Palestine Polytechnic University

College of IT and Computer Engineering
Department of Computer System Engineering

Graduation Project

[Autonomous Wheelchair Project]

Project Team

Akram abu ayyash
Islam warasna

Project Supervisor

Dr. mohammad aldesht

June 2021

Acknowledgements

We would like to express we deep gratitude to Dr.mohammad aldesht Our project supervisor, who helped us find the project idea and encouraged us to move forward.. I would also like to thank Eng.wael takrouri which helped us clarify some of the points that put us at the beginning of the road .
Thanks to all friends,colleagues and female colleagues.
We would also like to thank all the doctors and teachers who had it not been for their efforts we would not have reached what we have come to, a special thanks to Dr. Radwan Tahboub, Mazen Zalloum, Elayan Abu Gharbia, Nabil Arman, Alaa Al-Halawani, Amal Dweik, Hashem Al-Tamimi, Musa Faraj Allah. All doctors, even those we did not mention, that it was played a crucial role in achieving these results.

Abstract

Although the electric wheelchairs that are controlled by joysticks have improved the lives of many, there is a group of patients suffering from vision problems, amputation of hands, or quadriplegia that cannot control these chairs. Moreover, navigating electric wheelchairs through crowded environments without crashing into objects or people can be a challenging task. For these reasons, it became necessary to develop the electric wheelchair into autonomous wheelchair .

The user job is to enter commands via the mobile phone, either by voice (the name of a location such as a kitchen , or move forward) or by touch (determine a location on the map or controlling the movement of the chair such as moving forward for example), and after the user enters the command ,and with the help of the stored map The chair can locate it on the map, then locate the goal location, then determine the optimal path to reach the goal while avoiding obstacles.

After the research that we conducted and consulting with some doctors and teachers, it became clear to us that there is more than one way (technology) to obtain an autonomous wheelchair. There are easy ways, but less intelligent and less effective, such as using RFID stickers or QRcode stickers and other methods, and there is a more difficult, but more intelligent and effective method, which is to use ultrasonic sensor and encoder with raspberry pi, so that we take advantage of the ultrasonic sensor to draw a map and take advantage of the encoder sensor to locate the robot and know the distance and speed at which the robot moved, and all this is done through raspberry pi, and thus downloading ROS (Robot Operating System) to raspberry pi to write and execute algorithms (mapping, localization, and path planning,) and link them with the chair.

Table of contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Overview

An autonomous wheelchair or smart wheelchair is “a uniquely modified powered wheelchair which is equipped with a control system and sensors. It is also called a mobile robot base to which a seat has been attached. Smart wheelchairs are designed to provide assistance to users who are suffering with severe mobility impairment, visual impairments, etc. The primary purpose of the autonomous wheelchair is to reduce or eliminate the user's full responsibility on moving the wheelchair. They are also specifically designed with respect to the user’s situations and disabilities. A smart wheelchair comes in two different forms, one is a standard power wheelchair to which a computer and a collection of sensors have been added and the second one is a mobile robot base to which a seat has been attached.

1.2 Motivation

The autonomous mobility of humans will be one of the critical issues of the next century as the number of individuals with impaired mobility continues to increase worldwide. Hence, wheelchairs are one of the important devices in the contemporary world since it plays a vital role in determining the quality of life for individuals with mobility impairments such as wheelchair users .

1.3 Problem statement

Driving a wheelchair in indoor environments is a difficult task even for ordinary people and it becomes more difficult for people with disabilities in the arms or hands, or people with quadriplegia or people with blindness are completely unable to operate the joystick, and therefore they always need an escort To help them navigate.

1.4 System objectives

* The development of a manual electric chair into a self-driving chair.
* The chair receives commands through a phone application, and the phone receives commands by voice.
* That the chair be able to avoid any obstacles it faces in its path.
* That the user be able to call the chair to the place he is in.
* The user can give an order for the chair to move it from his place to the place he wants.
* That the user be able to define the chair on two or more specific locations as needed

1.5 Short description of system

This system consists of three main parts: the chair, the user, and the environment in which the user moves,
The autonomous word means self-control or independent , and autonomous wheelchair means that self-driving electric wheelchair that receives commands by voice, or by commands.
And the user is a patient who suffers from paralysis or amputation in the feet or hands or both,
And the environment in which the chair moves is an internal environment such as a home, university, or workplace ... etc.

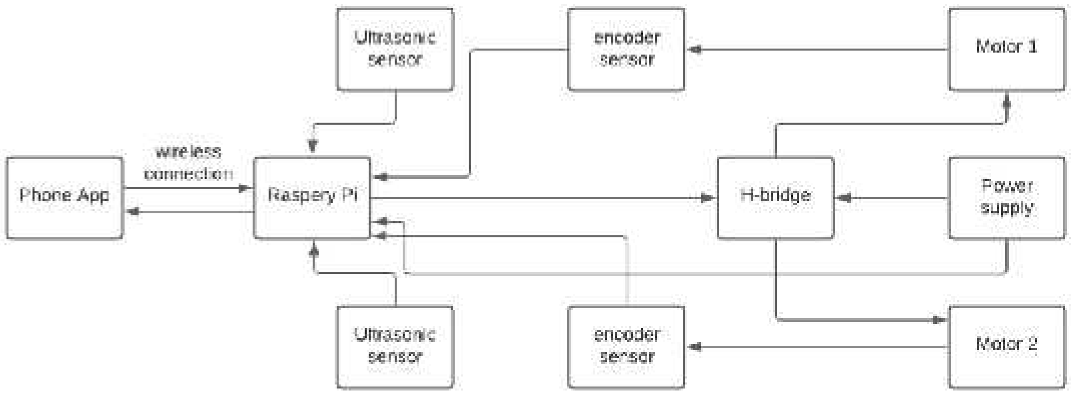

figure 1.1 - general figure for the system .

1.6 Overview of the rest of report section

In chapter 2 , "Background", contains the theoretical background about the algorithm and explains the set of micro algorithms that are used in the algorithm of the wheelchair
In chapter 3 "Design", includes a detailed conceptual description of the system, detailed design, structural diagrams, block diagrams, , and any necessary information about the design
In chapter 4 "Software", includes a description of the implementation, implementation challenge, implementation issues and description of the method used to validate the system, validation results
In chapter 5 "System Analysis and Discussion", includes analysis and discussion about the results.
In chapter 6 "Conclusion", includes, challenges, future directions, , summary and future work

Chapter 2: Background

Overview

This chapter describes briefly the theoretical background of the project, we will introduce a short description of the hardware and software components that are used in the system, and alternatives between them, this chapter contains 3 sections .

Robot Navigation

"Robot navigation means the robot's ability to determine its own position in its frame of reference and then to plan a path towards some goal location. In order to navigate in its environment, the robot or any other mobility device requires representation, i.e. a map of the environment and the ability to interpret that representation" [1].
Navigation can be defined as the combination of the four fundamental competences :

- Perception
- Localization
- Cognition
- Motion Control

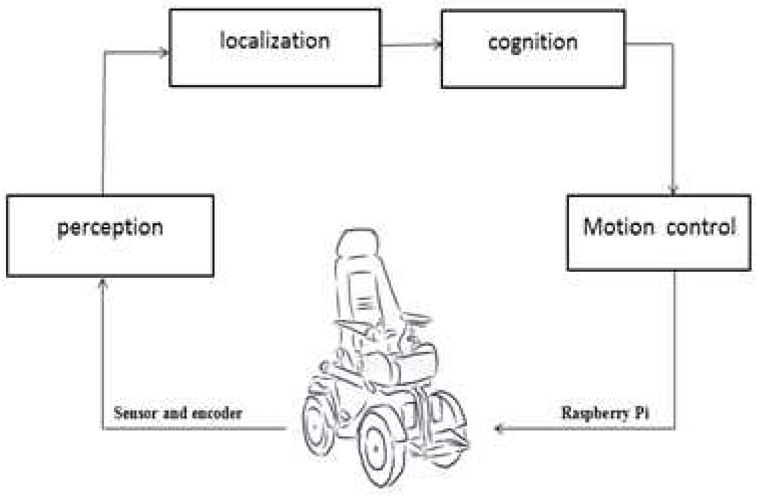

Figure 2.1– Planning Based Control

2.1: Hardware Components

A) Proprioceptive Sensors

1) Compass

It is a sensor that senses the rotational motion of the device added to it.

Figure 2.2 – Compass sensor

2) Encoders

It is a sensor that senses the distance and speed at which the robot has moved.

Figure 2.3– Encoder sensors

3) Accelerometers

it is a sensor that measures the acceleration forces acting on an object, in order to determine the object's position in space and monitor the object's movement.

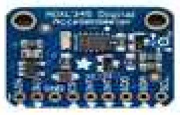

Figure 2.4 – accelerometer

B) Exteroceptive Sensors

1) Vision Systems

A camera (vision system) is added to the robot to help it avoid obstacles.

Figure 2.5– Vision Systems

2) Range Sensors

The sensor can measure the distance between the robot and the closest object to it, so that the reading is between 1 - 255 cm, but it loses the accuracy of the measurement if the object is closer than 5 cm.and there are several types of it:

A. Ultrasonic sensor

it is the most common type of distance measuring sensor, it detects the distance to objects by emitting high-frequency ultrasonic waves .

Figure 2.6– Ultrasonic Sensor.

B. IR sensor

it is sensor it does distance or proximity sensing through emitting IR beam and calculating angle of reflection.

Figure 2.7 - IR sensor.

C. LIDAR sensor

It is a sensor that measures the range of targets through light waves from a laser .

Figure 2.8 - LIDAR sensor.

3) Positioning System (e.g. GPS)

This system consists of two segments master and slave , connected by GPS.

Figure 2.9 – GPS

C) Electronic platforms

1) Raspberry Pi

“The Raspberry Pi is single-board computers (SBCs) , when connected to a monitor, USB keyboard and mouse, the Raspberry Pi is just like an entry-level Linux computer,the system features 2GB or 4GB of RAM, as well as USB-C port for power, and two micro HDMI ports for connecting to up to two displays. There are also USB 2.0 and USB 3.0 ports to connect peripherals, as well as a gigabit Ethernet port and a WiFi & Bluetooth module for respectively wired or wireless networking connectivity”[3].

Product	**SoC**	**Speed**	**RAM**	**USB Ports**	**Ethernet**	**Wireless**	**Bluetooth**
Raspberry Pi Model A+	BCM2835	700MHz	512MB	1	No	No	No
Raspberry Pi Model B+	BCM2835	700MHz	512MB	4	100Base-T	No	No
Raspberry Pi 2 Model B	BCM2836/7	900MHz	1GB	4	100Base-T	No	No
Raspberry Pi 3 Model B	BCM2837A0/B0	1200MHz	1GB	4	100Base-T	802.11n	4.1
Raspberry Pi 3 Model A+	BCM2837B0	1400MHz	512MB	1	No	802.11ac/n	4.2
Raspberry Pi 3 Model B+	BCM2837B0	1400MHz	1GB	4	1000Base-T	802.11ac/n	4.2
Raspberry Pi 4 Model B	BCM2711	1500MHz	2GB	2xUSB2, 2xUSB3	1000Base-T	802.11ac/n	5.0
Raspberry Pi 4 Model B	BCM2711	1500MHz	4GB	2xUSB2, 2xUSB3	1000Base-T	802.11ac/n	5.0

Raspberry Pi 4 Model B	BCM2711	1500MHz	8GB	2xUSB2, 2xUSB3	1000Base-T	802.11 ac/n	5.0
Raspberry Pi Zero	BCM2835	1000MHz	512MB	1	No	No	No
Raspberry Pi Zero W	BCM2835	1000MHz	512MB	1	No	802.11 n	4.1
Raspberry Pi Zero WH	BCM2835	1000MHz	512MB	1	No	802.11 n	4.1
Raspberry Pi 400	BCM2711	1800MHz	4GB	1xUSB2, 2xUSB3	1000Base-T	802.11 ac/n	5.0

Table 2.1 - Raspberry Pi versions [2].

2) Arduino

“Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing” [4].

3) NodeMcu

“NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits” [5].

2.2 : Software Components

A) Localization Techniques

The ability of the chair to know its current location within the stored map.

1) By vision (كاميرات المراقبة)

This method is a camera that is installed with the robot to help it know its location through the pictures it took during its movement, and the robot compares the images it takes with the images stored in the database so that it can know its current location.

2) RFID

In this method, stickers are placed on the ground so that when the robot walks over the sticker and reads it, it helps him to know where it is.

3) QR code

In this method, QR Code stickers are attached to the floor and directed to the ceiling so that the robot can be provided with location information. This method has the advantage of a shorter calculation time so that the current position of the robot can be estimated in less time.

4) Wheel encoder

This method is one of the most used methods for knowing the location of the robot and its direction, so that encoder sensors are installed with the motors, these sensors help to know the distance and speed at which the robot is moving and then know the current location on the stored map.

B) Path Planing Techniques

Searching for the goal location, and with the help of the stored map, the chair can create the optimal path between the current chair location and the goal while avoiding obstacles

1) Pure Pursuit

   It is a tracking algorithm that determines the path from the current location of the robot to a specific goal point without avoiding obstacles.

2) A* Planning Algorithm

   It is a tracking algorithm that determines the path from the current location of the robot to a specific goal point while avoiding obstacles and giving greater priority to the goals that are closer with lower costs.

3) D* Planning Algorithm

   The A * algorithm assumes that the entire environment is known, but there may be moving obstacles. To include this, D * has been proposed, as it aims to plan the effective course of the unknown and dynamic environments.

2.3 Alternatives and functions

A) Hardware Alternatives

1 : Sensors

*** Compass**

In our project, we don't need to use this sensor, we can calculate the angular motion through encoder sensors.

*** Encoders**

In our project, we will use this sensor that realizes the straight and angular motion of the robot.

*** Accelerometers**

In our project, it does not need to be used because, regardless of speed and acceleration, the robot must know the distance it has traveled.

*** Vision Systems**

This form of sensor is usually applied to robots that operate in factories, In addition to being costly, so we did not add it to our project.

*** Range Sensors**

A. Ultrasonic

In our project, most likely we will not use this type of sensor because the detection range in it is limited in addition to its low accuracy and not suitable for detecting fast-moving targets.

B. IR

In our project, we will not use this type of sensor because its detection range is limited in addition to it being affected by environmental conditions and solid objects.

C. LIDAR

Although this type of sensor has a higher cost compared to ultrasonic and infrared sensors, it is suitable for detecting fast-moving objects and is good at detecting small objects, in addition to being able to detect 3D structures, so there is a high probability of our use This type of sensors

*** Positioning System (e.g. GPS)**

The positioning system is used for outdoor environments, either in indoor environments because its accuracy is low, so we will not use it in the project.

2 : Electronic platforms

*** Raspberry Pi**

Raspberry Pi is the most powerful and versatile platform, contains the most resources, and is ideal for projects requiring a PC or a powerful Linux gateway. It's also the easiest platform to get started yet, and it's the piece that we'll be using in our project because we need to install the ros packages and algorithms and associate them with the chair,and the Raspberry Pi version we'll be using is Raspberry Pi 4 Model B with 4GB RAM .

* **Arduino**

We cannot complete the project using the Arduino because we need a complete computer system.

* **NodeMcu**

We cannot complete the project using the NodeMcu because we need a complete computer system.

B) Software Alternatives

1. Localization Techniques

* **By vision**

This method is best method ,but it is expensive to implement, so we did not use it in the project

* **RFID**

This method needs to stick a lot of signs in the environment in which the robot is moving, in addition to that, contacting it requires the robot to stand on it, meaning it is short-range, and therefore it is a method that is considered less intelligent if compared to other methods.

* **QR code**

We do not intend to use this method because ambient lighting or protection obstacles can affect the localization results. In addition, this method requires put of labels, so it is considered less intelligent and ineffective.

* **Wheel encoder**

This method is simple and inexpensive, which is the method that we will use in the project, And the algorithm we intend to use :

SLAM (Simultaneous Localization and Mapping):
is a tool that allows a robot to build and locate it on that map at the same time, used by autonomous robots. SLAM algorithms allow the robot to map unidentified environments. The robot uses the map data to perform tasks such as preparing the route and avoiding obstacle [6].

2. **Path Planning Techniques**

We will be using the D * algorithm because the chair must be able to update to the map when there are moving objects.

ROS :

“Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms”[12].

RVIZ :

“rviz is a 3d visualization tool for ROS applications. It provides a view of your robot model, capture sensor information from robot sensors, and replay captured data. It can display data from camera, lasers, from 3D and 2D devices including pictures and point clouds”[14].

Chapter 3: Design

Overview

In this chapter, we will cover a detailed explanation of how to design our project, which is entitled the autonomous wheelchair, so that the explanation will be about the hardware and software components ,and this chapter contains 2 sections .

3.1 : detailed design for hardware components

3.1.1 Electrical Wheelchair

"A motorized wheelchair, powerchair, electric wheelchair or electric-powered wheelchair (EPW) is a wheelchair that is propelled by means of an electric motor rather than manual power. Motorized wheelchairs are useful for those unable to propel a manual wheelchair or who may need to use a wheelchair for distances or over terrain which would be fatiguing in a manual wheelchair. They may also be used not just by people with 'traditional' mobility impairments, but also by people with cardiovascular and fatigue-based conditions,But it cannot be driven by people with quadriplegia or people suffering from blindness" [7].

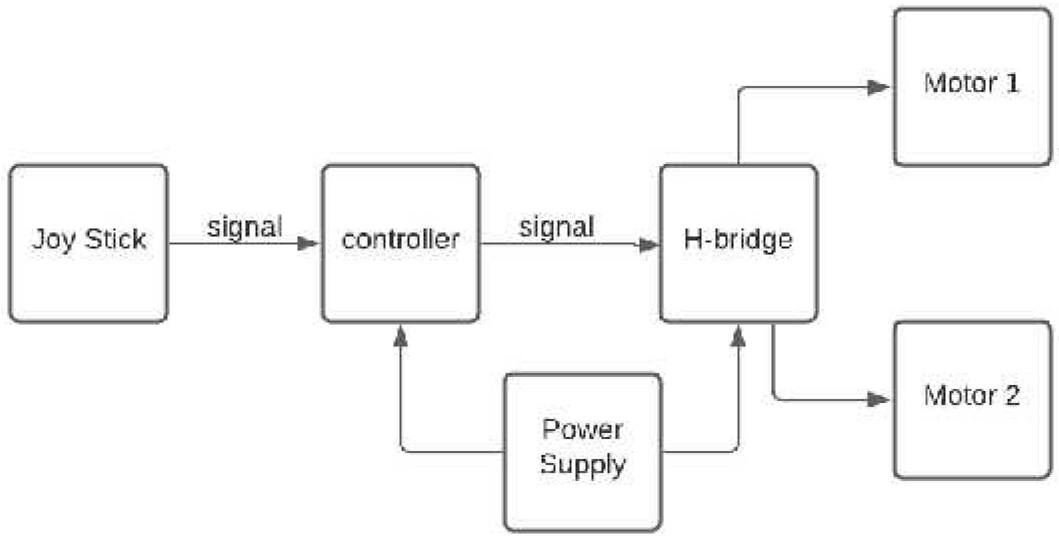

Figure 3.1 – Electrical wheelchair context diagram .

When moving the joy stick to control the wheelchair, signals are sent with values limited to the range 0-1023, so that the controller in the Arduino or the bic switches the signal to the range 0-255 and controls the H-bridge through the existing code to determine the speed and direction that the motors must implement.

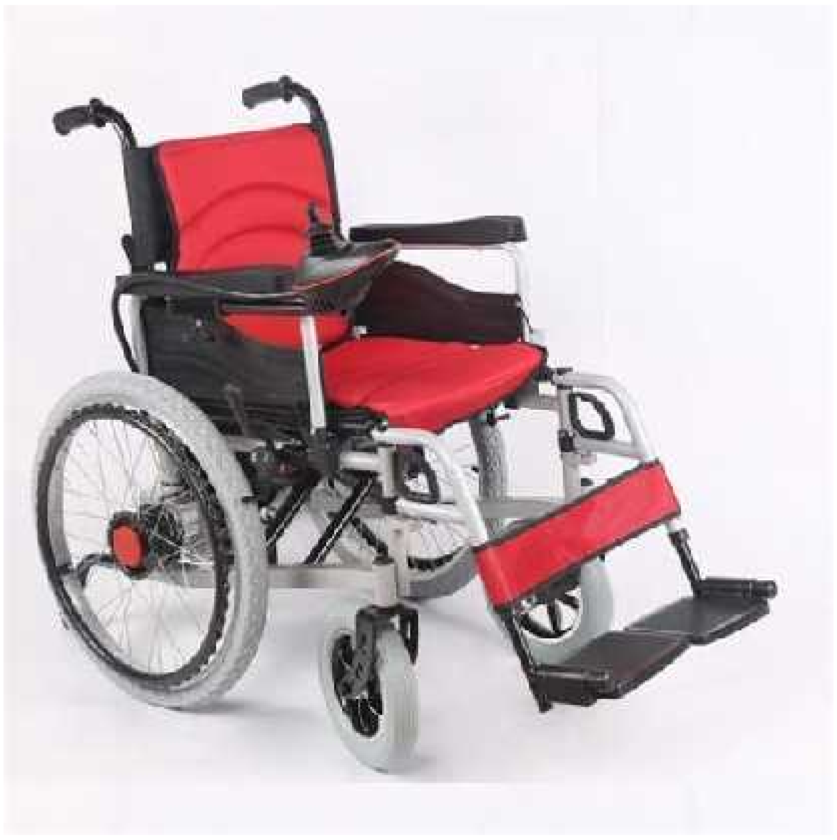
Figure 3.2 - electrical wheelchair image

3.1.2 Autonomous Wheelchair

"An autonomous wheelchair or smart wheelchair is "a uniquely modified powered wheelchair which is equipped with a control system and sensors. Smart wheelchairs are designed to provide assistance to users who are suffering with severe mobility impairment, visual impairments, etc. The primary purpose of the autonomous wheelchair is to reduce or eliminate the user's full responsibility on moving the wheelchair" [8].

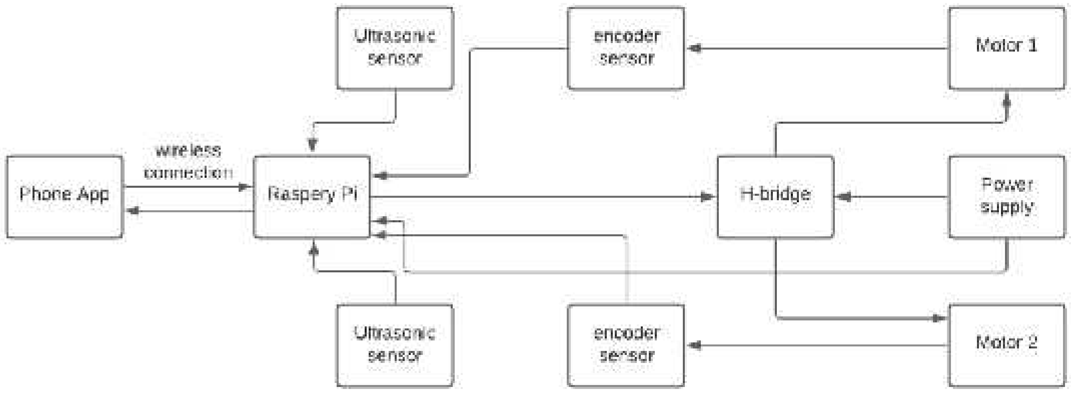

Figure 3.3 - Autonomous wheelchair context diagram .

In the beginning, the map is drawn by moving the wheelchair in the internal intended environment, with the help of an ultrasonic sensors, and sending the data through raspberry pi to the mobile application, And encoder sensors helps the wheelchair to know it location in the map .
The ultrasonic sensors helps the wheelchair avoid hitting any obstacle it faces while on the move .
raspberry pi receives commands from the mobile application and based on these commands the destination that the wheelchair must reach is determined.

3.1.3 Electrical Wheelchair Vs. Autonomous Wheelchair

Mode of Operation	Wheelchair Functionality	User Input Required
Electrical Wheelchair	Motorised wheelchair with filtered joystick input.	Joystick control.
Autonomous Wheelchair	Drive automatically to the destination selected.	Destination selection on phone App.

Table 3.1 - Operation Modes

Autonomous wheelchair = electrical wheelchair + ultrasonic sensors + encoder sensors + raspberry pi + phone app and wireless connection .

3.1.4 Phone App

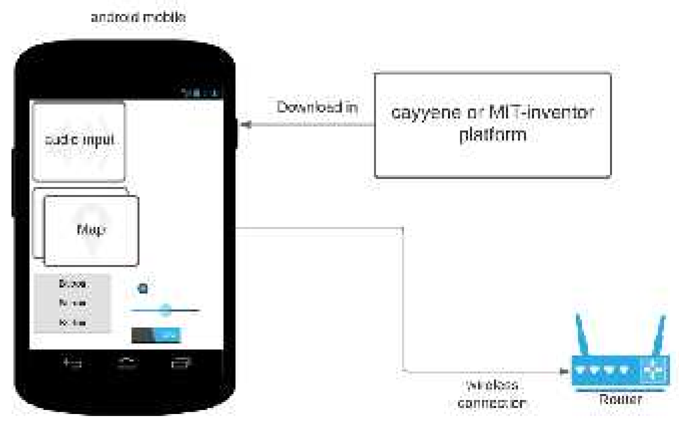

Figure 3.4 – Phone app implementation

After installing the cayenne or MIT-inventor platform on the Android phone, the app is created in several channels:
1) The possibility of entering voice commands, such as the names of the sites.
2) Show the map made with Raspberry PI,and the possibility of adding other maps on demand.
3) The possibility of controlling the chair with touch commands .
And the phone is connected to the Internet through the existing router to visualize the interaction with the raspberry pi Located with the wheelchair.

3.1.5 Raspberry Pi

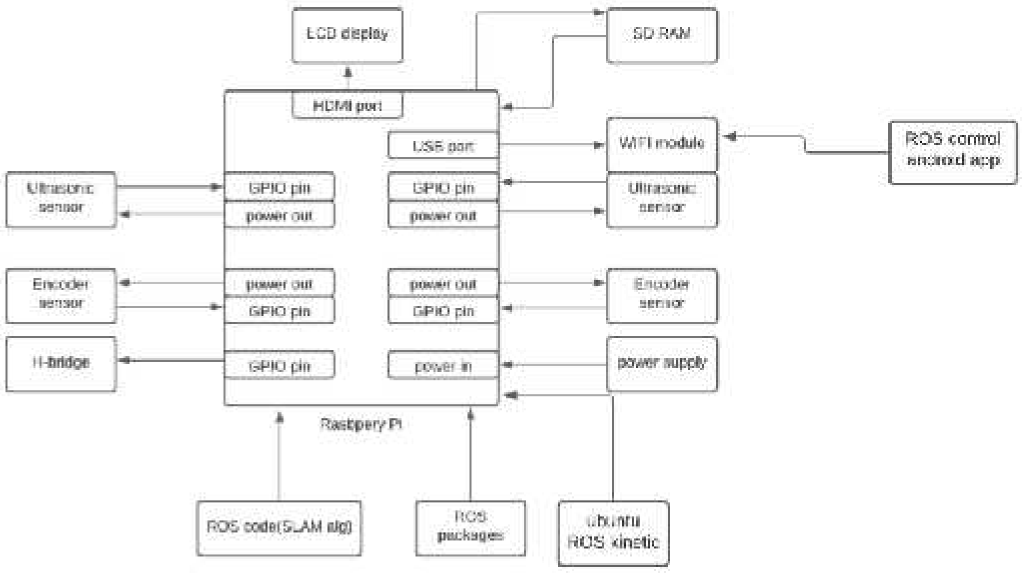

Figure 3.5 - Raspberry Pi context diagram .

Initially the raspberry pi OS , ROS packages and SLAM alg. are loaded onto the SD RAM located on the raspberry pi peace ,and by using the Wi-Fi segment, the widget connects to the network to interact with the mobile application, during this interaction, voice and touch commands are sent from the phone to the raspberry pi .
Through the ultrasonic sensors and according to the uploaded algorithm on the piece, the map is drawn in the environment in which the wheelchair moves, and thus the map is sent from the piece to the phone so that the user can interact with it,and through the ultrasonic sensors also, and according to the uploaded algorithm, the wheelchair can avoid collision with any obstacle it faces.
Through the encoder sensors, and according to the uploaded algorithm, the raspberry pi piece can determine the location of the wheelchair in the map, and the user can know where he is in the map and determine the destination he wants to go to either by voice or touch commands through the phone application.

3.1.6 H-bridge

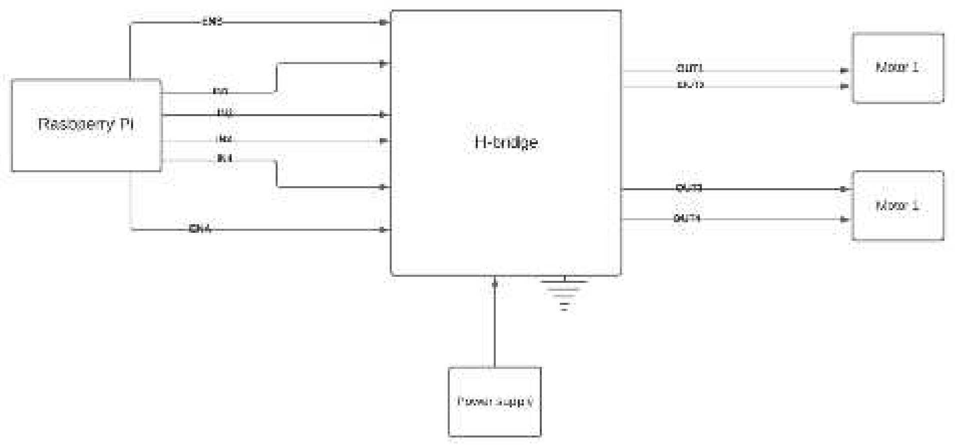

Figure 3.6 – H bridge context diagram .

There are four digital signals with values 0 and 1, namely IN1, IN2, IN3, and IN4 which the H-bridge uses to control the direction of the motors .
In addition to two analog signals , namely ENA and ENB used by the H-bridge to control the speed of the motors .

3.2 detailed design for software components

3.2.1 Mapping

A method for creating a map of the internal world in which the robot would move.after drawing it, it is saved to assist the robot in determining its position within the area in which it is moving, in addition to able to locate the target that the user has specified.

3.2.2 Localization

A method of assisting the robot in locating itself inside the stored map.

3.2.3 Path Planning

A mechanism in which the robot is assisted in locating the target within the stored map, as well as drawing the shortest path between the robot's current position and the target location selected by the user while avoiding obstacles.

3.3 Adjustments

When we started working on the project, we ran into some issues that forced us to make some changes:

1. **Electrical Wheelchair**

   The electric wheelchair on which we planned to work the project was swapped out for a Kobuki robot (Turtlebot 2) because the electric wheelchair was incompatible with the robotics operating system (ROS).

Figure 3.7 – from electrical wheelchair to turtlebot.

2. **Raspberry pi**

   The Raspberry Pi, which we had intended to use in the project, was replaced by a laptop because the laptop is easier to handle and the Raspberry Pi needs a screen and a cooler.

Figure 3.8 – from raspberry pi to Dell laptop.

3. Phone App

To communicate between the phone application and the ROS on the laptop, we used bluetooth instead of Wi-Fi, which necessitated the provision of the following parts:

- Two pieces of HC-05.
- Bread Board.
- Arduino Uno.
- Wires.

So the first HC-05 segment brings data from the phone app to the Arduino segment, and the second HC-05 segment sends this data directly to ROS on the laptop.

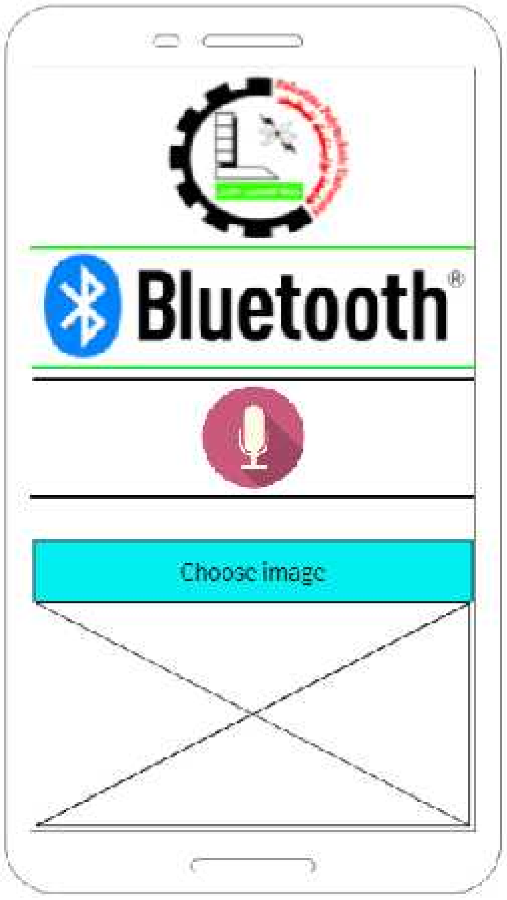

Figure 3.9 – my_app.

Chapter 4: Software

The system software is represented in various sections, including the program that controls the ability of mobile application to communicate with ROS. Furthermore, there is a program that acts as an intermediary between the phone application and ROS, as well as the algorithms that are run by ROS and are responsible for directly controlling the robot. The software is thus divided into three main sections:

1. Phone Application(my_app)

The main function of this program is to send specific numbers based on the words uttered via Bluetooth which the user must contact in advance, in addition to the ability to display two images in the application.

my_app pseudocode:

```
 Begin

when ListPicker1.BeforePicking

do {

      set ListPicker1.Elements to BluetoothClient1.AdressesAndNames

   }

End



Begin

when ListPicker1.AfterPicking

do {

      if call BluetoothClient1.Connect to address ListPicker1.Selection

      then set ListPicker1.Elements to BluetoothClient1.AdressesAndNames
   }
End

Begin
when clock1.Timer
do { if BluetoothClient1.IsConnected
       then set Label1.Text to Connected
       else set Label1.Text to Not Connected}
End

Begin
when Button1.Click
do {call SpeechRecognizer1.GetText}
End

Begin
when SpeechRecognizer1.BeforeGettingText
do {set Label2.Text to ""}
End

Begin
when SpeechRecognizer1.AfterGettingText
do { set Label2.Text to SpeechRecognizer1.Result
       if contains text "حمل"
       then call BluetoothClient1.SendByteNumber 1
```

```
        call SpeechRecognizer1.GetText

        else if contains "مطبخ"

        then call BluetoothClient1.SendByteNumber 2

        call SpeechRecognizer1.GetText

        else if contains "صالون"

        then call BluetoothClient1.SendByteNumber 3

        call SpeechRecognizer1.GetText

        else call SpeechRecognizer1.GetText

End
```

2. Mediator

This program receives data from the phone application via the first HC-05 segment, which has been connected to the phone via Bluetooth and sends the data directly to the Listener python code on the laptop via the second HC-05 segment, which's connected to the laptop via Bluetooth. The data reception and transmission processes are carried out with the aid of the Arduino piece.

Mediator pseudocode:

```
// Call up a library SoftwareSerial

Import SoftwareSerial

// Defining EEBlue as software serial on ports 2 and 3

SoftwareSerial EEBlue(2,3)


void setup() {

  Serial.begin(9600);

  EEBlue.begin(9600); // Default Baud rate
}


void loop() {
  if (EEBlue.available()){
       int c=int(EEBlue.read());
       Serial.write(c);
  }
}
```

3. ROS algorithms

The following are the algorithms that are responsible for fully directing the robot:

· Gmapping

The map is drawn by the movement of the robot in a specific location, with distance sensors reading the distances between the robot and nearby obstacles, and during the drawing process, the black color of the obstacles and the gray color of the empty areas are included.

A. Inputs:

· Distance sensors readings.

· Encoders sensors readings.

B. Outputs:

· The map

C. Source:

http://wiki.ros.org/Robots/TurtleBot

D. How to use Gmapping:

· Bring up a TurtleBot2 simulation (gazebo)

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

- make a map from a robot with laser publishing scans on the scan topic:

```
$ roslaunch turtlebot_gazebo gmapping_demo.launch
```

- Launch the rviz program

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

- To control the turtlebot by keyboard

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

· Localization and Path Planning

A process that helps the robot know its location within the map, in addition to the robot's ability to draw the path between its location and the location of the target that has been identified by the user, and reached it.

A. Inputs:

- Distance sensors readings.
- Encoders sensors readings.

B. Outputs:

- Find out the robot for its location
- Draw the path
- Reach the goal

C. Source:

http://wiki.ros.org/Robots/TurtleBot

D. How to use Localization and Path Planning

- Bring up a TurtleBot2 simulation (gazebo)

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

Localize using laser data on the scan topic:

```
$ roslaunch turtlebot_gazebo amcl_demo.launch
```

Launch the rviz program

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

To control the turtlebot by keyboard

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

· Voice control

This procedure is carried out in three stages:

1. **Talker**

At this stage, the link between the listener on laptop and the HC-05 Bluetooth segment is established.

Talker Pseudocode:

```
#!/usr/bin/env python

import rospy // ROS Python library

from std_msgs.msg import String

import bluetooth // Import the python-bluez library

// Bluetooth parameters

robot_bluetooth_mac_address ='00:13:EF:00:D9:2D'

port = 1

pc_bluetooth_handle = None

data_size = 3000

ultrasonic_handle = rospy.Publisher(

      'obstacle_distance', String, queue_size=100)
```

```
// Launch the ROS node

rospy.init_node('talker', anonymous=True)

rospy.loginfo("Starting Talker Node")

// Connect the PC's Bluetooth to the robot's Bluetooth

define connect():

  global pc_bluetooth_handle

  while(True):

       try:

       pc_bluetooth_handle = bluetooth.BluetoothSocket(

                                  bluetooth.RFCOMM)

       pc_bluetooth_handle.connect((

                     robot_bluetooth_mac_address, port))

       break;

       except bluetooth.btcommon.BluetoothError as error:

       pc_bluetooth_handle.close()

       rospy.logwarn(

       "Could not connect: ", error, "; Retrying in 10s...")

       rospy.sleep(10)

  return pc_bluetooth_handle

pc_bluetooth_handle = connect() // Connect to robot's Bluetooth

// Main code

// If this file is the main (driver) program you are executing

if __name__ == '__main__':

  while not rospy.is_shutdown():

       try:

       // Keep reading data from the robot
```

```
      incoming_data_from_robot = pc_bluetooth_handle.recv(
                          data_size)
      rospy.loginfo(incoming_data_from_robot)
      ultrasonic_handle.publish(incoming_data_from_robot)
      rospy.sleep(0.05)
 except bluetooth.btcommon.BluetoothError as error:
      rospy.logerr("Caught BluetoothError: ", error)
      time.sleep(5)
      pc_bluetooth_handle = connect()
      pass
  pc_bluetooth_handle.close()
```

2. Listener

At this stage, the data sent from the HC-05 segment is received and sent to Reach code.

Listener Pseudocode:

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
from geometry_msgs.msg import Pose, Point, Quaternion


def listener():
       def callback(data):
       number = data.data


       // Print the data that is heard from the ROS topic
```

```
        rospy.loginfo(
                rospy.get_caller_id() + " I heard %s", data.data)


        if number == '1':
                pos = {'x': -2.58, 'y': -2.41}
                quat = {'r1': 0.000, 'r2': 0.000, 'r3': 0.000, 'r4': 1.000}
        elif number == '2':
                pos = {'x': 2.82, 'y': -2.15}
                quat = {'r1': 0.000, 'r2': 0.000, 'r3': 0.000, 'r4': 1.000}




        goal_pose = Pose(Point(pos['x'], pos['y'], 0.000),
                                Quaternion(quat['r1'], quat['r2'], quat['r3'],
quat['r4']))


        pub = rospy.Publisher('desired_goal_pose', Pose, queue_size=10)
        rate = rospy.Rate(10)
        pub.publish(goal_pose)
        rate.sleep()


        // Publish to desired_goal_pose topic


        // Initialize the node
        rospy.init_node('listener', anonymous=True)


        // Subscribe to the obstacle_distance topic
        rospy.Subscriber("obstacle_distance", String, callback)


        // keeps python from exiting until this node is stopped

        rospy.spin()


if __name__ == '__main__':

        try:

        listener()

        except rospy.ROSInterruptException:

        pass
```

3. Reach

At this stage, the data that is sent from the listener is received, and based on this data the robot is moved to reach specific coordinates in the map.

Reach Pseudocode:

```
#!/usr/bin/env python

import rospy

from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

import actionlib

from actionlib_msgs.msg import all

from geometry_msgs.msg import Pose, Point, Quaternion

class GoToPose():

        def __init__(self):

        self.goal_sent = False

        //What to do if shut down (e.g. Ctrl-C or failure)
        rospy.on_shutdown(self.shutdown)

        // Tell the action client that we want to spin a thread by default

        self.move_base = actionlib.SimpleActionClient("move_base",
MoveBaseAction)

        rospy.loginfo("Wait for the action server to come up")

        // Allow up to 5 seconds for the action server to come up

        self.move_base.wait_for_server(rospy.Duration(5))

        def goto(self, pose):

        // Send a goal

        self.goal_sent = True

        goal = MoveBaseGoal()

        goal.target_pose.header.frame_id = 'map'

        goal.target_pose.header.stamp = rospy.Time.now()

        goal.target_pose.pose = pose

        // Start moving

        self.move_base.send_goal(goal)

        // Allow TurtleBot up to 60 seconds to complete task

        success = self.move_base.wait_for_result(rospy.Duration(60))

        state = self.move_base.get_state()

        result = False

        if success and state == GoalStatus.SUCCEEDED:

                // We made it!

                result = True

        else: self.move_base.cancel_goal()

        self.goal_sent = False

        return result

        def shutdown(self):
```

```
        if self.goal_sent:

                self.move_base.cancel_goal()

                rospy.loginfo("Stop")

                rospy.sleep(1)

if __name__ == '__main__':

        try:

        def callback(data):

                desired_pose = data

                navigator = GoToPose()

                success = navigator.goto(desired_pose)

                if success:

                rospy.loginfo("Hooray, reached the desired pose")

                else:  rospy.loginfo("The base failed to reach the desired
pose")

        rospy.init_node('nav_test', anonymous=False)

        // Subscribe to the desired_goal_pose topic

        rospy.Subscriber("desired_goal_pose", Pose, callback)

        // keeps python from exiting until this node is stopped

        rospy.spin()

        except rospy.ROSInterruptException:

        rospy.loginfo("Ctrl-C caught. Quittin
```

Chapter 5:Validation

Problems

The problem that we faced in the project at the beginning was the idea of developing an electric chair into a chair that can be controlled by sound or touch, we prepared all the parts necessary for the project, then we found out that it takes a long time because the chair is incompatible with the operating system of robots, then we tried to implement the same idea but by using a TurtleBot robot, we also discovered that the robot's camera had problems, then we applied the idea to the Turtlebot robot through simulation and they were able to meet all the requirements.

System testing

The system has been validated, which means that all components of the system have been inspected successful, and we will show the details as follows:

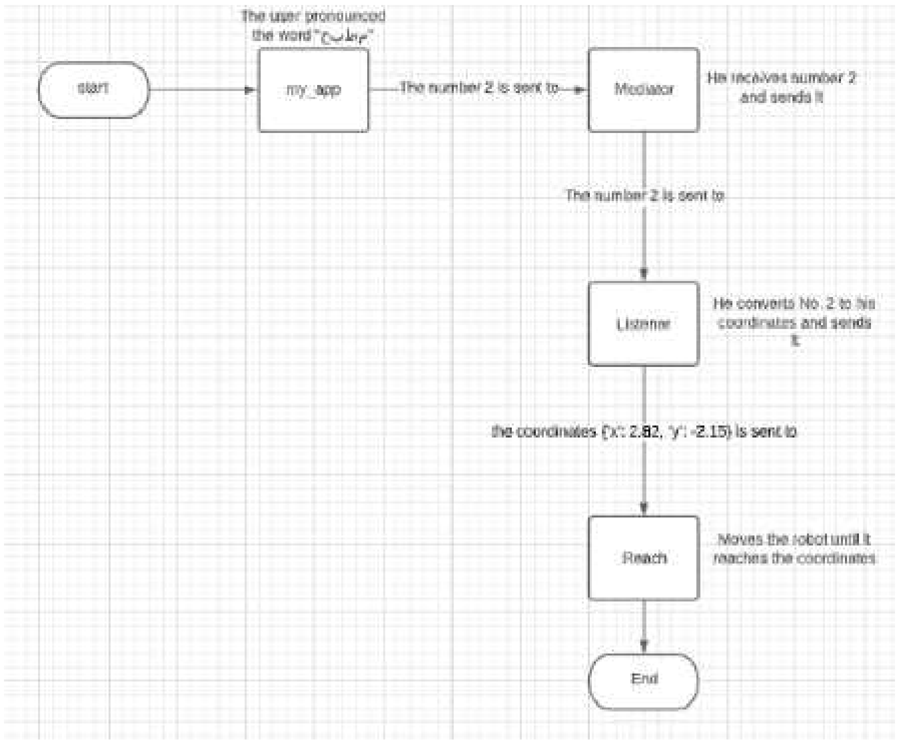
The user pronounced the word "مطبخ"
start
my_app
The number 2 is sent to
Mediator
He receives number 2 and sends it
The number 2 is sent to
Listener
He converts No. 2 to his coordinates and sends it
the coordinates {'x': 2.82, 'y': -2.15} is sent to
Reach
Moves the robot until it reaches the coordinates
End

Figure 3.6 – system testing chart.

1. At first, the user pronounces a specific word such as (حمام مطبخ، صالون) and if the user pronounces the word مطبخ, number 2 is sent to the Mediator via the Bluetooth piece HC-05.

2. When the number 2 reaches the Mediator, it is sent directly via the second Bluetooth piece HC-05 to the Listener code on the laptop.

3. Finally, when the number 2 arrives at Listener, the Listener converts the number 2 into its coordinates that have been determined by the programmer and then sends them to Reach code so that the function of this code is to give a command to the robot Turtlebot to go to the coordinates that have been selected.

Conclusion

We have successfully simulated operating the robot TurtleBot such that it can drive by the method that the user wants, either through the RVIZ computer program or using a phone application that is connected to the computer by Bluetooth and support voice commands.

If we wish to use the RVIZ computer software to drive the robot, we will choose a position on the stored map that appears in front of us on the RVIZ program, and after determining the location, the robot creates a path between its current location and the target site and travels there, but if we want to control the robot through voice commands, there are several words such as (حمام، مطبخ، صالون) if we said it then the application sends a specific number such as (1, 2, 3), and when the number reaches the computer, the target is determined automatically based on the number that was sent, so that each number has specific coordinates.

The computer program is a system created by combining existing algorithms responsible for the localization, obstacle avoidance, and navigation. Using data detected by robot sensors, these algorithms can successfully map a route to a known destination.

This project was going to be done on a real wheelchair, but we couldn't apply the algorithms to it because the wheelchair was incompatible with ROS, so we chose to work on the real Turtlebot in university, but due to some hardware problems, we were also unable to apply the necessary algorithms on it. Therefore, we were satisfied with the implementation of the project idea on the robot Turtlebot via simulation.

Finally, the robot is able to accept and execute commands via the RVIZ computer program or the phone application. We have only accomplished the project idea on the Turtlebot robot through simulation. However, we can easily transfer this idea to a real Turtlebot and follow the same processes, but to implement this idea in a wheelchair, it requires creating algorithms specific to the chair that correspond to its weight, length, width, speed of motors, and other things.

References

[1] robot navigation , “https://en.wikipedia.org/wiki/Robot_navigation ", Nov , 2020 .
[2] Raspberry pi versions , “https://www.raspberrypi.org/documentation/faqs/ ", Jan, 2021 .
[3]Raspberry-Pi,“https://www.cnx-software.com/2020/03/24/know-the-differences-between-raspberry-pi-arduino-and-esp8266-esp32/" , Nov , 2020 .
[4] Arduino , “https://www.arduino.cc/en/Guide/Introduction" , Nov , 2020 .
[5] NodeMcu , “https://en.wikipedia.org/wiki/NodeMCU" , Nov , 2020 .
[6] SLAM, “https://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-7-10“ , Nov , 2020 .
[7] Electrical Wheelchair , “https://en.wikipedia.org/wiki/Motorized_wheelchair “ , Nov , 2020
[8] Autonomous wheelchair ,“https://www.researchgate.net/publication/3731481_Autonomous_wheelchair_for_disabled_people “   , Nov , 2020
[9] mapping ., “https://liu.diva-portal.org/smash/get/diva2:1218791/FULLTEXT01.pdf “ , Dec / 2020

[12] D* Planning Alg pseudocode , “file:///C:/Users/islam/Downloads/4515-Article%20Text-13497-1-10-20181213.pdf ", Jan, 2021 .
[13] ROS , “https://www.ros.org/about-ros/", Mar, 2021 .
[14]RVIZ , “https://docs.aws.amazon.com/robomaker/latest/dg/simulation-tools-rviz.html ", Mar, 2021 .
[15]Gmapping alg. flowchart , “https://www.researchgate.net/figure/Flowchart-of-the-ProcessScan-function-of-the-GMapping-algorithm-which-is-called-every_fig4_267336764 ” , May, 2021.
[15]Localization alg. flowchart , “https://www.hindawi.com/journals/complexity/2018/2327637/ ” , May, 2021.
[15]Path planning alg. flowchart , “https://www.researchgate.net/figure/Flowchart-of-path-planning-algorithm_fig3_270675888 ” , May, 2021.