

**ORDER MANAGEMENT SYSTEM AND VISION BASED AUTONOMOUS DELIVERY ROBOT FOR RESTAURANTS**

Graduation project report

Submitted in partial fulfillment of the requirements for the

**Honors Degree of Bachelor of the Science of Engineering**

in

The Department of Electrical and Telecommunication Engineering

Faculty of Engineering

South Eastern University of Sri Lanka

University Park, Oluvil, #32360

Sri Lanka

by

**A. N. F. M. Begum (SEU/IS/13/EG/002)**

**G. S. U. AHMADH (SEU/IS/13/EG/011)**

**I. I. S. PREMASRI (SEU/IS/14/EG/032)**

Supervised by:

Dr. W.G.C.W. Kumara

Mr. M.F.M. Abdul Cadar

20 January 2019

# DECLARATION

**Declaration by Project Group**

We hereby declare that the project work entitled “Order Management System and Vision Based Autonomous Delivery Robot for Restaurants” system and work carried out at this university, is a record of an original work done by our team under the guidance of the supervisors Dr. WGCW Kumara & Eng. MFM. Abdul Cader, and the partial fulfillment of the requirements for the Honors Degree of Bachelor of the Science of Engineering. The results expressed in this report have not been submitted to any other university or institute for any degree or diploma or other similar titles.

………………………

ANFM. Begum

………………………

GSU. Ahmadh

………………………

IIS. Premasri

…………………………….

Date

**Declaration by Supervisors**

I/ we hereby declare that the preparation of this project work for the submission of the degree was supervised and accepted in accordance with the guidelines.

|  |  |
| --- | --- |
| …………………………….  Dr. WGCW Kumara  ………………………  Eng. MFM. Abdul Cader | …………………………….  Date  …………………………….  Date |

# APPROVAL

This Final Year Project Report entitled "Order Management System and Vision Based Autonomous Delivery Robot for Restaurants" was submitted in partial fulfilment of the requirements for Honors Degree of Bachelor of the Science of Engineering, in the Department of Electrical and Telecommunication Engineering, Faculty of Engineering, and was approved by,

|  |  |
| --- | --- |
| ……………………………………….  Head, Department of Electrical and  Telecommunication Engineering  Supervisor: Dr. WGCW Kumara  Signature: ……………………………  Supervisor: Eng. MFM. Abdul Cader  Signature: …………………………… | Date:……………………….  Date:…………………………….  Date:……………………………. |

# ABSTRACT

In the recent years, application of robots in daily life has become very normal. Robot autonomy has been exploited to extend their usage in several daily life. For mobile robots to have an autonomy, a very good and a reliable navigation system is vital. Object avoidance, path planning, obtain maps and other information about their environment are very much required for such an autonomous navigation. The process of determining the location of the robot and generating a map of its environment has been termed in the literature as Simultaneous Localization and Mapping (SLAM). There are various approaches by making use of Light Detection and Ranging (LiDAR), Sound Navigation and Ranging (SONAR) sensors, and depth cameras in order to achieve this purpose. However, a vision-based approach is more preferable in replacement of these techniques as it enables the paths for adopting to suddenly changing environments. In vision-based approaches for the robot navigation, most of the existing systems are working based on a single camera mounted in the front side of the robot to sense the immediate environment in front of it. They use Markers like ARUCO markers and similar other various approaches. Considering their inefficiency, inaccuracy and prone to errors, there are some advanced systems which are deploying an external camera to sense the entire environment and to build the map and thus having the path planned according to the overall status of the environment. Our approach is to build such a system and to implement it on a real application.

This project consists of developing such centralized-external camera-based navigation system and to deploy it on a restaurant delivery robot. Existing restaurant waiter robots are mostly involved some classical navigation systems such as visible or magnetic line following. Therefore, this was chosen for the implementation of the system. In addition to that, our project also includes a web-based order management system for customers to reserve tables and order foods and thus the navigation system can acquire the delivery table details and navigate the robot to the table.

# ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our supervisors Dr. WGCW Kumara and Eng. MFM. Abdul Cader for their precious guidance, continues encouragement, support and help they provided us towards the completion of this project in despite of their busy schedules.

We extend our gratitude for the head of the department of Electrical and Telecommunications Engineering, Dr. PMK Alahakoon, The final year undergraduate project coordinator Eng. R. Hirshan and all the other academic and non-academic staffs of the Electrical and Telecommunication Engineering Department and Department of Mechanical Engineering and their workshop staffs for their constant support to our project.

Finally, we would like to thank all of our friends, relations and all others who helped us in numerous ways in the completion of our project within the limited time frame.

# TABLE OF CONTENTS

DECLARATION ii

APPROVAL iii

ABSTRACT iv

ACKNOWLEDGEMENT v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES ix

1. INTRODUCTION 10

1.1 BACKGROUND 10

1.2 OBJECTIVES 10

2. LITERATURE REVIEW 12

2.1 Mobile Robot Self-Localization System Using Single Webcam Distance Measurement Technology in Indoor Environments 12

2.1.1 Image Processing 12

2.1.2 Robot Localization 14

2.2 Path planning algorithm for robot navigation 16

2.3 Restaurant delivery robot design and control system 17

2.3.1 Design of the Robot Mechanism 18

2.3.2 The Fixed Path Guided Systems 18

3. MATERIALS AND METHODS 19

3.1 Customer order management system – For customers to reserve the table and place the orders. 19

3.2 External vision-based path planning – Planning a path for the robot to navigate to the delivery table based on the current objects in the environment 20

3.2.1 Capturing and Processing the Images of the Room 20

3.2.2 Detection of Objects in the Room and Classifying of Moveable Paths and Obstacles 20

3.2.3 Generating the Path for the Robot to Its Destination 21

3.2.4 Regeneration of Path When a New Obstacle Is in the Previous Path 21

3.3 Robot’s hardware and control system 22

3.3.1 Defining Requirements of the robot 22

3.3.2 Hardware Design 22

3.3.3 Robot drive mechanism 22

3.3.4 Control System 23

3.4 Communication between the system 25

3.5 Bill of Materials 25

4. RESULTS AND DISCUSSION 27

4.1 Order management system 27

4.2 Navigation system 30

4.2.1 Image capturing and calibration of camera 30

4.2.2 Object detection and obstacle-path map building 31

4.2.3 Path planning and obstacle avoidance by re-planning of path 32

4.2.4 Communication between MATLAB and navigation system 33

4.2.5. Navigation System Structure 33

4.3 Robot Design, weight estimation and building 34

4.3.1 Designing Robot model 34

4.3.2 Total Weight Estimation 35

4.3.3 Calculation of motor torque 35

4.3.4 Center of Gravity Calculation 36

4.3.5 Hardware Selection 38

4.3.6 Robot Building and Control 39

4.3.7 Stepper Motor control and robot navigation 41

5. CONCLUSIONS 50

REFERENCES 51

# LIST OF FIGURES

Figure 1: Image before and after camera calibration [1] 13

Figure 2: Map of the experimental location [1] 14

Figure 3: Schematic diagram of the IBDMS [1] 15

Figure 4: Schematic diagram of the PLDMS [1] 16

Figure 5: Path planning in workspace with many obstacles [3] 17

Figure 6: The first generation of the restaurant service robot [4] 18

Figure 7: System diagram of the overall proposed system 19

Figure 8: Ultrasonic sensor detects immediate obstacle in the front [5] 21

Figure 9: Proposed differential drive system for the robot 23

Figure 10: Flow chart showing the control system for the robot 24

Figure 11: Connection between the subsystems 25

Figure 12: Homepage of the OMS system with food list 27

Figure 13: Order selection and conformation 27

Figure 14: Login page 28

Figure 15: Database for OMS with relevant tables 29

Figure 16: Orders are listed on the admin interface. 29

Figure 17: Table reservation information 30

Figure 18: Original bird view image on the left and the calibrated and corrected image on the right 31

Figure 19: Image segmentation and clustering. Image on the left shows the identification of objects and moveable path. And the image in the left shows after applying a contour. 31

Figure 20: Image with moving objects on the right and the final processed image after applying segmentation and clustering 32

Figure 21: Binary image of objects and the moveable path is on the left, generated path is in the right 32

Figure 22: Sudden introduction of a new object on the path of the robot is on the left and the regenerated path is on the right 33

Figure 23: Communication Message Structure 33

Figure 24: 3D model of the proposed robot design 35

Figure 25: 3D model of the actual robot design 36

Figure 26: Single line diagram shows the weight distribution 38

Figure 27: NEMA 23 Stepper motor and its properties 39

Figure 28: TB6600 Stepper motor controller and its properties 39

Figure 29: Final design of the robot 40

Figure 30: Bottom plate of the robot 40

Figure 31: Aluminum plate of the robot 41

Figure 32: Electronics and actuators mounting on the robot 41

Figure 33: System diagram 42

Figure 34: Turning angle calculation 43

Figure 35: Wheel rotation calculation 43

Figure 36: Current coordinate calculation of the robot 44

Figure 37: Obstacle detection by ultrasonic sensors 44

# 

# LIST OF TABLES

Table 1: Bill of Materials for the robot design 25

Table 2: Estimated weight for the components and total weight of the robot design 35

# 1. INTRODUCTION

## 1.1 BACKGROUND

Path planning and navigation of autonomous robots and vehicles have always been one of the mostly researched areas in robotics and automobile industry. Giving the autonomous robots a power to navigate to their target destination without having a collision nor a mis-navigation is a very essential task. Indoor robots, unlike autonomous vehicles on outdoor environment, their configuration space is not typical and also the obstacles are mostly unpredictable. Number of indoor robots like restaurant service robot, elder care robots and room cleaning robots etc., are increasing to take their places everywhere. Some of them have cameras to see their immediate surroundings and some have range sensors to detect immediate obstacles. But none of them give the robots an insight of what's beyond their line-of-sight and thus their path planning methods are very ineffective and irrespective of other obstacles which cannot be seen or detected with the sensors built-in.

In order to overcome this limitation and thus giving indoor robots a possibility to get their paths planned according to the overall status of their configuration spaces, our approach will make use of cameras mounted on top of the room to get an over-sight of the entire room. With the use of those cameras, the robot will be able to get a map of the entire room with obstacles marked on it. Therefore, it will be able to navigate to its final destination considering the entire configuration space.

## 1.2 OBJECTIVES

The purpose of this project to compensate for the issues faced in a real time indoor navigation system by providing a solution that is irrespective of the environment and the type of robot by making this system to be able to consider the varying factors in the environment such as moving objects like human and even displacements of objects like furniture in the room.

While the system itself would be having its own hardware including camera and a computer for processing, we planned this project to have the deliverables as a complete application used in a real environment. Such navigation system is very much applicable for service robots. Therefore, our project will bind this system over a real application and thus a working model will be demonstrated. A restaurant delivery robot will be built and its navigation will be handled by the aforementioned navigation system of our project to ease its navigation to the delivery table. In addition to automating the delivery process, it is planned to build an order management system to receive orders from the customers and to reserve tables for them. This system will be integrated to the navigation system to get the target destination for the robots.

As a summary, our deliverables are as follows,

i. Customer order management system – For customers to reserve table and place orders,

ii. External vision-based path planning – Planning a path for the robot to navigate to the delivery table by capturing the environment, and

iii. Robot’s hardware and control system – Building the robot and making a control system for the robot for driving it along the generated path by avoiding any sudden obstacles.

# 2. LITERATURE REVIEW

## 2.1 Mobile Robot Self-Localization System Using Single Webcam Distance Measurement Technology in Indoor Environments

Many recent studies have focused on the problem of mobile robot self-localization. This paper discusses the concept under two main parts which are image processing technique to detect the robot in real time and the distance measurement system to localize the robot.

### 2.1.1 Image Processing

Image processing technique uses following photography methods in order to identify robot,

1. Camera Calibration,
2. Image Segmentation,
3. Morphological Image Processing, and
4. Connected-Components Labeling.

1. Camera Calibration:

Camera calibration is the process of calculating the extrinsic and intrinsic parameters of a camera. Once we calibrate it the 3-D information can be retrieved from 2-D image. World points are transformed to camera coordinates using the extrinsic parameters. The intrinsic parameters have the polynomial mapping coefficients of projection function. The camera coordinates are mapped into the image plan using the intrinsic parameters.

The transformation from world points to camera point is given by the following equation and the component R and T together will form the extrinsic parameters as .

(1)

where,

: Camera points,

: World points,

*R*: Rotation, and

*T*: Translation.

In order to obtain the accuracy of the robot location, the distortions should be removed. Using camera calibration, we can correct radial and tangential distortions in the images as given below.

|  |  |
| --- | --- |
|  |  |

Figure 1: Image before and after camera calibration [1]

2. Image Segmentation:

Image segmentation allows to isolate objects by converting the grayscale image into binary image. For this, a threshold value is used in the image’s pixel and each pixel is compared with it.

*if f1(x, y)> t* (2)

*Fbinary (x, y) = 255*

else

*Fbinary (x, y) =0*

*Ffg (x, y) = |fbg (x, y) – fcurrent (x, y) |* (3)

where,

*t*: threshold value, and

*f‑1(x, y)*: The luminance of the pixel.

3. Morphological Image Processing:

It helps to extract image components that are useful in the representation and description of region shape using morphological image processing operations such as dilation and erosion. They deal with discontinuous edges and noise in the images.

4. Connected-Components Labeling:

It is an algorithm which is used to identify the connected-components that share similar pixel intensity. It helps for pattern recognition.

### 2.1.2 Robot Localization

After the procedures of the image processing, the coordinates in the image domain can be received. Consider the robot’s environment with three webcams as shown in Figure 3. To find out the coordinates (*xi*, *yi*), the distance between the wall and webcam *wi*and the distance between wall and robot’s center *di* have to be calculated.

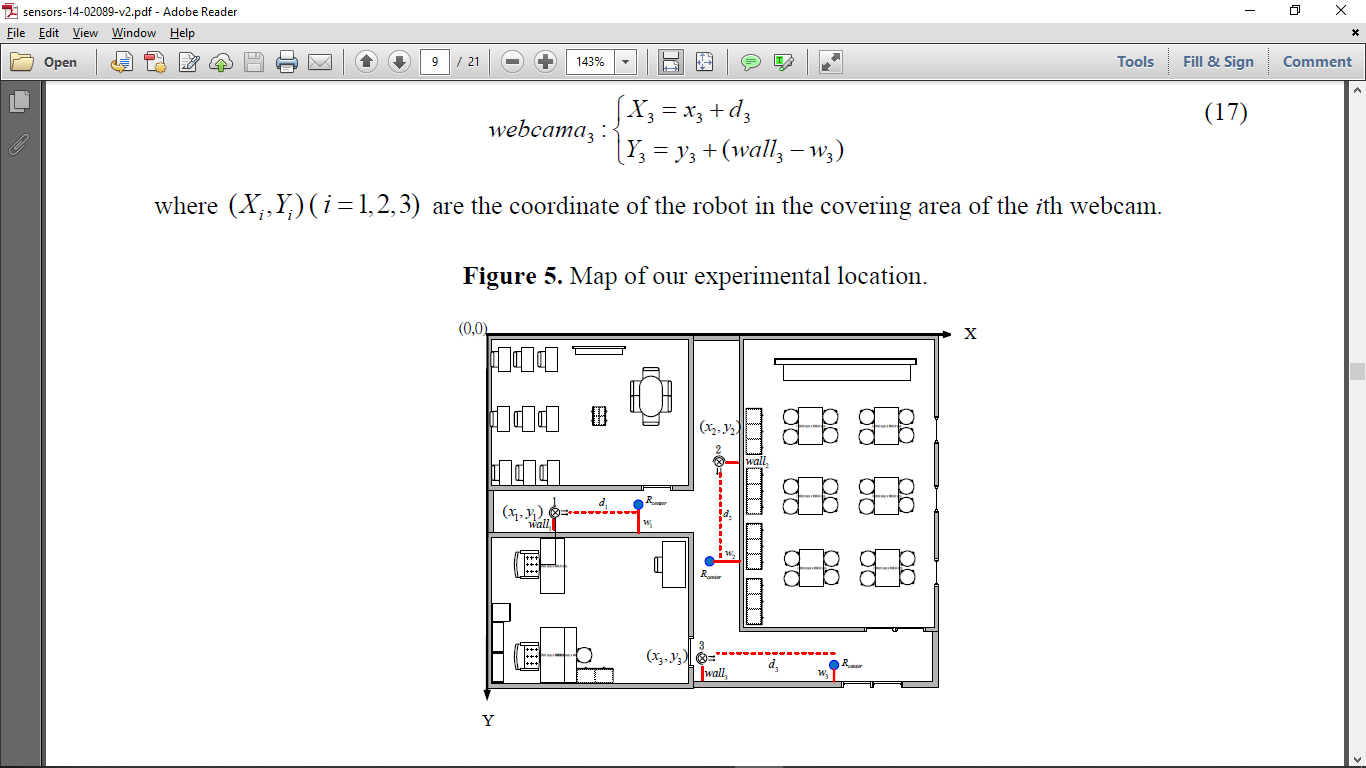


Figure 2: Map of the experimental location [1]

In order to do this, a single-webcam distance measurement technique for indoor robot localization is proposed in the paper which are Image-Based Distance Measurement System (IBDMS) and Parallel Lines Distance Measurement System (PLDMS). Here, only one webcam is needed for estimating the distance.

1. Calculation of *di* using IBDMS:

The idea behind the IBDMS is derived from the triangular relationship. An image is captured by combining the known dimensional rectangle at first. Then the proportion relationship between real and image dimension of the rectangular can be found. Hence, the distance between wall and webcam (*di*) is calculated as in the Equation (4).

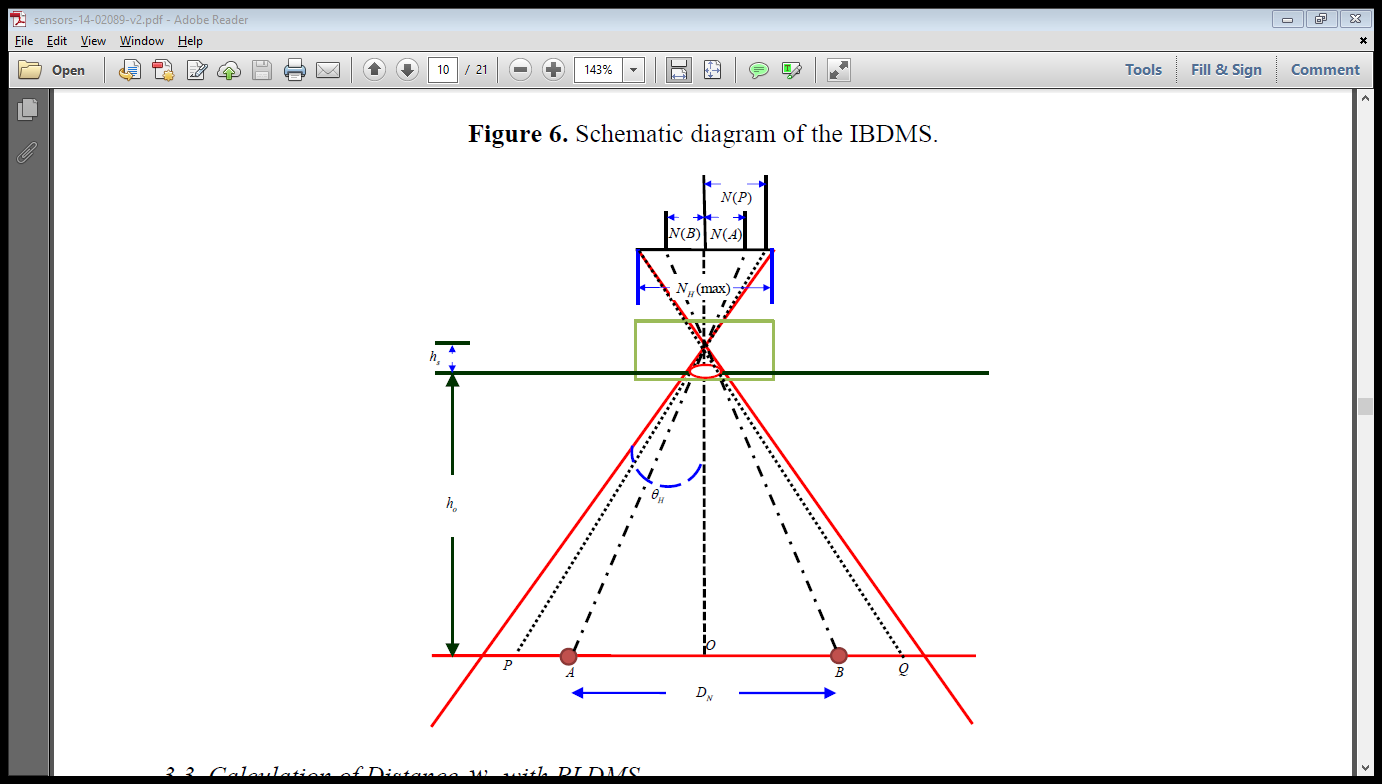


Figure 3: Schematic diagram of the IBDMS [1]

(4)

where,

*ho*: distance *di* (*i* = 1,2,3),

*hs*: constant parameter of the webcam,

*A, B*: given location points,

*N(A), N(B)*: pixel values in the captured image,

*NH max*: maximum pixel width values in the captured image,

*DN*: width between points *A* and *B*, and

*θH*: horizontal view angle.

2. Calculation of *wi* using PLDMS:

As in the Figure 5, there is a linear relationship between actual distance and image distance which similar to the concept of IBDMS. Therefore, the actual distance from the wall to the robot can be calculated from a simple formula as in Equation (5).

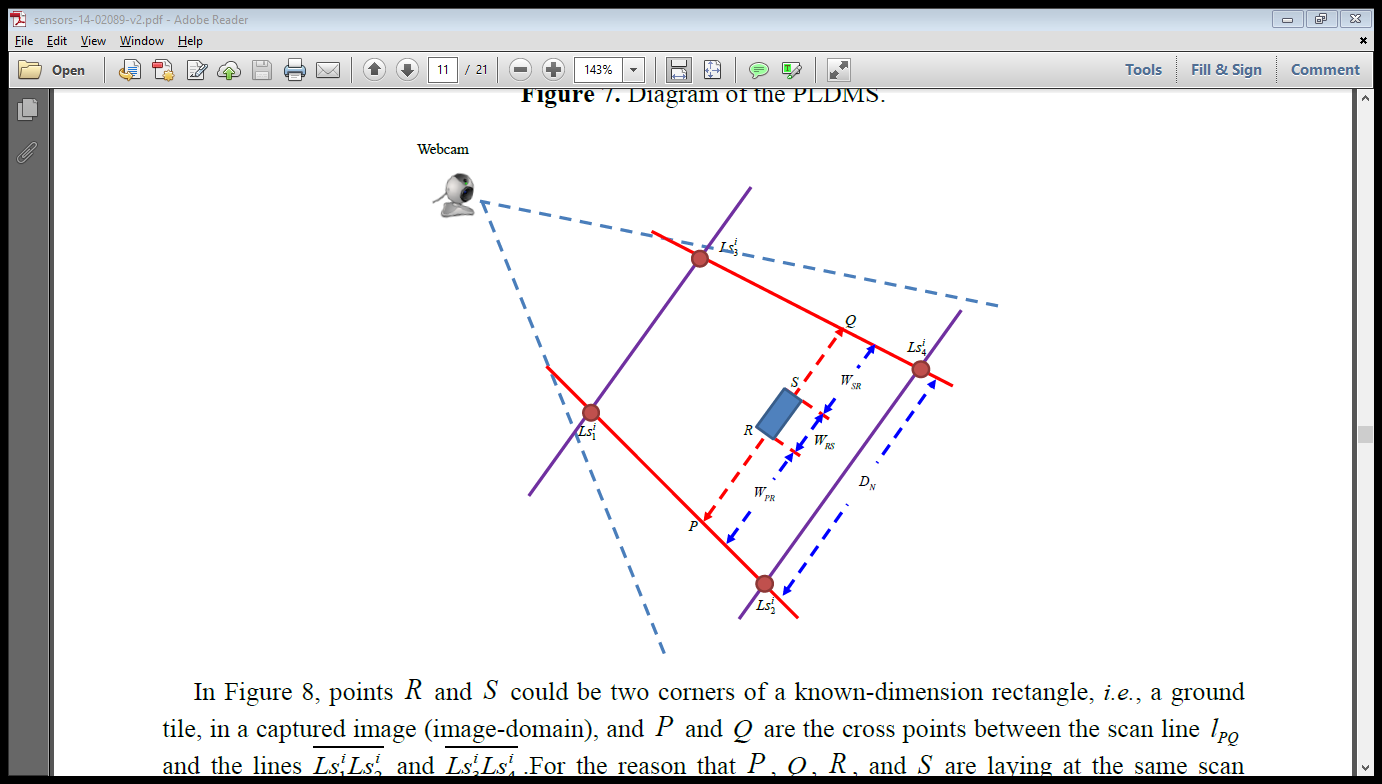


Figure 4: Schematic diagram of the PLDMS [1]

(5)

where,

*R, S*: corners of a known-dimension rectangle,

*WRS*: width between *R* and *S*,

*DN*: width between points *A* and *B*,

*NH(RS)*: number of pixels between *R* and *S*, and

*NH(PQ)*: number of pixels between *P* and *Q*.

The same concept is also used to calculate the width *WPR* and *WSQ*. Then the distance between wall and robot will be calculated by adding above three distance easily.

With the use of the external cameras, the robot will be able to get a map of the entire room with obstacles marked on it. Therefore, it will be able to navigate to its final destination considering the entire configuration space.

## 2.2 Path planning algorithm for robot navigation

The path planning of mobile robot in an unknown environment is the core technology in mobile robot research, which attracts a large number of researchers all over the world. It refers to find a path from the start point to the goal point in the work environment with unknown obstacles, and assures the safe, non-collisions and avoidance of all obstacles as mobile walking. According to the degree of integrity of the environmental information, robot path planning can be divided into local path planning and global path planning [2]

There are considerable researches on the first field, such as space-based partition of unity method, projection method, configuration space method and so on. In local planning, environment information is unknown, or is only partly known, and robot sensor system can only detect local environmental information it perceived. The most representative study in this field is artificial potential field and probabilistic roadmap. Artificial potential field is simple and innovative, which has been widely used in recent years, but is can easily lead to a partial die point, which makes the robots unable to go to the goal. Probabilistic roadmap method in complicated environments is difficult. [3]

As an in improved and a better consideration for the path planning, Rapidly Exploring Random Trees and its variations are being proposed by many researches.

The RRT approach to path planning introduces a technique of determining a planning path by selecting random points within a known environment and moving towards that point an incremental distance from the nearest node of an expanding tree. The movement from existing traversed points to random points in the environment will make a path that looks like a tree, and will cover most of the free space in the environment. A planned path will be found when a branch in the tree comes close the goal positions. [3]



Figure 5: Path planning in workspace with many obstacles [3]

## 2.3 Restaurant delivery robot design and control system

Mobile robots have the capability to move around in their environment and are not fixed to one physical location. For any mobile device, the ability to navigate in its environment is important. Avoiding dangerous situations such as collisions and unsafe conditions comes first, but if the robot has a purpose that relates to specific places in the robot environment, it must find those places. This section describes about robot building, Fixed Path Guided Systems and Laser positioning systems which are used to localize and navigate mobile robots.

### 2.3.1 Design of the Robot Mechanism

The design selected for the mobile robot should be able to fulfil the requirements of the task have to be archived. Here the design for a restaurant service robot is discussed. For this design, Differential drive system is used and main body of the service robot is consisted with five layers,

1. First layer (Bottom layer) - batteries and reflective type infrared sensors for obstacle avoidance,
2. Second layer - Sensors signal level converters, Voltage converters, Voltage and current meters and Motor Drivers,
3. Third layer - PC based robot controller, I/O card and USB to RS232 converter.
4. Fourth layer - The dish transmission system, and
5. Fifth layer (Top layer) - Touch screen, Wireless network module, camera, Speaker and Laser positioning system.



Figure 6: The first generation of the restaurant service robot [4]

### 2.3.2 The Fixed Path Guided Systems

The fixed path guided systems considered as old guided methods regarded as a low-cost guided method for robot navigation. Five infrared sensors have placed on the bottom facing the ground which can sense the location of the path and follows it. Obstacle Avoidance is done by eight reflective infrared sensors placed around the robot.

# 3. MATERIALS AND METHODS

For the achievement of the objectives the overall system is being proposed to be as illustrated in the Figure 8 with the system diagram of the overall proposed system. A User interface is at the end-user side which will be used by the customer for ordering food and to reserve their tables. The data will be sent to the main server and from there the management will receive the reserve and prepare the food and mark it ready to deliver once it is ready. Then, the delivery table number will be notified to the navigation system for its use for planning the path for the robot and depending on the logic explained further below the robot will be navigated to the target destination.

Figure 7: System diagram of the overall proposed system

Each components of the project are further three main parts of the project is further described below.

## 3.1 Customer order management system – For customers to reserve the table and place the orders.

A web-based order and reserve management system will be built with the following tools,

* In the back-end:
  + PHP – Scripting Language
  + MySQL – Database
  + Apache – Server
* In the front-end:
  + HTML – Markup language
  + CSS - Styles
  + JavaScript – Browser level processing

Management staff will have the available foods updated on the database and for security purposes, admin interface will have a login interface that blocks any unauthorized access to the back-end. The customer on their mobile device, will access the restaurant’s portal and reserve their table and order their foods.

When an order is placed, staff gets notified in their portal and once the food is ready, they mark it ready to deliver. Based on reservations and food orders, the database is automatically updated by marking the reserved tables as not-available and reducing the available food count.

As a summary, the following will be available on the system,

* User interface for the order to view available food and tables and thus making reservations,
* Management staffs, updating availability of food and tables and receiving new reservation notification
* Staffs, updating the system when the food is ready to be delivered.

## 3.2 External vision-based path planning – Planning a path for the robot to navigate to the delivery table based on the current objects in the environment

### 3.2.1 Capturing and Processing the Images of the Room

A fish eye lens camera will be mounted on top of the room to cover a wide area. Images captured from the camera will be having distortions throughout the images. In order to get a fixed and a proper image a calibration is needed. A standard method of calibration is to use a known size of chess board to get the camera intrinsic and extrinsic parameters and thus correcting the image distortions. From the calibration process a matrix is generated and is used for further operations.

### 3.2.2 Detection of Objects in the Room and Classifying of Moveable Paths and Obstacles

Once undistorted image of the room is achieved, then the objects in the room need to be identified in order to get a view of moveable path to allow the path planning algorithm to generate its paths in such a way it doesn’t go through any obstacles.

There are two kinds of objects in the room,

1. Fixed objects like furniture and pillars, and
2. Dynamic objects like human.

These objects will be identified and thus an occupancy grid map will be generated. Based on which the path planning will be performed.

### 3.2.3 Generating the Path for the Robot to Its Destination

Once the objects and moveable area in the environment are identified, now the path needs to be generated. For this purpose, the starting and destination nodes need to be defined. From the order management system, the delivery table number will be received and having its coordinates as the destination node and the robots resting point as the starting node, the map will be updated.

Using this map, navigation system will use Extended Rapidly Exploring Random Tree (RRT\*) algorithm for generating a path. This algorithm allows us to work with dynamically varying environment where path generation need to be re-done according to the moving obstacles.

Once this path is generated, it is transmitted to the robot and thus the robot follows the path. A built-in range sensor in the front of the robot will keep looking for any sudden obstacles come to collide on it. To avoid collision, the robot will stop moving if there is an object on its path. Once there is an obstacle on its path and it can’t move forward, the robot will allow a waiting period to check whether the object is moving away from the path. If not, the robot will need a new path and it requests the system for a new path.

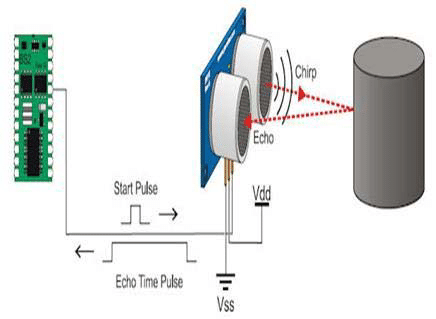


Figure 8: Ultrasonic sensor detects immediate obstacle in the front [5]

### 3.2.4 Regeneration of Path When a New Obstacle Is in the Previous Path

A regeneration of path will be done by simply changing the start point to the current position of the robot and generating the path again based on current parameters. And the newly generated path will be transmitted to the robot.

When the request for a path generation is received by the system it goes through all the previous processes of detecting obstacles and generating the binary image of obstacles and moveable area.

## 3.3 Robot’s hardware and control system

Building the robot and making a control system for the robot for driving it along the generated path by avoiding any sudden obstacles.

### 3.3.1 Defining Requirements of the robot

Before designing any robotic system, the first procedure is to identify its requirements. The following are a set of hardware requirements needs to be defined for this robot build.

* The robot should be able to move while carrying food to any table avoiding obstacles.
* The maximum payload robot should be able to carry.
* Speed that robot should be able to travel.
* Ground clearance of the robot.
* Robot height.
* Robot should be of low cost.

### 3.3.2 Hardware Design

The design is chosen such that it should be able to hold electronics components such as microcontrollers, sensors, battery and hold food we need to deliver. Based on the considerations mentioned in section 3.3.1, a model for the robot will be designed and the robot will be built. It will have a tray for carrying food and have spaces for accommodating necessary electronics and control elements.

### 3.3.3 Robot drive mechanism

A mobile robot needs locomotion mechanisms that enable it to move unbounded throughout its environment. One of the cost-effective solutions for mobile robot navigation is differential drive systems. The differential drive robot (Figure 10) consists of two wheels mounted on a common axis controlled by two separate motors and for balance and smooth riding, two caster wheels will be employed in the front and in the back.

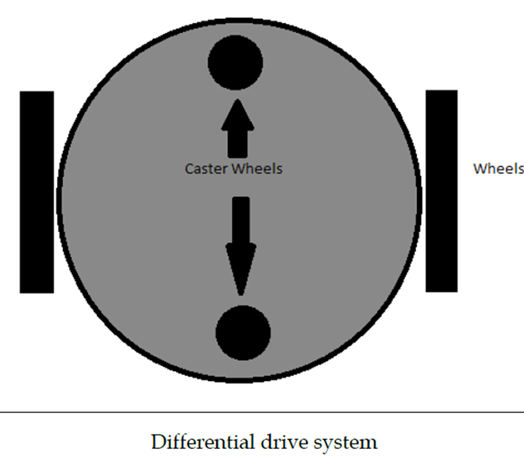


Figure 9: Proposed differential drive system for the robot

### 3.3.4 Control System

Robot needs to receive the coordinates from the navigation system continuously for the path and to navigate to each point. For this purpose, a wireless communication is needed. Once the coordinates for a point is received the motor speeds will be calculated differentially and the robot will move to the point.

Once it reaches the point, it will send a message to the navigation system and ask for the next coordinate and so on and so forth this will keep continuing until the desired delivery location is achieved.

The flow chart in Figure 11 portrays the logic flow that will power the robot’s control system.

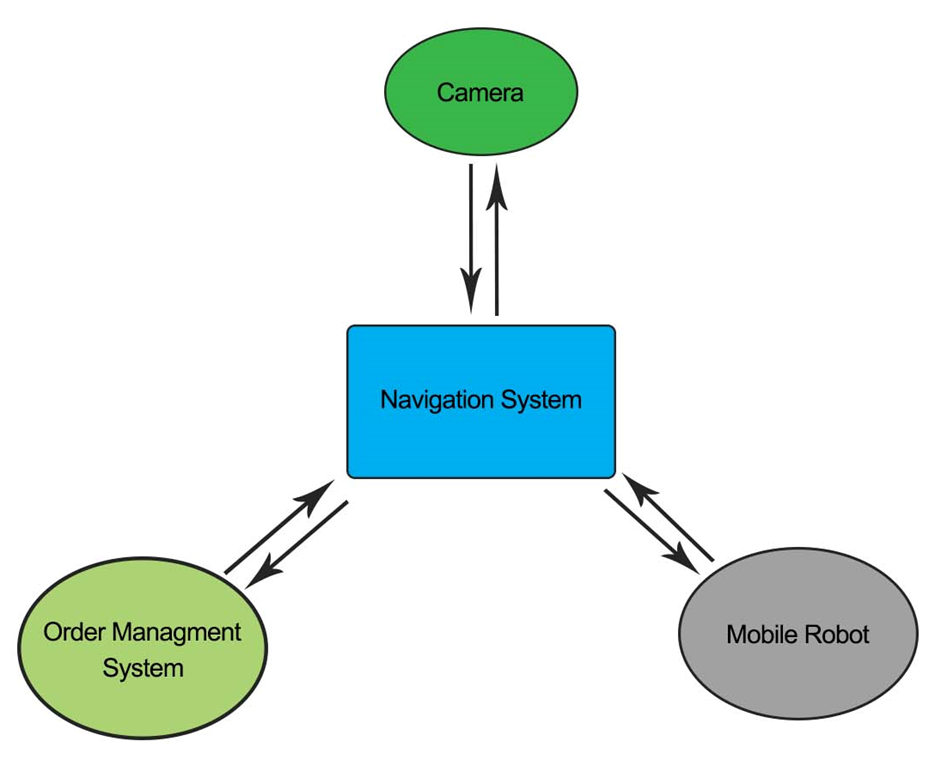


Figure 10: Flow chart showing the control system for the robot

## 3.4 Communication between the system

The system of the waiter robot is connected with mainly three subsystems. In order to complete the task, the communication is achieved as follows,

* Connection 1: The OMS is placed on the webpages and database and the navigation system runs on MATLAB. These two systems are connected with ODBC data source administrator and MATLAB data explorer.
* Connection 2: The data between MATLAB and robot’s hardware is transferred by the WebSocket.
* Connection 3: A camera mounted on the top of the room is connected to the Wi-Fi and the images are accessed from MATLAB to get into processing.



3

2

1

Figure 11: Connection between the subsystems

## 3.5 Bill of Materials

Table 1: Bill of Materials for the robot design

|  |  |  |
| --- | --- | --- |
| ITEM | QTY | PRICE |
| Nema 23 Stepper Motor | 2 | 7028.5 |
| TB6600 Upgraded Stepper Motor Driver | 2 | 2831.73 |
| Ball Bearings | 4 | 1807.95 |
| Aluminum Shaft Coupling | 2 | 678.89 |
| Arduino MEGA | 1 | 1,750.00 |
| Arduino UNO | 1 | 1000 |
| NodeMCU | 1 | 770 |
| logic level converter | 1 | 200 |
| Ultrasonic Sensors | 2 | 360 |
| M5 Stack Camera with Fish eye lenses | 1 | 5000 |
| Drive Wheels | 2 | 1000 |
| Caster Wheels | 2 | 700 |
| Aluminum fitting Shop - Fees |  | 1000 |
| Thread rod, nuts and bolts |  | 780 |
| Welding shop - Oluvil |  | 500 |
| Motor Mounts | 2 | 1100 |
| Nut and Bolts, Drill Bits, Sanding Paper |  | 1000 |
| LiPo Battery and Charger | 1 | 7000 |
| Wires and Connectors |  | 500 |
| **Total Expenses** |  | 35007.07 |

# 4. RESULTS AND DISCUSSION

## 4.1 Order management system

The Order Management System has been created to act as an interface between the user and system. When the customer browses the website, the dishes are shown in Figure 13. Then the orders which is selected by customer, are stored in the relevant database. There is also a popup message which is indicated order reference ID and order confirmation when the customer finishes the ordering process as in the Figure 14.

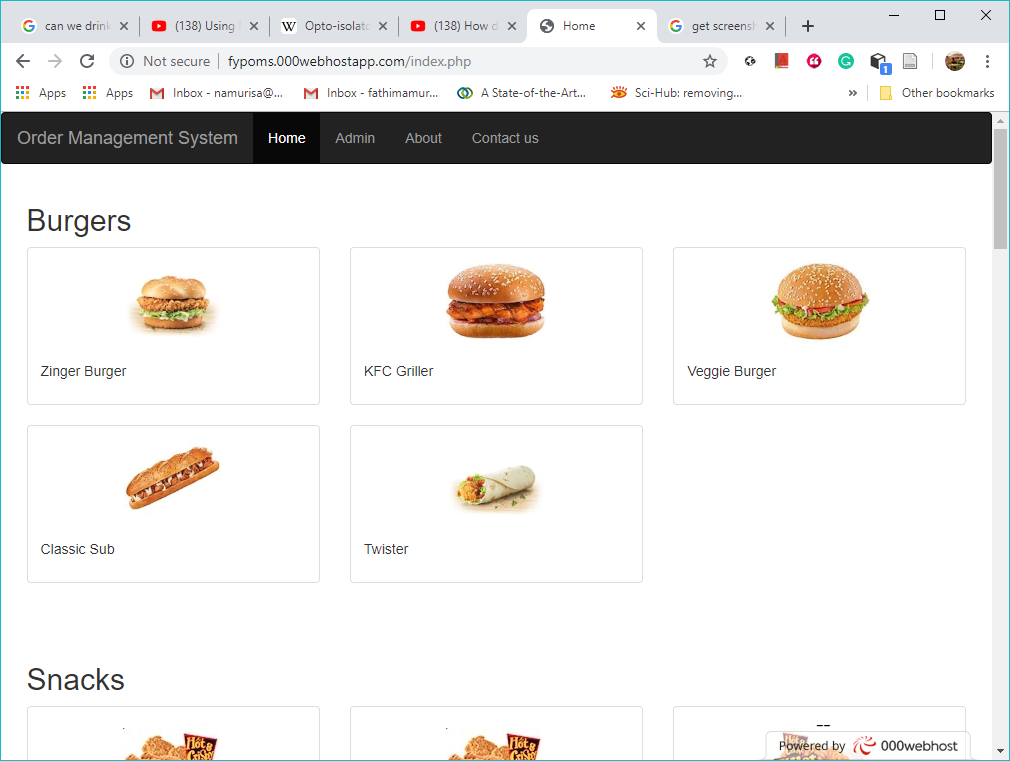


Figure 12: Homepage of the OMS system with food list

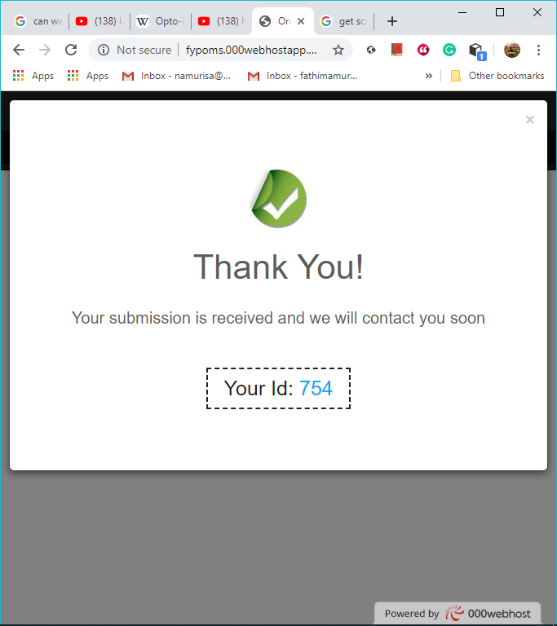
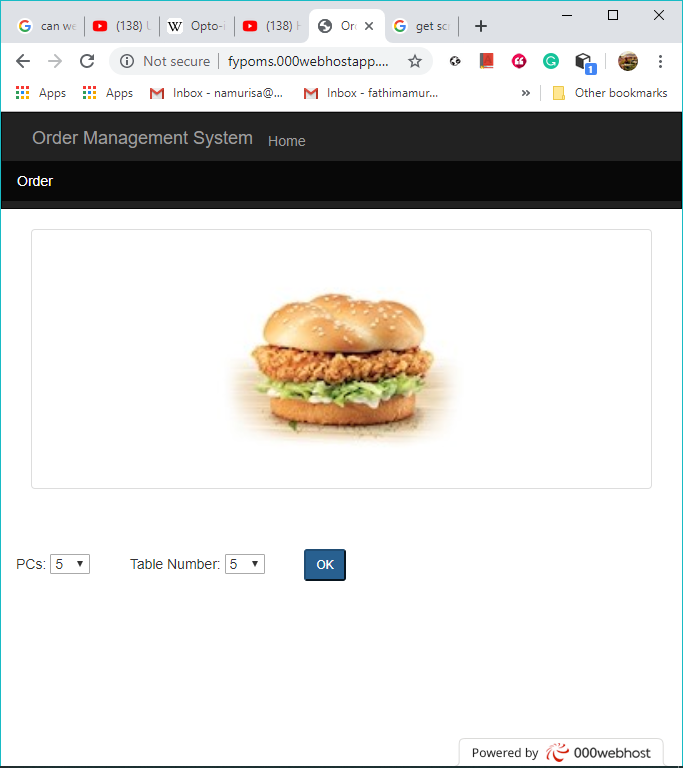


Figure 13: Order selection and conformation

The webpage is designed with two main interfaces: User Interface and Admin Interface. Login page is integrated to block unauthorized access to the admin interface (Figure 13).

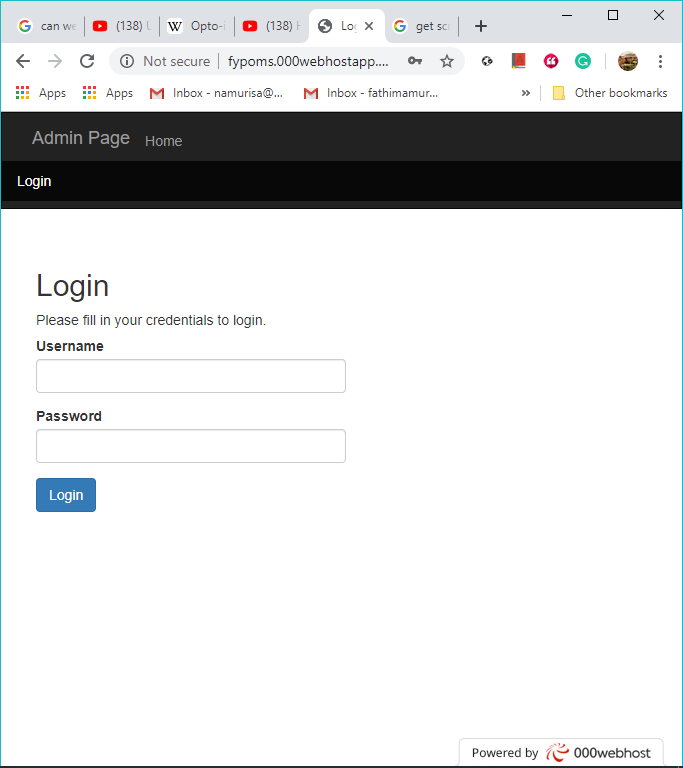


Figure 14: Login page

User interface is designed for customer who is asked to select the food items, enter the table number and give other payment details on the web page. Likewise, the admin updates the order details on the web page and food availability on the database. Since the food items that are displayed for ordering is directly taken from the database, customer can order only the available dish in the restaurant.

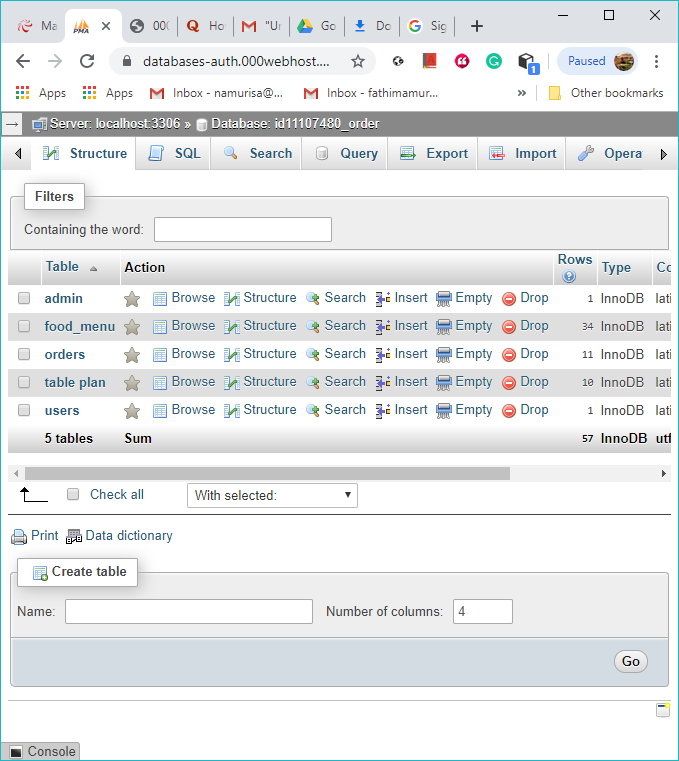


Figure 15: Database for OMS with relevant tables

Orders are listed in admin interface like in the Figure 15. Main purpose for having this listed item is to notify the table number to the robot and showing the food list to the chef.



Figure 16: Orders are listed on the admin interface.

In addition to that, there is a table reservation page which can be used to indicate the table status for admin, deliver details to find the order status and the payment details to inform whether the payment was made or not to the billing system.

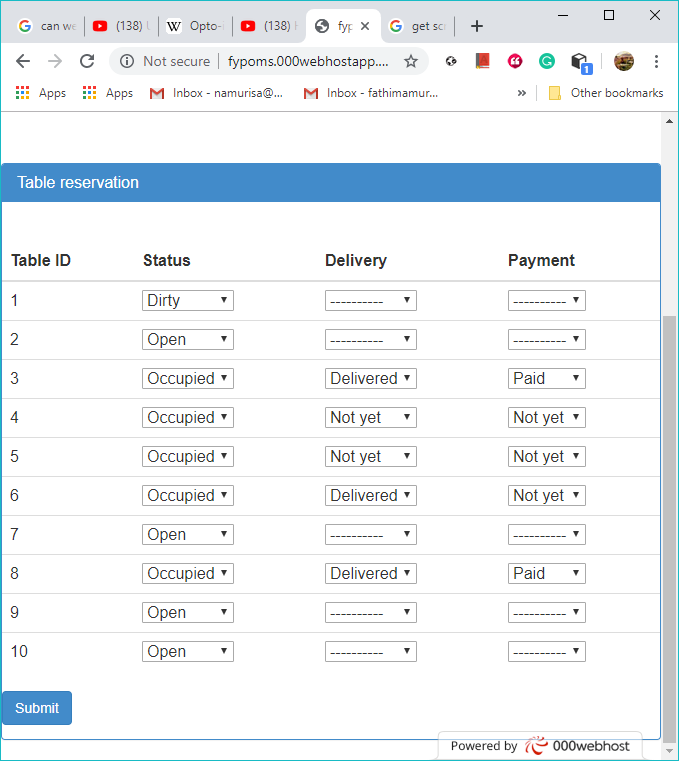


Figure 17: Table reservation information

## 4.2 Navigation system

For the centralized navigation system, initial experiments are done for the purpose choosing,

* The right camera for capturing most of the area of interest
* Well working algorithm that suits the environment conditions for object detection and map building
* Path planning algorithm that fits the circumstances where the environment changes and the conditions are not specific.

### 4.2.1 Image capturing and calibration of camera

After experimenting with various types of cameras, a camera with a wide-angle lens was chosen to cover most area in one shot in order to avoid further processing of images from different many cameras.

After mounting the camera on the top center of the faculty cafeteria, images were taken and it was observed to be distorted. For fixing them, a chess board having cells with known dimensions was used. From the images of the chess board of various orientations, matrix was generated which was later applied over all the images taken from the camera to have an undistorted image.

|  |  |
| --- | --- |
| WhatsApp Image 2019-07-26 at 21 | 2 |

Figure 18: Original bird view image on the left and the calibrated and corrected image on the right

### 4.2.2 Object detection and obstacle-path map building

Firstly, the captured image on which there were no presence of any dynamic objects was run through image segmentation and clustering and then a cost map was generated. Later, based on a threshold it was turned into a binary image marking objects as black and the moveable area as white.

But as this image was having inner white space between parts of objects which may be assumed by our path planning algorithms as a moveable path area, these spaces needed to be blacked out. For this purpose, detected objects were bounded into a contour.

|  |  |
| --- | --- |
| C:\Users\Umair Ahmadh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3.jpg | C:\Users\Umair Ahmadh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.jpg |

Figure 19: Image segmentation and clustering. Image on the left shows the identification of objects and moveable path. And the image in the left shows after applying a contour.

And as in a real environment, there will be moving objects. In order to give the robot a path, all objects in the environment at the time of generating the path need to be considered. To do so, the reference background image was used for identifying moving objects on top of it. The same process which was used in the above background reference generation was used here on top of this reference for this purpose.

|  |  |
| --- | --- |
| C:\Users\Umair Ahmadh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.5.jpg | C:\Users\Umair Ahmadh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3.jpg |

Figure 20: Image with moving objects on the right and the final processed image after applying segmentation and clustering

### 4.2.3 Path planning and obstacle avoidance by re-planning of path

As for experimenting the path planning with the proposed algorithm – RRT\*, the starting and target destination coordinates were manually fed into the algorithm and the paths were generated. For this purpose, the binary map of objects and moveable path that was generated earlier was utilized.

|  |  |
| --- | --- |
|  |  |

Figure 21: Binary image of objects and the moveable path is on the left, generated path is in the right

As for experimenting the sudden crossover of an objects in front of the robot image with moving objects on the cafeteria was used in the flow. And rerunning the path planning algorithm on it resulted a non-colliding path for the robot.

|  |  |
| --- | --- |
| 8.111 |  |

Figure 22: Sudden introduction of a new object on the path of the robot is on the left and the regenerated path is on the right

### 4.2.4 Communication between MATLAB and navigation system

Communication between MATLAB and mobile robot is handled by WebSocket connections. Data communicated structured as following.



Figure 23: Communication Message Structure

* Type 0 – Sent from MATLAB to mobile robot. Includes navigation data for robot.
* Type 1 – Sent from robot to MATLAB for path regeneration when an obstacle is detected.
* Type 2 – Sent from robot to MATLAB to acknowledge that robot has finished navigation task and ready for new orders.

### 4.2.5. Navigation System Structure

Navigation system is implemented in MATLAB and it has following tasks implemented in separate functions in separate files. (Attached in Annex 1)

* oms.m – Check for new orders in database and if a new detected, it will call getPath function with order data.
* RobotNavi.m - Called by oms.m with table number for delivery. This function coordinates entire navigation session for robot from MATLAB side including communication with robot and tacking actions according to message data received.
* getPath.m – Called by RobotNavi.m with image received from camera, start pose and goal pose for robot needs to navigate for the delivery. Function will return with 2 data arrays, degrees\_to\_turn and length\_in\_m which contains data for robot to navigate.
* Get\_image.m – Callled by oms.m when new order is detected and will return calibrated image received from camera.
* segmentImage.m – Called by getPath.m with RGB image received from camera and return segmented binary image.
* matlab\_\_py.py – handles WebSocket communication between MATLAB and mobile robot.

## 4.3 Robot Design, weight estimation and building

* Motor torque estimation
* Materials and component selection
* Building the robot

### 4.3.1 Designing Robot model

* A 3D model for the Robot design was built considering the requirements mentioned in 3.3.1.
* Materials for the build were chosen.



Figure 24: 3D model of the proposed robot design

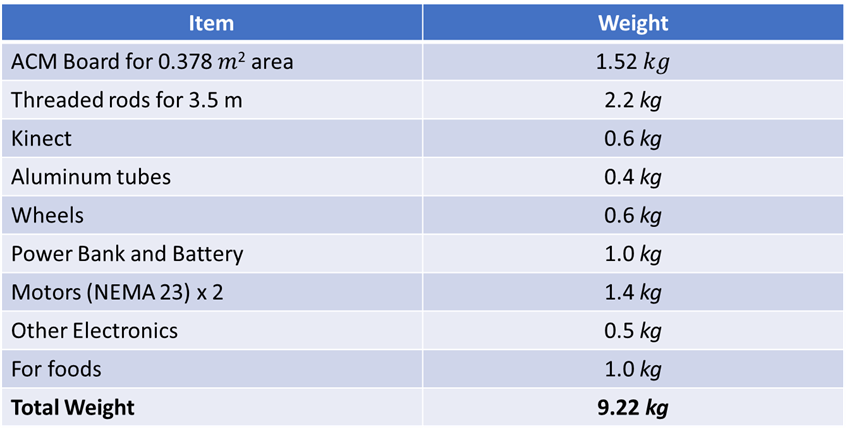
The final build of the robot as follows,



Figure 25: 3D model of the actual robot design

### 4.3.2 Total Weight Estimation

Table 2: Estimated weight for the components and total weight of the robot design

****

### 4.3.3 Calculation of motor torque

In our design we have 4 wheels and 2 motors. For calculation, we assume that the coefficient of friction is 0.6 and radius of wheel is 5 cm. The weight acting on the four wheels can be written as,

(6)

Where,

*N1*: weight acting on each caster wheel.

*N2*: on each motor wheel.

The maximum torque is required when the robot starts moving. It should also overcome friction. To calculate the maximum torque (*T*),

(7)

Where,

*μ* is the coefficient of friction.

*N* is the average weight acting on each wheel.

*r* is the radius of wheels.

*T* is the torque.

Assuming that the weight of the robot is equally distributed on all the four wheels,

(8)

Motor torque required,

(9)

### 4.3.4 Center of Gravity Calculation

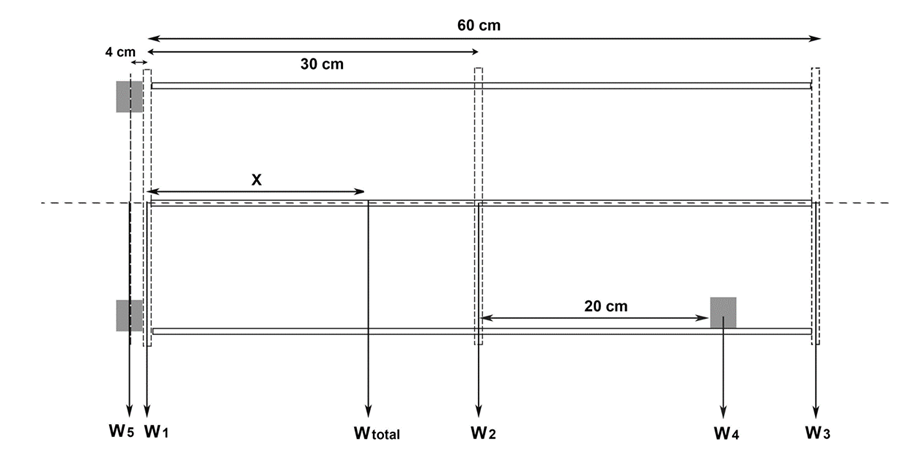


Figure 26: Single line diagram shows the weight distribution

Considering the moments around center of the bottom plate in clockwise direction,

*W1*= (0.8 kg + 0.95 kg + 1 kg + 0.5 kg) x

= 3.25 kg x

= 31.85 N

*W2* = (0.5 kg + 2.2 kg + 0.4 kg + 0.136 kg + 0.05 kg) x

= 3.286 kg x

= 32.20 N

*W3* = (0.5 kg + 0.2 kg + 1.0 kg) x

= 1.7 kg

= 16.66 N

*W4* = (0.6 kg + 0.15 kg) x

= 0.75 kg x

= 7.35 N

*W5* = (1.4 kg + 0.6 kg) x

= 2.0 kg x

= 19.6 N

*W1* x 0 + *W2* x 0.3 m + *W3* x 0.6 m + *W4* x 0.5 m - *W5* x 0.04m = *Wtotal* x *X* (10)

9.66 Nm + 9.996 Nm + 3.675 Nm – 0.784 Nm = 107.653 N x *X*

= (11)

*X* = 0.209 m

where,

*W1*: Steel Bottom plate with frame, Power bank, Battery and Other Electronics,

*W2*: ACM Board, Treaded rods, Aluminum tubes Jetson Nano and Arduino,

*W3*: ACM Board, Aluminum L profile and food,

*W4*: Kinect and holding plate, and

*W5*: Motors and wheels.

According to the calculation the Center of Gravity of the robot design is 0.209 m above the bottom plate. It is 9.1cm below the center.

### 4.3.5 Hardware Selection

Based on above calculations, the following hardware parts were chosen to be employed in the robot build.

* Drive Motor – NEMA 23 Stepper Motor

Holding torque - 1.0 Nm (Required )

Step Angle - 1.8 0

Current - 1.5A

Figure : NEMA 23 Stepper motor and its properties

* Motor Controller – TB6600 Stepper Motor Driver

****

Output Current - up to 4.0A

Micro Step - 1, 2/A, 2/B, 4, 8, 16 & 32

Figure 28: TB6600 Stepper motor controller and its properties

* Controller board for obstacle detection – Arduino Uno
* Controller board – Arduino Mega
* Ultrasonic Sensors
* NodeMCU – Wi-Fi Communication
* Camera – M5Stack Camera with Fish Eye lense

### 4.3.6 Robot Building and Control

Once the calculations were done, the selected materials were bought from different suppliers and put together the final body of the robot.



Figure 29: Final design of the robot

During the process, the bottom plate which is the main controller area was built with steel plates as planned. With the help of the staffs in the Mechanical Workshop in the Department of Mechanical Engineering and also a few helps from 3rd party welding shops, this part was completed successfully.



Figure 30: Bottom plate of the robot

Other plates of the robot were built utilizing the Aluminum Cladding Boards as follows. A local Aluminum Fitting Shop’s help was obtained during the process.



Figure 31: Aluminum plate of the robot

Once the building was done, the electronics and actuators were connected and a test run was run.

|  |  |
| --- | --- |
|  |  |

Figure 32: Electronics and actuators mounting on the robot

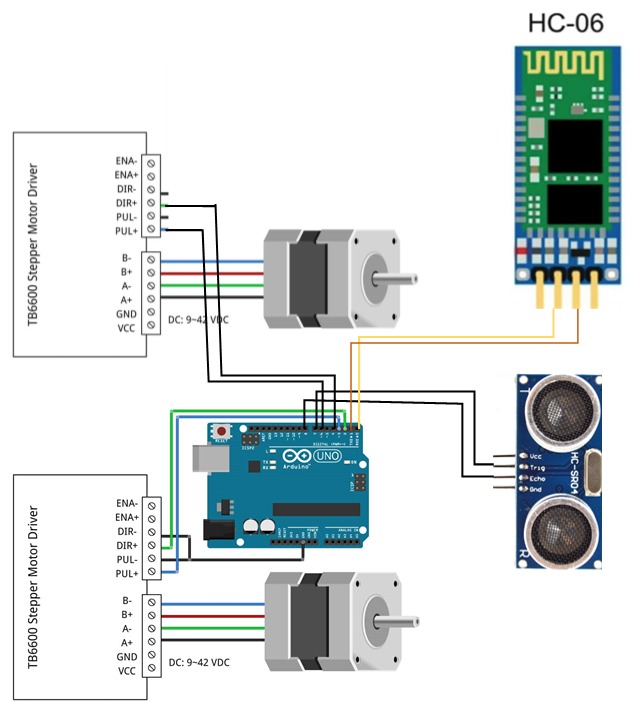


Figure 33: System diagram

By programming the microcontroller all the motors and movements were tested with random controls.

And then according to the following system diagram, the communication was tested, by remotely sending

### 4.3.7 Stepper Motor control and robot navigation

**Calculation** **of** **turning** **angle** **of** **the** **robot**

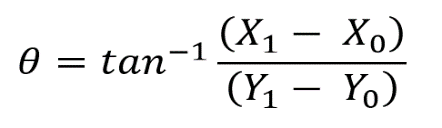
* Using the following equation, the angle of rotation of the robot is calculated. Based on which we will rotate the motors to turn the robot to be straight to the next coordinate.

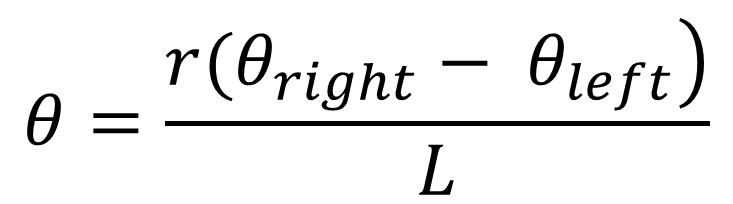
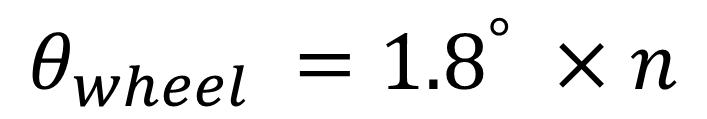


Figure 34: Turning angle calculation

**Wheel** **rotation** **angle** **calculation**



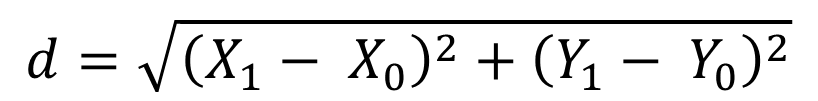
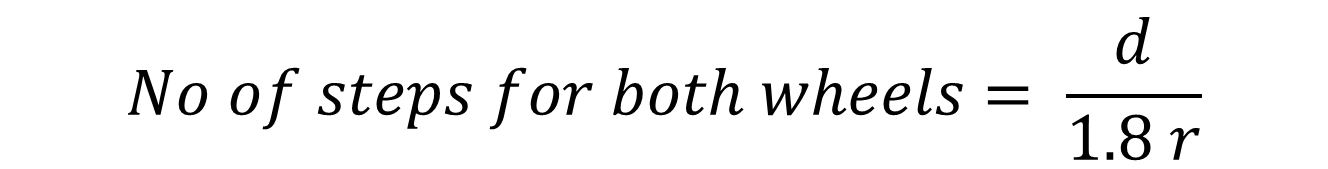
Figure 35: Wheel rotation calculation

**Calculation** **of** **straight-line** **travel** **distance** **for** **the** **robot**

* Using the following function, the distance the robot needs to travel in straight in order to reach the next coordinate can be calculated. Based on which we will rotate the motors to drive the robot in straight until reaching the target coordinate.



Figure 36: Current coordinate calculation of the robot

**Calculation** **of** **wheel** **rotations** **to** **travel** **straight** **to** **the** **next** **coordinate**

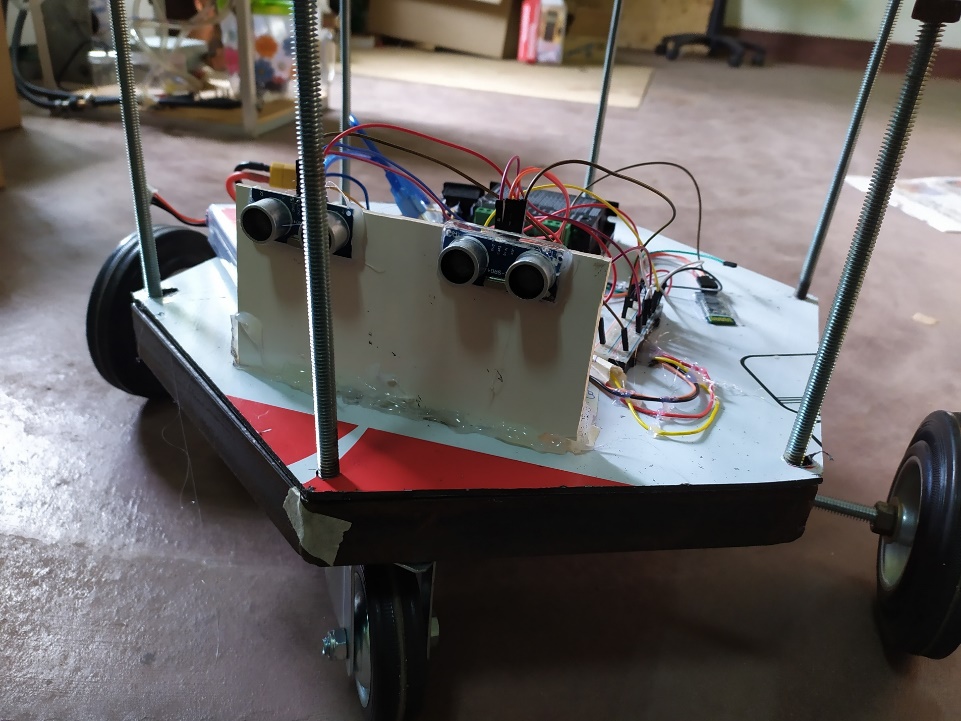


Figure 37: Obstacle detection by ultrasonic sensors

If any obstacles are encountered robot waits for a period and sends to the processing system how many steps it has yet to move from the command it received. Having it as the current position, navigation system regenerates a new path and pilot the robot according to it.

Based on the calculations, the logic was implemented in the Arduino Mega microcontroller as follows.

#include <AccelStepper.h>

#include <MultiStepper.h>

#include <ArduinoJson.h>

//Stepper motor configuration and object creation

AccelStepper stepper2(AccelStepper::FULL2WIRE, 4, 3);

AccelStepper stepper1(AccelStepper::FULL2WIRE, 7, 6);

MultiStepper steppers;

int attempt = 1; //attempt of obsacle scanning.

int stepCount = 0;

int stepRight = 0;

int stepLeft = 0;

const int obstacleAlertPin = 5;

bool alreadyWaited = 0;

void generateReplyMessage(int cmd, float x, float y, int angle) {

delay(10);

DynamicJsonDocument doc(200);

//cmd=1 : Regeneration request

//cmd=0 : Navigation completed

doc["cmd"] = cmd;

doc["x"] = x;

doc["y"] = y;

doc["deg"] = angle;

Serial.println(serializeJson(doc, Serial2));

// Serial2.println(serializeJson(doc, Serial));

//return serializeJson(doc, Serial);

}

//LiquidCrystal\_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

int num\_of\_data;

void receiveCommand() {

if (Serial2.available()) {

String buff = Serial2.readString();

//Serial2.println("Hi How are you?");

Serial.println(buff);

//Serial2.print("done");

data2arrays(buff);

}

}

void pathRegenerateRequest(float x, float y, int deg) {

generateReplyMessage(1, x, y, deg);

receiveCommand();

}

int stepperPosition(int rightStep, int leftStep) {

long positions[2];

positions[0] = rightStep;

positions[1] = leftStep;

steppers.moveTo(positions);

steppers.runSpeedToPosition();

}

int stepCountCalc(int type, int value) {

if (type == 0) { //Straight Movement to centimeters

stepCount = round((value + 0.1 \* value) \* 360 \* 8 / (1.8 \* 1.8 \* 6));

}

else if (type == 1) { //Rotation to Angle

stepCount = round((12975 / 180) \* value);

}

return stepCount;

}

void data2arrays(String pld4mnivi) {

DynamicJsonDocument doc(1024);

DeserializationError error = deserializeJson(doc, pld4mnivi);

if (error) {

Serial.print(F("deserializeJson() failed: "));

Serial.println(error.c\_str());

return;

}

num\_of\_data = doc["num"];

int angle\_data[num\_of\_data];

int dis\_data[num\_of\_data];

for (int i = 0; i < num\_of\_data; i++) {

angle\_data[i] = doc["angle"][i];

dis\_data[i] = doc["distance"][i];

}

stepper1.setCurrentPosition(0);

stepper2.setCurrentPosition(0);

Navigate(angle\_data, dis\_data);

}

void Navigate(int angle\_ar[], int dis\_ar[]) {

float Xdistance = 0; //mur

float Ydistance = 0; //mur

int FAngle = 0; //mur

for (int j = 0; j < num\_of\_data; j++) {

int type, value;

type = 0;

value = 0;

//rotating angle has no issue

stepper1.setCurrentPosition(0);

stepper2.setCurrentPosition(0);

type = 1;

stepRight = stepCountCalc(type, angle\_ar[j]);

stepLeft = -stepRight;

stepperPosition(stepRight, stepLeft);

delay(1000);

//go straight only if there is no obstacle

//going straight need t be sliced

stepper1.setCurrentPosition(0);

stepper2.setCurrentPosition(0);

type = 0; //straight mode is on

int slice\_length = 15; // dliving to this distance

int numberOfSlices = dis\_ar[j] / slice\_length; //going straight in slices this number of time

int remainingSliceLength = dis\_ar[j] % slice\_length; //going straight for the remaining

if (dis\_ar[j] > slice\_length) {

for (int slice\_loop = 0; slice\_loop < numberOfSlices; slice\_loop++) { //going straight in slices this number of time

alreadyWaited = 0;

while ( digitalRead(obstacleAlertPin) == HIGH && alreadyWaited == 0 ) {

delay(3000);

alreadyWaited = 1;

}

if (digitalRead(obstacleAlertPin) == HIGH) {

for (int angle\_index = j; angle\_index >= 0; angle\_index--) { //mur

FAngle += angle\_ar[angle\_index];

}

//pathRegenerateRequest();

pathRegenerateRequest(round(Xdistance), round(Ydistance), FAngle % 360);

}

else {

stepper1.setCurrentPosition(0);

stepper2.setCurrentPosition(0);

stepRight = stepCountCalc(type, slice\_length);

stepLeft = stepRight;

stepperPosition(-stepRight, -stepLeft);

Xdistance += slice\_length \* sin(angle\_ar[j] \* PI / 180); //mur

Ydistance += slice\_length \* cos(angle\_ar[j] \* PI / 180); //mur

// delay(100); //for debugging

}

}

//going straight for the remaining

if (remainingSliceLength != 0) {

alreadyWaited = 0;

while ( digitalRead(obstacleAlertPin) && alreadyWaited == 0 ) {

delay(3000);

alreadyWaited = 1;

}

if (digitalRead(obstacleAlertPin)) {

for (int angle\_index = j; angle\_index >= 0; angle\_index--) { //mur

FAngle += angle\_ar[angle\_index];

}

pathRegenerateRequest(round(Xdistance), round(Ydistance), FAngle % 360); //mur

}

else {

stepper1.setCurrentPosition(0);

stepper2.setCurrentPosition(0);

stepRight = stepCountCalc(type, remainingSliceLength);

stepLeft = stepRight;

stepperPosition(-stepRight, -stepLeft);

Xdistance += remainingSliceLength \* sin(angle\_ar[j] \* PI / 180); //mur

Ydistance += remainingSliceLength \* cos(angle\_ar[j] \* PI / 180); //mur

// delay(100); //for debugging

}

}

}

else {

alreadyWaited = 0;

while ( digitalRead(obstacleAlertPin) == HIGH && alreadyWaited == 0 ) {

delay(3000);

alreadyWaited = 1;

}

if (digitalRead(obstacleAlertPin) == HIGH) {

for (int angle\_index = j; angle\_index >= 0; angle\_index--) { //mur

FAngle += angle\_ar[angle\_index];

}

pathRegenerateRequest(round(Xdistance), round(Ydistance), FAngle % 360); //mur

}

else {

stepper1.setCurrentPosition(0);

stepper2.setCurrentPosition(0);

stepRight = stepCountCalc(type, dis\_ar[j]);

stepLeft = stepRight;

stepperPosition(-stepRight, -stepLeft);

Xdistance += dis\_ar[j] \* sin(angle\_ar[j] \* PI / 180); //mur

Ydistance += dis\_ar[j] \* cos(angle\_ar[j] \* PI / 180); //mur

}

}

}

generateReplyMessage(0, 0, 0,0);

receiveCommand();

}

void setup()

{

// lcd.init(); // initialize the lcd

Serial.begin(9600);

Serial2.begin(9600);

pinMode(obstacleAlertPin, INPUT);

stepper1.setMaxSpeed(1000);

stepper2.setMaxSpeed(1000);

steppers.addStepper(stepper1);

steppers.addStepper(stepper2);

}

void loop()

{

// read from port 1, send to port 0:

/\*if (Serial1.available()) {

int inByte = Serial1.read();

Serial.write(inByte);

}\*/

// read from port 0, send to port 1:

receiveCommand();

}

# 5. CONCLUSIONS

During the period of two semesters this project was carried out in the following phases**,**

1. Brainstorming of different ideas and features and proposal of the final objectives
2. Robot hardware building and Motion control
3. Order management system build and system communication
4. Navigation system – Path Planning and Communication
5. Final implementation of system

Initially the project works were carried out based on the plans, designs and calculation. And after having the hardware setup completed, we started the experimentation and implementation of the logics to achieve the objectives. The order management system was developed to accommodate the customer perspectives and to have the system automated. Based on the order from the customer the robot was supposed to navigate to the respective table from where the order was placed and deliver the food. It was the ultimate objective of the project. We experimented different cameras to check the area coverage, light conditions, etc. to have a smooth path planning system built. Finally, we concluded with the M5 Stack Fish Eye lens camera for the image processing works.

Once the programming and control part was successful, we ran a test run in the canteen to have further modifications. With further advancements, this project can be developed into a complete restaurant automation system.

# REFERENCES

|  |  |
| --- | --- |
| [1] | I.-H. &. C. M.-C. &. W. W.-J. &. S. S.-F. &. L. T.-W. Li, "Mobile Robot Self-Localization System Using Single Webcam Distance Measurement Technology in Indoor Environments," p. 21, 2008. |
| [2] | X. Z. Cai, H. g. He and Chen hong, "Some issues for mobile robot navigation under unknown environments [J]," in *Control and Decision. Vol.8*, 2002. |
| [3] | Liu Chang-an, Chang Jin-gang, Li Guo-dong and Liu Chu, "Mobile Robot Path Planning Based on an Improved," *International Conference on Automation and Logistics,* pp. 2375-2379, 2008. |
| [4] | T. Jyh-Hwa and L. Kuo, "The development of the restaurant service mobile robot with a laser positioning system," *IEEE - 27th Chinese Control Conference,* pp. 662-666, 2008. |
| [5] | S. B. Anila S, "INTELLIGENT SYSTEM FOR AUTOMATIC RAILWAY GATE CONTROLLING AND OBSTACLE DETECTION," 25 07 2019. [Online]. Available: https://www.researchgate.net/publication/318960459\_INTELLIGENT\_SYSTEM\_FOR\_AUTOMATIC\_RAILWAY\_GATE\_CONTROLLING\_AND\_OBSTACLE\_DETECTION. |
| [6] | H. M. L. Devin Connell, "Extended rapidly exploring random tree–based dynamic path planning and replanning for mobile robots," *International Journal of Advanced Robotic Systems,* 2018. |
| [7] | A. K. Z. H. Iram Noreen, "A Comparison of RRT, RRT\* and RRT\*-Smart Path Planning Algorithms," *International Journal of Computer Science and Network Security,* 2016. |
| [8] | C. J.-g. L. G.-d. L. C.-y. Liu Chang-an, "Mobile Robot Path Planning Based on an Improved," in *International Conference on Automation and Logistics*, Qingdao, China. |
| [9] | Zbigniew, "Mapping Camera Coordinates to a 2D Floor Plan," 15 May 2019. [Online]. Available: https://zbigatron.com/mapping-camera-coordinates-to-a-2d-floor-plan/. |
| [10] | R. H. A. M. G. I. &. C. A. Abiyev, "Robot pathfinding using vision based obstacle detection," *3rd IEEE International Conference on Cybernetics (CYBCONF) ,* 2017. |
| [11] | L. Joseph, Learning Robotics using Python, Packt Publishing, 2015. |
| [12] | R. Siegwart and I. Nourbakhsh, Introduction to Autonomous Mobile Robots, Cambridge: MIT Press, 2016. |
| [13] | S. K. L. Tzou Jyh-Hwa, "The development of the restaurant service mobile robot with a Laser positioning system.," p. 5, 2008. |

# 

**Annex 1 - Codes**

Getimage.m

function [undistortedImage] = Get\_image()

%cam = ipcam('http://192.168.43.105:8081/out.jpg');

%img = snapshot(cam);

%clear cam;

img = imread('G:\camera calib\10th\img25.jpg');

IntrinsicMatrix = [368.1660 0 0; 0 365.3690 0; 323.6920 235.6447 1.0000];

radialDistortion = [-0.2725 0.0535];

cameraParams = cameraParameters('IntrinsicMatrix',IntrinsicMatrix,'RadialDistortion',radialDistortion);

%img = imread('C:\cantin\edited.jpg');

J = imresize(img, [360 480]);

undistortedImage = undistortImage(J, cameraParams);

end

getPath.m

------------------------------------------------------------------------------------------------------------

%#codegen

function [degrees\_to\_turn,length\_in\_m] = getPath(I,startPose,goalPose)

%Path planner function

%for full image

%IntrinsicMatrix = [2.0939e+03 0 0; 0 2.1062e+03 0; 2.2971e+03 1.6890e+03 0.0010e+03];

%radialDistortion = [-0.2130 0.0428];

%for 480p image

IntrinsicMatrix = [218.0786 0 0; 0 219.3695 0; 239.4830 176.2512 1.0000];

radialDistortion = [-0.2130 0.0428];

cameraParams = cameraParameters('IntrinsicMatrix',IntrinsicMatrix,'RadialDistortion',radialDistortion);

undistortedImage = undistortImage(I, cameraParams);

J = imresize(undistortedImage, [360 480]);

figure

imshow(undistortedImage);

se = offsetstrel('ball',20,20);

erodedI = imerode(undistortedImage,se);

se = offsetstrel('ball',5,5);

dilatedI = imdilate(erodedI,se);

%figure

%imshow(dilatedI);

B = segmentImage(dilatedI);

% Segment the image.

%[C,scores,allScores] = semanticseg(J,net);

% Overlay free space onto the image.

%B = labeloverlay(J,C,'IncludedLabels',"Road");

% Display free space and image.

%figure

D = single(B);

costmap = vehicleCostmap( D );

%plot(costmap);

%startPose = [125 45 0];

%goalPose = [370 283 0];

planner = pathPlannerRRT(costmap);

refPath = plan(planner,startPose,goalPose);

%figure

%plot(planner)

%allPoses = connectingPoses(refPath);

isPathValid = checkPathValidity(refPath,costmap);

%allPoses(:,2) = [];

%controller.Waypoints = allPoses;

[refPoses,refDirections] = interpolate(refPath);

figure

hold on

plot(costmap);

approxSeparation = 35; % meters

numSmoothPoses = round(refPath.Length / approxSeparation);

[poses,directions,cumLength,curvatures] = smoothPathSpline(refPoses,refDirections,numSmoothPoses);

plot(poses(:,1),poses(:,2),'LineWidth',2,'DisplayName','Smooth path');

pixels\_per\_meter = 100;

cumLength = cumLength(2:end);

Length\_per\_step(1) = cumLength(1);

length\_in\_m(1) = (Length\_per\_step(1)/pixels\_per\_meter) \* 100;

for k = 2:length(cumLength)

Length\_per\_step(k) = cumLength(k) - cumLength(k-1);

length\_in\_m(k) = (Length\_per\_step(k)/pixels\_per\_meter) \* 100;

end

%length\_in\_m(end)=[];

for l = 2:length(poses)

Robot\_rotation\_angle(l-1) = atand((poses(l,2)-poses(l-1,2))/(poses(l,1)-poses(l-1,1)));

%Robot\_rotation\_angle = tan-1((x1-x0)/(y1-y0))

end

for m = 2:length(poses)

Robot\_rotation\_angle(l-1) = atand((poses(l,2)-poses(l-1,2))/(poses(l,1)-poses(l-1,1)));

%Robot\_rotation\_angle = tan-1((x1-x0)/(y1-y0))

qc = halfCirTest(poses(m-1,1),poses(m-1,2),poses(m,1),poses(m,2));

if (qc == 1 | qc == 4)

Robot\_rotation\_angle\_qc(m-1) = 90 - Robot\_rotation\_angle(m-1);

elseif (qc == 2 | qc == 3)

Robot\_rotation\_angle\_qc(m-1) = -(90 + Robot\_rotation\_angle(m-1));

end

end

degrees\_to\_turn(1) = Robot\_rotation\_angle\_qc(1);

for n = 2:length(Robot\_rotation\_angle\_qc)

degrees\_to\_turn(n) = Robot\_rotation\_angle\_qc(n) - Robot\_rotation\_angle\_qc(n-1);

end

%length\_in\_m(end)=[];

degrees\_to\_turn(length(degrees\_to\_turn)+1) = -sum(degrees\_to\_turn);

%length\_in\_m(length(length\_in\_m)+1) = 0;

length\_in\_m = [length\_in\_m, 0];

end

matlap\_py.py

------------------------------------------------------------------------------------------------------------

import websocket as ws\_lib

def Connection\_send\_Recieve(WebSocket\_Ip,Port,Command):

Str="ws://"+WebSocket\_Ip+":"+Port

ws = ws\_lib.create\_connection(Str)

ws.send(Command)

RecivedData=ws.recv\_data()

ws.close()

return RecivedData

oms.m

------------------------------------------------------------------------------------------------------------

conn = database('FYP','root','');

curs = exec(conn,'SELECT orderid,tableid,status FROM orders');

curs = fetch(curs);

orders=curs.Data(:,:)

for i=1:size(orders,1)

if string(orders(i,3)) == "Start"

CurrentTable=cell2mat(orders(i,2))

OrderId=cell2mat(orders(i,1))

break

end

end

if(CurrentTable ~= 0)

[angle2go,length2go] = getPath\_test(CurrentTable);

for i=1:size(orders,1)

if cell2mat(orders(i,2))==CurrentTable

processing = exec(conn,['UPDATE `orders` SET `status`=''Process'' WHERE tableid= ',num2str(CurrentTable)]);

end

end

%

% }

for i=1:size(orders,1)

if cell2mat(orders(i,2))==CurrentTable

processing = exec(conn,['UPDATE `orders` SET `status`=''Delivered'' WHERE tableid= ',num2str(CurrentTable)]);

end

end

CurrentTable=0

end

% % % % % % code for delivery

% {

%

%write the things as below at starting of delivery code

RobotNavi.m

------------------------------------------------------------------------------------------------------------

function [angle2go,length2go] = RobotNavi(table\_num)

close all;

%UNTITLED2 Summary of this function goes here

% Detailed explanation goes here

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%Setting Variables%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%startPose = [125 45 0];

%goalPose = [370 310 0];

Ws\_Ip='192.168.43.42';

Port='80';

delivery\_data = [1 2 3 4 5; 95 230 364 369 393; 205 245 301 171 51];

%startPose = [delivery\_data(2,table\_num) delivery\_data(3,table\_num) 0];

goalPose = [delivery\_data(2,table\_num) delivery\_data(3,table\_num) 0];

origin = [125 45 0];

startPose = origin;

%goalPose = [343 283 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%Start Robot Navigation%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

J = Get\_image();

[angle2go,length2go] = getPath(J,startPose,goalPose);

%[angle2go,length2go] = getPath(J,goalPose,startPose);

s = struct("num",length(angle2go),"angle",round(angle2go),"distance",round(length2go));

json = jsonencode(s)

Result=py.matlab\_\_py.Connection\_send\_Recieve(Ws\_Ip,Port,json)

test = extractBetween(char(Result(2)),"{","}")

char(test)

Result\_Jdec = jsondecode("{"+char(test)+"}");

%%%below this, work in progress. to test, you don't need this.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%Navigation Loop%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

done\_count = 0;

while (done\_count<2)

%%If obsticle detected

if (Result\_Jdec.cmd== 1)

%make new\_starPose from json

J = Get\_image();

new\_startPose = [(Result\_Jdec.x + startPose(1)) (Result\_Jdec.y + startPose(2)) Result\_Jdec.deg];

[angle2go,length2go] = getPath(J,new\_startPose,goalPose);

s = struct("num",length(angle2go),"angle",round(angle2go),"distance",round(length2go));

json = jsonencode(s);

Result=py.matlab\_\_py.Connection\_send\_Recieve(Ws\_Ip,Port,json);

Result

test = extractBetween(char(Result(2)),"{","}")

char(test)

Result\_Jdec = jsondecode("{"+char(test)+"}");

Result

%To return home

elseif(Result\_Jdec.cmd== 0)

done\_count = done\_count + 1;

if(done\_count == 1)

startPose = goalPose;

goalPose = origin;

J = Get\_image();

[angle2go,length2go] = getPath(J,startPose,goalPose);

s = struct("num",length(angle2go),"angle",round(angle2go),"distance",round(length2go));

json = jsonencode(s);

Result=py.matlab\_\_py.Connection\_send\_Recieve(Ws\_Ip,Port,json);

Result

test = extractBetween(char(Result(2)),"{","}")

char(test)

Result\_Jdec = jsondecode("{"+char(test)+"}");

end

end

end

end

segmentImage.m

------------------------------------------------------------------------------------------------------------

function [BW,maskedImage] = segmentImage(RGB)

%segmentImage Segment image using auto-generated code from imageSegmenter app

% [BW,MASKEDIMAGE] = segmentImage(RGB) segments image RGB using

% auto-generated code from the imageSegmenter app. The final segmentation

% is returned in BW, and a masked image is returned in MASKEDIMAGE.

% Auto-generated by imageSegmenter app on 23-Jul-2019

%----------------------------------------------------

% Convert RGB image into L\*a\*b\* color space.

X = rgb2lab(RGB);

% Auto clustering

sz = size(X);

im = single(reshape(X,sz(1)\*sz(2),[]));

MZ = zeros(172800, 3);

KZ = mean(im);

for r = 1:172800

MZ(r,:) = KZ(1,:);

end

im = im - MZ;

MY = zeros(172800, 3);

KY = std(im);

for r = 1:172800

MY(r,:) = KY(1,:);

end

im = im ./ MY;

s = rng;

rng('default');

L = kmeans(im,2,'Replicates',2);

rng(s);

BW = L == 2;

BW = reshape(BW,[sz(1) sz(2)]);

% Invert mask

%BW = imcomplement(BW);

% Create masked image.

maskedImage = RGB;

maskedImage(repmat(~BW,[1 1 3])) = 0;

end