, with change needed for things to take a turn for the better. F&B industry may consider relying less on manpower and more on automation.

III. Methodology

The ideal development framework for this project is to use the quickest way to deliver the project. The focus is more on working hardware and software rather than documentation. Working on this project would require planning on all robot components such as mechanical, electrical, and software. Each component should be fully tested and working properly so that the whole system would work when integrated.

The most appropriate software development methodology for this project is the Agile Scrum methodology. The Scrum methodology provides flexibility and uses the real-world progress of the project.

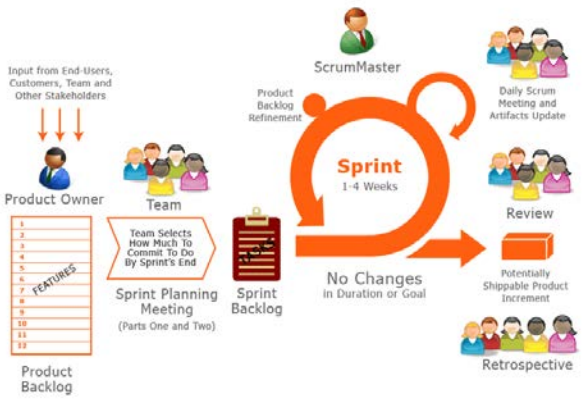


Figure 1: Scrum Methodology

The scrum approach was more suitable for this project as the development would require evaluation and experimentation. The whole project development required multiple iterations until the project is completed. Through the whole cycle of project development, there might be unexpected roadblocks that might be discovered in the design and changes to be made for improvement. This should be expected as some bugs and design flaws might appear during development or while on testing. Tasks and requirements may change in the next iterations and depending on the result of the current iteration.

A. Concept

Delibot was an AGV robot designed to handle and transport materials from one location to another. It was a general-purpose robot that can be used in warehouses, hospitals, manufacturing, agriculture, food and beverages industries, and agriculture. For this project, the robot is applied in F&B industries to transport food and dishes in a restaurant.

The main purpose of the project was to deliver the food directly to the customer without human intervention. Such a concept will benefit the business owner when it came to productivity, cost-saving, and safety of the

employee and customers. Below was the proposed Food Ordering and Serving flow with the use of a delivery robot (See Fig.3)

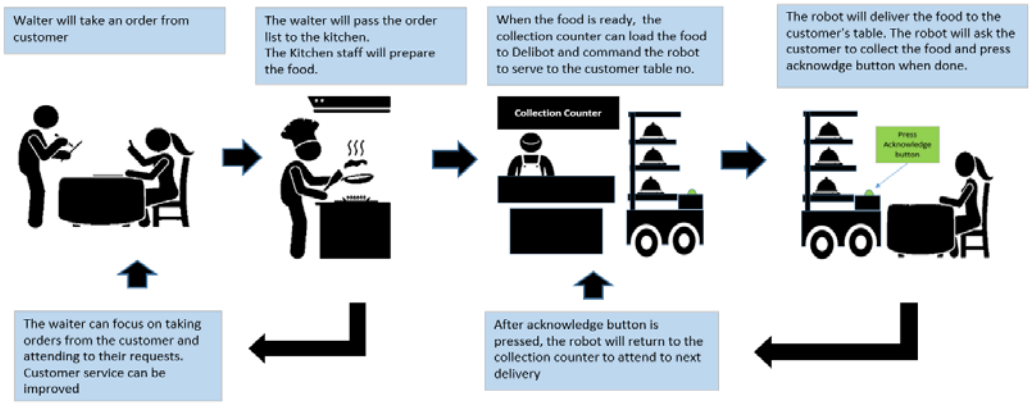


Figure 2: Improved Food Ordering and Serving Flow using Delibot

Below were the features of the delivery robot:

1. The robot would safely follow the line to deliver the dishes to the customer's table. In case the robot losses its track, the robot will automatically stop and alert the owner for recovery
2. The robot path can be programmed easily. The starting location was set to the Collection Counter. During teach mode, the robot followed the line and stopped when a junction was detected. The user selected the directions and continued moving until the robot reached its destination (Customer table). From the Customer table, the robot directions can be programmed to return to the Collection Counter. All these directions were saved into the controller's non-volatile memory. A maximum of 4 programs can be saved in the current memory.
3. The robot had a built-in ultrasonic range finder sensor. It detected any blockage on the robot's path and stopped to prevent any collision. The robot would play voice message to remove the blockage before it can continue to move.

4. The robot had a built-in Audio player saved with a pre-recorded message. The robot would use a human voice to communicate its current status and gave instructions to the user, making it more user-friendly.
5. The robot should be able to carry around more than manual delivery (10 kg loading capacity)
6. The robot can be controlled using an Infrared Remote Control for Teach and Run mode. A built-in keypad was added for Run mode functions only.

B. Methods

1. Review of Existing Systems

During the design stage, it's important to review the existing systems and technology. This would give the designer ideas that could be incorporated into his design. This would help mitigate the challenges that could be encountered during the testing stage.

Navigation Methods. According to Bluebotics[1] , when developing or upgrading an automated guided vehicle (AGV) or mobile robot, investing in the right autonomous navigation technology is key. Choosing the wrong AGV navigation method would have a dramatic effect on the cost, efficiency, and therefore success of the automated solution you produced. This was true both at the development stage when integrating navigation technology into your vehicle and, in the future, when your vehicle is on the market and you came to install that vehicle at customer sites.

There were numerous AGV navigation methods available. However, these can be broadly divided into two categories:

[1] AGV Navigation Methods 1: Line Following and Tags. Bluebotics, website

- Navigation technologies that guide vehicles along physical lines (Line following). With line following navigation technologies, AGVs were guided through a facility by a physical line, such as magnetic tape, inductive wire (installed in/under the floor), or painted lines.

Figure 3: Line following navigation

- Navigation technologies that guide vehicles along virtual (i.e. digital) paths, which exist only in the vehicle's software. Tag following works in much the same way as the following physical lines. However, in this case, AGVs were guided through a facility by tags such as QR codes, RFID tags, or magnetic points embedded in the floor. The behavior of the robot was defined in the software when the tag was detected.

Figure 4: Tag following navigation

Track Layout. The major steps in the design of an AGV system involved the choice of the track layout. The track would be installed according to the workspace and floor plan layout. The robot navigation can use line following, tag following navigation, or both. The movement of the robot can be controlled by adding markers to the track. Samples markers include stop markers, fork markers, and merge markers. Such design allowed the robot to move and change its direction and control its speed[2].

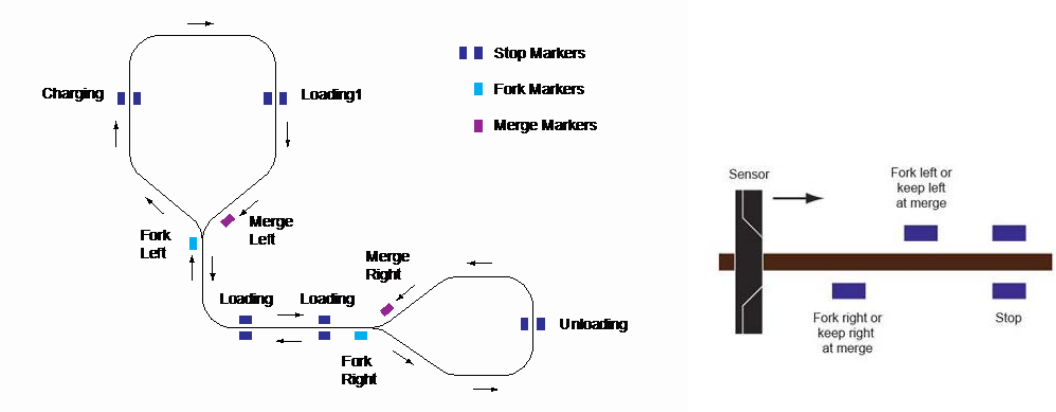
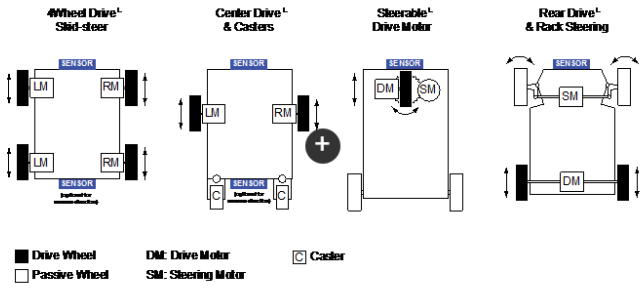


Figure 5: Line and Tags Track

Robot Steering. According to Roboteq[3], there were 4 basic ways of providing drive and steering (See Fig 8). Some types were easier to build, others have better steering characteristics. Two of these designs were fully symmetrical and may be operated in both directions. The table gave the list of characteristics of each design.

[3] Building a Magnetic Track Guided Vehicle. RoboteQ,, website https://www.roboteq.com/applications/all-blogs/18-building-a-magnetic-track-guided-agv



Design	Simplicity	Steering	Reversible
4Wheel Drive	Simplest	Coarse	Yes
Center Drive & Casters	Simple	Precise	Yes
Steerable Drive Motor	Medium	Very Precise	No
Rear Drive and Rack Steering	Complex	Precise	Difficult in reverse

Figure 6: Robot Steering Designs

2. Literature Review

An Automated Guided Vehicle (AGV) is an electric mobile robot that runs and navigates by itself without human intervention. It consists of one or more plc or pc controlled, wheel-based load carriers that navigate on the factory or warehouse floor without the need for an onboard driver.

According to AGVNetwork[4], an AGV is mainly composed of 5 elements: Navigation System, Safety System, Power System, Motion System, and Vehicle Controller.

a. The **Navigation System** is in charge of acquiring the info needed to drive and follow a given track or direction. There are several types of navigation: laser, natural, magnetic, etc. Choosing the right vehicle guidance technology is essential because it will influence the AGV robot system performance.

[4] Automated Guided Vehicle. AGVNetwork, website https://www.agvnetwork.com/what-is-automated-guided-vehicle-agv-robot

Figure 7: Types of Navigation Systems

b. The **Safety System** ensured that all the movements and maneuvers were performed in safe conditions. The main parts were the safety laser scanners and the safety plc. AGVs generally run smoothly and predictable, however on the rare occasion something doesn't work properly, the safety system will halt the vehicle.
c. The **Power System**. The AGV Battery provided the energy needed for vehicle movement and accessory functions. The main elements were the batteries and the charging solution.
d. The **Motion System.** These parts were in charge of converting the power into motion. Here we find the motors, motor-wheels, drivers, etc.
e. The **Vehicle Controller** is the brain of the AGV. It coordinates and processes all the info received from the other systems, thus, navigation, safety, motion, etc.

3. **Assessment of Existing Alternatives**

Based on the reviews of existing systems and comparing different alternatives, the design can be started by choosing the safety features and systems that are suitable for building the prototype. The choice of design should also consider the availability of resources, the cost, and the practicality of the design. For this project, the target prototype was a small-scale robot that could show the robot's capability as defined in the concept.

Navigation Method. The robot was designed to follow the track pattern to move from one location to another. The robot was using a proximity sensor

to track the lines as it moves. The sensor was able to detect the black line by measuring the surface reflectance of the track. A low-value reflectance indicates a black track is detected while a high reflectance indicates the sensor is out-of-track. The steering of the robot can be controlled through the algorithm written in the controller.

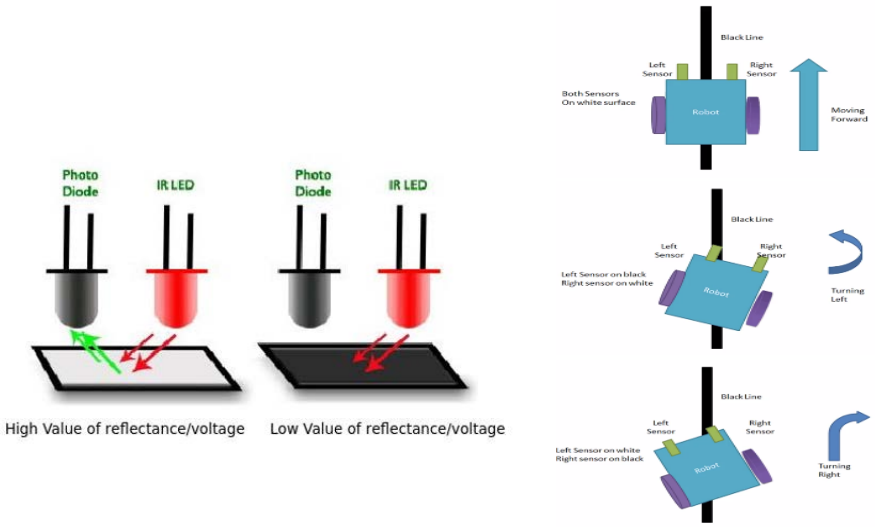


Figure 8: Track detection using proximity sensor

Safety System. The robot was equipped with a range finder to detect any object that blocks the path. It is using an ultrasonic transmitter/receiver to detect the blocking object. If the distance between the robot and the blockage is 20cm or less, it will automatically stop. If the distance of the robot and the block is 40 cm or more it will then continue to move forward. Another safety feature is in case the robot is not able to detect a track or out of track and was not able to recover for 3 sec, a timeout error will trigger and stop the robot's movement. This will prevent any accident to happen as a result of robot's failure.

Power System. The robot was powered by a 12V Lithium-ion battery. The battery will be replaced with a newly charged battery once the energy is drained.

Motion System. The implementation of this design was to use 4 wheels each driven by a DC motor for steering. The dc motor speed was driven by the motion driver module and controlled using Pulse width modulation signals coming from the controller.

Vehicle Controller. The brain of the robot will use the ATMEL2560 Arduino module board. The controller is equipped with Pulse Width Modulator (PWM) for controlling the motor speed. It has enough digital io to read sensors and switches status and control LCD and audio modules.

4. Production of Prototype

The production of a prototype involves initial designing and selection of components for building the prototype. The prototype is the product of research and review of existing systems and development to improve the system and make it better. The system design can be established through functional block diagrams and define the purpose of each component. The mechanical design showed the features and structure of the robot and what it will look like.

The software control and algorithm design show how the robot's movement is controlled and the operational features of the robot.

System Design

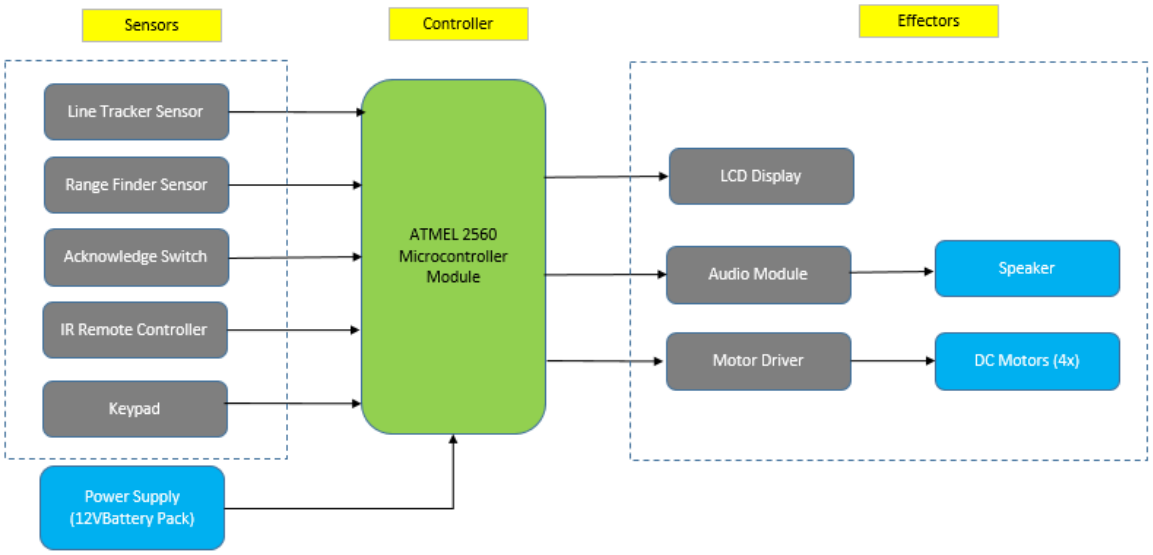


Figure 9: Parts of the Robot System Design

Below are the parts of the robot and their functions:

a. Microcontroller Module - Executes the control program (C program) for the robot to function. It has an EEPROM where the robot teaching program was stored, a nonvolatile memory so that it will not be lost when the robot is powered off. The control program was written in C language using Sketch environment, downloaded, and compiled to the microcontroller

b. Line Tracker Sensor - The proximity sensor detected the black lines that were placed on the track. The tracker was made of an LED infrared transmitter and receiver that detect the line using reflected light. It has 4 sensors to detect the Left & Right direction and the Forward & Reverse direction.

c. Acknowledge Switch – A push button was used to trigger the robot to execute after the user sent a command.

d. Range Finder Sensor – An ultrasonic sensor was used to detect any blockage on the robot's path. The distance from the robot to the blockage

was measured by emitting sound waves and converting the reflected sound into an electrical signal, similar to the principle of a radar system. If the distance between the robot and the blockage is 20cm or less, it will automatically stop. If the distance of the robot and the block is 40 cm or more it will then continue to move forward

e. Motor Driver - Control signals coming from the microcontroller were sent to the motor driver to make the robot move at a different speed. It drives the direction and speed of 4 motors that allow the robot to move forward, reverse, turn left, turn right and rotate

f. LCD Display - Liquid Crystal Display (LCD) display the readable status of the robot understandable by the user

g. Audio Module – drives the speaker to play pre-recorded human voice to convey information to the people in the environment

h. IR Remote Control - Infrared remote transmitter and receiver were used to control or teach the robot (for Supervisor user)

i. Keypad – The keypad was used to key in the table no. to deliver the food. To prevent any unauthorized access, the keypad is made active only when the robot is on the collection counter.

j. Power Supply (Battery Pack) – The battery pack used a 12V Li-Ion rechargeable battery to power the robot. The robot operation of the prototype lasted for 10 min. The actual robot is expected to run 8 – 10 hours a day and should be replaced once the power is drained.

Mechanical Chassis Design

The structure of the robot should be able to carry the load and navigate around the workspace. The mounting of electrical parts and control panel should be accessible to the user. The food tray should be accessible during

the loading and unloading of dishes. It is also important to consider the design's aesthetic qualities, ergonomics, strength, stability, rigidity, and safety.

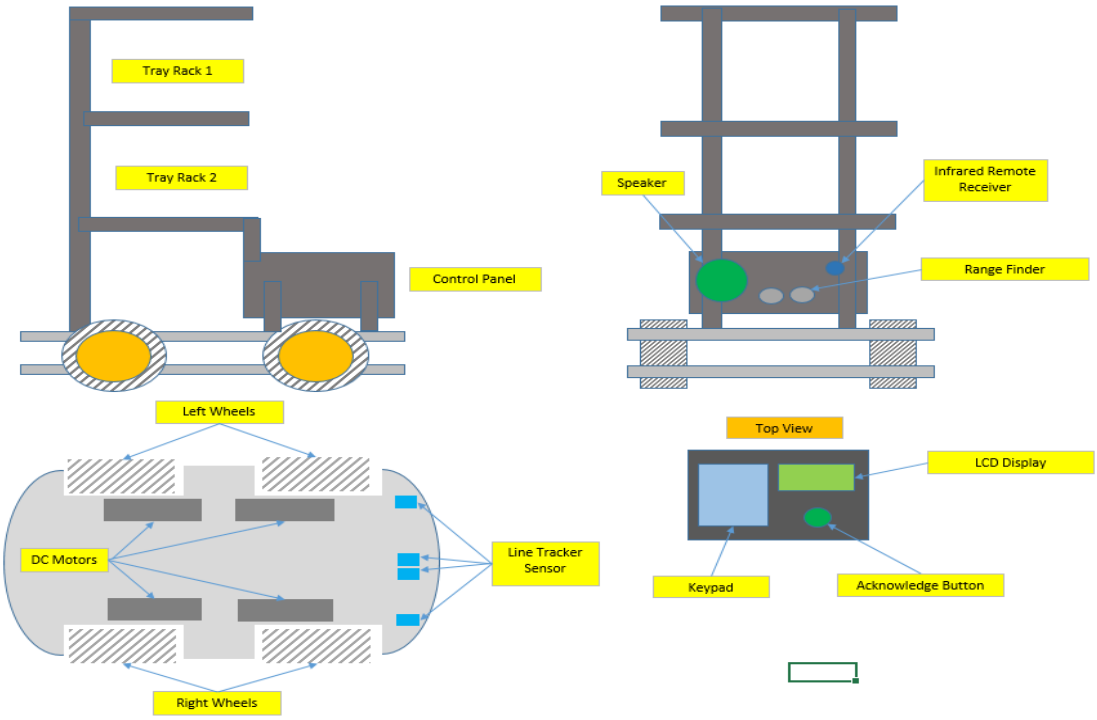


Figure 10: Mechanical design structure of Delibot

Software Control and Algorithm

The operational flow defines the behavior of the robot and the sequence of operation when performing the task assigned by the user. The robot is programmed to respond according to the sensor status and user inputs.

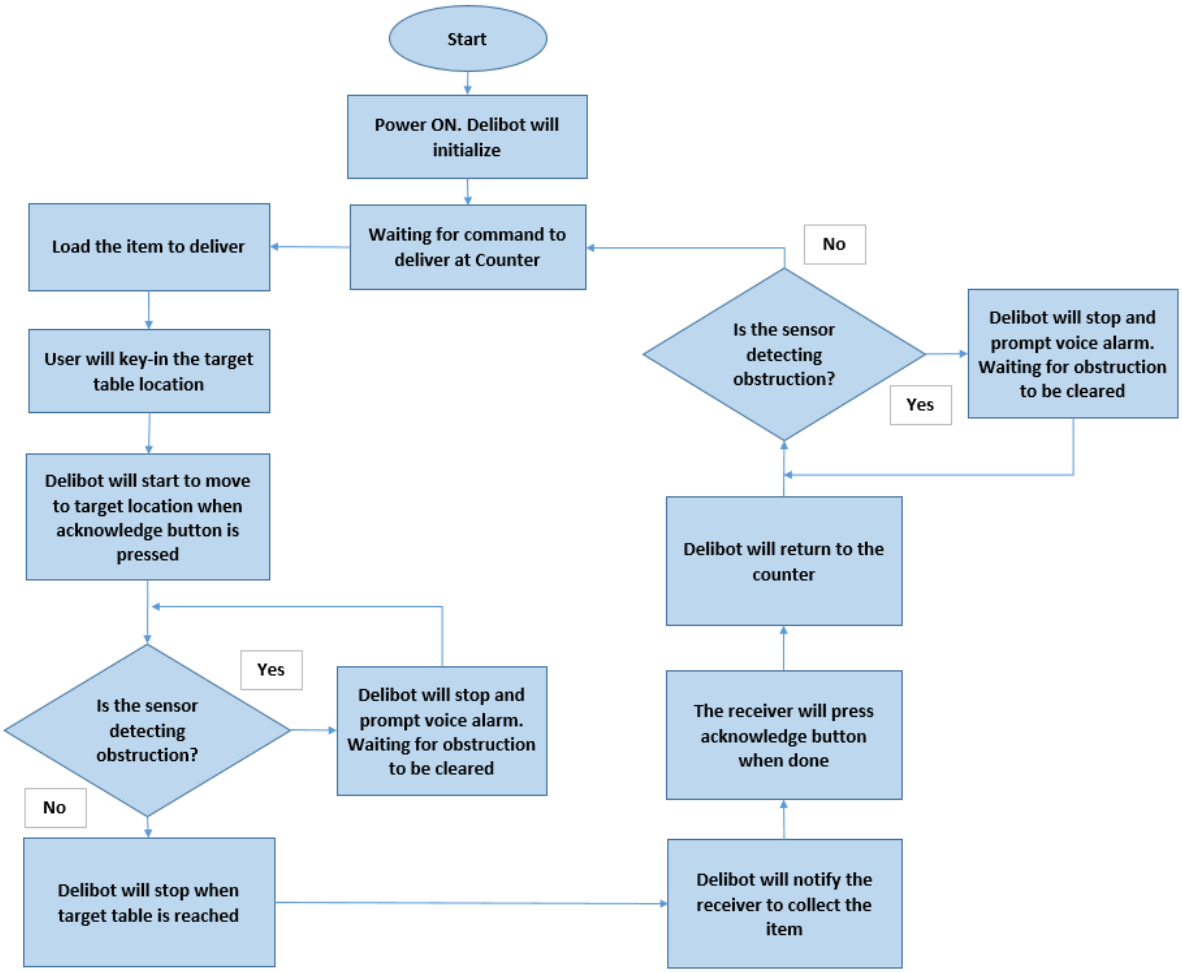
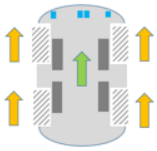
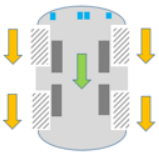
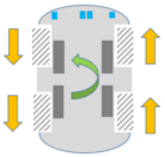
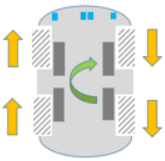
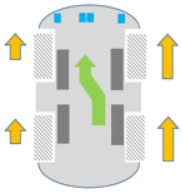
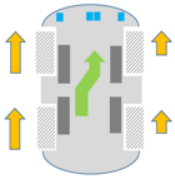


Figure 11: Operation Flow of Delibot

Movement Control. The robot should be able to move forward, turn left, turn right and turn in a reverse direction. The movement can be done by adjusting the direction of the motors to achieve the desired robot motion.

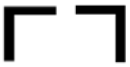
Table 1: Robot Movement Control Using Wheel Directions

No.	Robot Movement Control	Description	Wheel Directions	Applicable Track Patterns
1	Stop	All the wheels are not moving. The robot may stop when detected track pattern, no pattern or end of station		All Tracks / No tracks
2	Move Forward	All wheels are moving in forward (clock wise) direction. The robot will move in forward direction applied on straight tracks or cross tracks		Straight Tracks or cross tracks
3	Move Backward	All wheels are moving in reverse (counter clock wise) direction. This robot will move in backward direction applied on straight tracks or cross tracks		Straight Tracks
4	Turn Left	Left wheels are moving in reverse direction. Right wheels are moving in forward direction. The robot will rotate to the left from its current position, applied when changing directions on 90 deg tracks or Cross tracks		Left 90 deg track or Cross track
5	Turn Right	Left wheels are moving in forward direction. Right wheels are moving in reverse direction. The robot will rotate to the right from its current position, applied when changing directions on 90 deg tracks or Cross tracks		Right 90 deg tracks or Cross track

No.	Robot Movement Control	Description	Wheel Directions	Applicable Track Patterns
6	Tilt Left	Left wheels are moving in forward direction but at half speed. Right wheels are moving in forward direction at normal speed. The robot will tilt to the left from its current position while moving forward, applied to correct the robot position when the track is not aligned with right line sensor. Once the robot is aligned with the track, the robot will continue to move in forward direction		Straight Tracks and Curves
7	Tilt Right	Left wheels are moving in forward direction. Right wheels are moving in forward direction at half the speed. The robot will tilt to the right from its current position while moving forward, applied to correct the robot position when the track is not aligned with left line sensor. Once the robot is aligned with the track, the robot will continue to move in forward direction		Straight Tracks and Curves

Track Pattern. The robot can recognize different patterns using the line tracker sensors. The robot will respond accordingly to the track pattern. Unlike the existing AGV that uses a marker, Delibot was using the combination of pattern recognition and programmed directions to navigate. This will give Delibot flexibility in the movements and teaching of the robot.

Table 2: Track Pattern Designs and Robot Behavior

No.	Track Pattern	Description	Robot Behavior	Figures
1	Parking Stop	This pattern is used to identify if the robot has reached the parking station. Robot should stop and prompt message that it has reached the destination.	Stopped	
2	Horizontal / Vertical Straight line	When robot has identified the pattern, it can move in forward direction. The robot will move forward to follow the line until it reached a junction or parking stop pattern and stop	Moving Forward	
3	Curve line	Like the straight line, the robot will keep moving and follow the curve line until it reached a junction or park stop pattern and stop	Moving Forward	
4	90 deg Track	The robot will stop when this pattern is detected. It allows robot to rotate to left or right from its current position to change the direction.	Stopped	
5	T Track	The robot will stop when this pattern is detected. It allows robot to choose to rotate to left or right from its current position. During teach, the user must select the correct direction to follow.	Stopped	
6	Cross Track	The robot will stop when this pattern is detected. It allows robot to choose to rotate to left or right from its current position or to move forward. During teach, the user must select the correct direction to follow.	Stopped	

5. Testing of Prototype with the User

To set up Delibot in a working environment, a track should be lay-out on the workspace. The design should allow the robot to navigate around the kitchen and customer's table. For the demonstration purpose, the track design below was used in the Delibot demonstration video. (Fig. 15)

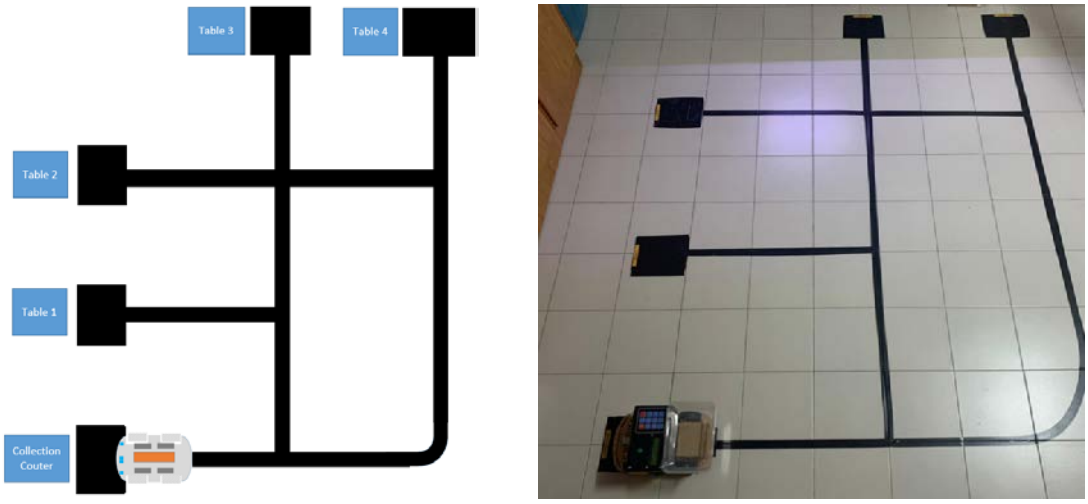


Fig 12. Delibot track layout for testing

There were two types of robot users, the operator, and the supervisor. The supervisor is allowed to teach the robot's program (Teach Mode) and command the robot to deliver (Run mode). The operator is allowed to command the robot to deliver in Run Mode only.
The robot's directions for each table were required to program and saved into the robot's memory. This function can be done when a robot is in Teach Mode using an infrared remote controller. (See Delibot Demo video - Teach Function)

Once the robot has completed the programming, it should be set to Run mode. In Run mode, the user can select a drop-off point from the collection counter to the selected table no. This could be done by key-in the table no. using the built-in keypad (operator user) or infrared remote controller (supervisor user) (See video: Delibot Demo video - Delivery Function)

6. Development of Assessment Instrument

To approve or buy off the robot, checklists have been provided to verify if the robot development was following the specifications defined during the

designing stage. The checklist includes Diagnostics and Hardware Testing, Delivery function test, Teach Track Path function, and Safety Features Test. (See Appendices)

7. Analysis of Assessment Data

Based on the testing and assessment, all the conditions defined in the checklist were assessed and able to pass. Although the intended design is a small-scale prototype, the proof of concept in the design was able to achieve the perspective that Delibot can travel autonomously by using the tracks and programmed directions and can be taught the program path using the remote control. The addition of voice messages proved to make the robot more user-friendly to non-technical users. The obstacle detection using the range finder was working well to keep the robot safe while traveling.

8. Revision of Prototypes

During the testing, there were some areas in the design that were highlighted by the user for improvement.

- Use metal structure and high-power DC motors to be able to carry and support heavier loads. The mechanical design can be improved in a large-scale design (next prototype) as the current prototype was intended to be used for proof of concept.

- Adjust the line tracker sensor positions for better and stable detection of track patterns.

- Improve the motion control like PID so that it can travel at a constant speed without wobbling. Adjustments were made in the software.

- Change the wheel design or material to prevent slippage causing the robot to move out of track position. The control system software was able to compensate for the position error but it would be better to fix the mechanical to make the robot stable. The mechanical change will be implemented in the next prototype.

C. User Testing and Assessment Plan

To test and verify the robot's functionality, a buy-off checklist has been prepared. The checklist contains the test cases and the expected result. If the expectation matched the robot's response, the test case is considered "Passed". The checklist should include robot diagnostics test, delivery function test, and teach function test. It needs to be updated if new bugs or features are added to the design.

Table 3 : User Test Cases and Results

No.	Test Case	Expected results	Status	Remarks
1	**Diagnostics and Hardware Testing**			The objective of this test is to verify if the hardware peripherals are functional and communicating with the microcontroller module. It is important to check if each hardware is working properly to ensure smooth operation when robot is performing its delivery task.
a	Power ON the robot (Start-up)	Robot should proceed to Run Mode after initialization	Passed	To check if microprocessor and software is working.
b	LCD Test	Robot should displaq message "Delibot V1.0" in the LCD upon Power ON	Passed	To check if LCD is functional
c	Audio Player Test	Voice message "My name is Delibot" should be played upon Power ON	Passed	To check if Audio module is functional
d	Keypad Test (Operator User)		Passed	To check if built-in Keypad is functional
	Press No. 1 in the Keypad	LCD will displaq drop-off point in Table No. 1 Voice message "Delibot will deliver in Table 1" should be played	Passed	
	Press No. 2 in the Keypad	LCD will displaq drop-off point in Table No. 2 Voice message "Delibot will deliver in Table 2" should be played	Passed	
	Press No. 3 in the Keypad	LCD will displaq drop-off point in Table No. 3 Voice message "Delibot will deliver in Table 4" should be played	Passed	
	Press No. 3 in the Keypad	LCD will displaq drop-off point in Table No. 4 Voice message "Delibot will deliver in Table 4" should be played	Passed	
	Press # or *	No response. Operator Users are not allowed to navigate other menu	Passed	
e	Infrared Remote Control Test (Supervisor User)			To check if Infrared Remote Control is working
	Press No. 1 in the Remote Control	LCD will displaq drop-off point in Table No. 1 Voice message "Delibot will deliver in Table 1" should be played	Passed	
	Press No. 2 in the Remote Control	LCD will displaq drop-off point in Table No. 2 Voice message "Delibot will deliver in Table 2" should be played	Passed	
	Press No. 3 in the Remote Control	LCD will displaq drop-off point in Table No. 3 Voice message "Delibot will deliver in Table 4" should be played	Passed	
	Press No. 3 in the Remote Control	LCD will displaq drop-off point in Table No. 4 Voice message "Delibot will deliver in Table 4" should be played	Passed	
	Press #	Robot will switch Teach mode. LCD will displaq Teach Menu. Pressing # again when in Teach Mode will switch back to Run Mode	Passed	
	Press * (Functional during Teach>Program and Teach>Location)	This function serves as <Enter> button to acknowledge changes in the program used during Teach	Passed	
	Arrow Up	The robot will move forward	Passed	
	Arrow Left	The robot will turn left	Passed	
	Arrow Right	The robot will turn right	Passed	
	Arrow Down	The robot will turn around	Passed	
f	Line Tracker Sensor Test			If sensor fails to detect the on-track and off-track status, the sensor need to adjust the sensitivity by rotating the trimmer in the sensor PCB to calibrate
	Move Left Line Detect Sensor to black track	Sensor LED should turn OFF when track is detected	Passed	
	Move Left Line Detect Sensor to outside black track	Sensor LED should turn ON when track is not detected	Passed	
	Move Right Line Detect Sensor to black track	Sensor LED should turn OFF when track is detected	Passed	
	Move Right Line Detect Sensor to outside black track	Sensor LED should turn ON when track is not detected	Passed	
	Move Left Tracker Sensor to black track	Sensor LED should turn OFF when track is detected	Passed	
	Move Left Tracker Sensor to outside black track	Sensor LED should turn ON when track is not detected	Passed	
	Move Right Tracker Sensor to black track	Sensor LED should turn OFF when track is detected	Passed	
	Move Right Tracker Sensor to outside black track	Sensor LED should turn ON when track is not detected	Passed	

IV. Results and Discussion

The checklist is considered a valid test report as it contains the items to test and the result of testing. In an equipment manufacturing environment, buy-off testing is done to ensure that every product assembled is working 100% before releasing it to the customer.

V. Conclusion and Recommendation

If the design will be implemented on a production scale, the following modifications are suggested for smooth operations:

- Use metal structure and high-power DC motor to be able to carry and support heavier loads.

- Improve the motion control like PID so that it can travel at a constant speed without wobbling. Adjustments were made in the program. Some mechanical parts like structure and wheel design need to change.

- Change the wheel design or material to prevent slippage causing the robot to move out of track position. The control system software was able to compensate for the position error but it would be better to fix the mechanical to make the robot stable

- Add a built-in charging circuit to charge the battery without removing it inside the robot.

- Increase the battery capacity so that the robot can last more than 8 hours of working without charging.

VI. References

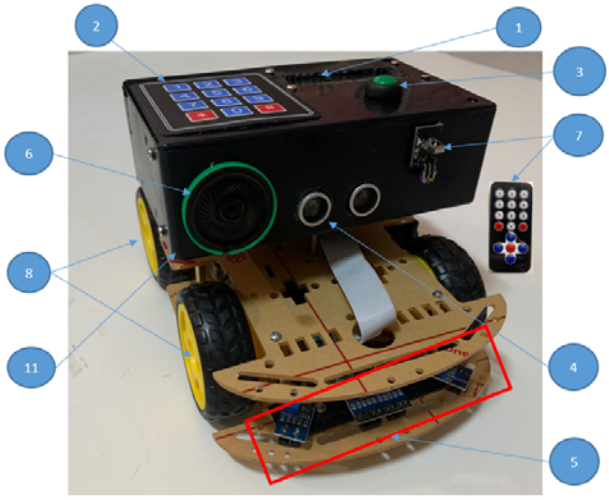
Building a Magnetic Track Guided Vehicle (2020)
https://www.roboteq.com/applications/all-blogs/18-building-a-magnetic-track-guided-agv
Automated Guided Vehicle (2021)
https://www.agvnetwork.com/what-is-automated-guided-vehicle-agv-robot
Benefits of automated guided vehicles (2016)
https://www.conveyco.com/7-cost-saving-benefits-automated-guided-vehicles-agvs/
Top 5 Ways Robotics Is Changing the Food Industry (2021)
https://blog.robotiq.com/top-5-ways-robotics-is-changing-the-food-industry

Covid 19 Policies : Depart of Health, Philippines (2021-2022)
https://doh.gov.ph/COVID-19-policies

Appendices

A. Deliverables and Milestones

1. Prototype



Parts of the Robot		
No.	Description	Function
1	LCD Display	Display the robot message and system menu in text
2	Keypad	Key-in the table no. to deliver the food
3	Acknowledge button	Press to acknowledge start of delivery and collection done
4	Range finder sensor	Detects object blocking the path and stop the motion
5	Line Tracker Sensor	Detects the line to follow
6	Speaker	Driven by audio module to announce voice message to the people around the environment
7	IR Remote Control Tx/Rx	Receives command from IR Remote Control Transmitter during Run and Teach Mode
8	Motor Driven Wheels	Drives the robot to move to follow the line path
9	Battery Pack	Provides 12VDC power to the robot
10	Motor Driver	Drives the motor driven wheels
11	Controller Box	Contains the microcontroller and audio module

2. Complete Program Listing

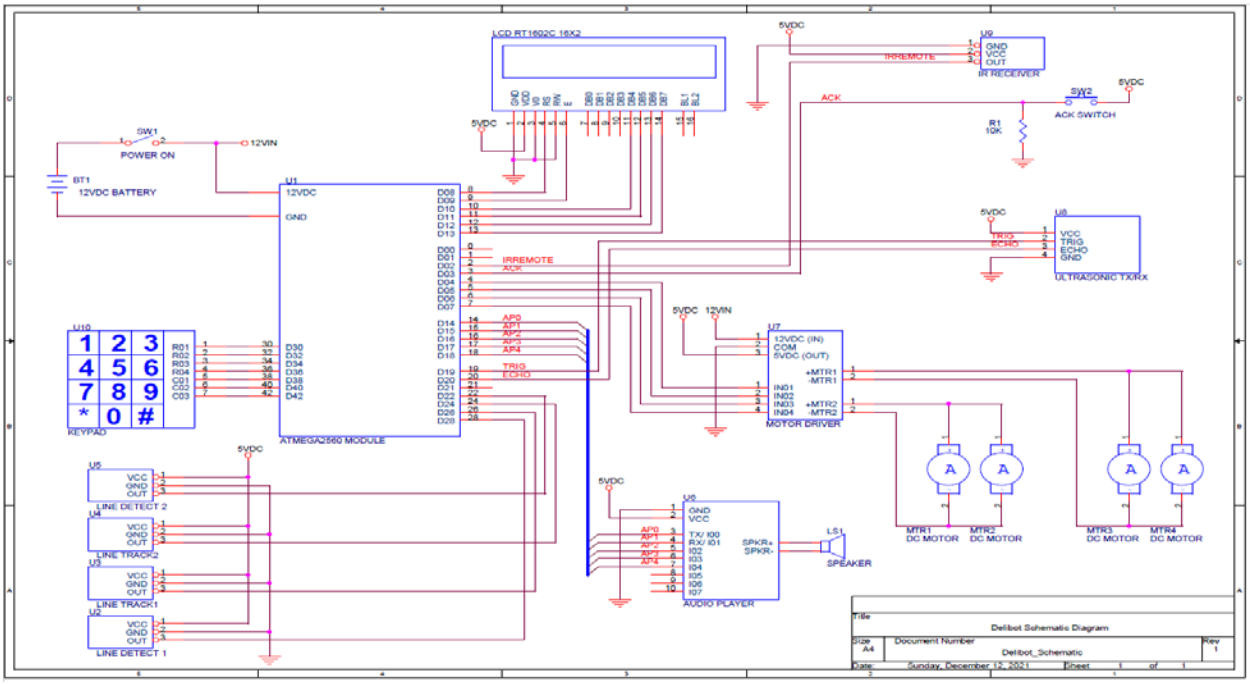
See Appendix A: Delibot System Source Code Listing V1.0

3. Technical References

A. Final System Specifications

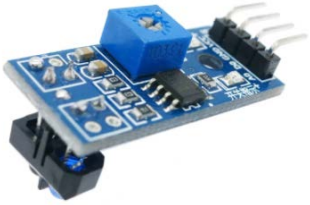
The robot is controlled by an Arduino Mega2560 microcontroller board based on an 8-bit ATMega2560 processor. It has a clock speed of 16MHz and 256kb Flash memory for storing the program. It has a built-in 4kb of EEPROM and 8kb SRAM for storing variables. It has an operating voltage of 5V. The controller is downloaded with system source code once to make it operational.

B. Schematic Diagram



C. Hardware and Software

Controller. The Arduino Mega2560 is a microcontroller board based on the ATmega2560 processor. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.	
Motor Driver. Motor driver board module (Model: HNV L298N) based from L298N DC Motor driver IC. It is capable of driving	

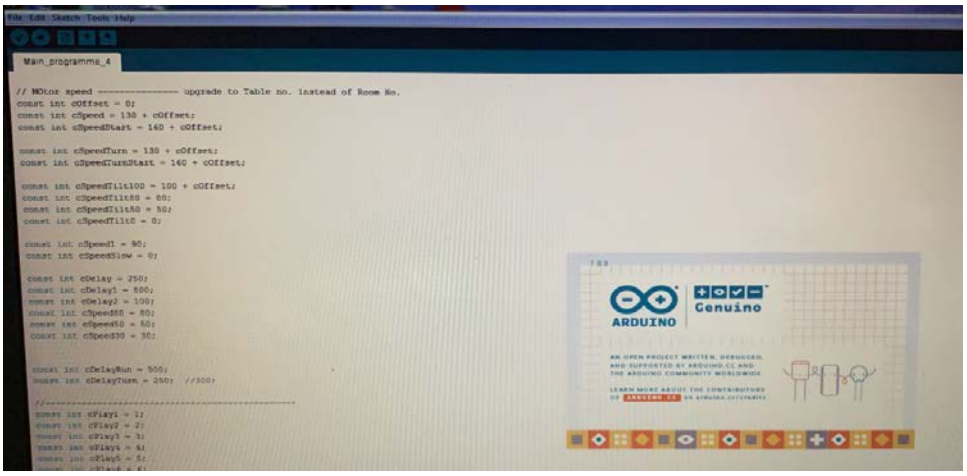
two motors in forward and reverse directions of different speeds using Pulse width modulation.	
LCD Display. (Model: LCD1602) LCD module with green screen and maximum of 16x2 Characters that can display	
IR remote control module. Infrared 17-Key IR Wireless Remote Control Receiver Module	
Voice Playback Module board. MP3 Music Player (Model No. DY-SV5W) 5W MP3 Playback Serial Control SD/TF Card	
Line tracker sensor. Infrared Reflective Sensor **(**Model No. TCRT5000) IR Photoelectric Switch Barrier Line Track Module	
Ultrasonic sensor Transmitter/receiver. TZT Ultrasonic sensor (Model No. HC-SR04) for distance and range detection	
Keypad. Consists of 12 keys (3x4) membrane switch connected to the controller	
4 Wheel Drive Smart car module. Function as the mechanical structure of the robot. It holds the control modules, dc motors and the food to be delivered using 2 layer acrylic plates	

D. Maintenance Plan for Software System

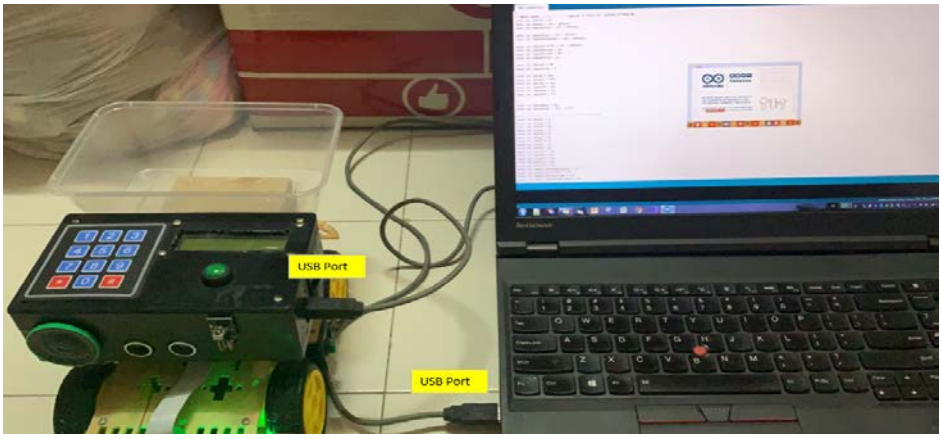
The robot's program does not require any software maintenance after the program was downloaded on the controller's non-volatile memory. In case of the controller breakdown and needed to be replaced, the Delibot system source code needs to be downloaded. Below are the procedures for downloading the source code to the robot.

The Delibot system source code is written in a text file and can be downloaded to the Arduino ATMel2560 board using the Sketch software. Below is the procedure for downloading Delibot source code to the robot.

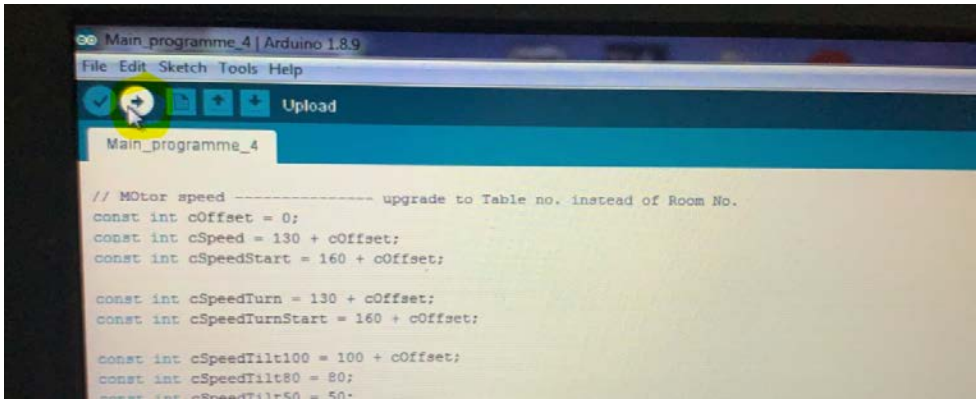
- Open the Sketch application and load the Delibot source code



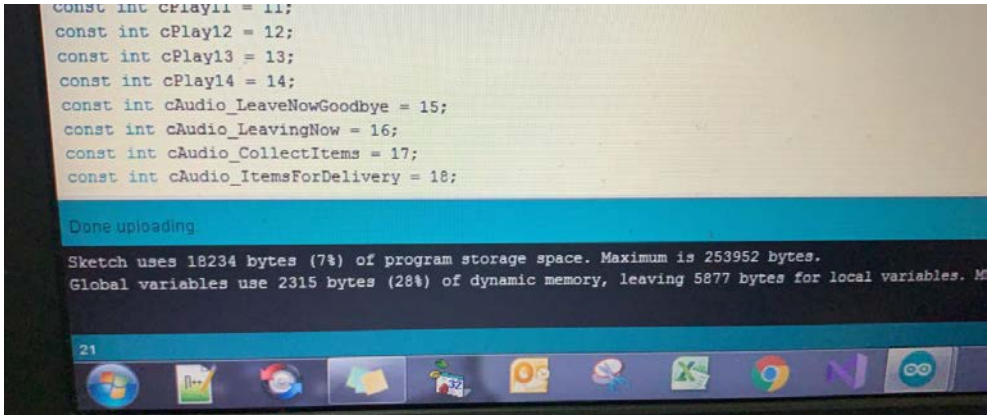
- Connect a USB cable from Delibot's USB port to the laptop's USB port



- Click the upload button. The application will connect to the robot and start uploading the program



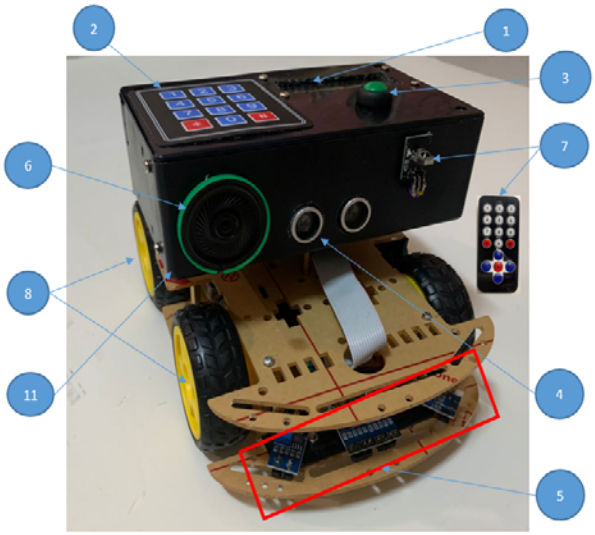
- The upload is complete, the message “Done uploading” will appear



- Delibot will initialize and run the uploaded code

4. User Manual

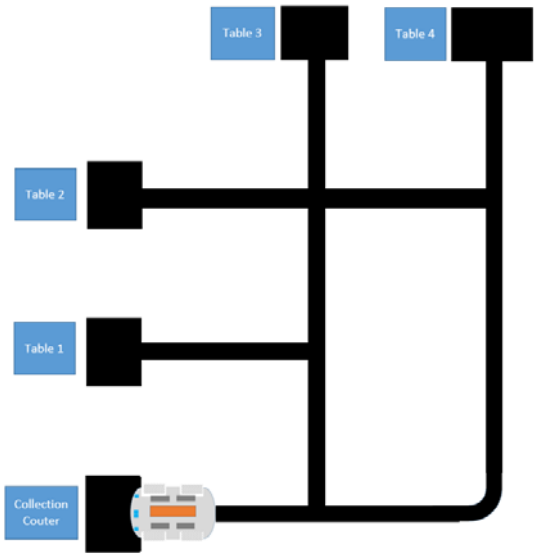
Below are the parts of the robot for system operations and their functions.



Parts of the Robot		
No.	Description	Function
1	LCD Display	Display the robot message and system menu in text
2	Keypad	Key-in the table no. to deliver the food
3	Acknowledge button	Press to acknowledge start of delivery and collection done
4	Range finder sensor	Detects object blocking the path and stop the motion
5	Line Tracker Sensor	Detects the line to follow
6	Speaker	Driven by audio module to announce voice message to the people around the environment
7	IR Remote Control Tx/Rx	Receives command from IR Remote Control Transmitter during Run and Teach Mode
8	Motor Driven Wheels	Drives the robot to move to follow the line path
9	Battery Pack	Provides 12VDC power to the robot
10	Motor Driver	Drives the motor driven wheels
11	Controller Box	Contains the microcontroller and audio module

Line Track Set-Up

Before the robot can be put into operation, the line track mapping should be planned and installed. Parking stations for collection counter and table location must be identified. The tracks and parking stations can be connected using black tape. Below is an example of the track layout used in this project.



System Operations

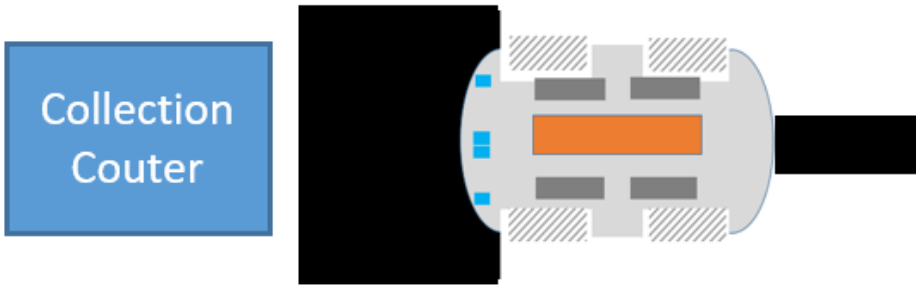
Delibot has two modes of operations, the Teach mode, and Run mode.

Teach mode. This mode allows the robot to program the path where the robot will follow to reach different table locations. For each table location, different sets of the path can be set when going to and from the table and collection counter.

Run Mode. This mode allows the robot to perform a delivery function by following the program done during teach mode. The user is allowed to key in the specific table location where items will be delivered. Upon confirmation after pressing the acknowledge button, the robot will move to the table location to deliver the item. When the robot reached the table, it will stop and play a message to the customer to collect the item. When items are collected and acknowledge button is pressed, the robot will move back to the collection counter.

Teach Mode Set-up

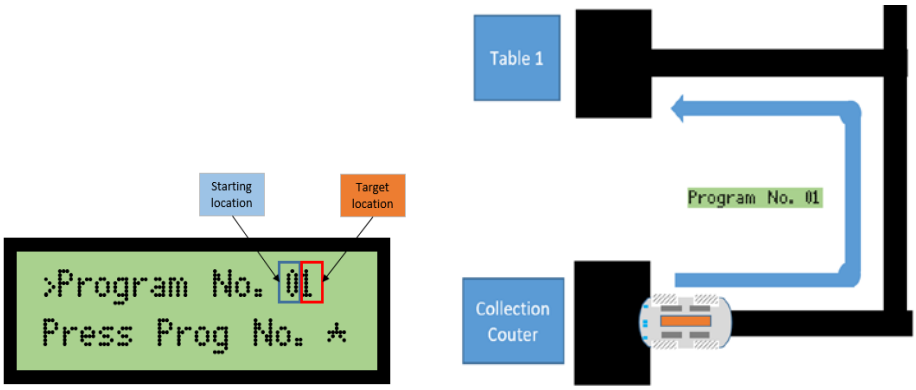
The robot's starting position is at the Collection Counter (Table 0). All 4 line tracker sensors should be detecting the black lines of the park position. In teaching the robot, the IR remote controller should be used.



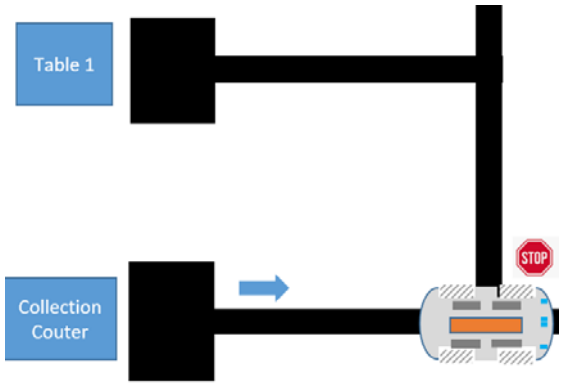
- To proceed to the Teach menu, press #

- Press 2 to define the path of the program

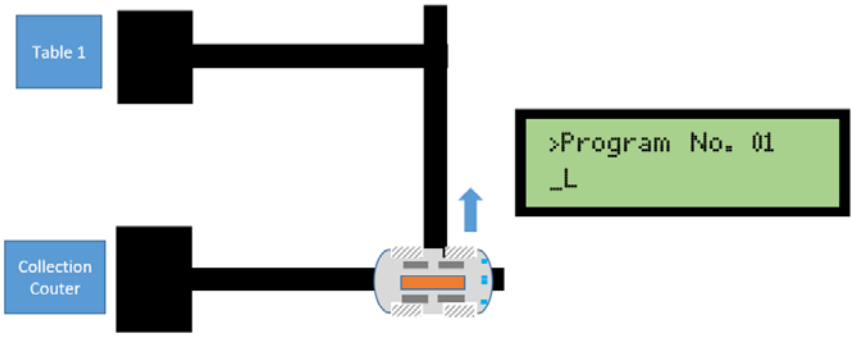
- Key-in the Program No. The first digit represents the starting location. The second digit represents the destination. For the example, the program below will teach the path from Collection Counter (Table 0) to Table 1 in the next step. Press * to proceed to the next step



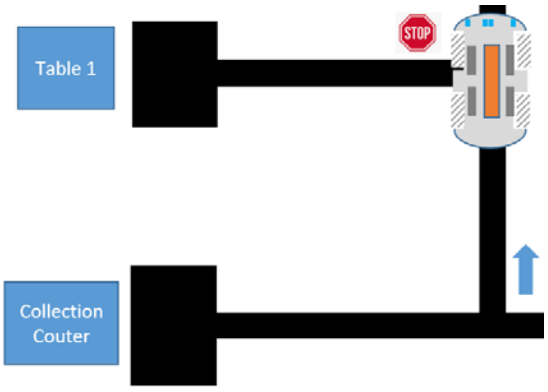
- When * is pressed, the robot will turn around and move forward. The robot will stop when a junction is detected.



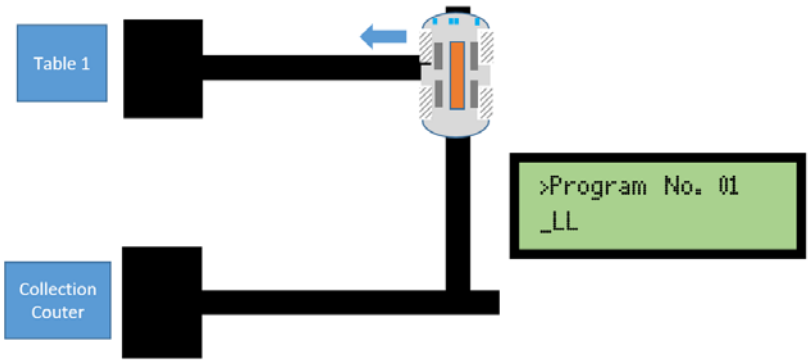
- Press Left Arrow to command the robot to turn Left. The LCD will display the updated command list.



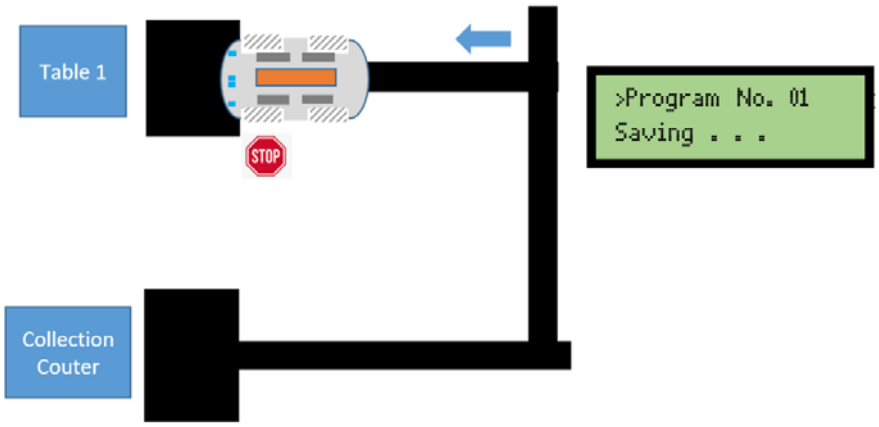
- Once the Left Arrow is pressed, the robot will turn left and move forward. The robot will stop when the next junction is detected



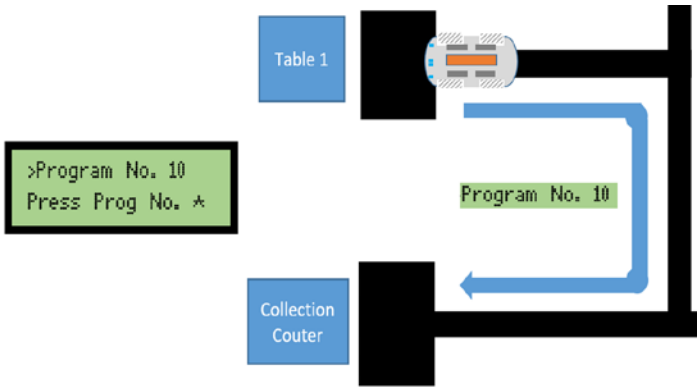
- Press Left Arrow to command the robot to turn Left. The LCD will display the updated command list.



- Once the Left Arrow is pressed, the robot will turn left and move forward. The robot will stop when the next junction is detected. If the parking station is detected, the robot will stop and save the command list in the controller's non-volatile memory. The teach for Program 01 is considered complete.

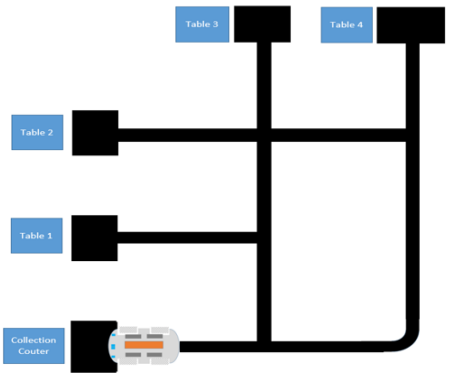


- To teach the robot to return from Table 1 to Collection Counter, key in the program no. as Program 10 and proceed with the next to teach steps



- To complete the teach for 4 Tables track layout, the following path directions should be performed and saved into the controller's non-volatile storage.

Program No.	Starting Location	Target Location	Path Directions
01	0	1	LL
10	1	0	RR
02	0	2	LFL
20	2	0	RFR
03	0	3	LFF
30	3	0	FFR
04	0	4	FFF
40	4	0	RLFR



Run Mode Set-up

From the main menu, the robot can perform the delivery by key-in the selected table no. using the keypad or IR remote controller. The starting location of the robot as default is the Collection Counter. The robot will proceed to Run mode as default when power is turned ON.

- To command the robot to deliver to Table 1, press “1”. The LCD will display the selected drop-off point.

- Press the Green button or “*” to confirm. The robot will start to move for delivery.

- The robot will stop when it reached Table 1. It will prompt a voice message to collect the items

- After collecting the items, the user should press the green button to confirm the collection is finished. The robot will start to move to return to the collection counter

- The robot will stop when it reached the collection counter. It will wait for the next command.

- The robot has a range finder sensor that detects any obstruction along its path during travel. The robot will stop if there’s a blockage along its path and prompt a voice message to remove it. Once the obstruction is removed, the robot will continue its way.

A. Budget

The project was developed using a small-scale prototype to lower the cost. The proponent has the skillsets to integrate the components to build a functional delivery robot according to the scope and features.

Table 4: Estimated Budget Cost

No.	Particulars	Remarks	Estimated Cost (P)
	Labor Cost		
1	Program manager	Gathers requirements and design concept	Proponent
2	Electrical and Control Designer	Select eletrical components and design schematics	Proponent
3	Mechanical Designer	Select mechanical comonents and design mechanical structure	Proponent
4	System Integrator / Software Control Engineer	Integrate Electrical and Mechanical designs to Controller to control the robot according to concept design using software	Proponent
	Non-Labor Cost		
5	Electrical Components	Include components used in R&D, prototyping, testing and evaluation	5,000.00
6	Mechanical Components	Include components used in R&D, prototyping, testing and evaluation	4,000.00
7	Transportation		1,000.00
8	Miscellaneous (Documentation, Manuals, etc)		2,000.00
	Total Estimated Budget		12,000.00

B. Qualifications

The author is a graduate of Computer Engineering with more than 20 years of experience in Automation working as a software and mechatronic engineer. He had worked on different projects (semiconductor and manufacturing equipment design) from multinational companies. He is currently based in Singapore working as a Senior Software Engineer.

Working with robotic projects was usually done by a team with different skills that comprise a mechanical designer, electrical designer, software engineer, and project leader. The experience of integrating the electrical and mechanical components and adding software algorithms to control the robot is challenging. Requirement gathering, designing, programming, and testing would be time-consuming and costly as modifications of design may take place during testing if a problem occurs. Different cases should be tested to know the limitations of the design and features to improve.

The learning agenda for this project is to study and apply robotics automation in the food industry. The challenge is to integrate electrical, mechanical, and software and make them work together to perform tasks that could help humans and make life better.

C. Resources

The following are the resources needed for documentation and development of the project:

- Laptop computer for software development
- Internet access for research and references
- Electronic tools (Multi-meter, Soldering iron, pliers, and cutters)
- Mechanical tools (Electric drill, Saw, Diamond file set, Screwdrivers)
- PCB Prototyping board and wires
- Power supply (12VDC)

Appendix A: Delibot System Source Code Listing V1.0

```
const int cOffset = 0;
const int cSpeed = 130 + cOffset;
const int cSpeedStart = 160 + cOffset;

const int cSpeedTurn = 130 + cOffset;
const int cSpeedTurnStart = 160 + cOffset;

const int cSpeedTilt100 = 100 + cOffset;
const int cSpeedTilt80 = 80;
const int cSpeedTilt50 = 50;
const int cSpeedTilt0 = 0;

const int cSpeed1 = 90;
const int cSpeedSlow = 0;

const int cDelay = 250;
const int cDelay1 = 800;
const int cDelay2 = 100;
const int cSpeed80 = 80;
const int cSpeed50 = 50;
const int cSpeed30 = 30;


const int cDelayRun = 500;
const int cDelayTurn = 250;  //300;

//-------------------------------------------------
const int cPlay1 = 1;
const int cPlay2 = 2;
const int cPlay3 = 3;
const int cPlay4 = 4;
const int cPlay5 = 5;
const int cPlay6 = 6;
const int cPlay7 = 7;
const int cPlay8 = 8;
const int cPlay9 = 9;
const int cPlay10 = 10;
const int cPlay11 = 11;
const int cPlay12 = 12;
const int cPlay13 = 13;
const int cPlay14 = 14;
const int cAudio_LeaveNowGoodbye = 15;
const int cAudio_LeavingNow = 16;
const int cAudio_CollectItems = 17;
const int cAudio_ItemsForDelivery = 18;
const int cAudio_ProgramSave = 19;
const int cAudio_ProgramCancel = 20;
const int cAudio_DeliverRoom1 = 21;
const int cAudio_DeliverRoom2 = 22;
const int cAudio_DeliverRoom3 = 23;
const int cAudio_DeliverRoom4 = 24;
const int cAudio_DeliverRoom5 = 25;
const int cAudio_DeliverRoom6 = 26;
const int cAudio_ObstructionDetected = 27;
const int cAudio_ObstructionCleared = 28;
const int cAudio_RangeFinderON = 29;
const int cAudio_RangeFinderOFF = 30;
const int cPlay31 = 31;
```

```
//---- Using Keypad --------

//Module: Audio MP3 Player
const int Aud00 = 14;   //1
const int Aud01 = 15;   //2
const int Aud02 = 16;   //4/
const int Aud03 = 17;   //8
const int Aud04 = 18;   //16

//Module Range finder
const int echoPin = 19;          //echo Utlrasonic rangefinder HC-SR04
const int trigPin = 20;          //trigger Ultrasonic rangefinder HC-SR04


// Module: Infrared remote
const int IrReceiverPin = 2;        // Turn the variable "IrReceiverPin"
const int IN3 = 3;                  //Acknowledge button

//MOdule motor driver
const int PWM1 = 4;    //rev L
const int PWM2 = 5;    //fwd L
const int PWM3 = 6;;   //rev R
const int PWM4 = 7;    //fwd L

// MOdule Proximity Sensor


const int SensorLeft = 28;
const int SensorTrackLeft = 26;
const int SensorTrackRight = 24;
const int SensorRight = 22;



// IR remote control Command
const int cLeft = 1;
const int cRight = 2;
const int cUp = 3;
const int cDown = 4;
const int cOk = 0;
const int cStar = 5;
const int cHash = 6;
const int cLoading = 0;
const int cRoom1 = 1;
const int cRoom2 = 2;
const int cRoom3 = 3;
const int cRoom4 = 4;


// EEPROM address
int key = 0;
byte command[10];       //Command list to execute
byte program[10][10][10]; //Multiple program lists
int cmdIndex = 0;
int cmdLastIndex = 0;
int iMenu = 0;
int iSource = 0;
int iDestination = 0;
int nSource = 0;
int nDestination = 0;
int nProgramNo = 0;

//Proximity SEnsor status
boolean IsSensorTrackLeftDetectON  = false;
```

```
boolean IsSensorTrackRightDetectON  = false;
boolean IsSensorLeftLineDetectON   = false;
boolean IsSensorRightLineDetectON  = false;
boolean IsMovingForward  = false;
boolean IsMovingReverse  = false;
boolean IsMovingLeft  = false;
boolean IsMovingRight  = false;
boolean TiltLeft  = false;
boolean TiltRight = false;
boolean TurnLeft  = false;
boolean TurnRight = false;
boolean lineDet = false;
boolean EnableSideSensor = false;
boolean IsLineDetected = false;
boolean IsRunMode = false;
boolean IsTeachMode = false;
boolean IsStartTeach = false;
boolean CanAddTeach = false;
boolean IsAtStation = false;
boolean IsTimeout = false;

//Range finder status
boolean IsBlocked = false;
boolean IsLatched = false;
boolean IsRangeEnabled = false;

//Timer
unsigned long lapMillis = 0;
unsigned long startMillis = millis();
unsigned long startMillis1 = millis();



// Initialise distance
long duration;
int distance = 200;
int distance1 = 200;
int distance2 = 200;
int distance3 = 200;
int distance4 = 200;
int distance5 = 200;

// LIBRARY to control arduino modules
#include <LiquidCrystal.h>
#include <IRremote.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
IRrecv irrecv(IrReceiverPin);         // create a new instance of "irrecv" and save this instance in variabele "IRrecv"
decode_results results;               // define the variable "results" to store the received button code
const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
byte rowPins[ROWS] = {31, 33, 35, 37}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {39, 41, 43}; //connect to the column pinouts of the keypad
```

```
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

//Function: PlayAudio
//Description: Command to play MP3 audio based fron iAudio No.
//
void PlayAudio(int iAudioNo = 0) {
 digitalWrite(Aud00, !(iAudioNo & 1)); digitalWrite(Aud01, !(iAudioNo & 2));   digitalWrite(Aud02, !(iAudioNo & 4));
 digitalWrite(Aud03, !(iAudioNo & 8));   digitalWrite(Aud04, !(iAudioNo & 16));   delay(500);
 digitalWrite(Aud00, HIGH);  digitalWrite(Aud01, HIGH);   digitalWrite(Aud02, HIGH);   digitalWrite(Aud03, HIGH);   digitalWrite(Aud04, HIGH);
}

//Function: LineTracing
//Description: aligns the robot on track
void LineTracing()
{
 IsSensorTrackLeftDetectON = digitalRead(SensorTrackLeft); //New Sensor
 IsSensorTrackRightDetectON = digitalRead(SensorTrackRight); //New Sensor
 IsSensorLeftLineDetectON =  digitalRead(SensorLeft);  //New sensor
 IsSensorRightLineDetectON = digitalRead(SensorRight); //New Sensor

 if (IsSensorTrackRightDetectON && IsSensorTrackLeftDetectON)     //Straight - Line Detected
 {
  lineDet = true;
  IsLineDetected = true;
  TiltLeft = false;
  TiltRight = false;
 }
 else if (IsSensorTrackLeftDetectON && IsSensorTrackRightDetectON == false)    //Tilt left
 {
  TiltLeft = false;
  TiltRight = true;
  lineDet = true;
  IsLineDetected = true;
 }
 else if (IsSensorTrackRightDetectON && IsSensorTrackLeftDetectON == false)    //Tilt right
 {
  TiltRight = false;
  TiltLeft = true;
  lineDet = true;
  IsLineDetected = true;
 }
 else if (IsSensorTrackRightDetectON == false  && IsSensorTrackLeftDetectON == false)  //Straight - No Line Detected, follow last command
 {
    if (TiltRight && lineDet)
    {
        if (IsLineDetected)
        {
          TiltRight = true;
          TiltLeft = false;
          IsLineDetected = false;
        }
    }
    else if (TiltLeft && lineDet)
    {
        if (IsLineDetected)
        {
          TiltRight = false;
          TiltLeft = true;
          IsLineDetected = false;
        }
    }
    else
    {
```

```
      TiltLeft = false;
      TiltRight = false;
    }
  }

}


// function: setup
// Description : HArdware configuration assigns the pin for input or output
void setup()
{
 Serial.begin(9600);               // Initialise the serial monitor
 pinMode (IN3, INPUT);
 pinMode (SensorTrackLeft, INPUT);
 pinMode (SensorTrackRight, INPUT);
 pinMode (SensorLeft, INPUT);
 pinMode (SensorRight, INPUT);
 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input
 pinMode (PWM1, OUTPUT);
 pinMode (PWM2, OUTPUT);
 pinMode (PWM3, OUTPUT);
 pinMode (PWM4, OUTPUT);
 pinMode (Aud00, OUTPUT);
 pinMode (Aud01, OUTPUT);
 pinMode (Aud02, OUTPUT);
 pinMode (Aud03, OUTPUT);
 pinMode (Aud04, OUTPUT);
 digitalWrite(Aud00, HIGH);
 digitalWrite(Aud01, HIGH);
 digitalWrite(Aud02, HIGH);
 digitalWrite(Aud03, HIGH);
 digitalWrite(Aud04, HIGH);

 lcd.begin(16, 2);
 lcd.setCursor (0, 0);
 lcd.print("I'm Delibot V1.0");
 Serial.println("Starting IR-receiver...");
 Serial.println("IR-receiver active");
 irrecv.enableIRIn();
 SetCommand(0,0);
 delay(2000);
 lcd.setCursor (0, 0);
 lcd.print(">Run Mode       ");
 digitalWrite(Aud00, HIGH);  digitalWrite(Aud01, HIGH);   digitalWrite(Aud02, HIGH);   digitalWrite(Aud03, HIGH);   digitalWrite(Aud04, HIGH);

 for (nSource=0;nSource<10;nSource++)
 {
  for (nDestination=0;nDestination<10;nDestination++)
  {
   for (nProgramNo=0;nProgramNo<10;nProgramNo++) // Able to put in 10 commands source(0-9), Destination(0-9)
   {
     program[nSource][nDestination][nProgramNo] = EEPROM.read(nSource*100+nDestination*10+nProgramNo);
     //EEPROM.write(nSource*100+nDestination*10+nProgramNo,0);
   }
  }
 }
 nSource = 0;
 nDestination = 0;
 nProgramNo = 0;

 TiltRight = false;
 TiltLeft = false;
```

```
 lineDet = false;
 IsLineDetected = false;
 Stop();
 lcd.setCursor (0, 1);
 lcd.print ( "Waiting for Item  ");
 PlayAudio(1);                    //-------------------------------------------------------------------------
}

//function: StopMotion
// description: Command the motor to stop
//            under teach mode, the command is added to the program table (  [program[iSource][iDestination][nProgramNo]) and update the lcd
display
void StopMotion()
{
 analogWrite(PWM1, 0);
 analogWrite(PWM2, 0);
 analogWrite(PWM3, 0);
 analogWrite(PWM4, 0);
 if (IsTeachMode)
 {
   if (CanAddTeach && IsStartTeach)
   {
     switch(key)
     {
      case cLeft:
               lcd.print ("R");
               IsStartTeach = false;
               program[iSource][iDestination][nProgramNo] = cLeft;
               nProgramNo++;
               break;
      case cRight:
               lcd.print ("L");
               IsStartTeach = false;
               program[iSource][iDestination][nProgramNo] = cRight;
               nProgramNo++;
               break;
      case cUp:
               lcd.print ("F");
               IsStartTeach = false;
               program[iSource][iDestination][nProgramNo] = cUp;
               nProgramNo++;
               break;
      case cDown:
               lcd.print ("R");
               IsStartTeach = false;
               program[iSource][iDestination][nProgramNo] = cDown;
               nProgramNo++;
               break;
     }
   }
 }
 else
 {
    lcd.setCursor (0, 1);
    lcd.print ( "Stopped ");
 }
}

// Function: Stop
// description: initialise the variables when stop
void Stop()
{
 StopMotion();
 IsMovingForward = false;
```

```
 IsMovingReverse = false;
 IsMovingLeft = false;
 IsMovingRight  = false;
 EnableSideSensor = false;
 CanAddTeach = false;
 key = cOk;
}

//function: SetCommand
// description: Copy command from program list based from Source and Destination
void SetCommand(int Source, int Destination)
{
 iSource = Source;
 iDestination = Destination;
 nProgramNo = 0;

 int i=0;
 while (i<10)
 {
  command[i] = program[iSource][iDestination][i];
  i++;
 }
 cmdIndex = 0;
}

//function:ReadInfraredKey
// description : read the signals from IR receiver and interpret as command to the robot
void ReadInfraredKey()
{

 unsigned long KeyValue = 999;

 if (irrecv.decode(&results)) {
  // Determine which button has been pressed
  // Resume the IR-receiver to listen for new signals
  irrecv.resume();
  KeyValue = results.value;
 }

 char keyp = keypad.getKey();
 if (keyp!=NO_KEY){
  switch (keyp) {
    case '*':
      KeyValue = 11;
      break;
    case '#':
      KeyValue = 10;
      break;
    case '0':
      KeyValue = 0;
      break;
    case '1':
      KeyValue = 1;
      break;
    case '2':
      KeyValue = 2;
      break;
    case '3':
      KeyValue = 3;
      break;
    case '4':
      KeyValue = 4;
      break;
    case '5':
```

```
      KeyValue = 5;
      break;
    case '6':
      KeyValue = 6;
      break;
    case '7':
      KeyValue = 7;
      break;
    case '8':
      KeyValue = 8;
      break;
    case '9':
      KeyValue = 9;
      break;
  }
}

 if (KeyValue==999) return;

  switch (KeyValue) {
    case 0x8C22657B:  // button right
    case 0xFF10EF:
      key = cRight;
      break;

    case 0x449E79F:  // button left
    case 0xFF5AA5:
      key = cLeft;
      break;

    case 0x3D9AE3F7:  // button up
    case 0xFF18E7:
      key = cUp;
      break;

    case 0x1BC0157B:  // button down
    case 0xFF4AB5:
      key = cDown;
      break;

    case 0x488F3CBB:  // button ok
      key = cOk;
      StopMotion();
      break;

    case 0xE318261B:   //1   0 -> 1
    case 0xFFA25D:
    case 1:

      if (iMenu==1) // TEACH
      {
        iMenu=2;    // TEACH > SET LOCATION
        lcd.setCursor (0, 0);
        lcd.print ( ">Set Location: ");
        lcd.print ( iSource );
        lcd.setCursor (0, 1);
        lcd.print ( "Press Tbl no.     ");
        break;
      }
      if (iMenu==2)
      {
        iSource = 1;  // TEACH > SET LOCATION > KEY IN LOCATION NUMBER
        lcd.setCursor (0, 0);
        lcd.print ( ">Set Location: ");
```

```
   lcd.print ( iSource );
   lcd.setCursor (0, 1);
   lcd.print ( "Press Tbl no.    ");
   SetCommand(iSource,0);
   break;
  }
  if (iMenu==3)
  {
   iSource = iDestination;        // TEACH > PROGRAM
   iDestination = 1;
   lcd.setCursor (0, 0);
   lcd.print ( ">Program No.");
   lcd.print (iSource);
   lcd.print (iDestination);
   lcd.setCursor (0, 1);
   lcd.print ( "Press prog no. *");
   break;
  }
  if (iMenu==0)
  {
   lcd.setCursor (0, 1);
   lcd.print ( "Drop-off: Table1"); // RUN MODE > Table 1
   PlayAudio(cAudio_DeliverRoom1);   // delivery to room 1
   SetCommand(0,1);
   break;
  }
  break;

case 0x511DBB:   //2
case 0xFF629D:
case 2:
  if (iMenu==0)
  {
   lcd.setCursor (0, 1);
   lcd.print ( "Drop-off: Table2");  // RUN MODE > Table 2
   PlayAudio(cAudio_DeliverRoom2);  // delivery to room 2
   SetCommand(0,2);
   break;
  }
  if (iMenu==1)                      // TEACH > PROGRAM
  {
   iMenu = 3;
   lcd.setCursor (0, 0);
   lcd.print ( ">Program No.");
   lcd.print ( iSource );
   lcd.setCursor (0, 1);
   lcd.print ( "Press prog no.    ");
   break;
  }
  if (iMenu==2)                      // TEACH > SET LOCATION
  {
   iSource = 2;
   lcd.setCursor (0, 0);
   lcd.print ( ">Set Location: ");
   lcd.print ( iSource );
   lcd.setCursor (0, 1);
   lcd.print ( "Press Table No. ");
   SetCommand(iSource,0);
   break;
  }
  if (iMenu==3)                       // TEACH > SET LOCATION > KEY IN LOCATION NUMBER
  {
   iSource = iDestination;
   iDestination = 2;
```

```
   lcd.setCursor (0, 0);
   lcd.print ( ">Program No.");
   lcd.print (iSource);
   lcd.print (iDestination);
   lcd.setCursor (0, 1);
   lcd.print ( "Press prog no. *");
   break;
  }
  break;

case 0xEE886D7F: //3
case 0xFFE21D:
case 3:
  if (iMenu==0)
  {
   lcd.setCursor (0, 1);
   lcd.print ( "Drop-off: Table3");                    // RUN MODE > Table 3
   PlayAudio(cAudio_DeliverRoom3);    // delivery to Table 3
   SetCommand(0,3);
  }
  if (iMenu==2)
  {
   iSource = 3;
   lcd.setCursor (0, 0);
   lcd.print ( ">Set Location: ");
   lcd.print ( iSource );
   lcd.setCursor (0, 1);
   lcd.print ( "Press Room no.     ");
   SetCommand(iSource,0);
  }
  if (iMenu==3)
  {
   iSource = iDestination;
   iDestination = 3;
   lcd.setCursor (0, 0);
   lcd.print ( ">Program No.");
   lcd.print (iSource);
   lcd.print (iDestination);
   lcd.setCursor (0, 1);
   lcd.print ( "Press prog no. *");
   break;
  }
  break;

case 0xFF22DD:   // 4
case 0x52A3D41F:
case 4:
  if (iMenu==0)
  {
   lcd.setCursor (0, 1);
   lcd.print ( "Drop-off: Table4");
   PlayAudio(cAudio_DeliverRoom4);    // delivery to Table 4
   SetCommand(0,4);
  }
  if (iMenu==2)
  {
   iSource = 4;
   lcd.setCursor (0, 0);
   lcd.print ( ">Set Location: ");
   lcd.print ( iSource );
   lcd.setCursor (0, 1);
   lcd.print ( "Press Tbl No.    ");
   SetCommand(iSource,0);
  }
```

```
 if (iMenu==3)
 {
  iSource = iDestination;
  iDestination = 4;
  lcd.setCursor (0, 0);
  lcd.print ( ">Program No.");
  lcd.print (iSource);
  lcd.print (iDestination);
  lcd.setCursor (0, 1);
  lcd.print ( "Press prog no. *");
  break;
 }
 break;

case 0XFFE01F:   // 7 > RANGEFINDER WILL ON
case 0xF076C13B:
 PlayAudio(cAudio_RangeFinderON);
 IsRangeEnabled = true;
 break;

case 0XFFA857:   // 8 > RANGEFINDER WILL OFF
case 0xA3C8EDDB:
 PlayAudio(cAudio_RangeFinderOFF);
 IsRangeEnabled = false;
 break;


case 0xFF9867:    //0
case 0x97483BFB:
case 0:
 if (iMenu==2)
 {
  iSource = 0;
  lcd.setCursor (0, 0);
  lcd.print ( ">Set Location: ");
  lcd.print ( iSource );
  lcd.setCursor (0, 1);
  lcd.print ( "Press Tbl no.     ");
  break;
 }
  if (iMenu==3)
 {
  iSource = iDestination;
  iDestination = 0;
  lcd.setCursor (0, 0);
  lcd.print ( ">Program No.");
  lcd.print (iSource);
  lcd.print (iDestination);
  lcd.setCursor (0, 1);
  lcd.print ( "Press prog no. *");
  break;
 }
 break;

case 0xC101E57B:    // * > ENTER MENU
case 0xFF6897:

 if (iMenu==0)
 {
  key = cStar;
  break;
 }
 if (iMenu==3)
 {
```

```
      IsStartTeach = false;
      nProgramNo = 0;
      iMenu = 4;
      lcd.setCursor (0, 0);
      lcd.print ( ">Program No.");
      lcd.print (iSource);
      lcd.print (iDestination);
      lcd.setCursor (0, 1);
      lcd.print ( "                ");
      lcd.setCursor (0, 1);
      lcd.print ( "_");
      key = cDown;
      break;
     }
     break;

    case 0xF0C41643:   // #
    case 0xFFB04F:
     key = cHash;
     if (iMenu==0)
     {
      iMenu=1;          //Teach
      lcd.setCursor (0, 0);
      lcd.print ( ">Teach Mode       ");
      lcd.setCursor (0, 1);
      lcd.print ( "1-Loc 2-Program    ");
      IsTeachMode = true;
      IsStartTeach = false;
     }
     else if (iMenu==1)
     {
      iMenu=0;          //Run Mode
      lcd.setCursor (0, 0);
      lcd.print ( ">Run Mode         ");
      lcd.setCursor (0, 1);
      lcd.print ( "                  ");
      IsTeachMode = false;
      IsStartTeach = false;
     }
     else if ((iMenu==2) || (iMenu==3))
     {
      iMenu=1;          //Teach
      lcd.setCursor (0, 0);
      lcd.print ( ">Teach Mode       ");
      lcd.setCursor (0, 1);
      lcd.print ( "1-Loc 2-Program    ");
     }
     else if (iMenu==4)
     {
      iMenu=3;
      lcd.setCursor (0, 0);
      lcd.print ( ">Program No.");
      lcd.print (iSource);
      lcd.print (iDestination);
      lcd.setCursor (0, 1);
      lcd.print ( "Press prog no. *");
      PlayAudio(cAudio_ProgramCancel) ;
     }
     break;
   }
  }
//}
```

```
void ReadRangeFinder() {
  if (IsRangeEnabled && EnableSideSensor && !IsTeachMode && key==cUp)
  {
    // Clears the trigPin
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculating the distance
    distance= duration*0.034/2;

    if (distance>200)
    {
     distance = 200;
    }

    distance5 = 200; //distance4;
    distance4 = 200; //distance3;
    distance3 = 200; // distance2;
    distance2 = distance1;
    distance1 = distance;

    if (distance1<distance2 && distance1<distance3 && distance1<distance4 && distance1<distance5)
    {
      distance = distance1;
    }
    else if (distance2<distance1 && distance2<distance3 && distance2<distance4 && distance2<distance5)
    {
     distance = distance2;
    }
    else if (distance3<distance1 && distance3<distance2 && distance3<distance4 && distance3<distance5)
    {
     distance = distance3;
    }
    else if (distance4<distance1 && distance4<distance2 && distance4<distance3 && distance4<distance5)
    {
     distance = distance4;
    }
    else if (distance5<distance1 && distance5<distance2 && distance5<distance3 && distance5<distance4)
    {
     distance = distance5;
    }

    lcd.setCursor (0, 1);
    lcd.print ( "Distance:");
    lcd.print (distance);
    lcd.print (" cm  ");

    if (distance<20 && !IsBlocked)
    {
     IsBlocked = true;
     IsLatched = false;
    }
    else if (distance>40 && IsBlocked)
    {
     IsBlocked = false;
     PlayAudio(cAudio_ObstructionCleared);
     delay(3000);
    }
  }
```

```
   else
   {
     IsBlocked = false;
   }

}
//------- Main Program -------------------------------------

void loop() {

 ReadInfraredKey();       // CALL COMMAND FOR INFRARED

 if (digitalRead(IN3) || (key==cStar))     //Acknowledge Switch or press star - start execute command
 {
  if (IsTeachMode==false)
  {
    lcd.setCursor (0, 0);
    if (iDestination==0)
    {
      lcd.print ( "Run Mode: Ctr      ");
    }
    else
    {
      lcd.print ( "Run Mode: Table");
      lcd.print (iDestination);
    }
    SetCommand(iSource,iDestination);
    lcd.setCursor (0, 1);
    lcd.print ( "Leaving now...       ");
    if (iSource==0)
    {
      PlayAudio(cAudio_LeaveNowGoodbye);
      delay(2000);
    }
    else
    {
     PlayAudio(cAudio_LeavingNow);
     delay(4000);
    }
    IsRunMode = true;
    key = cDown;
    cmdIndex = 0;
  }
 }

 switch (key) {
  case cLeft:
    if (!IsMovingLeft)
    {
      if (IsTeachMode)
      {

      }
      else
      {
        lcd.setCursor (0, 1);
        lcd.print ( "Turn Left           ");
      }
      IsMovingLeft = true;
      IsMovingRight  = false;
      delay(cDelayRun);
      startMillis = millis();
      EnableSideSensor = false;
      IsTimeout = false;
```

```
  }
  if (IsTimeout)
  {
   analogWrite(PWM1, cSpeedTurnStart );
   analogWrite(PWM2, 0);
   analogWrite(PWM3, 0);
   analogWrite(PWM4, cSpeedTurnStart );

   IsSensorTrackLeftDetectON = digitalRead(SensorTrackLeft);     //New Sensor
   IsSensorTrackRightDetectON = digitalRead(SensorTrackRight);   //New Sensor
   if (IsSensorTrackLeftDetectON || IsSensorTrackRightDetectON)   //Check line sensors
   {
       CanAddTeach = IsTeachMode;
       Stop();
       if (IsRunMode || IsTeachMode)
       {
         lineDet = true;
         IsLineDetected = true;
         key = cUp;
         delay(cDelayRun);
       }
       TiltLeft = true;
       TiltRight = false;
   }
  }
  else
  {
   IsTimeout = (millis() - startMillis) > cDelayTurn;
   analogWrite(PWM1, cSpeedTurn );
   analogWrite(PWM2, 0);
   analogWrite(PWM3, 0);
   analogWrite(PWM4, cSpeedTurn );
  }
  break;

 case cRight:
  if (!IsMovingRight)
  {
   if (IsTeachMode)
   {

   }
   else
   {
    lcd.setCursor (0, 1);
    lcd.print ( "Turn Right               ");
   }
   IsMovingLeft = false;
   IsMovingRight  = true;
   delay(cDelayRun);
   startMillis = millis();
   EnableSideSensor = false;
   IsTimeout = false;
  }

  if (IsTimeout)
  {
   analogWrite(PWM1, 0);
   analogWrite(PWM2, cSpeedTurn );
   analogWrite(PWM3, cSpeedTurn );
   analogWrite(PWM4, 0);

   IsSensorTrackLeftDetectON = digitalRead(SensorTrackLeft);     //New Sensor
   IsSensorTrackRightDetectON = digitalRead(SensorTrackRight);   //New Sensor
```

```
  if (IsSensorTrackLeftDetectON || IsSensorTrackRightDetectON)   //Check line sensors
  {
      CanAddTeach = IsTeachMode;
      Stop();
      if (IsRunMode || IsTeachMode)
      {
        lineDet = true;
        IsLineDetected = true;
        key = cUp;
        delay(cDelayRun);
      }
      TiltLeft = false;
      TiltRight = true;
  }
 }
 else
 {
  IsTimeout = (millis() - startMillis) > cDelayTurn;
  analogWrite(PWM1, 0);
  analogWrite(PWM2, cSpeedTurnStart);
  analogWrite(PWM3, cSpeedTurnStart);
  analogWrite(PWM4, 0);
 }
 break;

case cDown:
  if (!IsMovingRight)
  {
   if (IsTeachMode)
   {

   }
   else
   {
    lcd.setCursor (0, 1);
    lcd.print ( "Turn Around                ");
   }
   IsMovingLeft = false;
   IsMovingRight  = true;
   startMillis = millis();
   IsAtStation = false;
   IsTimeout = false;
  }

  if (IsTimeout)
  {
   analogWrite(PWM1, 0);
   analogWrite(PWM2, cSpeedTurn );
   analogWrite(PWM3, cSpeedTurn );
   analogWrite(PWM4, 0);

   IsSensorTrackLeftDetectON = digitalRead(SensorTrackLeft);     //New Sensor
   IsSensorTrackRightDetectON = digitalRead(SensorTrackRight);   //New Sensor
   if (IsSensorTrackLeftDetectON || IsSensorTrackRightDetectON)   //Check line sensors
   {
      Stop();
      if (IsRunMode || IsTeachMode)
      {
        lineDet = true;
        IsLineDetected = true;
        key = cUp;
        delay(cDelayRun);
      }
      TiltLeft = false;
```

```
      TiltRight = true;
    }
   }
   else
   {
    IsTimeout = (millis() - startMillis) > cDelay1;
    analogWrite(PWM1, 0);
    analogWrite(PWM2, cSpeedTurnStart );
    analogWrite(PWM3, cSpeedTurnStart );
    analogWrite(PWM4, 0);
   }
   break;

 case cOk:
   IsMovingForward = false;
   IsMovingReverse = false;
   EnableSideSensor = false;
   IsMovingLeft  = false;
   IsMovingRight  = false;
   IsRunMode = false;
   break;

 case cUp:
   if (IsAtStation)    //Robot at station
   {
    break;
   }

   if (!IsMovingForward)
   {
    if (IsTeachMode)
    {

    }
    else
    {
     lcd.setCursor (0, 1);
     lcd.print ( "Moving          ");
    }
    IsMovingForward = true;
    startMillis = millis();
    EnableSideSensor = false;
    IsBlocked = false;
   }

   LineTracing();          // CALL COMMAND FOR LINE TRACKER

   if (EnableSideSensor)
   {
    if ((IsSensorLeftLineDetectON || IsSensorRightLineDetectON))
    {
     delay(50);
     CanAddTeach = IsTeachMode;
     Stop();

     if (IsTeachMode)
     {
      IsStartTeach = true;
      delay(100);
      LineTracing();
      if (IsSensorTrackLeftDetectON && IsSensorTrackRightDetectON && IsSensorLeftLineDetectON &&IsSensorRightLineDetectON )
      {
       lcd.setCursor (0, 1);
       lcd.print ( "Saving ...");
```

```
      PlayAudio(cAudio_ProgramSave);
      for (nProgramNo=0;nProgramNo<10;nProgramNo++)
      {
        EEPROM.write(iSource*100+iDestination*10+nProgramNo,program[iSource][iDestination][nProgramNo]);
      }
      delay(1000);
      lcd.setCursor (0, 1);
      lcd.print ( "Press prog no. ");
      iMenu=3;
     }
    }
    if (IsRunMode)
    {
     delay(100);
     LineTracing();
     if (IsSensorTrackLeftDetectON && IsSensorTrackRightDetectON && IsSensorLeftLineDetectON &&IsSensorRightLineDetectON )
     {
      cmdIndex = 0;
      if (iDestination==0)
      {
       lcd.setCursor (0, 1);
       lcd.print ( "Waiting for Item  ");
       PlayAudio(cAudio_ItemsForDelivery);    //Do you have items for delivery?
      }
      else
      {
       PlayAudio(cAudio_CollectItems);    //Delibot have items for you. please collect and press RED button.
       lcd.setCursor (0, 1);
       lcd.print ( "Pls collect Items ");
      }
      SetCommand(iDestination,iSource);
     }
     else
     {
      delay(100);
      if (command[cmdIndex]>0)
      {
       key = command[cmdIndex];
       cmdIndex++;
      }
     }
    }
    break;
  }
}
else
{

  if ((IsSensorLeftLineDetectON && IsSensorRightLineDetectON) || ((millis() - startMillis) > cDelay2))
  {
    EnableSideSensor = true;
  }
  IsBlocked = false;
}
if (!IsBlocked)
{
    if (TiltLeft && !TiltRight)       //Tilt Left
    {
      if (EnableSideSensor)
      {
        analogWrite(PWM1, 0 );
        analogWrite(PWM2, cSpeed);        //cSpeed
        analogWrite(PWM3, cSpeedTilt80);   //cSpeedTilt100
        analogWrite(PWM4, 0 );
```

```
  }
  else
  {
    analogWrite(PWM1, 0 );
    analogWrite(PWM2, cSpeedTilt100);
    analogWrite(PWM3, cSpeedTilt100);
    analogWrite(PWM4, 0 );
  }

  break;
 }
 if (!TiltLeft && TiltRight )    //Tilt Right
 {
  if (EnableSideSensor)
  {
    analogWrite(PWM1, cSpeedTilt80);   //cSpeedTilt100
    analogWrite(PWM2, 0);
    analogWrite(PWM3, 0);
    analogWrite(PWM4, cSpeed);        //cSpeed
  }
  else
  {
    analogWrite(PWM1, cSpeedTilt100 );
    analogWrite(PWM2, 0);
    analogWrite(PWM3, 0);
    analogWrite(PWM4, cSpeedTilt100 );
  }

  break;
 }
 if (!TiltLeft && !TiltRight)     //Straight line detected - move straight
 {
  if (EnableSideSensor)
  {
    analogWrite(PWM1, 0);
    analogWrite(PWM2, cSpeed);  //fwd L
    analogWrite(PWM3, 0);
    analogWrite(PWM4, cSpeed);  //fwd R
  }
  else
  {
    analogWrite(PWM1, 0);
    analogWrite(PWM2, cSpeedStart);  //fwd L
    analogWrite(PWM3, 0);
```