ASHESI UNIVERSITY

THE DESIGN OF A LOWCOST AUTOMATIC GROUND VEHICLE FOR WAREHOUSE STOCK MANAGEMENT

CAPSTONE

B.Sc. Mechanical Engineering

Eugene Kudzai Jamu

2019

ASHESI UNIVERSITY

THE DESIGN OF A LOWCOST AUTOMATIC GROUND VEHICLE FOR WAREHOUSE STOCK MANAGEMENT

APPLIED

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University

in partial fulfillment of the requirements for the award of Bachelor of Science degree in

Mechanical Engineering

Eugene Kudzai Jamu

2019

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

……

Date:

……

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

……

Supervisor's Name:

Acknowledgements

My profound gratitude to Dr. Heather Beem, Dr. Elena Rosca, Dr. Ayorkor Korsah and Dr.Stephen K. Armah for their unwavering technical support and advice on this project.

Abstract

In this project the design of an automatic ground vehicle for use in warehouses is explained. Ecommerce has been growing rapidly across the globe because of the spread of the internet and due to the introduction and acceptance of secure digital payment systems. Volumes of products handled by online stores has been increasing exponentially because of increased demand. This has resulted in a need for online shops to automate some of their processes in order to make logistics more efficient. Automatic ground vehicles have been developed however they remain very expensive for small and medium businesses SMEs. In this project a low-cost automatic ground vehicle is developed to meet the needs of SMEs.

Table of Contents

Chapter 1: Introduction

1.1 Background and Introduction

With the exponential growth in ecommerce transactions and market penetration, large retail businesses are incurring huge revenue losses due to the time wasted by employees moving around the warehouse to pick required items [1]. The world's largest ecommerce businesses such as Amazon and Alibaba have started using thousands [1] of stock management robots and this has increased delivery efficiency considerably.

According to Business Insider, Kiva robots used at 13 of Amazon fulfilment centres have reduced expenses by about 20% which amounts to about US$ 22million per centre [2]. The robots have also reduced order delivery cycle time from an average of 70 minutes to only 15 minutes [2]. Kiva robots have also allowed Amazon to build narrower isles and get rid of some handling systems. The smarter use of space has increased their inventory space by over 50%.

For a 7.5 hours work shift at the Alibaba warehouse, workers would traditionally sort approximately 1,500 products after taking 27,924 steps. However, with the help of Quicktron robots the same workers now sort about 3,000 products having taken only an average of 2,563 steps during the same period [3]. Instead of workers walking around the warehouse to find products, they stay in their assigned stations and send commands to the autonomous robots to bring them specific shelves. The workers then pick the required items from the shelf. Thus, instead of the worker going to the shelves to pick an item, the shelves are brought to them.

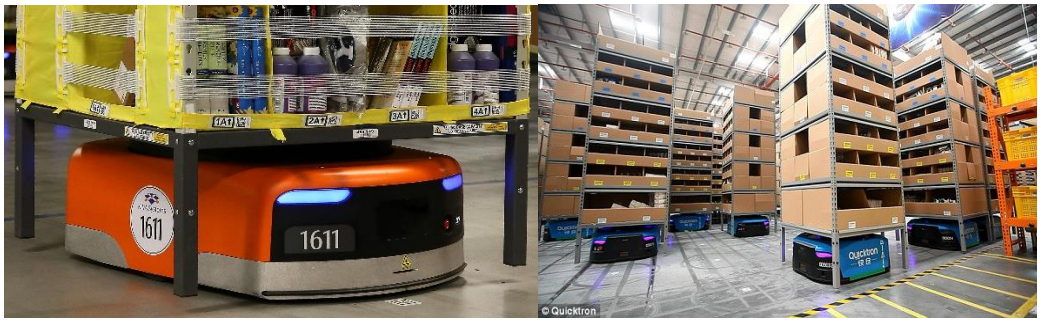

Figure 1. 1: KIVA (left) and QUICKTRON (right) warehouse management robots carrying shelves

The automated guided vehicles (AGVs) are connected to a wireless network and use barcode scanners on the floor to navigate the warehouse. The AGVs are also equipped with obstacle avoidance sensors and they use the wireless network system to plan paths and avoid collisions. Kiva and Quicktron robots also lift and rotate shelves, allowing workers to easily pick products with minimum movement. The AGVs can carry up to 500 kilograms above them. When the battery is low they take themselves to charging stations. A 5-minute charge can give the AGV enough power to operate for 4-5 hours [4]. At Alibaba, the autonomous robots are reported to do 70% of the work.

This new technology is not yet adopted by African countries because the robots are very expensive reaching as high as USD 50, 0000 per unit [4]. Ecommerce is also rapidly growing in Africa with almost double the global growth rate because of the globalization, the rise of startups and the spread use of the internet [5]. Large international online shops like Jumia (Nigeria), Takealot (South Africa) and Kilimall (Kenya) have extensively expanded their warehouses and may need to automate them.

1.2 Problem Definition

Given the rapid growth rate of African ecommerce businesses, there is a need to automate warehouse logistics in order to increase efficiency and cut costs. There is therefore a need to design and develop efficient and low-cost robots for ecommerce businesses.

1.3 The Objectives of the Project work

This project is focused on the design and fabrication of an automated ground vehicle for stock management in ecommerce warehouses. The robotic vehicle should increase the

efficiency of ecommerce warehouse logistics, thereby cutting down on costs. The AGV should be designed with minimised cost of production for increased affordability. The robot should quickly, accurately, and safely move stock to where it is needed. It should receive commands wirelessly on an internet-based system then plan and navigates its path to pick the required shelf and deliver it to a working station where a worker will pick the required items. This should be achieved with minimum changes to the warehouse environment and at low production costs. The project will also mainly focus on solving the problem of accurate localization.

1.4 Expected Outcomes of the Project Work

At the end of the project, I should deliver a working prototype of an automated ground robot for stock management. The low-cost robot is expected to safely, accurately and quickly pick a shelf and place it where it is needed. The robot will receive commands wirelessly.

1.5 Motivation and Justification for the Project Topic

Technology is advancing tremendously, however, the adoption of trending technology in Africa remains relatively low because of the high costs of operation. Automated stock management is one of the technologies which can be very profitable for African communities, particularly ecommerce businesses where most systems are manual and slow. I have had the opportunity to meet and discuss with the CEO of one of Africa's largest ecommerce businesses, Jumia. The business has been growing remarkably since its establishment in 2012. Started in Nigeria, Jumia now operates in 14 African countries including Ghana, Kenya and South Africa. The business now has about half a million customers [6]. In my discussion with the CEO for Ghana Jumia, he explained to me that they need a robot that can pick and place a shelf. He also explained that at the moment they cannot give a robot to hand pick products because they can easily break. They have asked for a proof of concept for the robot that can pick shelves. They promised to take on the

project and fund it if the proof of concept is satisfactory to the Jumia executive. This has given me the energy to invest time and resources on this project. It is a great opportunity to make a difference and apply my engineering skills.

Chapter 2: Literature Review

2.1 Automatic Guided Vehicles Overview

Automated or automatic guided vehicles (AGVs) are mobile robots that follow markers or other physical guides, vision or lasers and are often used in material handling, assembly lines, plants or manufacturing facilities [15]. A number AGVs such as Amazon's Kiva and Alibaba's Quicktron for warehouse management and related applications have been designed, fabricated, and implemented. The research is still in its premature stage and is ongoing both in academia and the industry. Most AGVs in development are however, mainly used for surveillance and inventory management.

AGVs are often battery powered [7] and are controlled from a central database. They can operate both indoors and outdoors. There is often a user interface where the operator can issue commands through wireless communication [8]. Host software in internal centralised controllers are used by the AGV to plan and navigate a path from the start position to the goal point. In other more complex systems, AGVs are connected to a supervisory software which facilitates effective communication between robots, control traffic routing for maximized efficiency and assign missions to the fleet. The supervisory software is often referred to as Warehouse Management System (WMS). Some AGVs periodically report to charging stations when the battery is drained.

2.2 Installation of the System

AGVs can be installed in both new and existing facilities. However, it is easier to incorporate and integrate designs before the facility is constructed. When considering installing an AGV system in a facility that was already established it is important that the changes that are required for it to be functional be kept at minimum. Every AGV is customised for a unique task and specific environment set up. There have however been

concerns of standardising and regulating AGVs so that compatibility becomes easier even with robots from different manufacturers.

2.3 Local Path Planning and Navigation

The vehicles need a way of knowing and navigating their environment. They achieve this by using precise maps which are usually in the form of occupational grids. An occupational grid contains vital information of an environment indicating where there are pathways, obstacles and other points of interest [9, i-Fork]. The occupation grid is often a nested array of binary numbers, where a '1' often represents an occupied space and '0' represents free coordinate. A pathway would therefore be a series of connected free spaces that link the robots, initial location and its planned destination. There can be many possible pathways, the robot needs to route the most efficient and cost-effective path. The AGVs are equipped with sensors such as LASER and LIDAR which help them sense unexpected objects and avoid crushing when there are sudden changes in the environment for example a pedestrian close to the robot's path.

2.4 Safety and Efficiency

It is important to consider safety implications when using robots especially where they share the environment with humans like in a warehouse [8]. Sensors installed on AGVs cannot detect and differentiate between objects. Therefore, AGVs are required to operate at low speeds in critical zones such that they can behave safely when responding to unpredictable situations [10]. It is also important for the vehicles to have the capability to replan and reroute in situations where an alternative path becomes necessary due to unexpected obstructions. Where this is not possible, the AGV would get stuck and would require human assistance [8].

2.5 Coordination and Fleet Management

In a system where many AGVs are simultaneously working, a central system would be in charge of coordinating routing for each vehicle thereby avoiding collisions. Path planning by one AGV can be very difficult [11] because of limited knowledge about the environment especially about non-stationary objects, it gets even more complex when using a centralised system and the challenge extends with increasing number of AGVs. A partially decentralised [13] coordination strategy was introduced. The approach consists of hierarchical architecture that implements path planning in two layers: topological layer and route map layer. Local negotiation will then take place between AGVs based on shared information about the state of the fleet.

2.5 Market Opportunities

AGVs are expected to diffuse in warehouse management and the current major competitors are manual forklifts. However, human driven forklifts account for most injuries and fatalities in the shop floors, due to human error. A survey conducted by Aberdeen Group revealed that 90% of 150 surveyed managers showed commitment to invest in automating their warehouses to increase efficiency, monitoring and performance [10]. Even though the market for AGVs is gradually growing, manufacturers face hurdles because of high initial installation costs and the changes required to make the system fit. On average installation costs for an individual vehicle average around US$ 125, 000 per vehicle [10]. Currently only big businesses can afford AGV technology installation and operation costs.

2.6 Major Challenges

Currently highly skilled and trained personnel are required for the operation and maintenance of AGVs. User interfaces are also not as user friendly. Furthermore, AGV systems on the market require extensive environment changes and their installation fees are very high. The simplification of the user interface, operation and maintenance will

undoubtably increase the adoption of AGVs reducing initial and operation costs will diffuse AGVs into small to medium enterprises (SMEs).

One other major challenge with AGVs is accurate localization. Sensors such as LIDAR and cameras that are required for high accuracy positioning are very expensive and require high speed processors. Using odometry sensors would increase the vehicle's accuracy, however, this method is expensive both in terms of the required computational time which makes processing slow and cost of required sensors which is high.

GPS is not being used on warehouse management robots because it does not work properly indoors. RFID technology has been one of the mainly employed technologies in addressing the localization problem. This method was successfully implemented in a design by Laiwatimena et al in their design for a mini fork lift based on an AtMega 8535 microcontroller [14]. Another advantage is there is minimal changes to the environment.

This project therefore will be focused on minimizing changes to the warehouse environment when installing AGVs. Accurate localization and easy operation of AGVs will also be addressed in this project. Most importantly this project seeks to reduce the cost of buying and installing AGVs.

Chapter 3: Design

3.1 Design Specifications

The design specifications of the automatic ground vehicle are as below:

- The AGV should have low production, installation and operational costs, each vehicle should cost less than US$3,500-5000 (based on the budget provided by the Jumia Ghana CEO) for purchasing and installation price
- There should be minimum changes to the environment within which the robot will operate
- The AGV should be easy and safe to operate without much training for personnel
- The AGV should receive wireless commands, plan its path and navigate safely and quickly
- The AGV should pick the accurate shelf and place it where it is needed
- The AGV should be able to avoid obstacles and plan an alternative path if necessary
- The AGV should be able to precisely follow a line and localize itself

3.2 The System Design

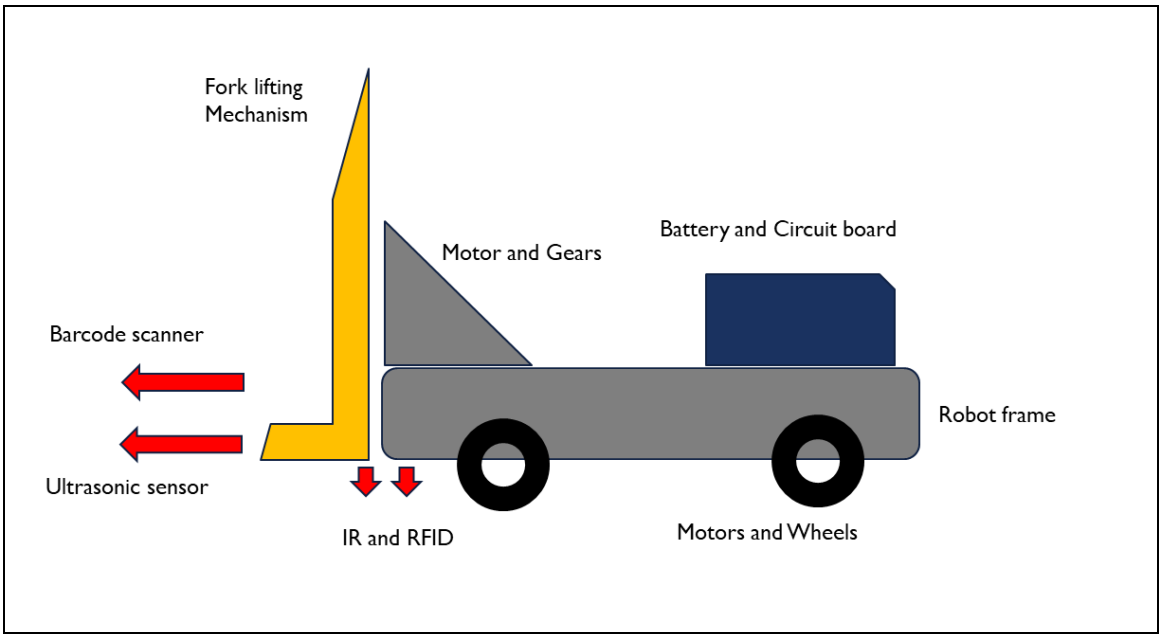

Figure 3.1: Diagram showing a simplified 2D design of the automated vehicle

The design shown in Fig 3.1 is the proposed solution to meet the design objectives in section 3.1. For optimised and simple localization, RFID floor mapping and line

following technics have been chosen since they are affordable and accurate. The methods also require minimum changes to the environment. The robot system is made up of a software system for navigation, frame for supporting all parts, mechanical lifting mechanism and electrical system for control. The robot would be controlled from a webserver; thus it would require a wireless communication system. As it navigates the environment it will avoid collision using an ultrasonic sensor. In the following section, the Pugh matrix is used to explain how components and materials have been selected for the design.

3.3 Components and Materials Selection

In an effort to achieve the design objectives stated in section 3.1 we have to carefully select the best materials and components for the project such that we can optimise resources and reduce cost. The Pugh Matrix has been used for the selection of components and materials. A scale of 1-5 has been used to rank the components. The values 1 and 5 represent poor and excellent respectively and every value will range between these two extremes.

3.31 Pugh Matrix for Microcontroller Boards

A microcontroller is needed to read values from sensors and control the motors. Most importantly the controller will be used for receiving commands, planning and navigating the path from start to goal position. There are four possible components for this task as listed on the matrix below. The Raspberry Pi is robust and very reliable; however, it is quite expensive and difficult to configure. The Arduino board is very affordable and easy to use but it does not have wireless connectivity unless an additional module is used. PIC microcontrollers are very difficult to configure and program, but they are very affordable. For this project the ESP32 development board was chosen because it is cheap, easy to program and it is itself a wireless board. Thus, it would be very easy and quick to send, receive and implement commands.

Material	Cost	Wireless Connectivity	Configuration And Programming	Processing Speed and Memory	Total
Scales	6	4	3	2	15
Arduino	1	-1	1	1	10
ESP32	1	1	0	1	12
Raspberry Pi	0	0	1	1	5
PIC	1	-1	-1	1	1

Table 3.1: Pugh matrix for microcontroller boards

3.3.2 Pugh Matrix for Frame Material

For the prototype design it is desirable to for its weight to be relatively small. Furthermore, it should be easy to manufacture. Acrylic and ABS plastic materials were chosen because of their low weight, easy manufacturability and availability. Parts for the frame and cases for sensors were 3D printed.

Frame Material	Cost	Manufacturability	Weight	Durability	Total
Scales	3	5	5	2	15
ABS (3D Printed)	1	1	1	0	13`
Acrylic	1	0	1	0	13
Steel	0	0	-1	1	-3
Aluminium	0	0	1	1	7.

Table 3.2: Pugh matrix for robot frame material

3.3.3 Pugh Matrix for Localization Sensors

Possible sensors for localization included cameras, RFID, IR, QR and Barcodes. For the purposes of floor mapping, the RFID was chosen because it does not require much changes to the environment for its installation. Similarly, the IR sensors were chosen for line following because this would not require major changes to the warehouse. Cameras would give very accurate localization however they are very expensive to purchase and use. Computer vision requires high processing speeds and thus a more robust microcontroller

and software program. This would be very difficult with the ESP32. QR and Barcodes are hard to configure with the ESP32 development board.

Localization	Cost	Accuracy	Configuration And Programming	Environment Changes	Total
Scales	5	3	4	3	15
Cameras	-1	1	-1	0	-9
RFID	1	1	1	1	15
QR	0	1	0	1	6
Bar Code	1	1	-1	1	6
IR	1	0	1	1	12

Table 3.3: Table 3.1: Pugh matrix for localization sensors

3.3.4 Pugh Matrix for Lifting Mechanism

A lifting mechanism will be required for lifting and placing shelves. The belts and pulley system is most suitable because the prototype motors are low power and weights to be lifted are relatively small. Because of the small weights a very strong locking mechanism is not as necessary. Most importantly this system is readily available and cheap to install.

Fork Lifting Mechanism	Cost	Locking Mechanism/ Efficiency	Weight	Durability	Total
Scales	4	4	4	3	15
Chains & Pulley	1	0	1	0	8
Belts & Pulley	1	0	1	0	8
Gears & Rack	0	1	1	0	8
Hydraulic	-1	1	0	1	3

Table 3.4: Pugh matrix for lifting mechanism

3.2 The Software Program

The robot will be controlled using the flow chart below. It will wait to receive a command via wireless connection (WiFi). The command can either be for the AGV to pick a shelf or for it to go the charging station. Once it has this information it will plan its path and begin navigation. The robot will keep relocalizing when it comes across an RFID tag at a corner. This improves its accuracy as it moves towards the goal. Upon reaching the goal, the robot will go on standby waiting for a command and the cycle repeats.

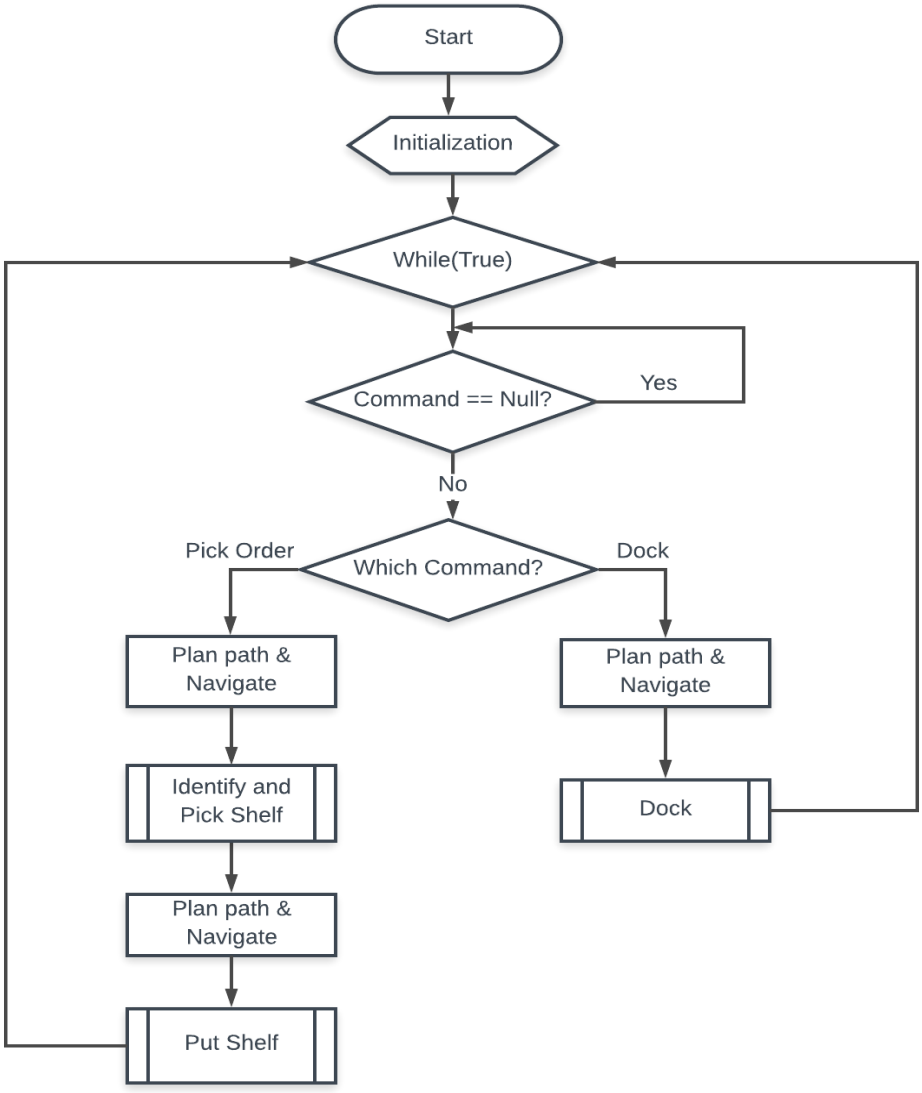

Figure 3.2: Flowchart showing how the software on the automatic ground vehicle works

3.4 The Automatic Ground Vehicle Frame Design

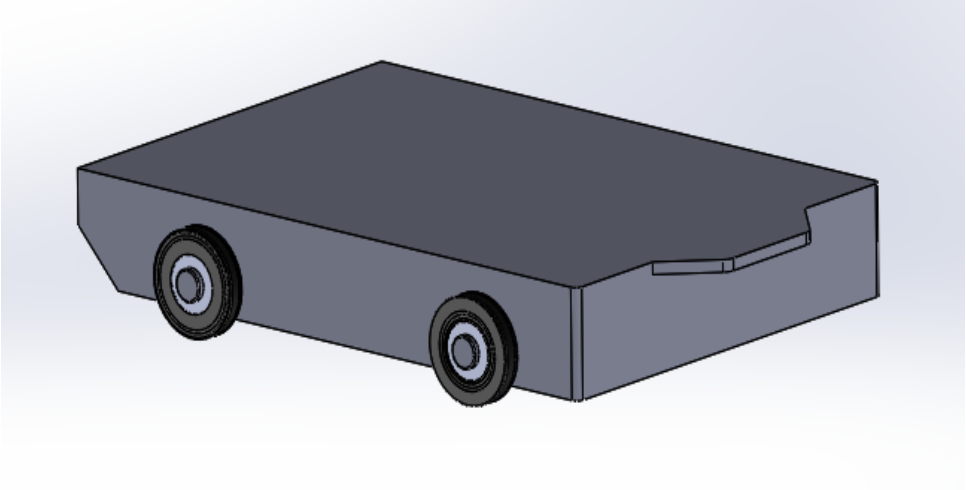

Figure 3.4: The AGV robot frame SolidWorks 3D design

The AGV would have a platform for mounting sensors in front and a lifting mechanism on top as shown in the figure above. The robot will be very close to the ground for stability and for increased sensor accuracy.

3.4.1 The First AGV Prototype

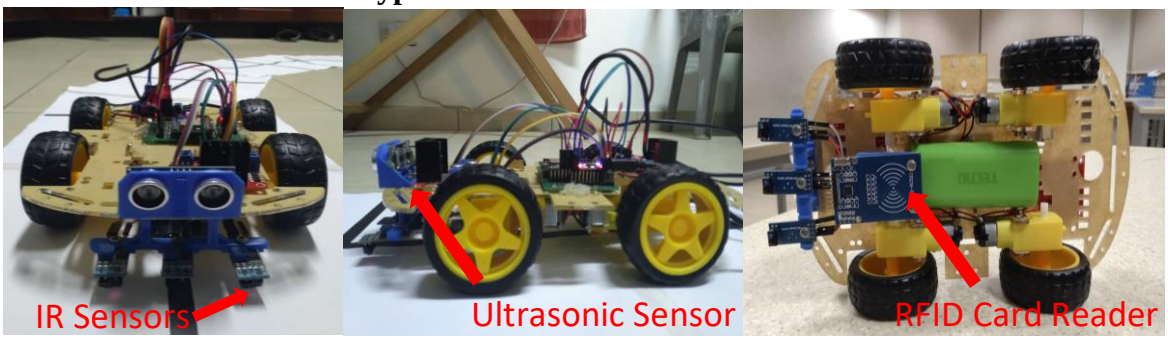

Figure 3.5: Images showing the arrangement of sensors on the first prototype AGV

Shown in the images above is the automatic ground vehicle prototype made with acrylic and 3D printed ABS material. The AGV is controlled using the ESP32 microcontroller. The ultrasonic sensor is installed in front together with infrared (IR) sensors below. The RFID reader is also very close to the front but below the robot. Three infrared sensors were used. The centre IR sensor should stay on the black line while the left and right IR sensors should always be on white, this way the robot will always follow the black line.

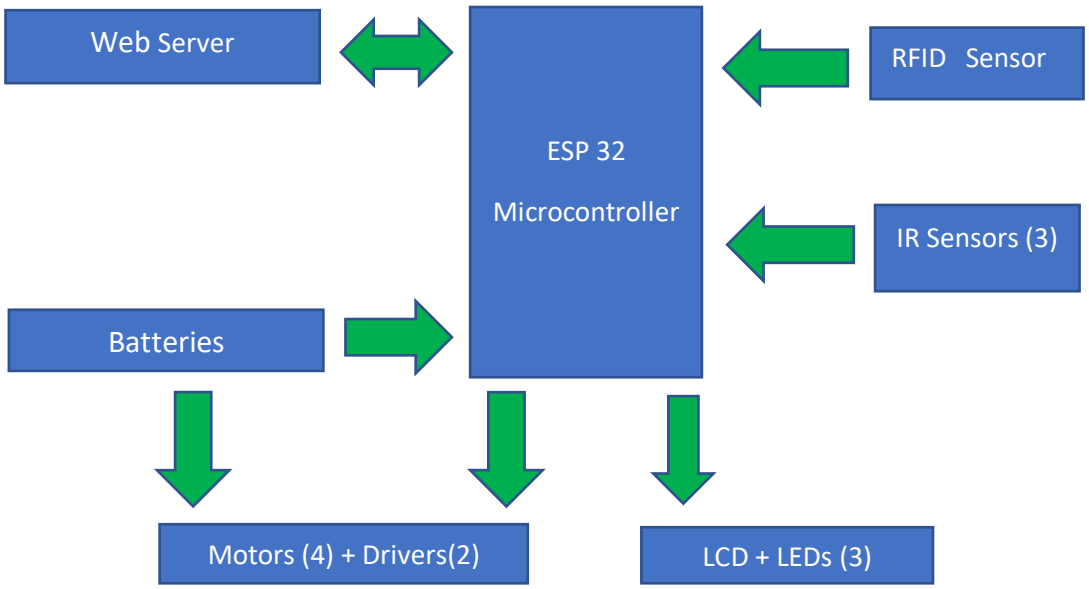

Figure 3.5: Block diagram showing how components relate

The block diagram below shows the connection of the AGV hardware. The major components are the ESP 32 microcontroller board, RFID sensors, and the IR sensors. The ESP32 controller will receive a command from a webserver, plan the path and use the IR sensors, RFID and motor drivers for navigation. The battery level sensor will always be checking battery capacity when it is below threshold the robot will stop receiving commands and drive to the nearest charging spot and wait to be recharged. The AVG will notify the database of its status – low battery. Shown below is the electrical circuit of the system.

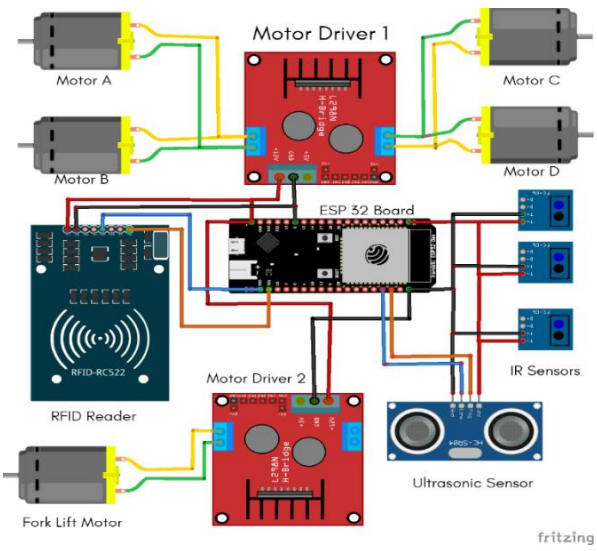

Figure 3.8: Circuit diagram showing all electrical connections between components

3.4.2. The Second AGV Prototype

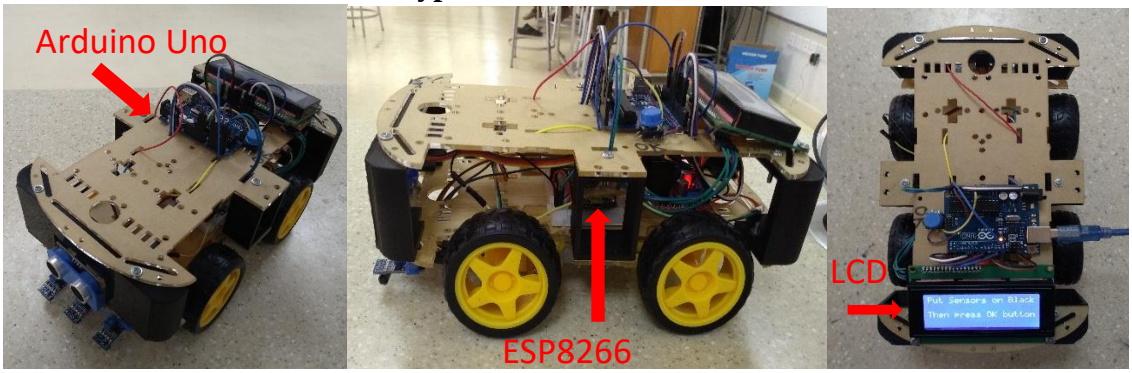

Figure 3.7: Images showing the microcontrollers and LCD on the second AGV prototype

The second AGV prototype is controlled using an ESP8266 and the Arduino Uno. This development was necessary because the RFID reader was not working with the ESP32 microntroller. In the second prototype the ESP8266 which has fewer input/output pins was used for wireless communication, localization and obstacle avoidance. The Arduino Uno was used for navigation; thus, it was used to control the motors. The ESP8266 and the Arduino Uno communicate using serial communication. The prototype 2 AGV also has a button which allows the user to calibrate the sensors using a white and black surface when the system boots. An LCD has also been added for the user or observer to understand how the robot is acting. The block diagram below shows the configuration of the system.

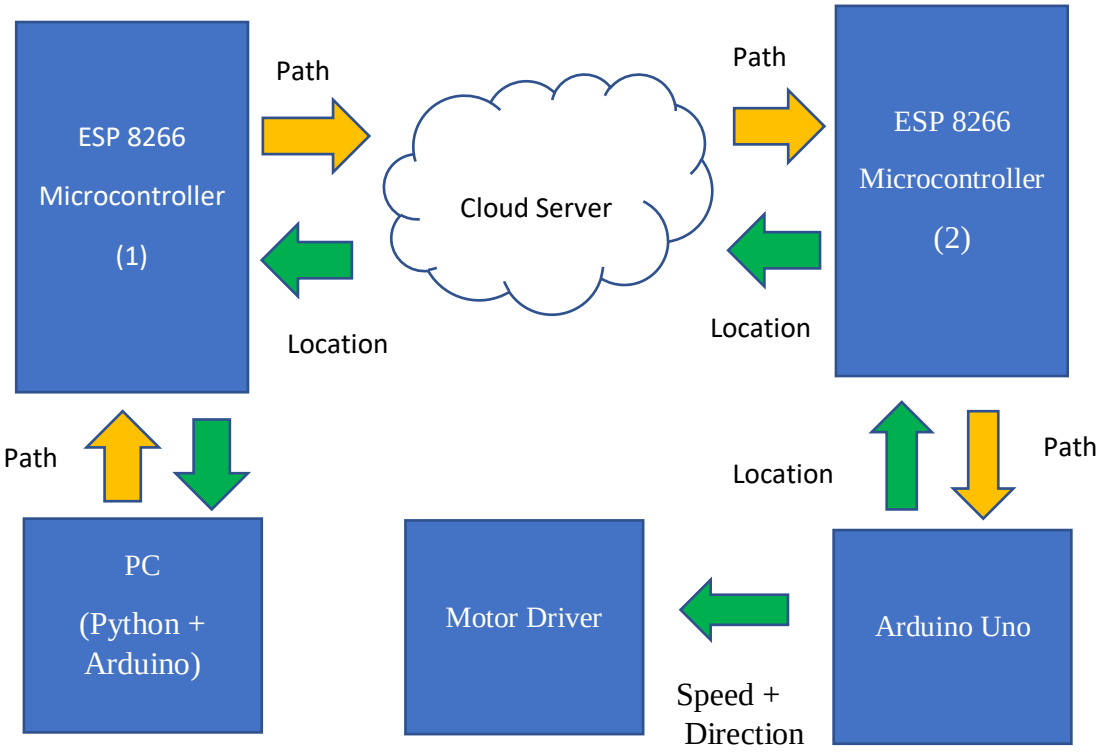

Figure 3.9: Block diagram showing the configuration of components on the second prototype

3.5 The Robot Working Environment

The working environment of the robot should be clearly defined with a model for simplification. An occupational grid will be used to make a map for the warehouse which the robot can easily interpret. The AGV mainly needs to know its position and areas where there are obstacles. Given this information the robot can know which areas it can safely move and which ones to avoid. Each coordinate on the grid represents nodes which can either be free or occupied. The software program which runs of the robot represents occupied nodes with 1s and free nodes with 0s.

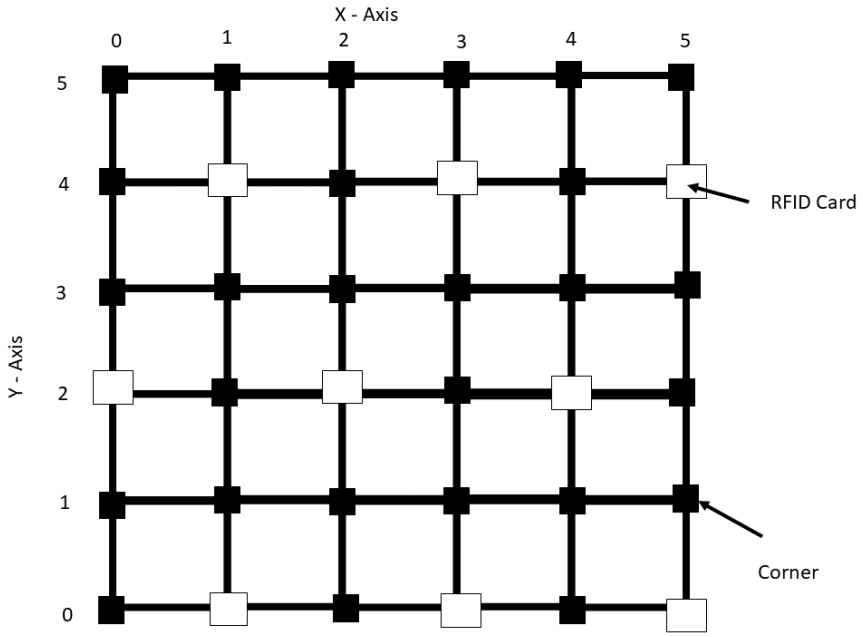

Figure 3.10: A typical grid which will be used for occupancy

3.6 The Path Planning and Navigation Algorithm

The AGV will implement the wavefront algorithm to navigate the warehouse work space. According to Ashesi Robotics the wavefront algorithm is a cell-decomposition path planning method in which the workspace is divided into equal polygons [16]. Each polygon is assigned (x, y) coordinates and the working space would be labelled with 0's and 1's where they represent free and occupied spaces respectively. A typical wavefront matrix is shown in figure [Figure XYZ] below:

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	1	1	0
2	0	0	1	0	0
3	start	1	0	0	goal

Figure 3.11: An example of a wavefront grid map

Given a workspace like the one in the diagram above, each square would be allocated a number. The start coordinate is the goal coordinate is allocated the value 2 and neighbouring cells will be numbered incrementally beginning at 3 until the start position has been labelled. These numbers represent the number of steps required to move from the start to goal position. Therefore, the shortest possible path will be the one with decreasing values from start to goal as in the figure below.

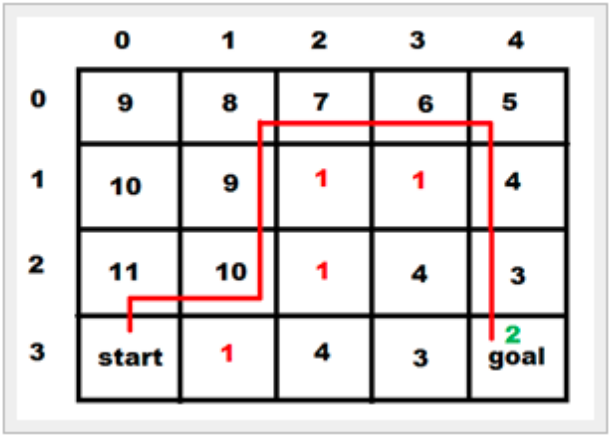

Figure 3.12: Wavefront grid showing the shortest path from start to goal

In this project the wavefront path planning algorithm will be implemented using Python. The path will be sent to the ESP8266 via serial communication and finally to the robot through the webserver.

Chapter 4: Methodology

4.1 Sensors and Motor Control

The robot will navigate the warehouse working space based on sensor values. Tests will be performed for line following, obstacle avoidance and localization. These tests are explained in the following subsections.

4.1.1 Line Following Test

The IR sensors output high voltage values when they reflect light on dark (black) surfaces, they also return low values for light (white) surfaces. These values will be mapped to either 1 or 0 respectively. Motors will be controlled based on these IR values in order for the robot to follow the black line. The table below summarizes how the robot would behave based on the IR sensor values.

Left IR Sensor	Centre IR Sensor	Right IR Sensor	Interpretation
1	1	1	Black Surface (Coordinate)
1	1	0	Turn Left to Align
1	0	1	Not Possible (Stop robot)
1	0	0	Turn Left to Align
0	1	1	Turn Right to Align
0	1	0	Forward (Robot on black line)
0	0	1	Turn Right to Align
0	0	0	White Surface (Coordinate + RFID)

Table 4.1: Table: Table showing expected robot behaviour based on IR sensor values

4.1.2 Obstacle Avoidance Test

The robot should be able to maintain a minimum distance of 0.2m from obstacles. Readings will be taken using the ultrasonic sensor and obstacles will be placed in front of the robot to observe how it reacts.

4.1.3 Localization Test

During navigation the robot should also be able to detect RFID tags placed at specific coordinates of the working space.

4.2 Communication Test

Paths for the robot will be computed on the PC and sent to the robot using a webserver. The information will be sent in the form of a string of commands. Communication between the ESP8266

(connected to the PC) and the webserver will be tested. Another test will be conducted for retrieving the path from the webserver by the ESP8266 on the robot.

4.3 Navigation Test

Once the ESP8266 on the robot receives the commands from the webserver it should begin navigation. The AGV will be tested in an environment set up as shown in Fig. 8 below. The commands from the webserver will instruct the actions of the robot. The commands will instruct the robot to navigate a path. Tests will be done for paths A (at the origin) to point B (shelf 2 at shelf row 2) and from B to point C (the Counter). The oberver will observe the robot navigate each od these two paths 10 times and record whether it reached the destination or not. A one value statistics T-Test will be conducted to anaylyse the results statistically.

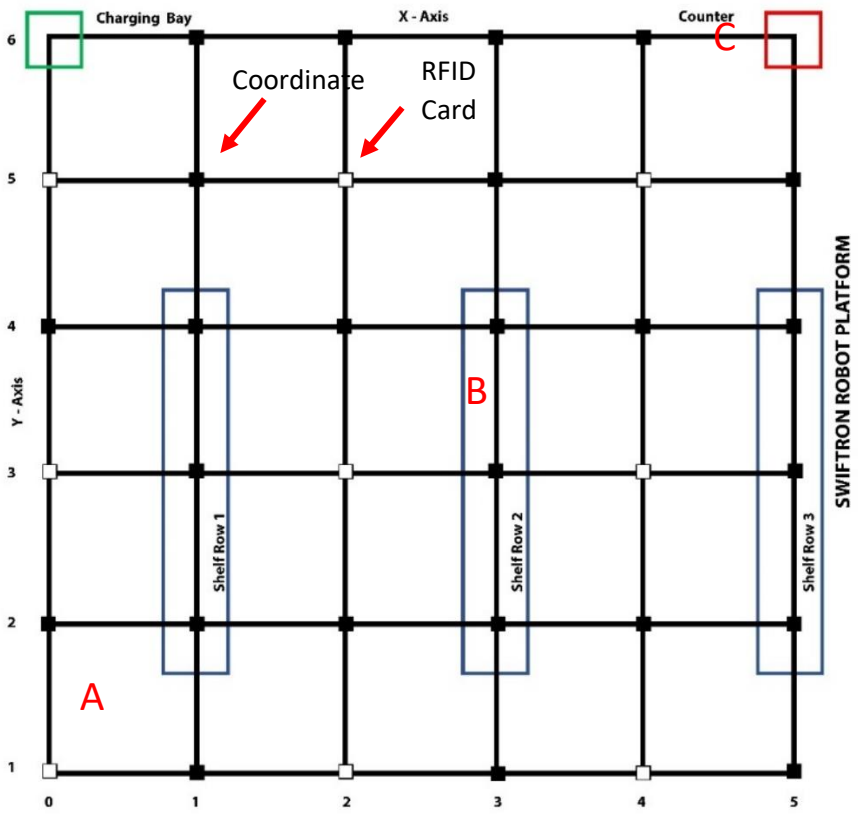

Figure 4.1: Occupancy grid with key points used to test the robot

The robot will have to navigate from its initial location to the chosen goal location. The observer will record whether the robot succeeded or failed to reach a specified goal. Several tests will be carried out for different start point and shelves coordinates.

Chapter 5: Results and Analysis

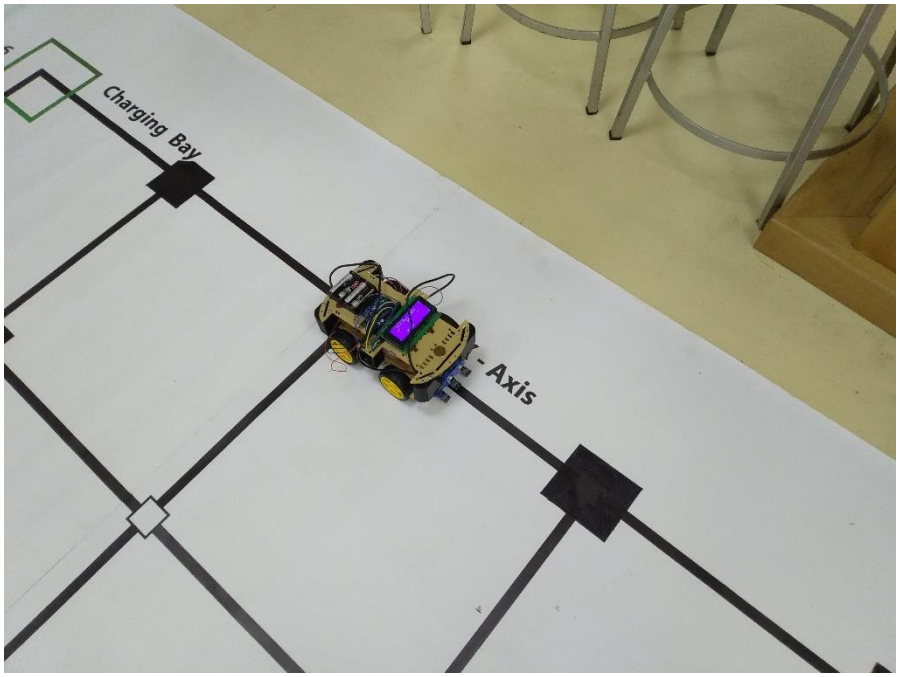

Table 4.1: Images showing the robot during line following tests

The robot was tested using the methodology explained in chapter 4. The results will be analysed on the followiing sections.

5.1 Line Following Results

The robot successfully followed the black lines. The AGV behaviour ws as expected, it would turn into the right direction when sensor values chnaged during navigation. Shown below are images from the test.

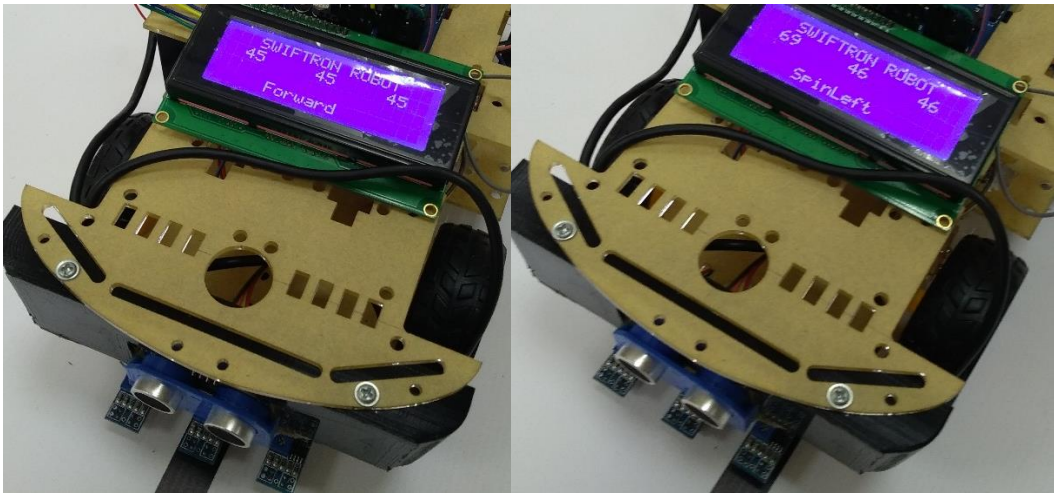

Figure 5.2: Images showing the robot taking decisions based on IR sensor values

5.2 Obstacle Avoidance Results

The AGV is able to avoid obstacles with a stopping distance. The table below summarises the values on the AGV's stopping distance when approaching obstacles.

Test	Desired Stopping Distance	Observed Stopping Distance
1	0.2m	0.23
2	0.2m	0.19
3	0.2m	0.18
4	0.2m	0.20
5	0.2m	0.22

Table 5.1: Results from the obstacle avoidance test

5.3 Localization Results

The robot ESP8266 is able to read the RFID information at a maximum range of 0.02m.

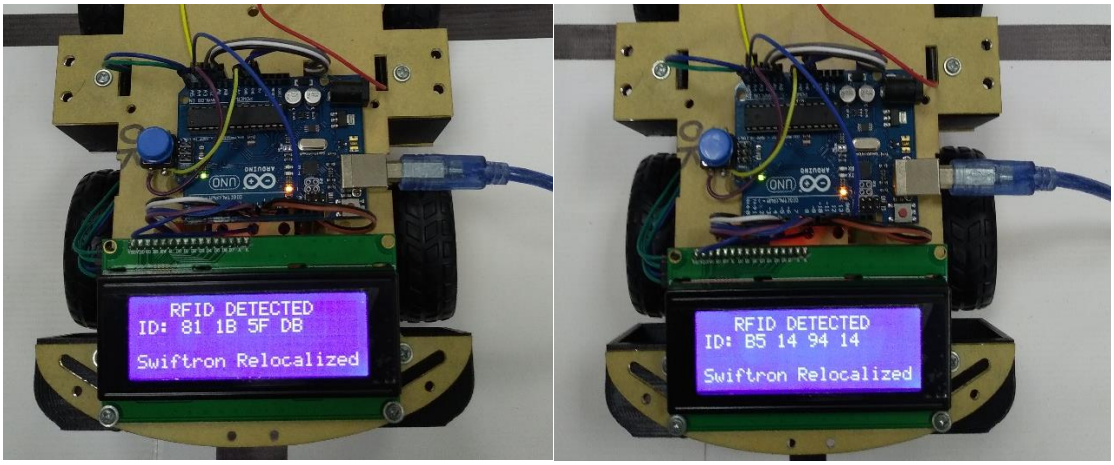

Figure 5.3: Example results from the RFID test

5.4 Communication Results

Communication between the ESP8266 on the PC and the webserver as well as between the ESP8266 on the robot and the server has been implememnted successfully. The path was sent and received as was expected. Ultrasonic sensor and RFID information has also be successfully sent between the the robot ESP8266 and the Arduino Uno.

5.5 Navigation Results

The navigation test could not be performed because it has been observed that the wheels of the robot are not turning 90 and 360 degrees because of insufficient friction between the

wheels and the robot. Initially it was assumed it was a power problem, it has taken long to realise this challenge. There is need for a solution to the friction problem.

Chapter 6: Conclusion

6.1 Discussion

The communication, sensing and path planning have been successfully implemented. However, the project could not yield expected results because of unforeseen challenges. There is need for a rougher surface or rough wheels to increase the friction between the two surfaces. Given the tests conducted with the communication, sensing and path planning the robot is working theoretically or in the sense of a simulation. There is need however to make it work in practice.

6.2 Limitations

There are a number of challenges involved with this project. It is very easy to design a system theoretically, but its implementation is usually limited with physical elements. Varying levels of intensity significantly affect how the robot differentiates between black and white using IR sensors because the threshold would have shifted. Another challenge has been with the barcode scanner which was available was not compatible with the microcontroller. Using RFID would not be as efficient because they have a very short and limited detection range. Furthermore, the ESP32 microcontroller which would be needed for connecting the barcode scanner or RFID reader had all its input/output ports exhausted. Friction has been the major challenge which has hindered progress of this project.

6.3 Future Work

There is need to seek a solution to the friction problem with this the robot can be tested conveniently. In order to minimise the number of microcontrollers used sufficient knowledge in required in C data structures with this path planning can be done locally on the robot which would receive commands directly from the webserver. A barcode scanner compatible with the ESP32, ESP8266 or Arduino would make it easier for the robot to identify shelves. Furthermore, this technology is common already in shops and would be

cheap for them to implement. A Raspberry-Pi could also be used for more reliability and compatibility with the number of sensors and processing required.

References

[1] C. Guillot, “4 types of autonomous mobile robots, and their warehouse use cases”, Supply Chain Drive, 2018, [Online], Available: https://www.supplychaindive.com/news/4-types-of-autonomous-mobile-robots-and-their-warehouse-use-cases/529548/

[2] Business Insider, “Kiva Robots Save Money for Amazon, 2016, [Online], Available: https://www.businessinsider.com/kiva-robots-save-money-for-amazon-2016-6?IR=T

[3] Daily Mail UK, “Wifi-equipped robots triple work efficiency at the warehouse of the world's largest online retailer”, 2017, [Online], Available:

https://www.dailymail.co.uk/news/article-4754078/China-s-largest-smart-warehouse-manned-60-robots.html

[4] Business Insider, “Inside Alibaba Smart Warehouse Robots Do 70 Percent of the Work”, Logistics Technology, 2017, [Online], Available:

https://www.businessinsider.com/inside-alibaba-smart-warehouse-robots-70-per-cent-work-technology-logistics-2017-9?IR=T

[5] World Economic Forum, “How Ecommerce is Booming in Africa”, 2015, [Online], Available:

https://www.weforum.org/agenda/2015/08/how-e-commerce-is-booming-in-africa/

[6] Small Starter, “The biggest players in Africa’s Fast Growing Ecommerce Market”, n.d, [Online], Available:

http://www.smallstarter.com/know-the-basics/the-biggest-players-in-africas-fast-growing-e-commerce-market/

[7] M. M. Oliveira, J. P. M. Galdames, K. T. Vivaldini, D. V. Magalhães and M. Becker, "Battery state estimation for applications in intelligent warehouses," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 5511-5516. doi: 10.1109/ICRA.2011.5980548

[8] S. Teller *et al.*, "A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments," *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010, pp. 526-533. doi: 10.1109/ROBOT.2010.5509238

[9] H. M. Barbera, J. P. C. Quinonero, M. A. Z. Izquierdo and A. G. Skarmeta, "i-Fork: a flexible AGV system using topological and grid maps," 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 2003, pp. 2147-2152 vol.2.

[10] F. Oleari, M. Magnani, D. Ronzoni and L. Sabattini, "Industrial AGVs: Toward a pervasive diffusion in modern factory warehouses," 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj Napoca, 2014, pp. 233-238.doi: 10.1109/ICCP.2014.6937002

[11] Naiqi Wu and MengChu Zhou, "AGV routing for conflict resolution in AGV systems," 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 2003, pp. 1428-1433 vol.1. doi: 10.1109/ROBOT.2003.1241792

[12] S. Prombanpong, W. Kiattiphatthananukul, A. Songsanan and A. Sukin, "NThe design of an AGV in the manufacturing cell," *2012 IEEE International Conference on Industrial Engineering and Engineering Management*, Hong Kong, 2012, pp. 1006-1009. doi: 10.1109/IEEM.2012.6837892

[13] A. Krnjak, I. Draganjac, S. Bogdan, T. Petrović, D. Miklić and Z. Kovačić, "Decentralized control of free ranging AGVs in warehouse environments," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 2034-2041.doi: 10.1109/ICRA.2015.7139465

[14] S. Liawatimena, B. T. Felix, A. Nugraha and R. Evans, "A mini forklift robot," The 2nd International Conference on Next Generation Information Technology, Gyeongju, 2011, pp. 127-131. doi: 10.1109/ROBOT.2003.1241911

[15] B. Y. Qi, Q. L. Yang and Y. Y. Zhou, "Application of AGV in intelligent logistics system," *Fifth Asia International Symposium on Mechatronics (AISM 2015)*, Guilin, 2015, pp. 1-5.
doi: 10.1049/cp.2015.1527

[16] Eddem Diaba and Abdelrahman Barakat, " Path Planning: Wavefront Algorithm" Ashesi Robotics, Ashesi University, Ghana, 2012, [Online], Available:

https://ashesirobotics.wordpress.com/2012/12/28/wavefront-planning-task-4/

Appendix

A.1 Arduino Navigation Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#include<SoftwareSerial.h> //Included SoftwareSerial Library
//Started SoftwareSerial at RX and TX pin of ESP8266/NodeMCU
//SoftwareSerial s(8,9);

LiquidCrystal_I2C lcd(0x27,20,4);

//IR Sensor Pins
int leftIR = A1;
int rightIR = A0;
int centerIR = A2;

//IR Sensor Values
int leftValue;
int rightValue;
int centerValue;

//IR Sensor Threshhold for B/W
int threshHold;

//FRFFFRFFFT
String path = "FFLFRFT"; //Initialized variable to store recieved data
String action;

//Motors
int motorA1 = 5; //IN1
int motorA2 = 6; //IN2
int motorB1 = 10; //IN4
int motorB2 = 11; //IN3

//Motor Control
int enable = 3;
int turn_delay = 500;
int turn_speed = 225;
int vSpeed = 120;

//US Sensor
int echoPin = 8;
int trigPin = 2;
long duration;
int distance;

//button pin
int button = A3;
int buttonValue;

//white and black averages
float blackAvg = 0;
float whiteAvg = 0;

//path index
int Idx = 0;
int test = 0;

void setup() {
  //Serial communication
  Serial.begin(9600);
  //s.begin(9600);

```

```
62.   //IR sensors
63.   pinMode(leftIR, INPUT);
64.   pinMode(rightIR, INPUT);
65.   pinMode(centerIR, INPUT);
66.
67.   //US sensors
68.   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
69.   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
70.
71.   //Motors
72.   pinMode(motorA1, OUTPUT);
73.   pinMode(motorA2, OUTPUT);
74.   pinMode(motorB1, OUTPUT);
75.   pinMode(motorB2, OUTPUT);
76.
77.   //Speed
78.   pinMode(enable, OUTPUT);
79.
80.   //LCD
81.   lcd.init();                      // initialize the lcd
82.   lcd.init();
83.   lcd.backlight();
84.
85.   //Calibrating Black Value
86.
87.   lcd.setCursor(0,0);
88.   lcd.print("Put Sensors on Black");
89.
90.   lcd.setCursor(0,2);
91.   lcd.print("Then press OK button");
92.
93.   buttonValue = analogRead(button);
94.   Serial.print("buttonValue");
95.   Serial.println(buttonValue);
96.
97.   Serial.print("Put IR Sensors on Black");
98.   Serial.println("And press the okay button");
99.
100.         while (buttonValue != 0){
101.
102.         leftValue = analogRead(leftIR);
103.         rightValue = analogRead(rightIR);
104.         centerValue = analogRead(centerIR);
105.
106.         delay(1000);
107.
108.         blackAvg = (leftValue + rightValue + centerValue)/3;
109.         buttonValue = analogRead(button);
110.         delay(1000);
111.          }
112.
113.         lcd.clear();
114.
115.         lcd.setCursor(0,0);
116.         lcd.print("Average Black Value ");
117.         Serial.print(" Average Black Value:");
118.
119.         lcd.setCursor(9,2);
120.         lcd.print(blackAvg);
121.         Serial.println(blackAvg);
122.
123.         delay(2500);
124.
125.         //Calibarting white value
126.
127.         lcd.clear();
```

```
        lcd.setCursor(0,0);
        lcd.print("Put Sensors on White");

        lcd.setCursor(0,2);
        lcd.print("Then press OK button");

        buttonValue = analogRead(button);

        Serial.print("Put IR Sensors on White");
        Serial.println(" And press the okay button");

        while (buttonValue != 0){

        leftValue = analogRead(leftIR);
        rightValue =  analogRead(rightIR);
        centerValue =  analogRead(centerIR);
        delay(1000);

        whiteAvg = (leftValue + rightValue + centerValue)/3;
        buttonValue = analogRead(button);

        }

        lcd.clear();

        lcd.setCursor(0,0);
        lcd.print("Average White Value");
        Serial.print("Average White Value:");

        lcd.setCursor(9,2);
        lcd.print(whiteAvg);
        Serial.println(whiteAvg);

        delay(2500);

        //Displaying threshold

        threshHold = (whiteAvg+blackAvg)/2;

        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Threshhold Value");
        Serial.print("Threshhold Value: ");

        lcd.setCursor(9,2);
        lcd.print(threshHold);
        Serial.println(threshHold);
        delay(2000);

        //Initializing System
        lcd.clear();
        lcd.setCursor(2,0);
        lcd.print("SWIFTRON SYSTEM");

        lcd.setCursor(4,2);
        lcd.print("INITIALIZED");

        Serial.println("SWIFTRON SYSTEM INITIALIZED");
        delay(2000);
       }


      void loop() {
          pathFollower();
          stopMotors();
        }
```

```
194.
195.
196.       void pathFollower(){
197.
198.           test = 0;
199.
200.           action = path[Idx];
201.
202.           Serial.println(action);
203.
204.           //display to LCD and Serial
205.           screen();
206.
207.           lcd.setCursor(0,3);
208.
209.           if (action == "F"){ //+ while idx
210.
211.             while (test == 0){
212.
213.             //Read from sensors
214.             leftValue = analogRead(leftIR);
215.             rightValue =  analogRead(rightIR);
216.             centerValue =  analogRead(centerIR);
217.
218.             screen();
219.             lcd.setCursor(9,3);
220.
221.                  //Forward
222.               if ((leftValue < threshHold) && (centerValue > threshHold) && (ri
    ghtValue < threshHold)){ //(ir3 > threshHold) &&
223.                   forward(vSpeed);
224.                   Serial.print("Forward");
225.                   lcd.print("Forward");
226.                   }
227.
228.                 //Left, leftIR on black
229.               else if((leftValue > threshHold) && (rightValue < threshHold)){
230.                   spinLeft(turn_speed,turn_delay);
231.                   Serial.print("SpinLeft");
232.                   lcd.print("SpinLeft");
233.                   }
234.
235.                  //Right, rightIR on black
236.               else if((leftValue < threshHold) && (rightValue > threshHold)){
237.                   spinRight(turn_speed,turn_delay);
238.                   Serial.print("SpinRight");
239.                   lcd.print("SpinRight");
240.                   }
241.
242.                  //All sensors on black
243.               else if((leftValue > threshHold) && (centerValue > threshHold) &&
    (rightValue > threshHold)){
244.                   Serial.print("Black");
245.                   lcd.print("Black");
246.                   //exit;
247.                   test = 1;
248.
249.                   forward(vSpeed);
250.                   delay(150);
251.                   stopMotors();
252.                   ++Idx;
253.                   }
254.
255.                  //All sesnsors on white
256.                else if((leftValue < threshHold) && (centerValue < threshHold) &
    & (rightValue < threshHold)){//(centerIR < threshHold) &&
```

```
                    Serial.print("White");
                    lcd.print("White");
                    //exit;
                    test = 1;

                   forward(vSpeed);
                   delay(150);
                   stopMotors();
                    ++Idx;

                    //look for RFID, if not found stop

                    if (s.available()>0) {

                           //s.listen();
                           RFID = s.readString(); //Read the serial data and stor
    e it
                           Serial.println(RFID);
                            lcd.clear();
                            lcd.setCursor(3,0);
                            lcd.print("RFID DETECTED");

                            lcd.setCursor(0,1);
                            lcd.print("ID: ");

                            lcd.setCursor(3,1);
                            lcd.print(RFID);

                            lcd.setCursor(0,3);
                            lcd.print("Swiftron Relocalized");


                           }
                    }

                  else{
                     stopMotors();
                     Serial.print("Unknown");

                     lcd.setCursor(0,3);
                     lcd.print("Error: Robot Stopped");
                     }

                delay(200);
                }

            }


          else if(action == "L"){

            spinLeft(turn_speed,1250);
            lcd.print("Turning Left: 90");
            Serial.println("Turning Left: 90");
            ++Idx;

            }

          else if(action == "R"){

            spinRight(turn_speed,1250);
            lcd.print("Turning Right: 90");
            Serial.println("Turning Right: 90");
            ++Idx;

            }
```

```
322.
323.         else if(action == "T"){
324.
325.           spinRight(turn_speed,2500);
326.           lcd.print("Turning 360");
327.           Serial.println("Turning 360");
328.           ++Idx;
329.
330.           }
331.
332.         else{
333.
334.           stopMotors();
335.           delay(2500);
336.           lcd.print("MISSION COMPLETE");
337.           Serial.println("MISSION COMPLETE");
338.
339.           }
340.
341.           delay(3500);
342.
343.        }
344.
345.       void screen(){
346.          lcd.clear();
347.           lcd.setCursor(3,0);
348.           lcd.print("SWIFTRON ROBOT");
349.
350.           //Read from sensors
351.           leftValue = analogRead(leftIR);
352.           rightValue =  analogRead(rightIR);
353.           centerValue =  analogRead(centerIR);
354.
355.           //IR Values Display on LCD
356.           lcd.setCursor(2,1);
357.           lcd.print(leftValue);
358.
359.           lcd.setCursor(9,1);
360.           lcd.print(centerValue);
361.
362.           lcd.setCursor(16,1);
363.           lcd.print(rightValue);
364.
365.           lcd.setCursor(3,2);
366.           lcd.print("ThreshHold: ");
367.
368.           lcd.setCursor(15,2);
369.           lcd.print(threshHold);
370.
371.           Serial.println(threshHold);
372.
373.           //IR Values Display on PC
374.           Serial.print("Left: ");
375.           Serial.println(leftValue);
376.
377.           Serial.print("Right: ");
378.           Serial.println(rightValue);
379.
380.           Serial.print("Center: ");
381.           Serial.println(centerValue);
382.
383.           Serial.print("");
384.           Serial.println("");
385.
386.         }
387.
```

```
388.        void forward(int Speed){
389.
390.          Serial.println("[] SWIFTRON going forward");
391.          digitalWrite (motorA1,LOW);
392.          digitalWrite(motorA2,HIGH);
393.          digitalWrite (motorB1,LOW);
394.          digitalWrite(motorB2,HIGH);
395.
396.          analogWrite(enable, Speed);
397.          }
398.
399.        void reverse(){
400.          Serial.println("[] SWIFTRON reversing");
401.          digitalWrite (motorA1,HIGH);
402.          digitalWrite(motorA2,LOW);
403.          digitalWrite (motorB1,HIGH);
404.          digitalWrite(motorB2,LOW);
405.
406.          analogWrite(enable, turn_speed);
407.          }
408.
409.        void stopMotors(){
410.          Serial.println("[] Motors Stopped");
411.          digitalWrite (motorA1,LOW);
412.          digitalWrite(motorA2,LOW);
413.          digitalWrite (motorB1,LOW);
414.          digitalWrite(motorB2,LOW);
415.          }
416.
417.        void spinLeft(int Speed, int Delay){
418.              Serial.println("[] SWIFTRON turning LEFT");
419.              digitalWrite(motorA2, LOW);               // GET /H turns the LED on
420.              digitalWrite(motorA1, HIGH);
421.
422.              digitalWrite(motorB2, HIGH);
423.              digitalWrite(motorB1, LOW);
424.
425.              analogWrite(enable, Speed);
426.              delay(Delay);
427.              stopMotors();
428.          }
429.
430.        void spinRight(int Speed, int Delay){
431.              Serial.println("[] SWIFTRON turning RIGHT");
432.
433.              digitalWrite(motorA2, HIGH);               // GET /H turns the LED o
    n
434.              digitalWrite(motorA1, LOW);
435.
436.              digitalWrite(motorB2, LOW);
437.              digitalWrite(motorB1, HIGH);
438.
439.              analogWrite(enable, Speed);
440.              delay(Delay);
441.              stopMotors();
442.          }
443.
```