ParkingEaze

Parking IOT system scale model

Navkiran Kaur

Saina Kapoor

Harleen Kaur Saini

Computer Engineering and Technology

14 April,2020

Status

/1 Hardware present?

/1 Title Page

/1 Declaration of Joint Authorship

/1 Proposal (500 words)

/1 Executive Summary

Declaration of Joint Authorship

We, Navkiran Kaur, Saina Kapoor and Harleen Kaur Saini confirm that this work submitted is the joint work of our group and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included. The work breakdown is as follows: Each of us provided functioning, documented hardware for a sensor or effector. Navkiran Kaur provided the bipolar stepper motor that will be able to move the barriers in the car parking in order to allow the entrance and exit of the vehicle using the rotational movement controlled by the driver and raspberry pi Broadcom. Saina Kapoor provided the PCF8574 LCD with HC-SR04 ultrasonic distance sensor in order to measure the distance of the car and display it on the LCD screen on the raspberry pi. Harleen Kaur Saini provided Ek1245 infrared sensor on the raspberry pi will detect the presence of any vehicle parked in the parking spot. In the integration effort Saina Kapoor is the lead for further development of our mobile application, Navkiran Kaur is the lead for the Hardware, and Harleen Kaur Saini is the lead for connecting the two via the Database.

Proposal

We have created a mobile application, worked with databases, completed a software engineering course, and prototyped a small embedded system with a custom PCB as well as an enclosure (3D printed/laser cut). Our Internet of Things (IoT) capstone project uses a distributed computing model of a smart phone application, a database accessible via the internet, an enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB as well as an enclosure (3D printed/laser cut), and are documented via this technical report targeting OACETT certification guidelines.

Intended project key component descriptions and part numbers
Development platform: Raspberry Pi 3B+
Sensor/Effector 1: PCF8574 LCD effector/ HC-SR04 ultrasonic sensor

PCF8574 LCD module is controlled by I2C bus to convert the parallel interface to a serial one. This needs only 2 wires SDA & SCL, apart from the power connections and ground. There is a blue model which adjusts the contrast of the LCD.

HC-SR04 ultrasonic sensor is a distance measuring transducer sensor which needs power supply of 5V DC and working current is 15mA. It has both transmitter and receiver modules and offers excellent non-contact range detection with high accuracy and stable readings

Part number: PCF8574 LCD module - IF-modulen_11747

Part number: HC-SR04 ultrasonic sensor- EL-SM-001

Sensor/Effector 2: EK1254 IR sensor

EK1254 infrared sensor is used in obstacle avoidance car, line count etc. It can be used for 3-5V DC power supply modules and detects the distance between 2 to 30cm and the output port OUT sensor module can be directly connected to the microcontroller IO port.

Part number: EK1254 infrared sensor- EK1254

Sensor/Effector 3: DRV8825 Bipolar Stepper motor

Bipolar stepper motor is the Nema 17 stepper motor that uses the full coil having two leads per phase that is four leads in total, having high torque, low noise, low self-inductance reactance. The motor is driven by **DRV8825** driver with 45 V maximum supply voltage which has 4-layer, 2 oz copper PCB for improved heat dissipation which can interface directly with 3.3 V and 5 V systems.

Part Number: Stepper motor- Usongshine motor1, ASIN: B07C1MVTZC

Part Number: DRV8825- 606321399948, ASIN: B07L2TKLPW

We will continue to develop skills to configure operating systems, networks, and embedded systems using these key components to make our project work successfully. Daily parking, navigation problems and time constraint were taken into consideration to give a prominent solution so by the end of this project a better parking facility will be delivered to the customers using our simple application. Sensors will collect the data and will show in the application about the count of spaces available. Within one click, user will get booked location in the app and will park the car at the destination without any delay. A major change in the navigation is that the directions will be available as soon as the driver enters the parking lot in order to direct to the correct parking spot. Our hardware and software components offer a better usability and testability that will also store user's information in the database along with time slots, credit card information, login credentials, license plate number, etc. This will prevent the traffic congestions as everything will be done already in the app.

Our project description/specifications will be reviewed by the Department of Public Safety, Humber College, ideally an employer in a position to potentially hire once we graduate. They will also ideally attend the ICT Capstone Expo to see the outcome and be eligible to apply for NSERC funded extension projects. This typically means that they

are from a Canadian company that has been revenue generating for a minimum of two years and have a minimum of two full time employees.

The small physical prototypes that we build are to be small and safe enough to be brought to class every week as well as be worked on at home. In alignment with the space below the tray in the Humber North Campus Electronics Parts kit the overall project maximum dimensions are 12 13/16" x 6" x 2 7/8" = 32.5cm x 15.25cm x 7.25cm.

Keeping safety and Z462 in mind, the highest AC voltage that will be used is 16Vrms from a wall adapter from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will not exceed 20 Watts. We are working with prototypes and that prototypes are not to be left powered unattended despite the connectivity that we develop.

Executive Summary

Parking is a worldwide problem, especially when people are unaware about spaces, costs and policies that results in waste of time, crowding and frustration. Our project aims to solve these problems by using such sensors and effectors which in coordination with our mobile application makes it easier for our users to park and pay in a fast and efficient manner. With the help of our technical skills and professor's guidance we used the features and key points of different sensors and effectors such as infrared sensor, ultrasonic sensor, Bipolar stepper motor, LCD module and customizable development of mobile application in order to deliver a trustworthy finished product under a low-cost budget.

Contents

List of Figures

1.0 Introduction

Traffic flow, distribution, and accessibility of parking space, in these days, have been a major concern to every person in daily routine which makes it an important issue. There is a lot of consummation of time and resources for drivers who spend their precious time driving along the streets in search of parking space and then also going through the long process to pay the parking fees. To solve this necessity of vehicle parking, we introduce a parking project called ParkingEaze. The ParkingEaze Project enables customers/drivers to online reserve a parking space using the mobile application. The purpose of this project is to provide the user the convenience to park their vehicle as per their requirement using the app. It allows the user to view the parking status and its location in the android device application. This project is developed to tackle the traffic congestion, collision, and delays faced by the people in their daily life which results in a lot of headaches, frustration, and wastage of time. The project includes the software and hardware integration that will determine the execution of our project. The spots for booking that we include and consider for reservation belongs to the Humber college. The hardware part of the project would be having raspberry pi as a development platform. Along with that infrared sensor, ultrasonic sensor, bipolar stepper motor, LCD module contribute as a major portion for project development. The bipolar stepper motor will allow the entrance and exit of the vehicle while the LCD will show the booked parking spots. The infrared sensor will detect the presence of the vehicle in the parking area also determines the obstacles. Coming to the software, an android app called ParkingEaze is designed on the android studio which users can log in to reserve the vacant slots online. This android application called ParkingEaze is created for the user to reserve an empty parking spot from the locations mentioned above. The mobile application is constructed through java language which can run on android devices using the android studio as a platform for its development. The app provides rich user experience and eases to utilize the time. Also, the issues of losing the paper ticket, waiting in lines to park are solved while using this system. After booking the parking spot in the app, the user will select the date and duration of the time during which the

vehicle needs to parked and then an online payment method is performed. Afterward, the user will be allotted a parking spot where the vehicle can be parked. The project is scheduled to be accomplished in a certain time in which the mobile application and the individual hardware projects are developed in the first four months. The hardware and software integrations are performed in the last four months. Therefore, the project aims at solving a real-world problem by designing an android application system that will enable the customers to make a reservation of available parking space. The project mainly focusing on assisting the users and helping them to balance their working life easier by providing them a solution to the problem that has been faced by them daily in their lifestyle.

1.1 Scope and Requirements

In this project basically, we are going to combine our hardware and software courses that we have done in the previous semester. It is an Internet of Things (IoT) capstone project that uses a distributed computing model of a smartphone application, a database accessible via the internet, enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB as well as an enclosure (3D printed/laser cut), and is documented via an OACETT certification acceptable technical report. Also, there are some limits of the project like through our the app we will be able to book the parking spots of only the Humber college parking and what will not be done (CSA testing) in this project: we are not going to add any real-time card transactions, we were thinking to add license plate detector but that will not fit in our budget, so we are thinking to drop the idea.

Development platform specifications:

- We are using Raspberry pi 3(Rpi3) Model B+ having 64 bit quad-core 1.4 GHz,1 GB RAM.

Hardware specifications:

We are using different sensors in our project:

- Infrared sensor – to detect the presence of any vehicle parked in the parking spot.
- Ultrasonic distance sensor - to measure the distance of the car.
- LCD – to display it on the LCD screen on the raspberry pi.
- Bipolar stepper motor - to move the barriers to car parking.

the Android device requirements:

- Our app will be able to run on the android devices whose minimum API will be 23 i.e. marshmallow.

database specifications/protocols:

- We are going to use firebase, where we can store, extract and add the data.

Report

/1 Hardware present?

/1 Introduction (500 words)

/1 Scope and Requirements

/1 Background (500 words)

/1 References

2.0 Background

We would like to thank the Department of Public Safety and James Irvine, manager at Transportation and parking services, Humber College for supporting this project. We chose this project to solve the day to day problems related with parking, so for the smart solutions we researched about various sensors and effectors which can be taken into consideration to deliver a finished and efficient project called Smart Parking. We chose different sensors and effectors such as EK1254 IR sensor, an infrared sensor which sense the cars at the parking spot and shows if the parking spot is available or occupied (Ms.S.Srikurinji, 2016), the movement of Bipolar stepper motor,Nema 17, allows the vehicle to enter and exit the parking lot (The journal of design and technology, 2000).The bipolar stepper motor is driven by DRV8825 driver that will be capable of driving the motor .Lastly the HC-SR04 ultrasonic sensor that will the sense the object combined with the LCD module displays the distance of the cars to the parking spaces and the user is able to see what is displayed (GauravKumarMaurya, 2016). The idea of the application of smart parking is to provide ease to the customers within one click, user will get booked location in the app and will park the car at the destination without any delay. Our hardware components will collect the data and will be displayed in our application with certain features like the updating the parking lots and the location will take only few seconds (Kahonge, 2013).The development platform is the raspberry pi Broadcom that will be used to develop the parking project interacting with the sensors and effectors which are stated in the project. The project is developed on the basis of the requirements to fulfill the project needs for resolving the problem faced in the real world. The ParkingEaze application asks the user to register first and then login and all the data is added to the database. The information of the customer such as contacts, name, password etc. are the part of the database as it will be stored them as we are using the firebase for the project. Our project development will help us gain necessary skills in order to gain more knowledge and experience through exploration of ideas used in the hardware and software. The project explores the wide picture of the innovative and creative ability of integrating the components which will led to a successful accomplishment that will benefit the society.

3.0 Methodology

The project is called parkingEaze and here the required resources are mentioned below

3.1 Required Resources

The components that comprise the hardware project are bipolar stepper motor driven by drv8825 ,EK1254 infrared sensor and the lcd module with ultrasonic sensor

Report

/1 Parts/components/materials (500 words)

/1 PCB, case (500 words)

/1 Tools, facilities (500 words)

/1 Shipping, duty, taxes (250 words)

/1 Working time versus lead time (250 words)

3.1.1 Parts, Components, Materials

We are using Raspberry pi 3(Rpi3) Model B+ having 64 bit quad-core 1.4 GHz,1 GB RAM as our Broadcom.

Sensors or effector that we are using are:

Sensor/Effector 1: Gikfun IR Infrared Sensor EK1254

The output port OUT sensor module can be directly connected to the microcontroller IO port, you can directly drive a 5V relay; The module detects the distance 2 -30cm, detection angle 35 °, the distance can detect potential is adjusted Can be used for 3-5V DC power supply modules. When the power is turned on, the red power indicator light. This sensor will sense if the parking spot is available or not. This sensor will send the information to the LCD to display which parking lot is available. This is our one of the effectors that we will discuss in upcoming paragraphs

Sensor/Effector 2: Bipolar stepper motor driven by DRV8825 driver on the raspberry pi Broadcom

Bipolar stepper motor: It is the Nema 17 stepper motor that uses the full coil having two leads per phase that is four leads in total, having high torque, low noise, low self-inductance reactance.

The DRV8825 stepper motor driver has output drive capacity of up to 45V and lets you control one bipolar stepper motor at up to 2.2A output current per coil. This driver carrier is a breakout board for DRV8825 micro stepping bipolar stepper motor driver.

This sensor will be used at the entrance and exit of the parking area to raise up or down the boom barrier.

Sensor/Effector 3: pcf8574 LCD module: LCD module yellow-green screen PCF8574 IIC/I2C 1602 LCD. the dedicated IIC bus control, it only takes two IO. Adjustable contrast and backlight controllable Yellow and green screen.

This sensor will display the information of parking spots as if parking area is full or which parking spot is empty.

HC-SR04 ultrasonic sensor: Ultrasonic hc-sr04 distance measuring transducer sensor. HC-SR04 consists of ultrasonic transmitter, receiver, and control circuit. When trigged it sends out a series of 40KHz ultrasonic pulses and receives echo from an object having 5V DC, current less than 2Ma.

This sensor will sense the vehicle if it is coming nearby and will be connected with the bipolar stepper motor that will raise up the boom barrier when the vehicle is at certain distance.

Headers: Different headers are used that will be connected on the PCB so that our sensors can be easily connected or removed from the PCB board.

Resistors: We have used the resistors so that our sensors or effector does not get damage by high voltage. They are directly connected on the PCB through soldering.

Jumper Wires: They are used to connect the different parts of the bipolar stepper motor our stepper which can't be directly connected with board.

Acrylic sheets: We need acrylic sheet for the enclosure. Although we'll get sheets from the next door that is the prototype lab, but still this will be one of the requirements.

3.1.2 Manufacturing

3.1.2.1 PCB

To display a working module, we first designed the breadboard design so that it can be implemented on the printed circuit board in order to deliver finish working project. We used Fritzing software to draft the pcb which had multiple options to check if the connections were correct and exported with the help of Gerber files and it was made in the prototype lab. Starting with Saina's project it consists of 1 HC-SR04 sensor which is connected with the LCD module and two resistors to display the distance of the car, to design the pcb it took multiple attempts to plan the correct board. In the very first attempt pcb was in the opposite direction of the raspberry pi which again led to second attempt in which Via points were not in correct order so the sensor was responding late, third attempt was to basically position the LCD in the horizontal manner which led to the fourth and the final attempt for the pcb which had all the requirements such as small and all the correct connections. Navkiran's project includes bipolar stepper motor, capacitor, driver and power supply. It took hardly two attempts to design the pcb, in the first try the parts were disorganised and the pcb was huge which was taken forward to the second attempt, a mistake was made because of so many wires one wire was missed so it was soldered manually and the possibility of short was taken care of. Harleen's project consist of an EK1254 infrared sensor which had a very simple connection with raspberry pi, this also took 2 attempts in order to fix the via points and to make the pcb smaller. Hence, the three printed circuit boards were made in the hardware production technology and these hardware will be integrated in to single project in this semester.

3.1.2.2 Case

To design the enclosure case, we used Corel draw software which was again printed in the prototype lab using the laser cut technique. Saina's raspberry pi case was taken from one of the projects mentioned in the thingiverse website. All the sides except the top was made from white acrylic sheet and the top was made from clear acrylic sheet in order to see the readings on the LCD. Also, a notch was made for the sensor to take the readings. Harleen's enclosure was also derived from thingiverse website and it was

made from clear acrylic sheet however, a slit was made in the sidewise direction to make room for the sensor to sense the object. Lastly, Navkiran’s case fitted all the elements accurately but the bipolar stepper motor was connected through the jumper wires and it was placed outside the box because the motor was placed in the 3D printed case which was again inspired from one of the projects available at the thingiverse website.Therefore, for the enclosure, we have considered both laser cut printing and 3D printing which are learning resources gained from the hardware production course.

3.1.3 Tools and Facilities

The individual hardware projects which are involved in this project are accomplished using various tools and facilities. These privileges mainly include the facilities provided by Humber College itself. The prototype lab plays a vital role in the completion of our hardware projects. The prototype lab prepares a printed circuit board for students after getting the Gerber files from them. The prototype lab provides facility of the headers and sockets, soldering the pcb, jumper wires, wire cutters which are the required resources in our project. Moreover, the prototype also delivers the service of laser cutting and 3D printing. This is helpful at the time of the enclosure as the components are securely attached in the acrylic case enclosure. The prototyping lab staff were also very helpful in giving the instructions and guidance regarding the queries. They would also assist the students in terms of solving the issues related to hardware. Also, Humber faculty also helped in 3D printing of the bipolar stepper motor used in our project. The parts crib plays an important part in giving access to students with the essential tools needed for the practise and students understand and get to know about their functions in particular. The labs contain the soldering devices, multimeters, power supplies, oscilloscopes, and function generators etc. which are easily accessible to us while working on our hardware projects. These facilities contributed a lot in the execution of our projects and they will be helpful furthermore in the integration too. Without these facilities in college, it would have been very hard to work on the project from outside which would be very time consuming and costly. In addition to it, the soldering has been done in a safe environment as we are always asked to wear our safety glasses and proper and necessary equipment such as vacuum are available at the soldering spots. There is the Idea Lab in the college designed to support digital literacies at Humber College. The Idea Lab has three components: studio spaces, workshops, and online resources. They provide the facility of 3D printing and also guide us regarding it through conducting a workshop in the last semester. Humber libraries have been helpful in regard with access to the reading resources and also how to make references in the documentation. Coming to the tools, we have used fritzing for designing the printed circuit board and also making the schematic and breadboard connections through it. The CorelDraw application helps to create a .cdr file that is required to build the acrylic case. In the

software application, we have used the android studio in order to design the mobile device application for the project. The android studio comprised of all the significant features that are capable of building the application for the users. Therefore, with the help of these tools and facilities, the hardware production technology projects and the android application has been accomplished. Hence, these resources perform a major role in the production of hardware and the software.

3.1.4 Shipping, duty, taxes

Before planning for the project, we worked on our individual budget and ordered the parts required from various sources and mentioned the taxes, duty and shipping cost in our final budget. Saina's parts costed CAD $89.71 and tax of CAD $15.2 which made a total of CAD $104.91 all these items were bought from Amazon however; the shipping cost was free because of the subscribed member of the prime in Amazon. In Harleen's project, apart from the sensor additional things were bought such as keyboard and HDMI to VGA converter which made a total of CAD $121.95 without any additional charge such as shipping, taxes etc. because of the prime membership with Amazon. Navkiran's total cost amounted to CAD $173.89 of which the taxes were $3.63 and the shipping was all free as the components are ordered from amazon which also has the amazon prime facility for students. Some of the products are also bought in person from the store such as mouse and keyboards as they also contributed towards taxes.

3.1.5 Time expenditure

The working time has been different than the lead time which contributes to the whole time consumed by the project. The printed circuit board has taken around few hours to create design on the fritzing. Afterwards the time for pcb board to be made took around 1 and half day and then working on pcb such as soldering take half hour. For the ordered parts such as ordering on amazon take maximum 2 days as the amazon prime service was offered and available for the students. So, it took less time for delivering the products. However, product such as LCD module was delivered almost after 15 days as it took longer because it was shipped from another country. The schedule was prepared on the basis of the Gantt chart and the deadlines and due dates are considerably taken into effect with respect to time period. Although, the lead time mattered in the project yet the working time was appropriately adjusted and compensated according to the increased lead time. The lead time for the 3D printed motor was around a week which was more than the working time. The working time for the motor 3D printing only constituted for gathering 3D printing files and sending them for the creation. For the laser cutting, it took more time to create design in CorelDraw than actually having it laser cut that amounts to 2 or 3 days. The laser cut printing only took 15 minutes. For the software application, the working was more consuming basically while ensuring the connectivity to the database. Therefore, the time expenditure comes out as a vital factor and a determent of the outcomes for the hardware production and the software application.

3.2 Development Platform

3.2.1 Mobile Application

As a part of capstone project, we choose the parking problem as our project to solve the real-world problem. In the software project we develop an android application called ParkingEaze to provide the users with a facility to book a parking spot using this android application in order to tackle the traffic issues and delays in the working world. The android mobile application is developed on the android studio with the features that are contained within the android studio and implemented considering the requirements.

Test cases and plans implemented while developing the application

- Various test cases are implemented which includes all the validation for the email field and the password field such as in the register area. The test cases are passes that consists of the specifications.
- Password containing at least one number, an uppercase letter, a lowercase letter, the special character and the fields could not be empty.
- A splash screen is displayed when the user clicks on home icon of the app for a duration of 3 seconds.
- A dialog box appears if user click on soft back key, asking for either to exit or stay in the app.
- Around 11 test cases are implemented and tested in the android project
- There is a run time permission that asks the user to give the location access to the mobile application through displaying a dialog box when the application is started in the beginning.

We set some requirements for our android application which are further divided into functional and non-functional requirements

<u>Functional requirements</u>

- Mobile Application for end users requirements so they will be capable of using it
- Register for the parking and input personal and vehicle details.

- Finding a parking area from the list of areas
- View the details of a selected parking area.
- Reserve an available parking lot and specifying duration of reservation.
- For example, getting a confirmation once a slot is booked.

Non-functional requirements

- Availability: Independent platform so that it can be accessible and store app data
- Ability of server to handle simultaneous requests from different users.
- Confidentiality for user data using database encryption
- High availability and high accuracy in the locations finding
- Performance as to be started within a click
- Reliability for acknowledging the customer for any important action
- Ability to render its layout to different screen sizes

Mock-up discussion before developing the real application

- Mock-up discussion is based on the working sample for reviewing format and layout of the whole project.
- Used for experimental and instructional purpose

The mobile application is developed using the features such as runt time permission, location services, navigation drawer layout, java language, minimum api 23 which is marshmallow have been connecting it firebase but accomplishments to be made in this semester.The connection has been tried using the firebase but we are unable to connect it as we are getting the errors. We have researched regarding this issue but the connection to the firebase cannot be established despite the fact that we have tried a lot of attempts. We have consulted the software project professor in order to get further help regarding connectivity to database so the problem for this issue can be resolved.

Splash screen

When the user clicks on the homescreen of the application, the splash screen will appear for the duration of 3 seconds followed by a login screen that will ask user to login.

Figure 1: Splash screen view of the android application

Login Activity

Then comes the login screen where the user will have ability to enter the username and the password in order to login the application and it also has the register option which on the click will ask the user for registration if the user is not registered yet.

Figure 2 login screen view of the android application

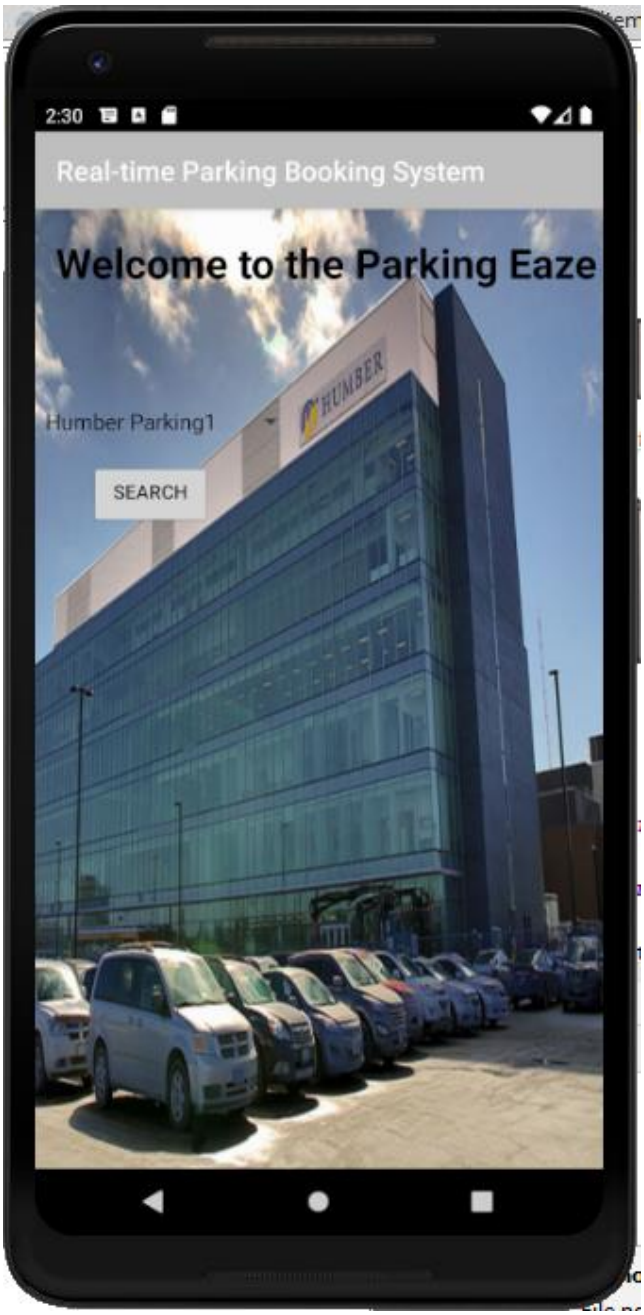


Figure 3 ;user asked to choose the parking location

Data visualization activity

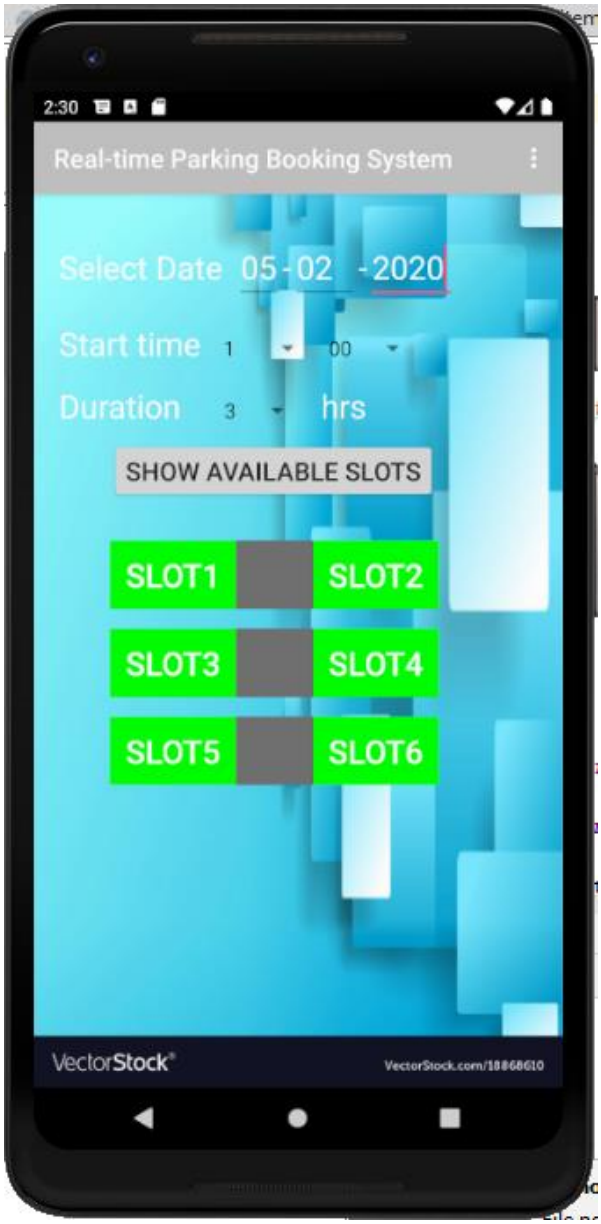


Figure 4the parking slots are shown which are empty

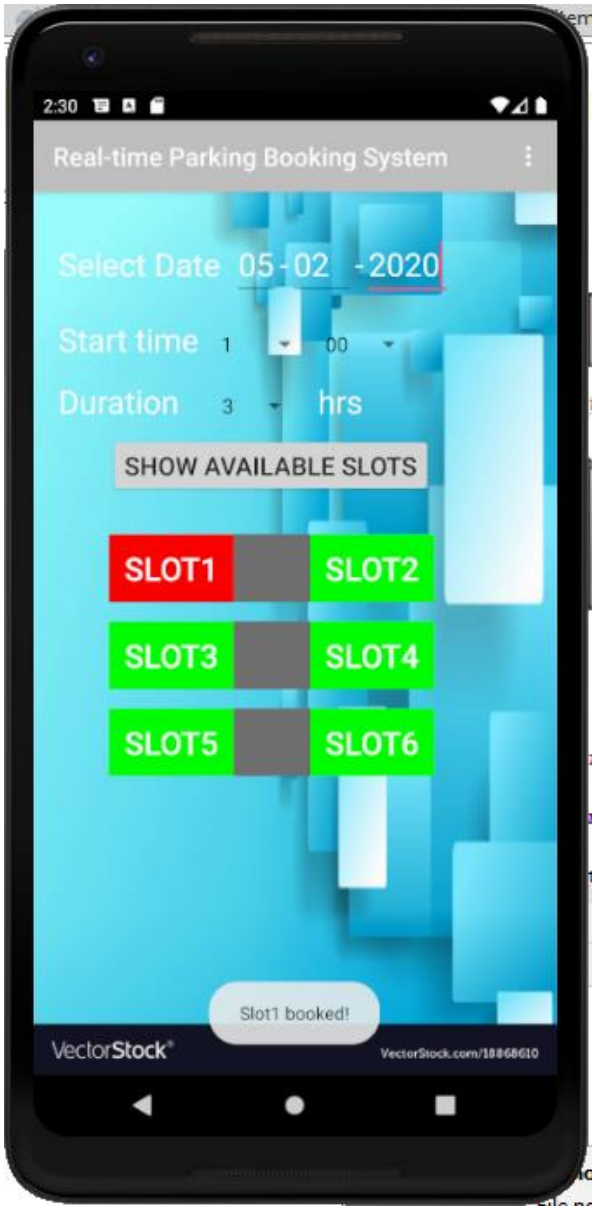


Figure 5 the user booked the parking slot which turns red

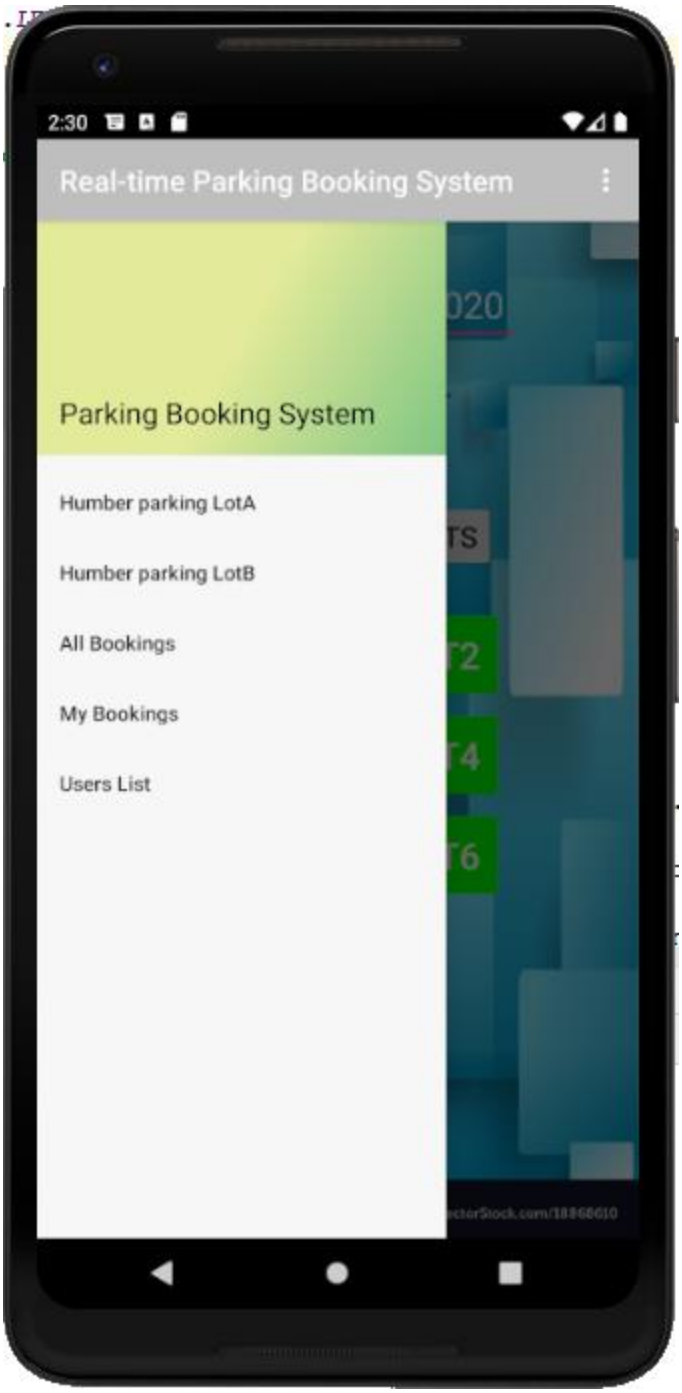


Figure 5 the navigation drawer layout has been implemented

Action Control Activity

After the user is logged in the mobile application, the location for the parking the vehicle are to be chosen from three options which are Humber Parking1, Humber Parking2 and Humber Parking3(Figure 3). The dialog box is displayed which shows these choices to user When the user clicks on one of the options then a new screen appears which asks the user to choose an empty parking spot(Figure4). The slot turns green to red (Figure5)after the user booked it and therefore the spot to park the vehicle has been booked by the customer through the android application.

Status

/1 Hardware present?

/1 Memo by student A + How did you make your Mobile Application? (500 words)

/1 Login activity

/1 Data visualization activity

/1 Action control activity

3.2.2 Image/firmware

This documentation depicts the structure of the code and connectivity to the raspberry pi. In the Navkiran's project, bipolar stepper motor that is Nema 17 is driven by the DRV8825 driver on the raspberry pi development platform. The Raspberry Pi is used to control bipolar stepper motors on a Raspberry Pi in Python using a DRV-8825 stepper motor driver. Saina’s project is to measure the distance of the car and display it on the LCD screen. The LCD screen will display the available parking spots and will give indication and direction to the customer and the HC-SR04 the ultrasonic sensor is also used with the LCD module. Harleen’s project is to reduce parking problems using the Ek1254 sensor. Ek1245 infrared sensor will detect if there is any car parked in the parking spot. These three-hardware production projects are developed on the raspberry model 3B+. The raspberry has been set up using connecting the raspberry with power supply, HDMI cable from raspberry pi to the HDMI to VGA cable and that VGA side to the monitor. Connecting the separate mouse and the keyboard with the pi. Following the steps below:

- Download Raspbian to be installed on your raspberry pi.
- Use SDCardFormatter to format your SD card so for getting the pi to work
- Inserting the SD card in raspberry pi and ensuring the connection all the above-mentioned parts to be connected appropriately.
- We downloaded the Raspbian image file that we download onto an SD card which in turn can be used to boot your Raspberry Pi and Via APC into the Raspbian operating system. Using a Raspbian image is the easiest way for a new user to get started with Raspbian.

- Switch on the power and finish the further setup including changing the settings.
- Giving the commands on the terminal that require installation like sudo updates etc.

These steps were followed in the last semester for setting the raspberry pi to the hardware.

Connection of VNC viewer to the raspberry pi

Cloud connections are convenient and encrypted end-to-end, and highly recommended for connections over the Internet. There's no firewall or router reconfiguration and you don't need to know the IP address of your Raspberry Pi or provide a static one that is why we used VNC viewer. For the authentication, to VNC viewer we needed to complete either a direct or cloud connection to VNC Server. After that, we entered the user name and password to log on to the user account on the Raspberry Pi. This is how our firmware got connected with the VNC viewer. This enables the wireless connectivity and the internet availability on raspberry pi should be similar to that of VNC viewers and that will ensure the connectivity. We are using raspberry pi and bipolar stepper motor which need the power supply as separate in order to operate as we need power cables for them. The foremost and major problem that we have faced is connecting the pi to the VNC viewer with the secure Wi-Fi network and still, there is the process of troubleshooting going on with the connection. The code used for the working of the hardware projects used in the last semester is written using the python language. The

separate codes used for our individual projects are posted on the GitHub and we have provided the link below to access the software code.

https://github.com/NavkiranKaur/ParkingEaze/tree/master/firmware

Status

/1 Hardware present?

/1 Memo by student B + How did you make your Image/firmware? (500 words)

/1 Code can be run via serial or remote desktop

/1 Wireless connectivity

/1 Sensor/effector code on the repository

3.2.3 Breadboard/Independent PCBs

The below description depicts the hardware that had been made for the project.

The effector used by Navkiran is the bipolar stepper motor along with the DRV8825 driver. The bipolar stepper motor that is Nema 17 is driven by the DRV8825 driver on the raspberry pi development platform. The stepper motor will be able to move the barriers in the car parking in order to allow the entrance and exit of the vehicle using the rotational movement controlled by the driver and raspberry pi. The development for the motor's hardware project, the instruction based on this link is followed. The link that followed is https://www.rototron.info/raspberry-pi-stepper-motor-tutorial/ and based on this website, the schematic is designed in the fritzing (Figure 1). The other designs that are constructed in the fritzing software are the breadboard (Figure2) and PCB(Figure3). The schematic design is constructed based on the following connections which are listed below: The connections between the raspberry pi and the stepper motor

Raspberry pi pin 3v: DRV8825 SLEEP, DRV8825 RESET

- Where connecting the sleep and reset together whereas connecting pi to one of them
- Raspberry pi pin GND: DRV8825 GND1
- Raspberry pi pin GPIO21: DRV8825 STEP
- Raspberry pi pin GPIO20: DRV8825 DIRECTION
- Raspberry pi pin GPIO14: DRV8825 MS1

- Raspberry pi pin GPIO15: DRV8825 MS2
- Raspberry pi pin GPIO18: DRV8825 MS3

The connections for the driver with the raspberry pi and the bipolar stepper motor

IN DRV8825 -DRV8825 GROUND 1 to be connected with GROUND2 of DRV8825 itself -In order to bind the motor wire pins with the drv8825

• The motor has two coil pairs Coil pair 1, coil pair 2

- Coil pair 1 includes A and C
- Coil pair 2 includes B and D
- MOTOR A WIRE PIN: DRV8825 OUT 2B
- MOTOR C WIRE PIN: DRV8825 OUT 2A
- MOTOR B WIRE PIN: DRV8825 OUT 1A
- MOTOR D WIRE PIN: DRV8825 OUT 1B
- MOTOR POWER SUPPLY ADAPTOR OF 12 V

Connecting the capacitor

- Capacitor (positive side): DRV8825 VOLTAGE PIN: POWER SUPPLY POSITIVE (12V)
- Capacitor (negative side): DRV8825 GROUND: POWER SUPPLY GROUND

• The pin can be connected vice versa but make sure coil pair pins goes with the pair on the driver, not interchanging if connecting a pair both pins to be connected to other driver pair itself. Don’t connect the pin in the pairs differently.

Under these considerations, breadboard connections are made (Figure 4). After that, the actually printed circuit board is soldered (Figure5 and Figure6). The connections with PCB with all the hardware components are shown in figure7.

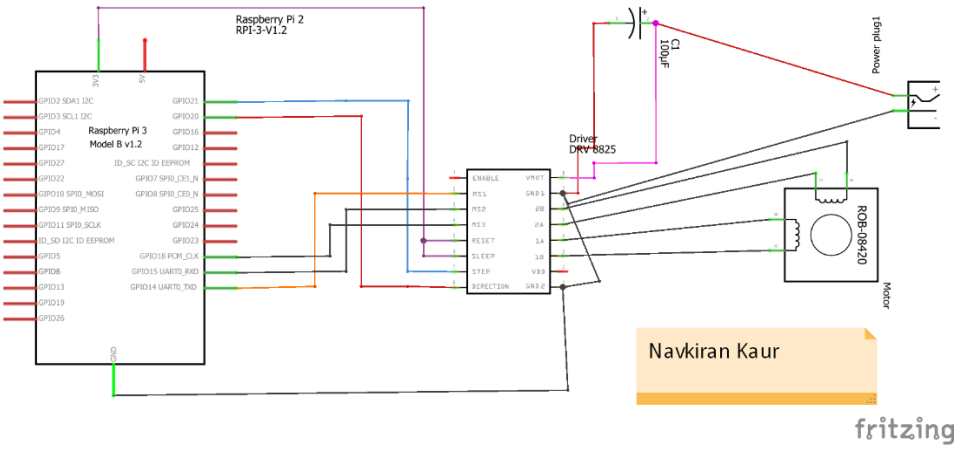


Figure 6.The schematic design for the stepper motor

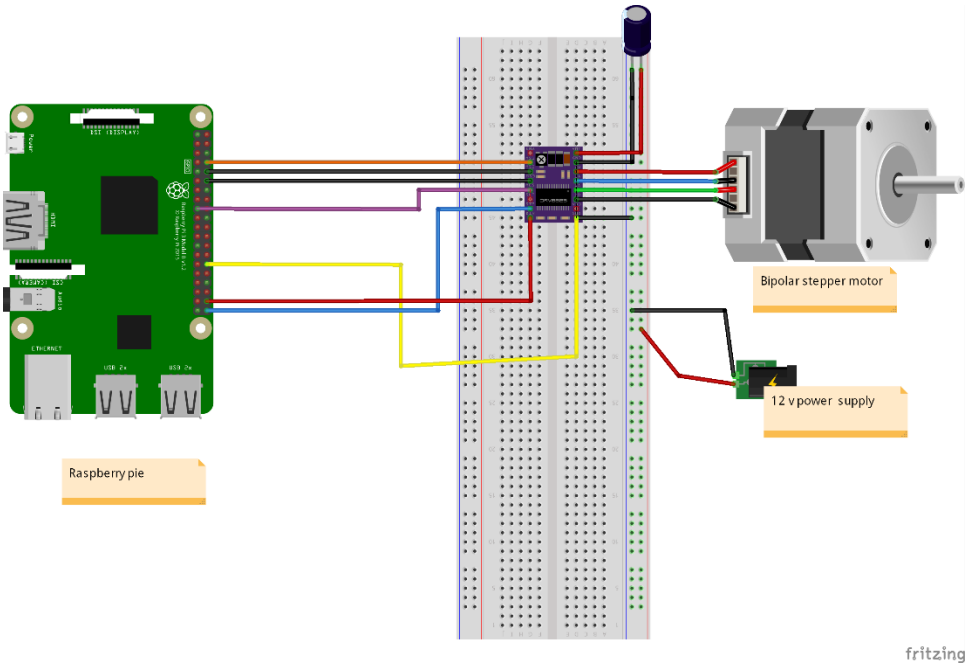


Figure 7 The breadboard design made using fritzing for the stepper motor

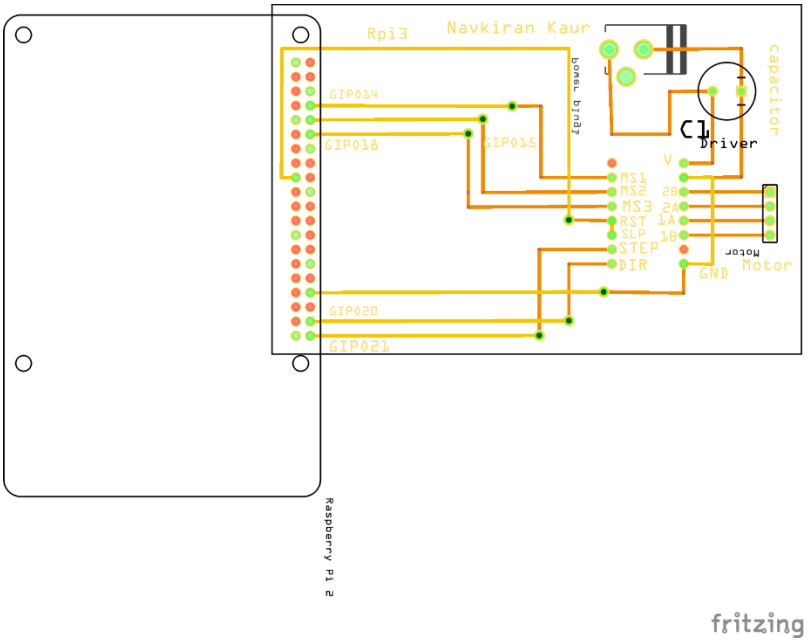


Figure 8 The pcb design for the stepper motor

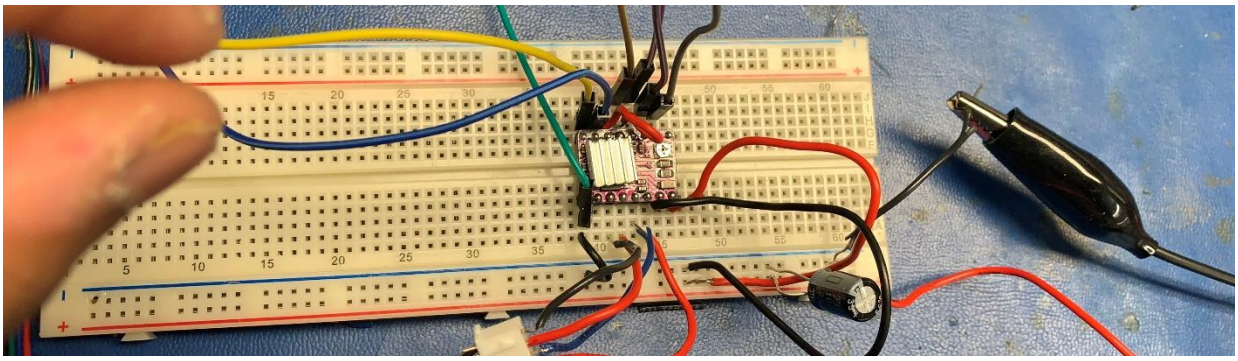


Figure 9 The breadboard connection

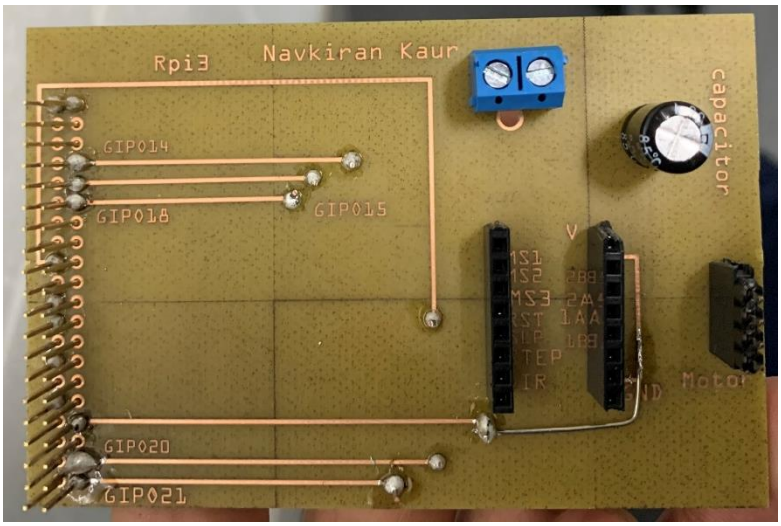


Figure 11 The printed circuit board

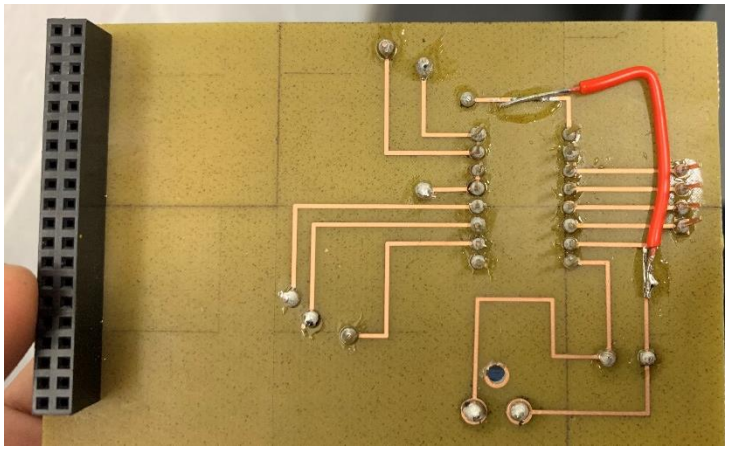


Figure 10 The other side of motor's printed circuit board

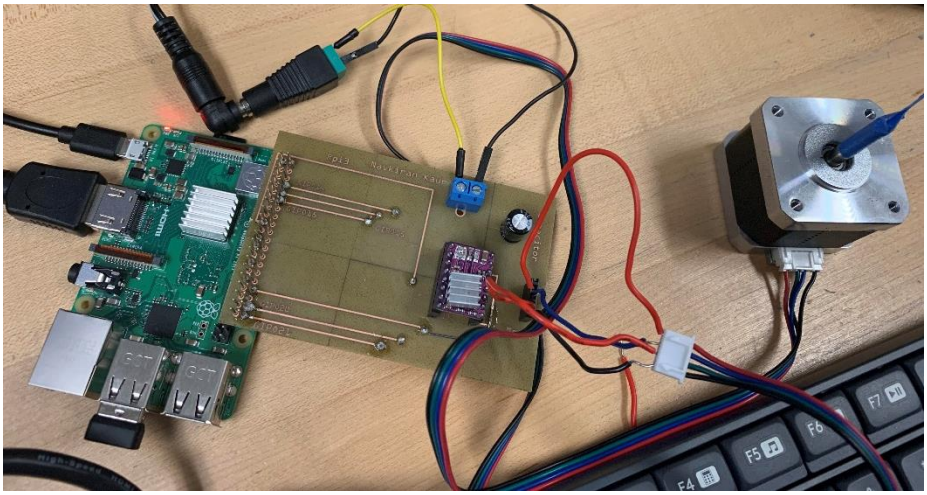


Figure 12 The connections made with the printed circuit board with the motor, pi and the driver

In Saina's project, two components were used one is the sensor and the other is the effector which is LCD. The main idea of using these two components was to measure the distance of the car and display it on the LCD but many things were kept in mind before doing that. The first major thing is the voltage of LCD should be compatible with the raspberry pi, it should not exceed more than the voltage supplied by
a raspberry pi. Second thing is, the LCD has a PCF8574 backpack which is IOT so apart from power and
ground it will also include SDA and SCL. For the HC-SR04 ultrasonic sensor, it was not directly connected to raspberry pi, two resistors were used as a voltage divider.

For HC-SR04 sensor

- Sensor VCC to 5V on Raspberry Pi.
- Sensor GND to GND on Raspberry Pi.
- Sensor TRIG to Pin 16 GPIO23.
- Sensor ECHO to 1k resistor, another leg of 1k resistor to Pin 18 GPIO24.

For PCF-8574 LCD module

- LCD VCC to 5V on Raspberry Pi.
- LCD GND to GND on Raspberry Pi.
- LCD SDA to Pin 3 GPIO2 / SDA
- LCD SCL to Pin 5 GPIO3 / SCL

The breadboard connections are made with the LCD module, raspberry pi and ultrasonic sensor (Figure 10). After that, the actually printed circuit board is soldered

The connections with PCB with all the hardware components are shown in figure13.

Also, Figure 14 shows the PCB that has been made in the fritzing software.

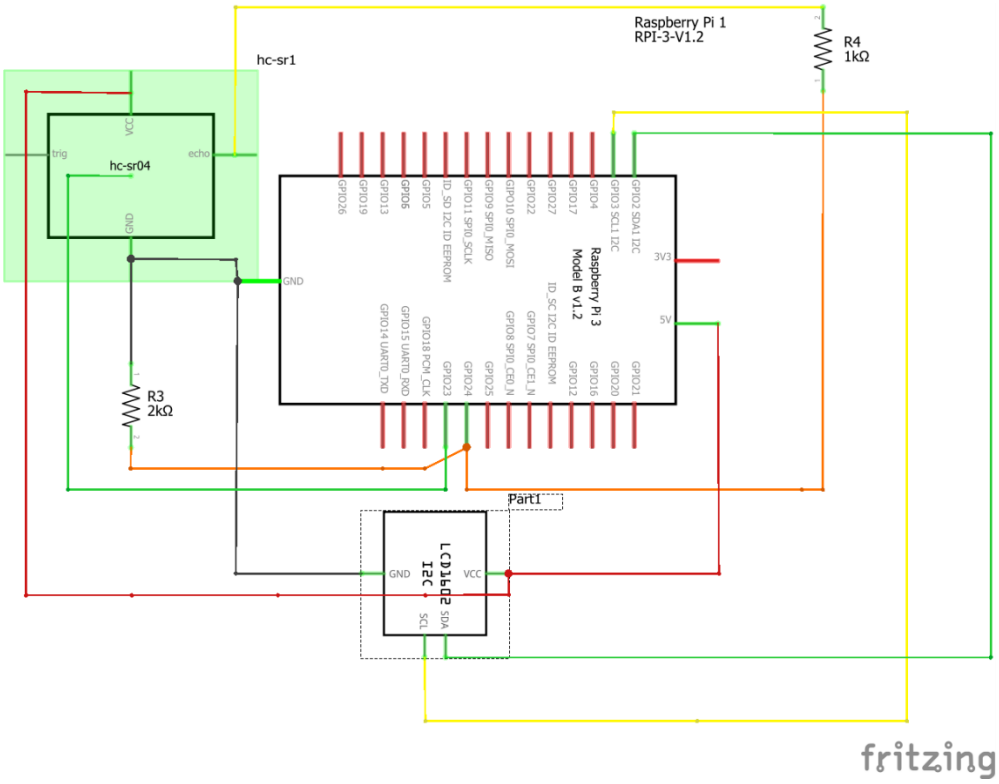


Figure 13 The schematic design

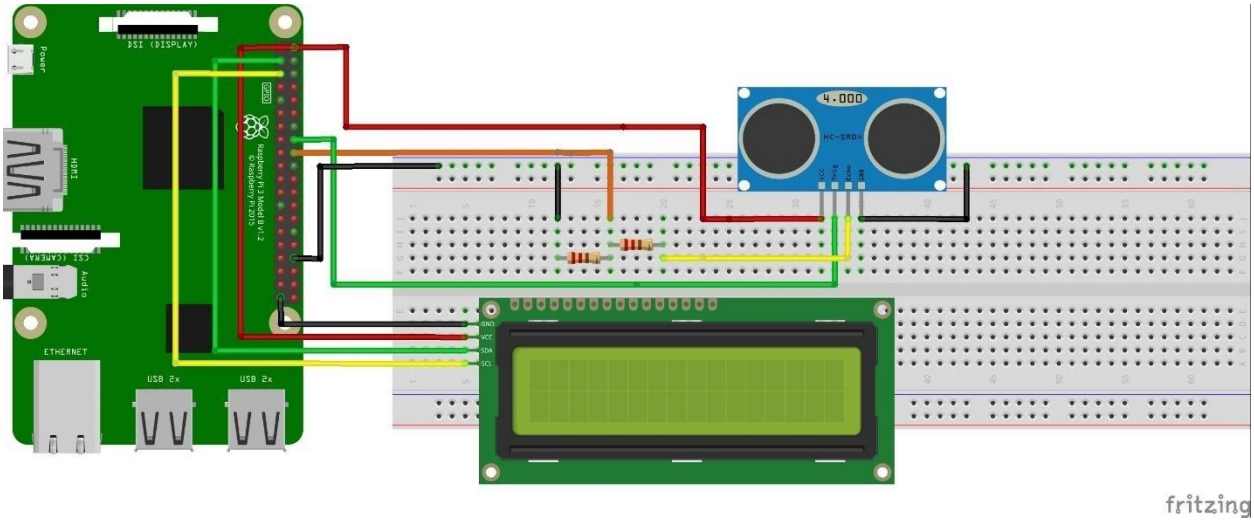


Figure 14 The breadboard design for LCD

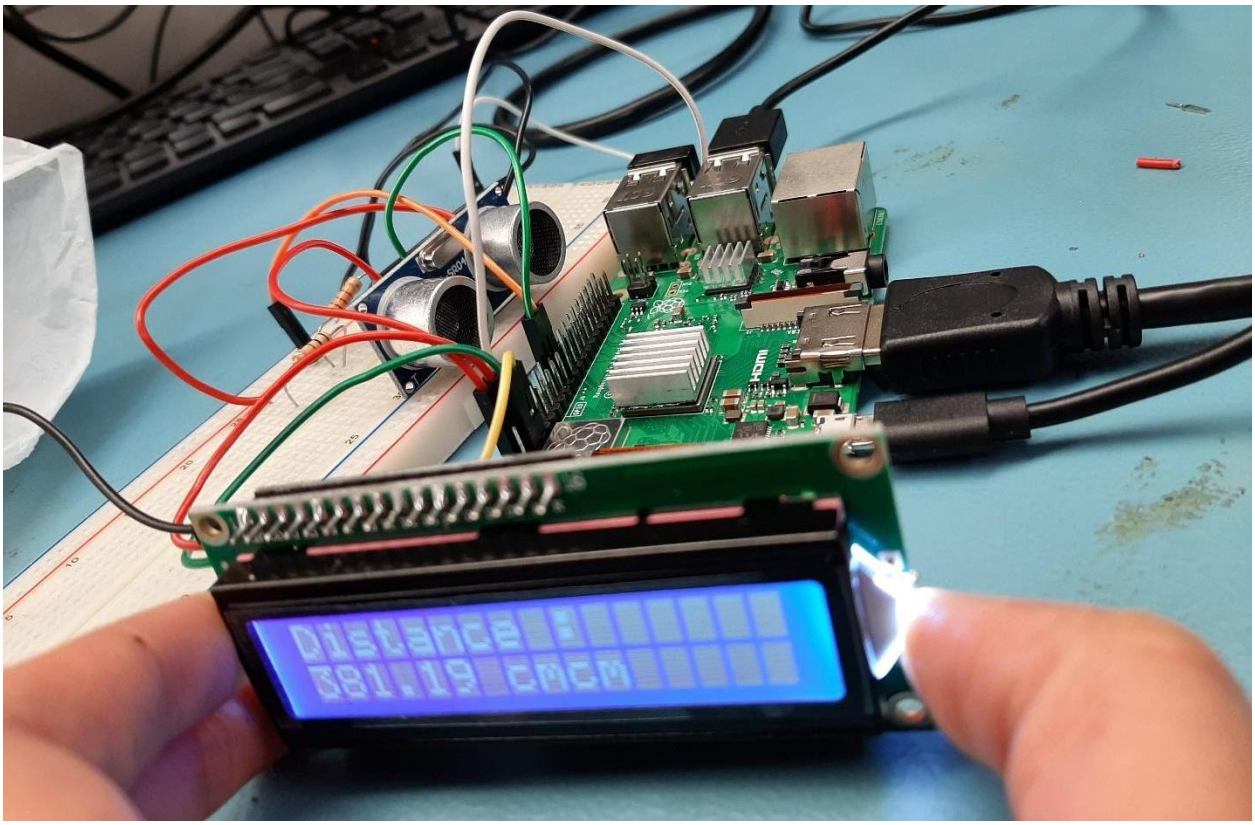


Figure 15 The connection made on breadboard for LCD module,pi and the ultrasonic sensor.

.

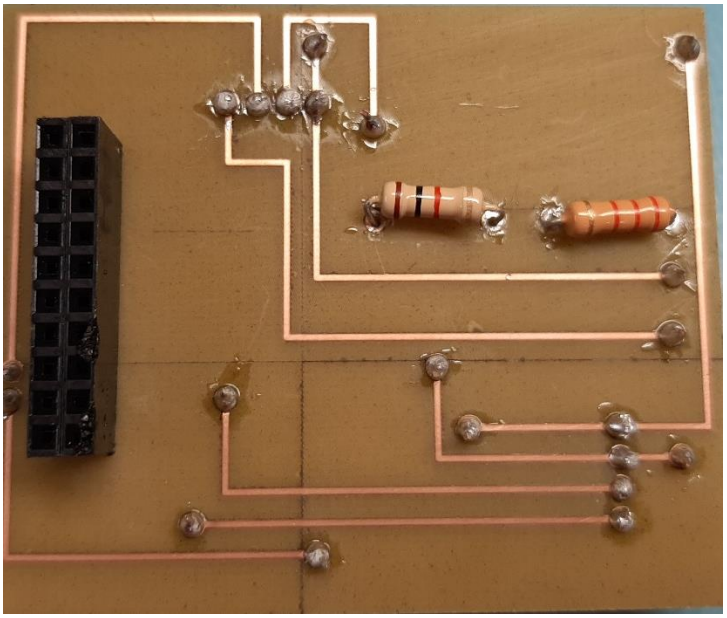


Figure 16 The pcb soldered for the LCD module

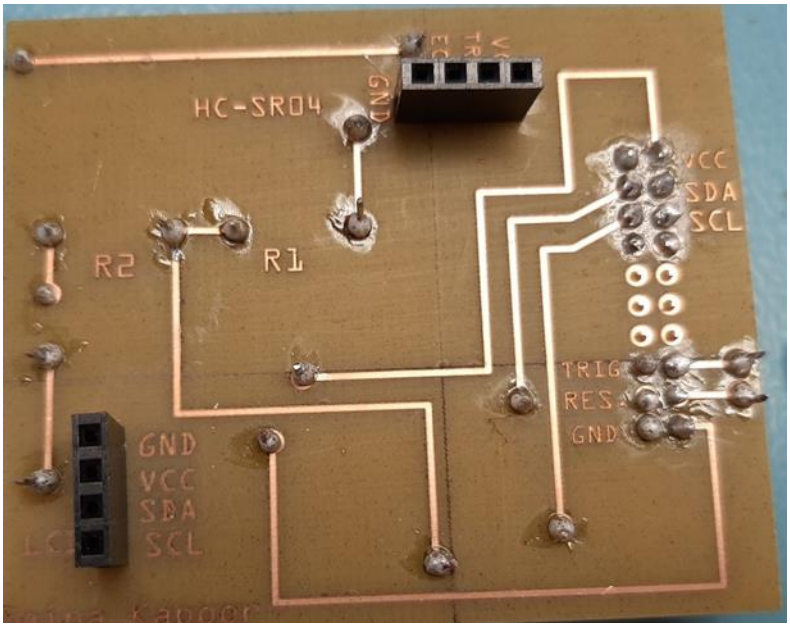


Figure 17 The pcb for LCD

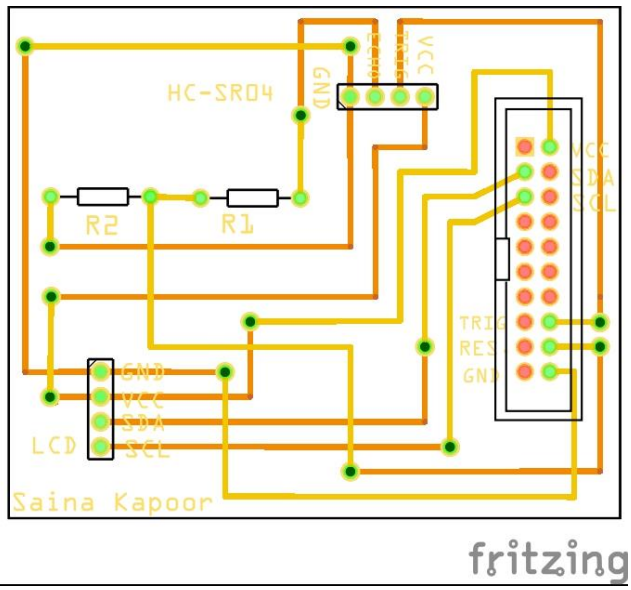


Figure 18 The fritzing design

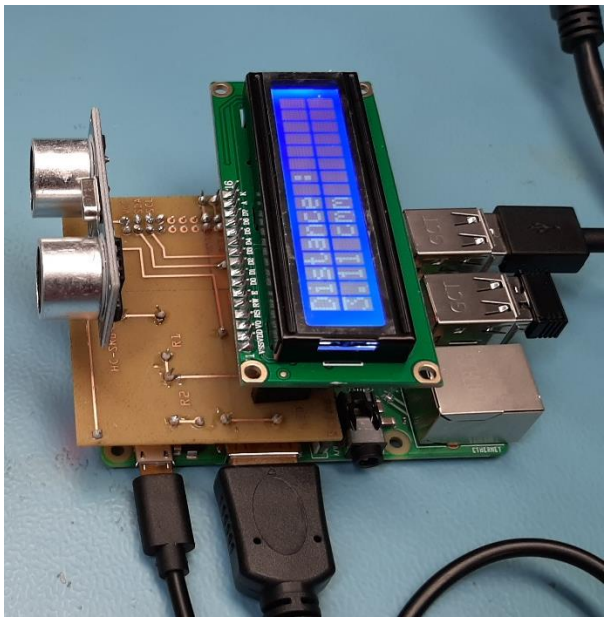


Figure 19 The pcb mounted on raspberry pi

In Harleen's project there is one sensor which is IK1254 Infrared sensor which is used to check if the parking spot is available or not. Firstly, schematic [Figure14], breadboard(Figure15),PCB(Figure 16) was made on fritzing and used the below given wiring.

- Raspberry Pi 3V to sensor VCC
- Raspberry Pi GND to sensor GND
- Raspberry Pi GPIO4 to sensor OUT

After making the schematic fritzing generated breadboard design which was used to make the real breadboard. Then wirings were tested first on breadboard so that before making the final pcb further changes can be made if something went wrong in the wirings. Jumper wires were used to make connections from sensor to breadboard and breadboard to raspberry pi. The breadboard test was successful in a single shot. Sensor was working fine when tested on breadboard (Figure 17). Then

that the printed circuit board is soldered and connections are made with the hardware components are shown in figure18.

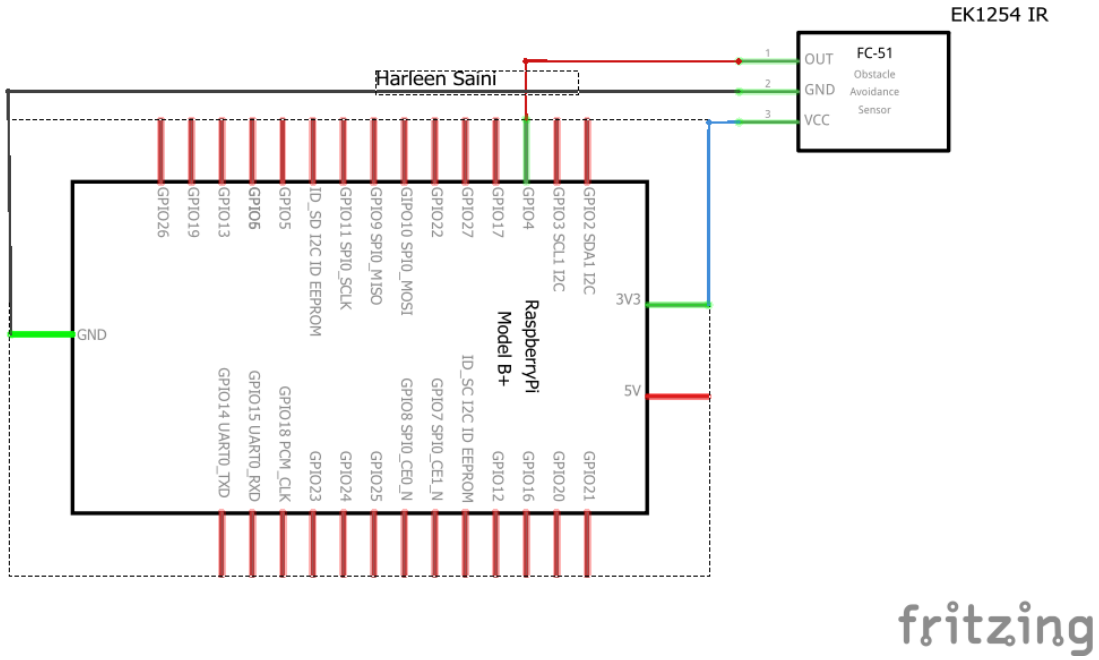


Figure 20 The schematic design for infrared sensor

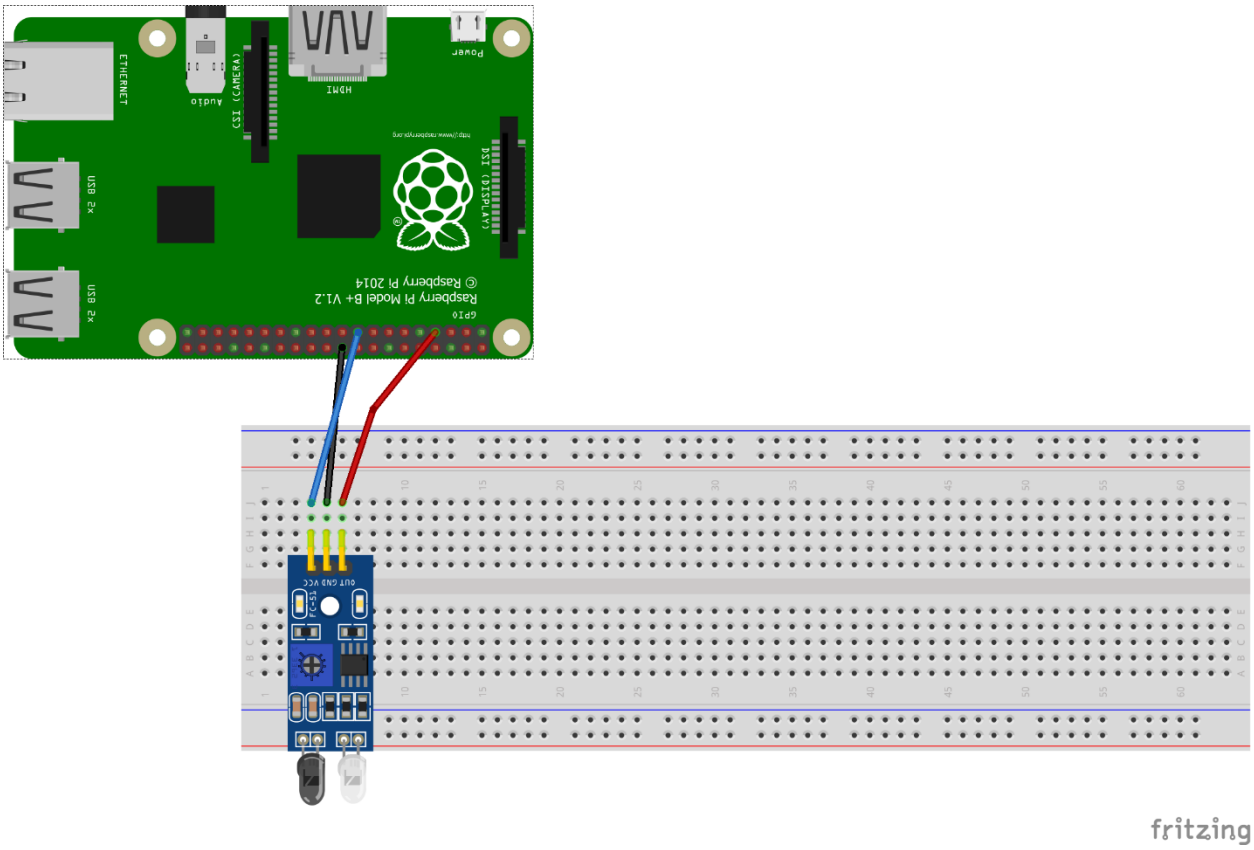


Figure 21The breadboard design including the infrared sensor

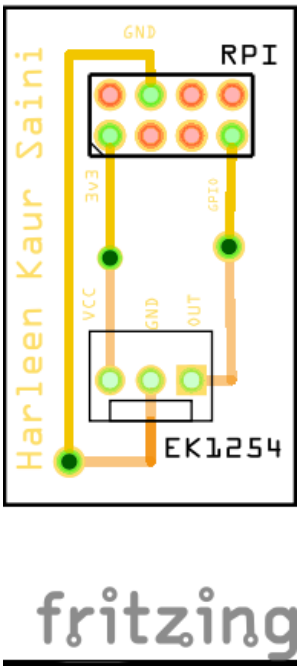


Figure 22 The printed circuit board design in fritzing

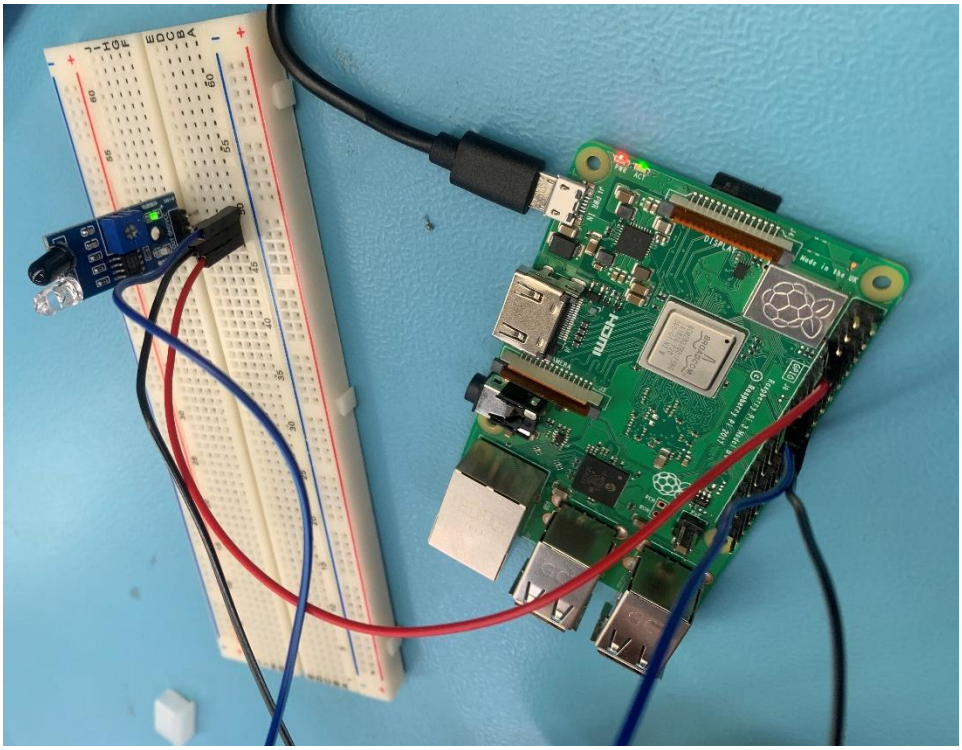


Figure 23 The breadboard connections with the wiring with hardware components

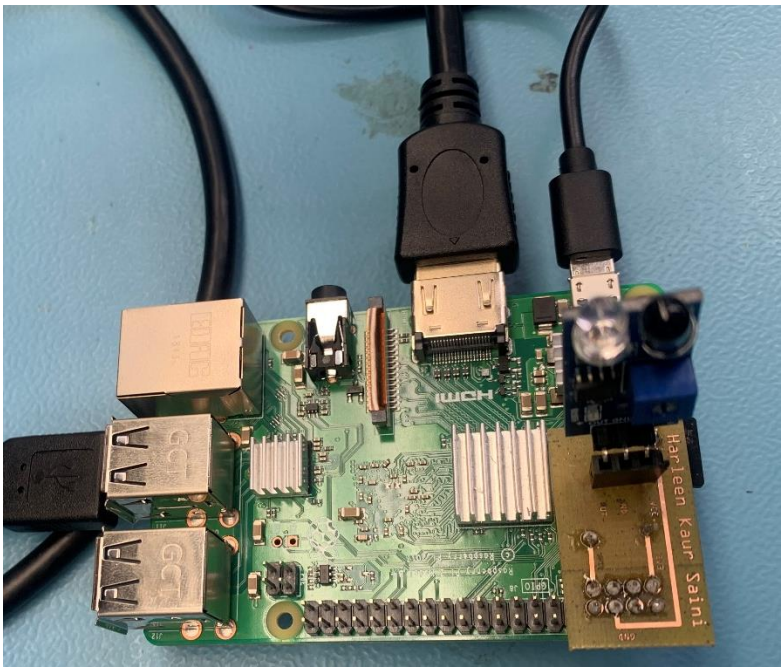


Figure 24 The printed circuit board powered up with the pi and sensor

Status

/1 Hardware present?

/1 Memo by student C + How did you make your hardware? (500 words)

/1 Sensor/effector 1 functional

/1 Sensor/effector 2 functional

/1 Sensor/effector 3 functional

3.2.4 Printed Circuit Board

Demo

/1 Hardware present?

/1 PCB Complete and correct

/1 PCB Soldered wire visible but trim, no holes or vacancies

/1 PCB Tested with multimeter

/1 PCB Powered up

How did you build your Prototype: PCB?

The printed circuit board is made through some amendments in the three individual printed circuit boards which were made by the group members in their hardware projects in the last semester. Then these printed circuits boards are combined into one PCB after testing these connections on the breadboard using jumper wires. The PCB of Navkiran ,that contains the connection of motor and driver, is combined with the addition of two sensors of Harleen which are infrared sensors. The connection of the PCB consisting of the LCD module and ultrasonic sensor is also combined through making the connections to raspberry pi within the same printed circuit board. An additional ultrasonic sensor, two resistors with reference to the previous connection in Saina's PCB are added in the new printed circuit board. The via and the bend points are securely made in the fritzing to make complex connections simpler. The raspberry pi is soldered on the top layer while all the other components are soldered within the bottom layer. The connection wires on PCB are mostly bent at right angles in order to ensure clarity for the connections. The PCB is acquired after a week from the prototype lab, even though the Gerber files are sent a week before the PCB is received. The PCB is soldered and then tested with the multimeter. The design in the fritzing for the printed circuit board had no errors and we make sure the wire do not get intersect with the wire of same color. The intersecting wire connections are resolved through via and bend points.

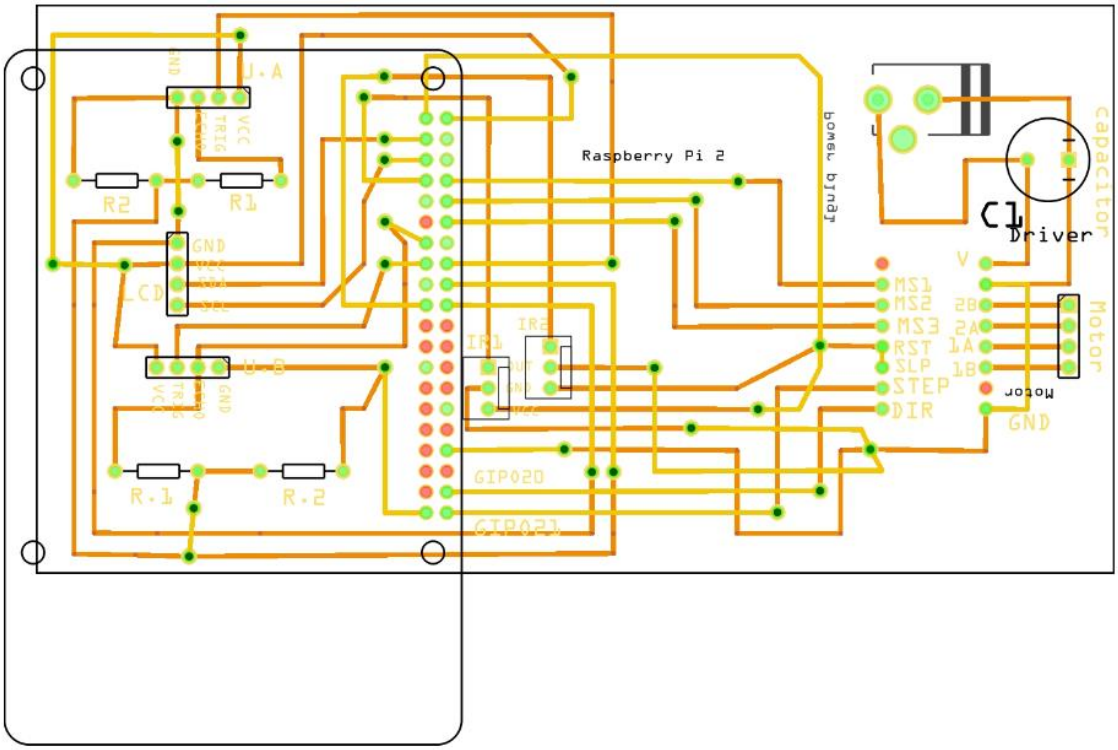


Figure 25 The pcb design in the fritzing

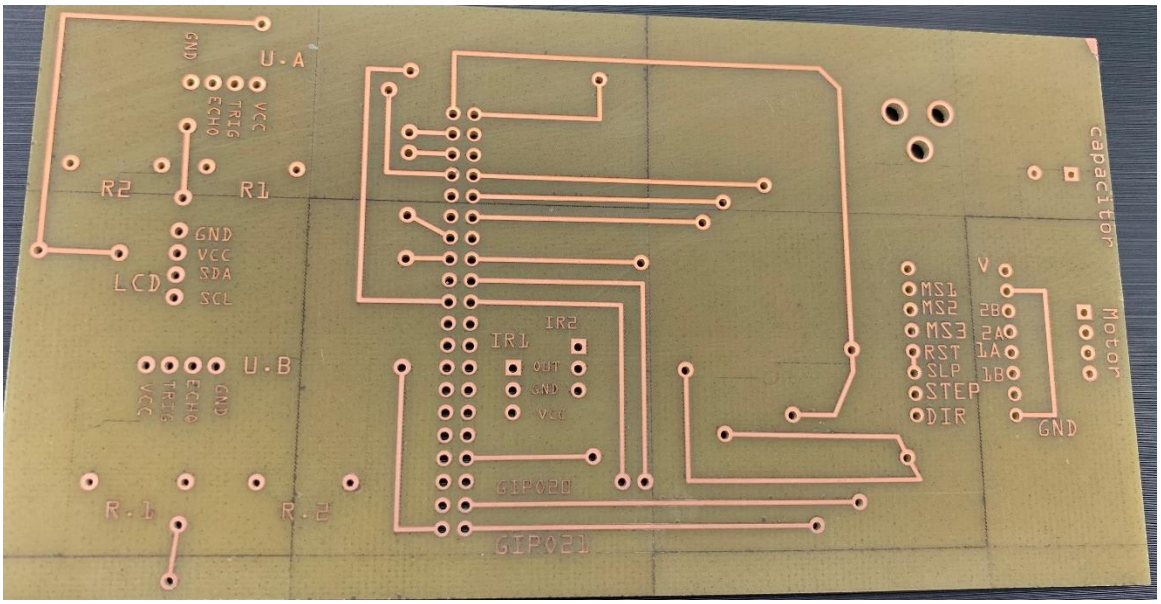


Figure 26 The image of the received pcb from prototype lab

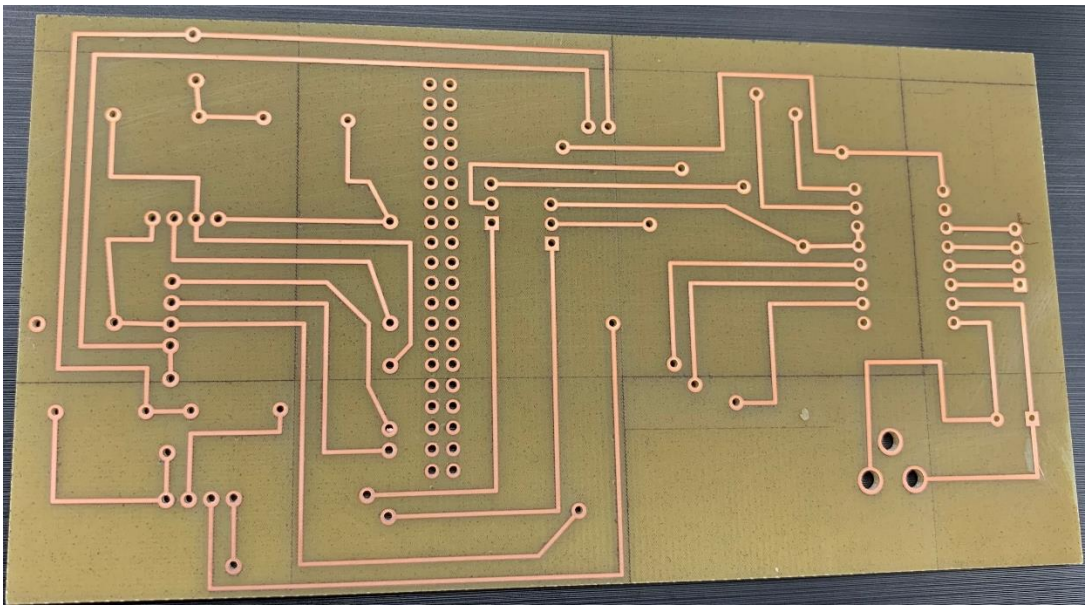


Figure 27 The bottom side of the pcb

PCB tested with multimeter

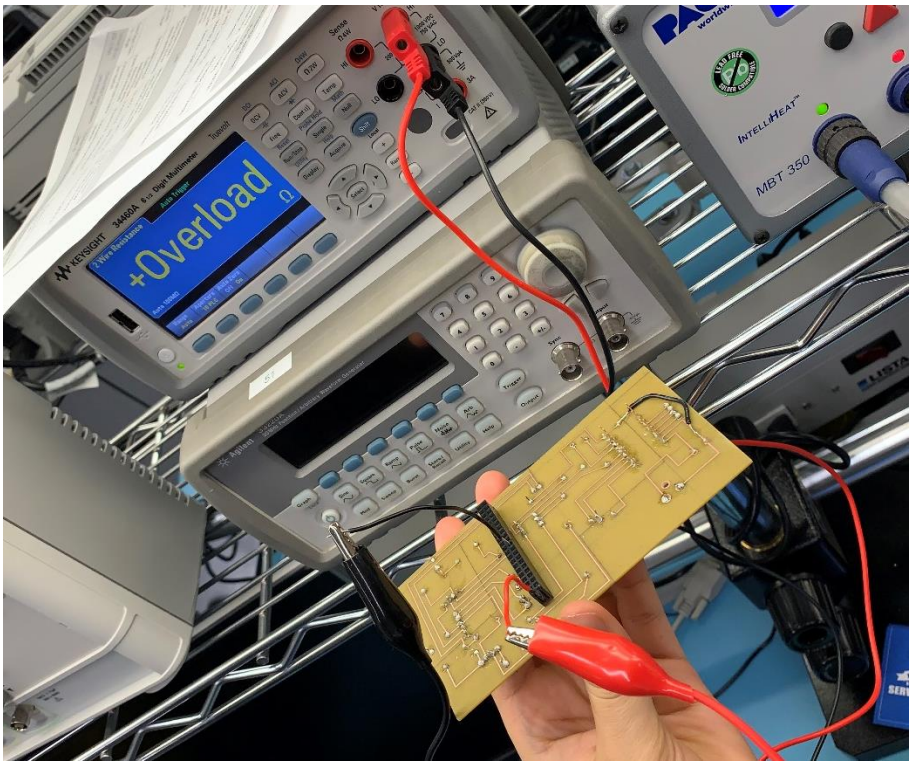


Figure 28 The testing of the pcb with the multimeter from voltage and ground

The below image shows the surface mounting of the wire and hence tested with the multimeter.

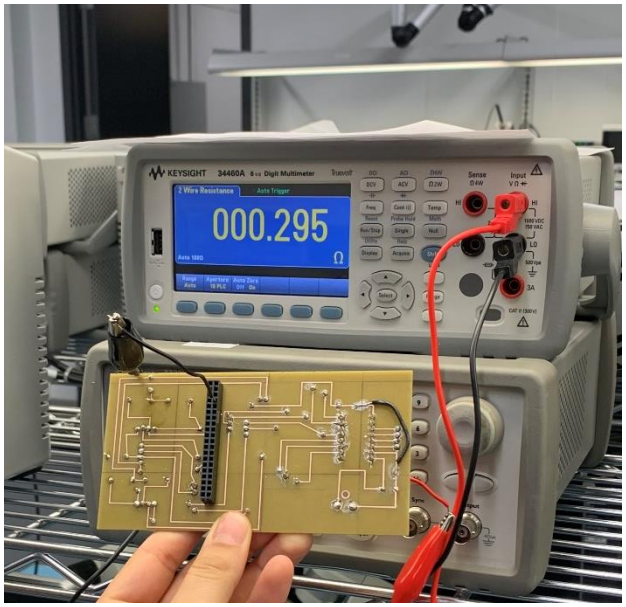


Figure 29 The multimeter connected to vcc and ground for ensuring surface mounting wire

PCB powered up

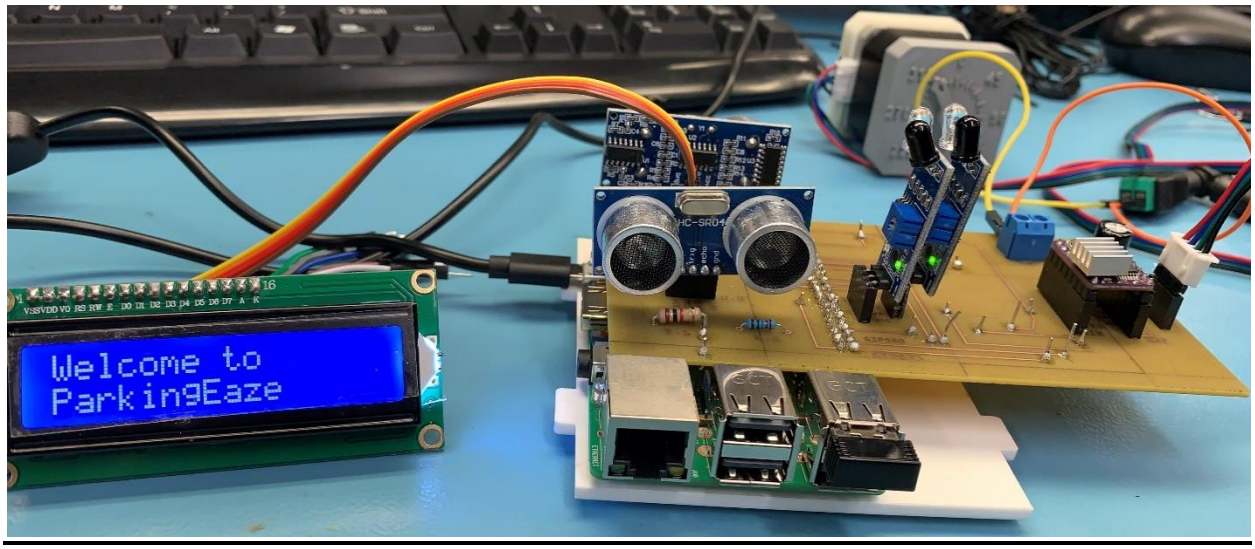


Figure 30 The powering up of the printed circuit board

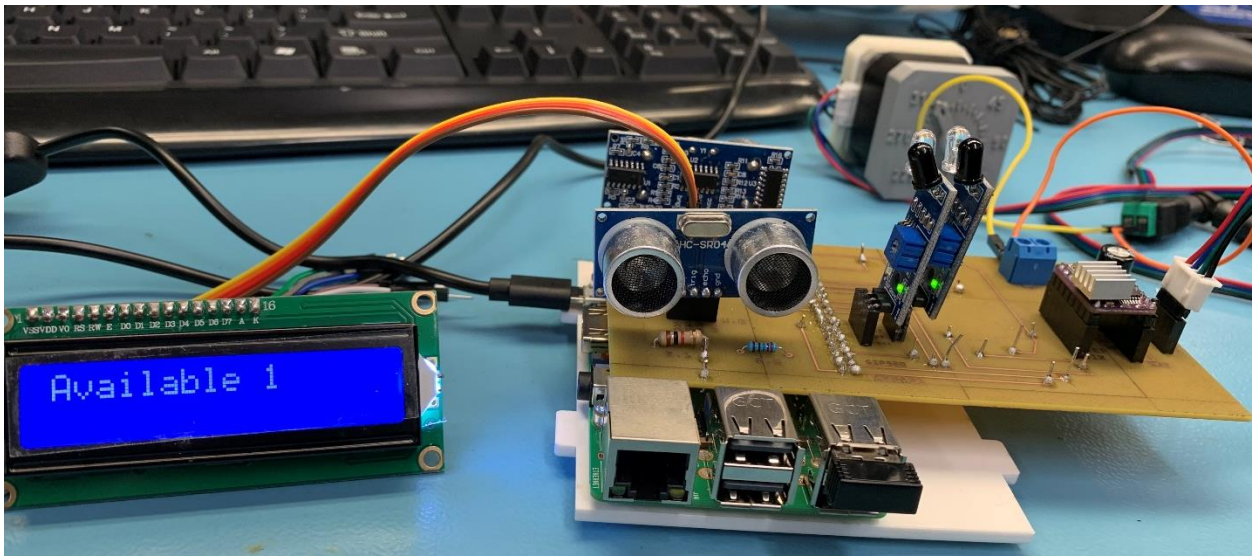


Figure 31 The LCD shows the number of available parking spots after powering up

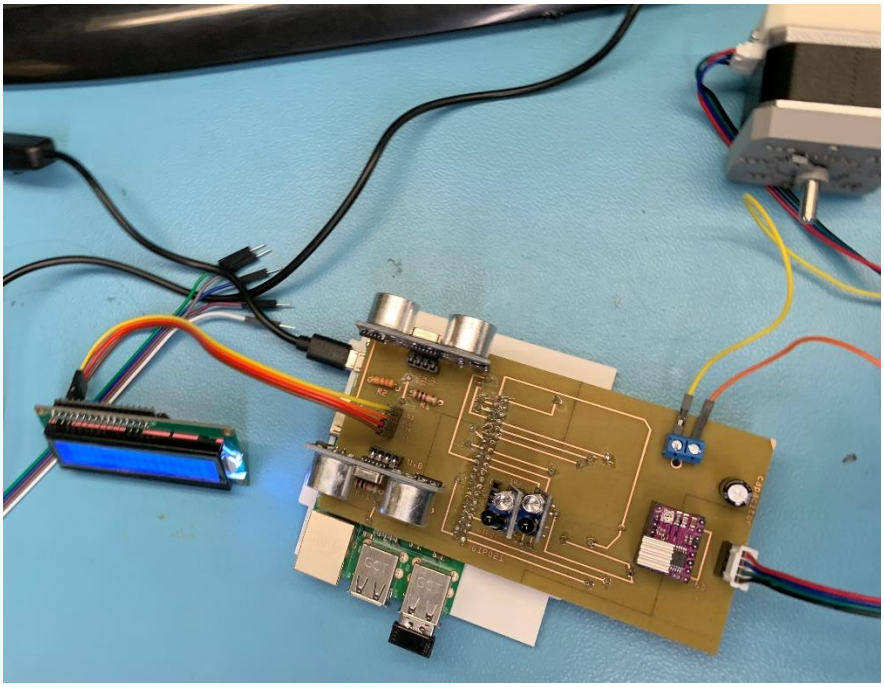


Figure 32 The powering up of pcb image representing all components

3.2.5 Enclosure

Demo

/1 Hardware present?

/1 Case encloses development platform and custom PCB.

/1 Appropriate parts securely attached.

/1 Appropriate parts accessible.

/1 Design file in repository, photo in report

3.2.5.1 Building the enclosure

The case encloses the printed circuit board and the development platform (raspberry pi) which is connected to the appropriate parts used in the project. The dimensions for the case were taken using a reference case already present in the prototype lab. The case is built using the laser-cut technique with the help of the prototype lab. The application which helped us to design the case was the MakerCase the only thing we need to enter was the dimensions of the box and in the CorelDraw, we made some necessary changes like added some holes and slits in the design. MakerCase is a free web tool for designing custom project cases and simple to use which requires entering the dimensions for the case. The barrier we had for the parking was too short to pass two vehicles for entrance and exit which was made in the previous semester as a 3d printing. Hence, a new acrylic barrier was designed using CorelDraw. We were thinking to make again 3D barrier but due to a shortage of time as 3D printing will take 2-3 days and the college was going to shut down from the following week, we dropped the idea of making the 3D barrier and continued with the acrylic design. The motor covers are made of 3D printing on the top and bottom but it is also covered with acrylic so it gets enclosed completely and properly. Two acrylic rectangular stands are also included in the design, one is kept at one of the corners of the base just for a better look of the parking lot and the other is designed and placed in the center to hold the LCD which displays the available parking spots. The bottom surface is transparent from where one can be able to view the raspberry pi, PCB and other parts attached. The pictures clicked for the enclosure are displayed below.

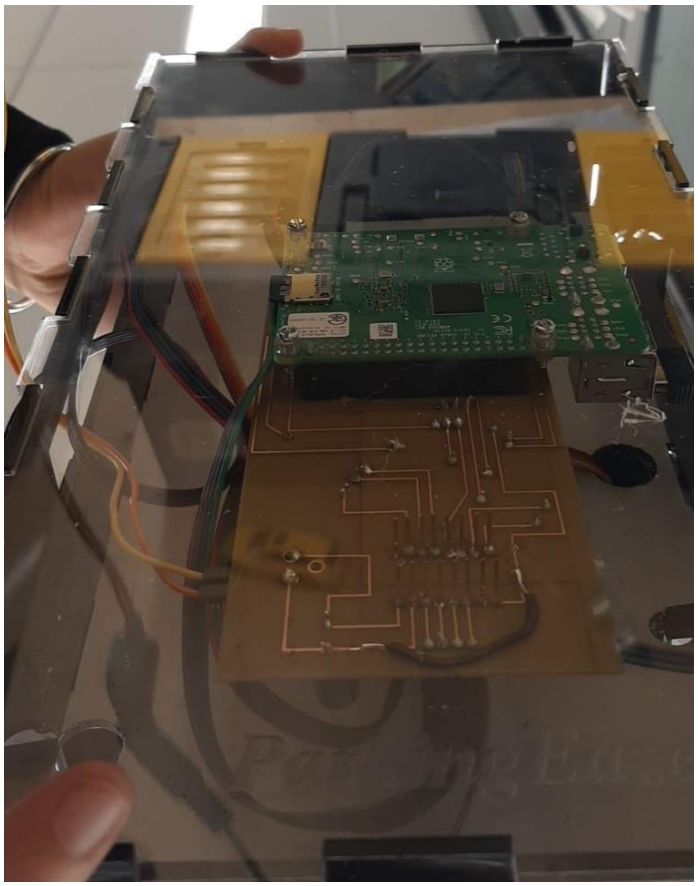
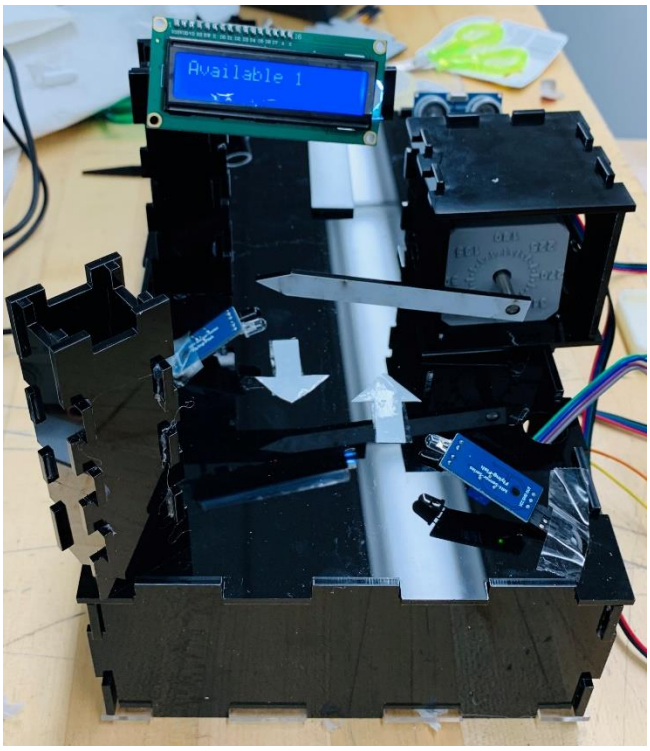


Figure 33 The bottom surface of the enclosure which is transparent

Figure 34 The barrier moves allowing the car to exit the parking lot



IFigure 35 The LCD placed in the center using a stand ıdisplays the available spots

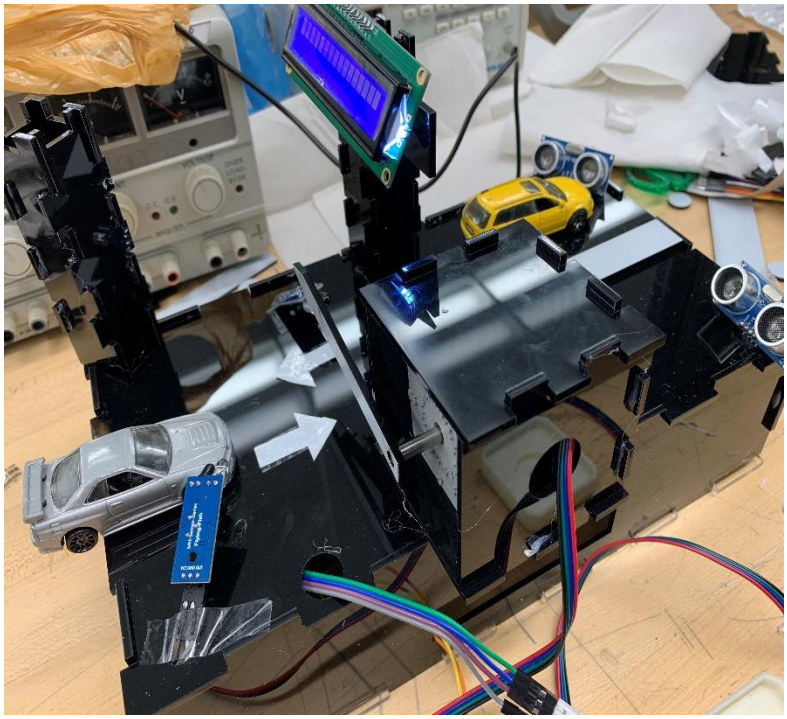


Figure 36 The figure shows the side view of the lot

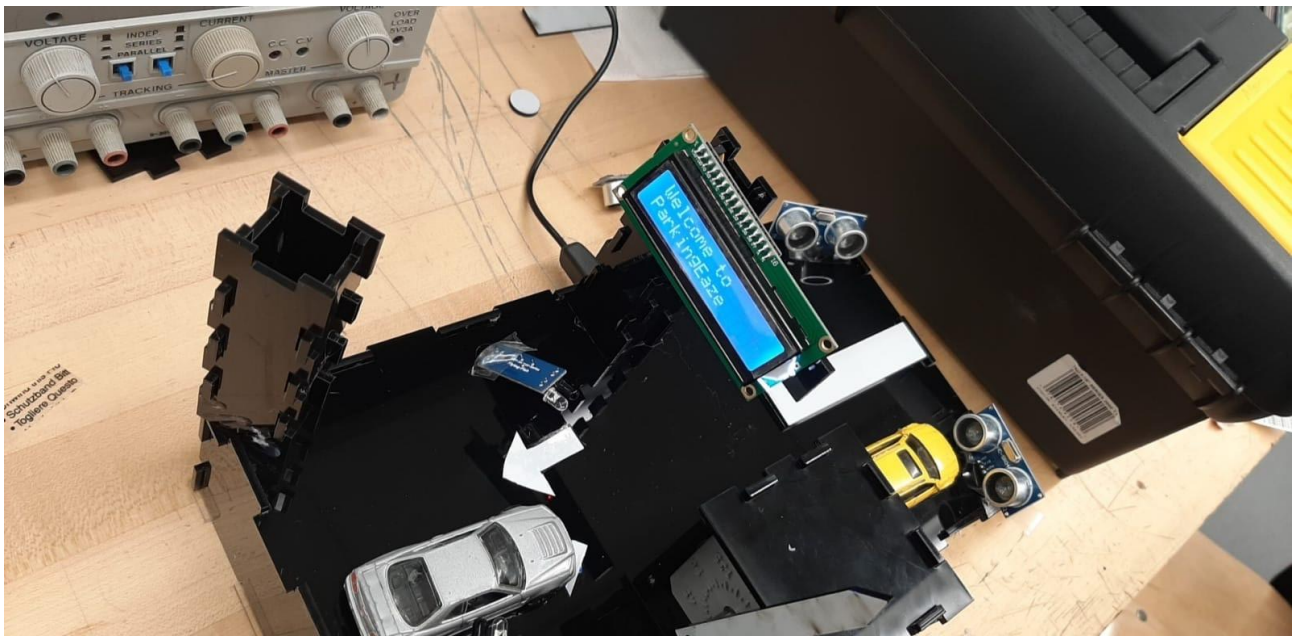


Figure 37 The figure shows the entrance of vehicle

3.3 Integration

Database connections functionality update

This document provides the information regarding the functionality of the database in the android mobile application. The demonstration has been held during the online class session for the database connection. The database is stored and saved in the firebase for the android application.

Grading for this milestone:

/1 Participated: Online session

/4 Online demo

/1 Addresses connection to enterprise wireless (v.s. home/open Wi-Fi/hotspot)

/1 Database configuration mentioned

/1 Security considered

/2 Unit and production testing considerations

3.3.1 Database Configuration

Database configuration

Firebase Implementation for our database

A firebase project is created using a username and password then firebase is added with the android application project in the android studio. The data is stored in the JSON format along with providing an offline facility of data storage by enabling the disk persistence. After creating firebase project, the android application can be registered to it. Firebase also supports to focus on performance and implementation issues by repairing bugs precisely from its backend solution.However,our group have experienced many problems in order to connect to firebase.Eventually,we are able to display the users in the firebase , who are logged in the application.

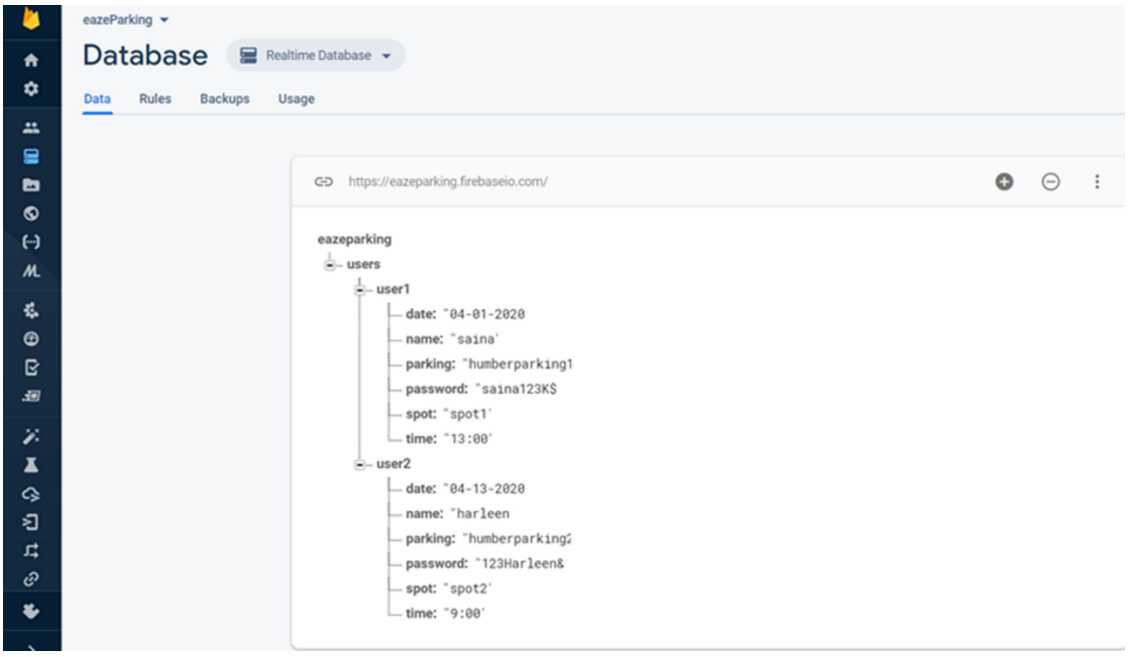


Figure 38 The information stored in the firebase is displayed

3.3.2 Enterprise Wireless Connectivity

The connection is established using the wireless network which is WIFI. The application uses the internet using the runtime permission feature in the android studio which is provided in the AndroidManifest.xml file. The connection can be established in order to access the internet.

<uses-permission android: name="android. permission. INTERNET"/>

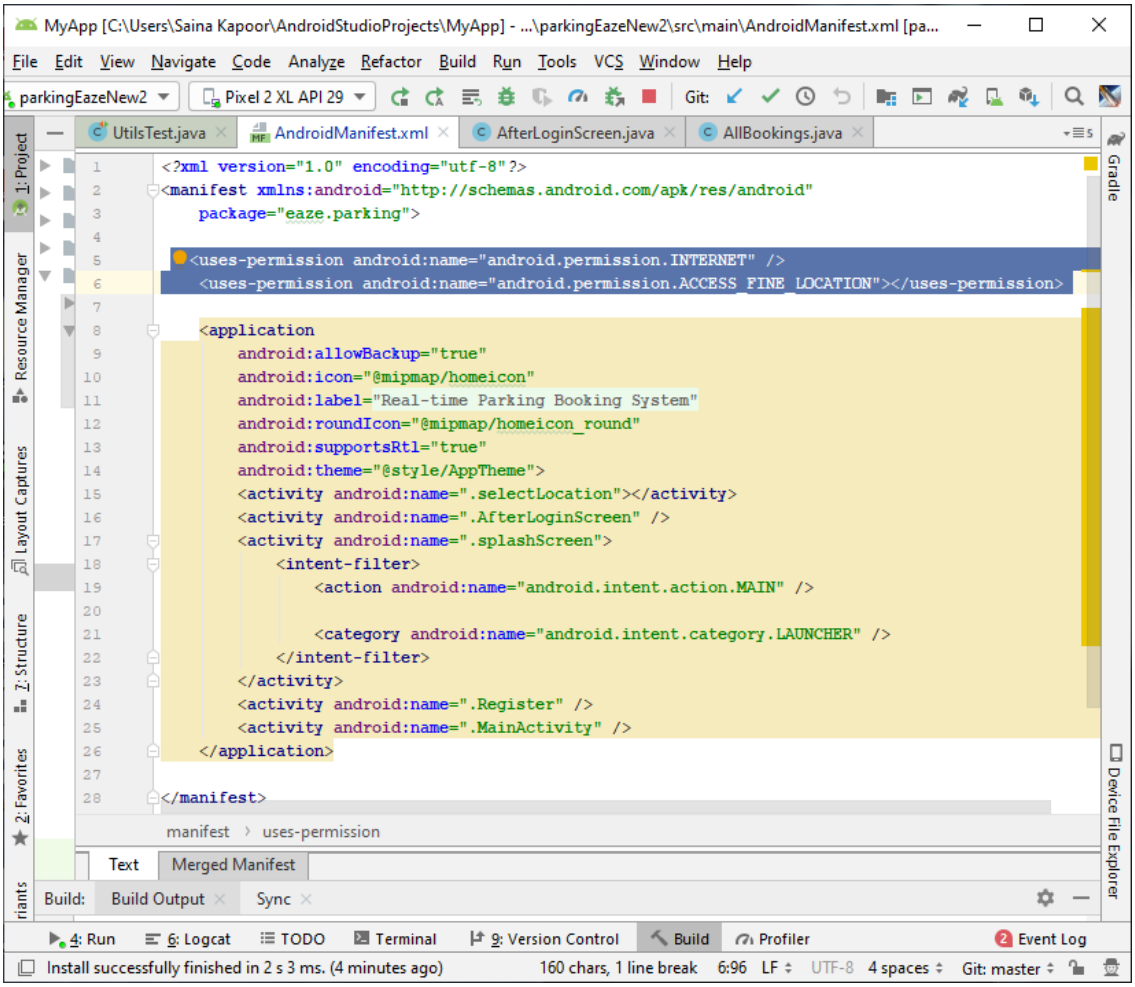


Figure 2 The AndroidManifest.xml file in android studio

3.3.3 Security

Consideration of security:

The security of the application used by the user is enhanced as shown in figure3 and Figure4 below. When the user opens the application, a dialog box appears which prompts a message whether the user wants to provide the location access to the application that follows user's privacy. It also assures and reminds them if they want to exit without the submission of any changes when the user clicks exit option. Hence,instead of instantly closing the app, a dialog box is prompted.

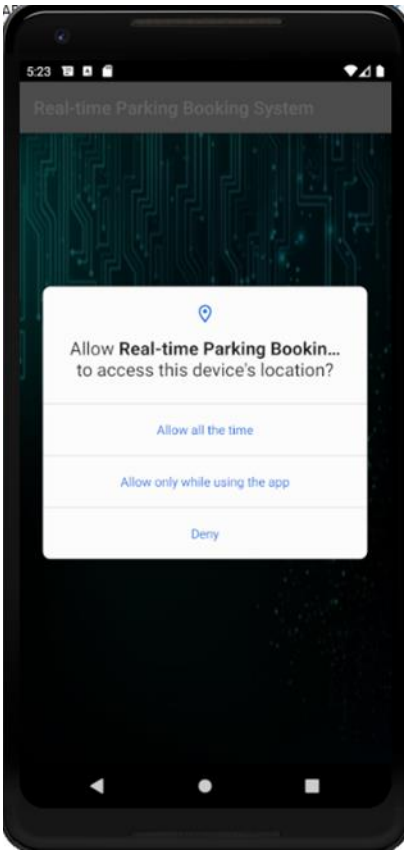


Figure 3 Asking permission to use the current location before seeing the nearby parking lots.

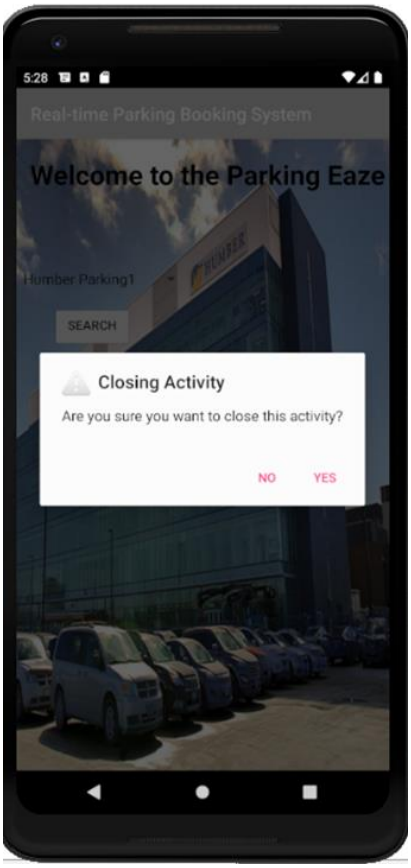


Figure 4 Asking the user if the user wants to leave without submitting their changes.

3.3.4 Testing

Unit Testing

In the figure below, the testing is done for software validation code of some cases. The test cases are implemented and tested step by step which are eventually passed. Performing unit tests is intended to be uncomplicated, generally the tests are written in the structure of functions that will decide whether a returned value equals the value you were anticipating when you entered the function in the software code.

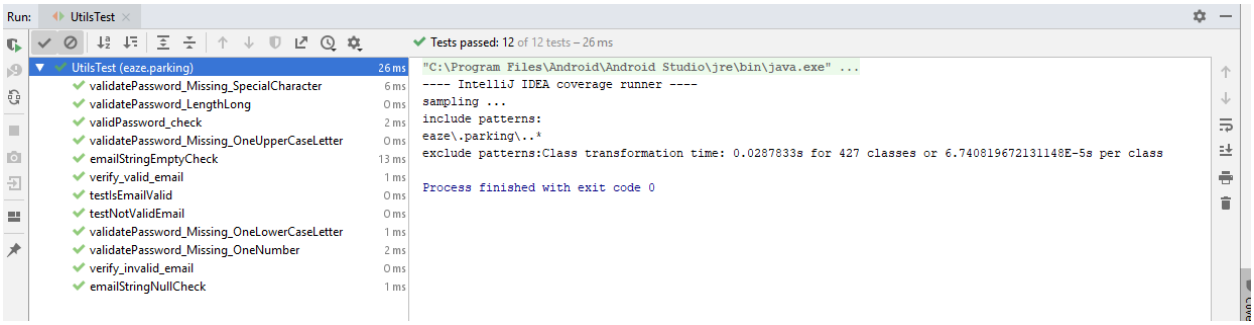


Figure 5 The test cases for the validation are tested and all of them are passed

Production Testing

The application connected to the firebase and once the application is complete and if it is eligible to be launched on the google play store, then the users can download it from there based on their requirement. The hardware consists of the sensors and effectors which comply differently according to the production testing.

- **UtilsTest.java contains testing functions from the code and the code can be found through this link**
- https://github.com/NavkiranKaur/ParkingEaze/blob/master/code%20files/UtilsTest.java

- **The main activity java file that contains firebase connection code can be found using this link below**
- https://github.com/NavkiranKaur/ParkingEaze/blob/master/code%20files/MainActivity.java

4.0 Results and Discussions

The ParkingEaze mobile application provides a reliable user interface and easy to navigate functionality in it. The app. first opens up with a splash screen with a time delay of 3sec to load and then provides the user to:

1. Register as a new user.

2. Login access with the same credentials.

3. Navigate between 2 different parking lots.

Once the user is logged in and selected the parking lot, it gives the user access to:

1. Select the date- The date user wants to park the car.

2. Start time – At what time the user wants to park the car in the parking slot.

3.Duration – For how long the user wants to park its car for.

After filling all the necessary details, on clicking the “Show available slots” 6 specific slots will be shown in green color meaning that the slots have not been booked. Once the user selects the slot, it turns the color to red and generated the toast message saying “Slot # booked!”. For more functionality, this app also provides navigation drawer with information such as “My bookings” where the user can see all the prior booked slots.

All the information and the selections made by the user are sent to the server-side database connected with the firebase. Here the admin can see the details of the user such as the name of the user, time and date the slot is booked, slot number.

5.0 Conclusions

The project for the parking is accomplished to an extent of showcasing it as a finished one. Further improvements can be done in the android application to make it more effective and user-friendly. Moreover, new features can be included in the design and functionality such as providing more options for the user. The experience of making the android application and hardware production provides the necessary skills which make us being capable of moving to advance steps for the project development. We are moving forward to fix the errors and troubleshooting the software application along with working on the database. Although hardware components are integrated and work accordingly yet further changes and modifications can be considered to grow the parking space for additional plans in the future. Overall, the project has allowed us to solve a real-world problem through this course as we learn vital technical skills and knowledge using both hardware and software.

Report

/1 Hardware present?

/1 Checklist truthful

/1 Valid Comments

/1 Results and Discussion (500 words)

/1 Conclusion

6.0 References

Gaurav Kumar Maurya, S. K. (2016). Smart Parking Control with LCD Display.

Kahonge, K. E. (2013). MOBILE PHONE –BASED PARKING SYSTEM.

Ms.S.Srikurinji, M. M. (2016, March). SMART PARKING SYSTEM ARCHITECTURE USING INFRARED DETECTOR. IJAICT.

The journal of design and technology. (2000).

7.0 Appendix

7.1 Firmware code

7.1.1 Modified code files

ultrasonic2_lcd.py

```
import lcddriver
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

display = lcddriver.lcd()

TRIG_1 = 23
ECHO_1 = 24
TRIG_2 = 22
ECHO_2 = 27

print ("Welcome!!")
display.lcd_display_string("Welcome to ", 1)
display.lcd_display_string("ParkingEaze", 2)
time.sleep(2)
#display.lcd_display_string("Parking slots", 1)
#display.lcd_display_string("Available :2", 2)
#time.sleep(2)
display.lcd_clear()
GPIO.setup(TRIG_1,GPIO.OUT)
GPIO.setup(ECHO_1,GPIO.IN)
GPIO.setup(TRIG_2,GPIO.OUT)
GPIO.setup(ECHO_2,GPIO.IN)
avgDistance_1=0
avgDistance_2=0
```

```
vacant = ['A','B']

try:
    while True:

        GPIO.output(TRIG_1, False)
        print ("Measuring Distance")

        time.sleep(2)

        GPIO.output(TRIG_1, True)
        time.sleep(0.00001)
        GPIO.output(TRIG_1, False)

        while GPIO.input(ECHO_1)==0:
          pulse_start = time.time()

        while GPIO.input(ECHO_1)==1:
          pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        distance_1 = pulse_duration * 17150

        distance_1 = round(distance_1, 2)

        avgDistance_1=avgDistance_1+distance_1

        avgDistance_1=avgDistance_1/5

#       if avgDistance_1 < 5:
#           display.lcd_display_string("Car is at A ", 1)
```

```
 #      else:
  #         display.lcd_display_string("Car is not A ", 1)

#for ultrasonic sensor 2
        GPIO.output(TRIG_2, False)
 #      print ("Measuring Distance")

        time.sleep(2)

        GPIO.output(TRIG_2, True)
        time.sleep(0.00001)
        GPIO.output(TRIG_2, False)

        while GPIO.input(ECHO_2)==0:
          pulse_start = time.time()

        while GPIO.input(ECHO_2)==1:
          pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        distance_2 = pulse_duration * 17150

        distance_2 = round(distance_2, 2)

        avgDistance_2=avgDistance_2+distance_2

        avgDistance_2=avgDistance_2/5

        if (avgDistance_1 < 5) and (avgDistance_2 < 5):
            display.lcd_display_string("Available 0 ", 1)
            time.sleep(2)
            display.lcd_clear()
```

```
    display.lcd_display_string("Welcome to ", 1)
    display.lcd_display_string("ParkingEaze", 2)
    time.sleep(2)
    display.lcd_clear()

elif (avgDistance_1 < 5) and (avgDistance_2 > 5):
    display.lcd_display_string("Available 1", 1)
    time.sleep(2)
    display.lcd_clear()
    display.lcd_display_string("Welcome to ", 1)
    display.lcd_display_string("ParkingEaze", 2)
    time.sleep(2)
    display.lcd_clear()

elif (avgDistance_1> 5) and (avgDistance_2 <5):
    display.lcd_display_string("Available 1 ", 1)
    time.sleep(2)
    display.lcd_clear()
    display.lcd_display_string("Welcome to ", 1)
    display.lcd_display_string("ParkingEaze", 2)
    time.sleep(2)
    display.lcd_clear()

else:
    display.lcd_display_string("Available 2", 1)
    time.sleep(2)
    display.lcd_clear()
    display.lcd_display_string("Welcome to ", 1)
    display.lcd_display_string("ParkingEaze", 2)
    time.sleep(2)
    display.lcd_clear()

#result = str(distance)+" cm"
```

```
#         display.lcd_disaply_string("Car has come", 2)
 #        display.lcd_display_string(result,2);

except KeyboardInterrupt: # If there is a KeyboardInterrupt (when you press ctrl+c), exit the program and cleanup
    print("Cleaning up!")
    GPIO.cleanup()
```

motor_ir.py

```
import RPi.GPIO as GPIO
from time import sleep
import time

DIR = 20   # Direction GPIO Pin
STEP = 21  # Step GPIO Pin
CW = 1     # Clockwise Rotation
CCW = 0    # Counterclockwise Rotation
SPR = 100  # Steps per Revolution (360 / 7.5)

GPIO.setmode(GPIO.BCM)
GPIO.setup(4,GPIO.IN)
GPIO.setup(10,GPIO.IN)
GPIO.setup(DIR, GPIO.OUT)
GPIO.setup(STEP, GPIO.OUT)
GPIO.output(DIR, CW)

MODE = (14, 15, 18)   # Microstep Resolution GPIO Pins
GPIO.setup(MODE, GPIO.OUT)
RESOLUTION = {'Full': (0, 0, 0),
                      'Half': (1, 0, 0),
```

```
        '1/4': (0, 1, 0),
        '1/8': (1, 1, 0),
        '1/16': (0, 0, 1),
        '1/32': (1, 0, 1)}
GPIO.output(MODE, RESOLUTION['1/16'])

while True:
                sensor = GPIO.input(4)
                sensor1 = GPIO.input(10)
                if (sensor ==0) or (sensor1==0):
                        print("car is here")
                        step_count = SPR * 8
                        delay = .0208 / 8
                        #step_count = SP
                        #delay = .0208
                        print("FORWARD")
                        GPIO.output(DIR, CW)
                        for x in range(step_count):
                                GPIO.output(STEP, GPIO.HIGH)
                                sleep(delay)

                                GPIO.output(STEP, GPIO.LOW)
                                sleep(delay)
                        print("REVERSE")
                        sleep(4)
                        GPIO.output(DIR, CCW)

                        for x in range(step_count):
                                GPIO.output(STEP, GPIO.HIGH)
                                sleep(delay)

                                GPIO.output(STEP, GPIO.LOW)
                                sleep(delay)
```

```
            print("DONE")

            sleep(1)

        elif (sensor ==1) or (sensor1 == 1):
            print("no car")
            sleep(1)
```

Demo

/1 Hardware present?

/3 Code runs concurrently for all sensors/effectors

/1 Project repository contains integrated code

Status

/1 Memo including updates

/1 Financial update

/1 Progress update

/1 Modified Code Files in Appendix

/1 Link to Complete Code in Repository

7.2 Application code

7.2.1 Modified code files in the appendix

MainActivity.java

```
//parkingEazeTeam
                package eaze.parking;
                import android.Manifest;
                import android.content.Context;
                import android.content.DialogInterface;
                import android.content.Intent;
                import android.content.pm.PackageManager;
                import android.location.Criteria;
                import android.location.LocationManager;
                import android.os.Bundle;
                import android.util.Log;
                import android.view.Menu;
                import android.view.MenuItem;
                import android.view.View;
                import android.widget.Button;
                import android.widget.EditText;
                import android.widget.Toast;

                import androidx.annotation.NonNull;
                import androidx.appcompat.app.AlertDialog;
                import androidx.appcompat.app.AppCompatActivity;
                import androidx.appcompat.widget.Toolbar;
                import androidx.core.app.ActivityCompat;
                import androidx.core.content.ContextCompat;

                import com.google.android.gms.tasks.OnCompleteListener;
                import com.google.android.gms.tasks.Task;
                import com.google.firebase.auth.AuthResult;
                import com.google.firebase.auth.FirebaseAuth;
                import com.google.firebase.auth.FirebaseUser;

                public class MainActivity extends AppCompatActivity {
                    EditText mTextUsername;
                    EditText mTextPassword;
                    Button mButtonLogin;
                    Button mButtonRegister;
                    private FirebaseAuth mAuth;
```

```
    private static final int REQUEST_CODE_ASK_PERMISSIONS = 300;
    LocationManager locationManager;
    Criteria criteria;
    private static final int REQUEST_LOCATION = 2;
    protected void onCreate(Bundle savedInstanceState) {
        //setTheme(R.style.AppTheme_NoActionBar);

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);


        findAllViewsfromLayout();
        handleLogin();

        locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        // Creating an empty criteria object
        criteria = new Criteria();

// Get the location from the given provider
        if (ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) ==
                PackageManager.PERMISSION_GRANTED
                || ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
                PackageManager.PERMISSION_GRANTED) {
            onResume();
        } else {

// request permission from the user
            // Check Permissions Now

            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {

            }

            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    REQUEST_LOCATION);

        }
    }
```

```
    private void findAllViewsfromLayout() {
        mTextUsername = (EditText) findViewById(R.id.edittext_username);
        mTextPassword = (EditText) findViewById(R.id.edittext_password);
        mButtonLogin = (Button) findViewById(R.id.button_login);
        mButtonRegister = (Button) findViewById(R.id.button_register);
        mButtonRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                createNewUser();
            }
        });
    }



    private void handleLogin(){
        // Initialize Firebase Auth
        mAuth = FirebaseAuth.getInstance();

        mButtonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loginUser(String.valueOf(mTextUsername.getText()),
String.valueOf(mTextPassword.getText()));
            }
        });




        /*signout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SignoutfromDatabase();
            }
        });*/
    }
```

```
    private void loginUser(String email, String password){


        if(email.length()==0 || password.length()==0)
        {
            Toast.makeText(getApplicationContext(), "Email & Password Can't be empty",
                    Toast.LENGTH_LONG).show();
            return;
        }
        // TODO: Login with Email and Password on Firebase.
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            // Sign in success, update UI with the signed-in user's information
                            // Log.d("MapleLeaf", "signInWithEmail:success");
                            FirebaseUser user = mAuth.getCurrentUser();
                            /*message.setText("User "+ user.getEmail() + " is now Logged In");
                            setButtonStatus(true);
                            gotoRead();*/
                            gotoMainReadingsPage();

                        } else {
                            // If sign in fails, display a message to the user.
                            Log.w("MapleLeaf", "signInWithEmail:failure", task.getException());
                            Toast.makeText(getApplicationContext(), "Success.",
                                    Toast.LENGTH_LONG).show();
                            Intent intent1 = new Intent(getApplicationContext(),
selectLocation.class);
                            startActivity(intent1);
                            finish();
                        }

                        // ...
                    }
                });


    }
    private void createNewUser(){
        Intent register = new Intent(getApplicationContext(), Register.class);
        startActivity(register);
```

```
        finish();

    }

    private void gotoMainReadingsPage() {
        // TODO : Start the read option After login
        Intent intent1 = new Intent(getApplicationContext(), AfterLoginScreen.class);
        startActivity(intent1);
        finish();
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
        switch (requestCode) {
            case REQUEST_CODE_ASK_PERMISSIONS:
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
// Permission Granted
                    onResume();
                } else {
// Permission Denied
                    finish();

                }
                break;
            default:
                super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }

    @Override
    public void onBackPressed() {
        new AlertDialog.Builder(this)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setTitle("Closing Activity")
                .setMessage("Are you sure you want to close this activity?")
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                        System.exit(0);
                    }

                })
```

```
                .setNegativeButton("No", null)
                .show();
        }



    }
```

Bookings.java

```
package eaze.parking;

import android.os.Bundle;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ListView;

import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.ArrayList;
import java.util.List;

import eaze.parking.AllBookingsAdapter;
import eaze.parking.Booking;
import eaze.parking.R;


public class MyBookings extends Fragment {
    private ListView mylistview;
    private AllBookingsAdapter myadapter;
    private ChildEventListener childEventListener;
    private DatabaseReference reference;
```

```
private FirebaseDatabase database;
private List<Booking> mybookings= new ArrayList<Booking>();
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    if (container != null) {
        container.removeAllViews();
    }
    View view= inflater.inflate(R.layout.fragment_my_bookings, container, false);
    database=FirebaseDatabase.getInstance();
    reference=database.getReference().child("Bookings");
    mylistview=(ListView) view.findViewById(R.id.mybookings);
    myadapter= new AllBookingsAdapter(getActivity(),mybookings,"user");
    mylistview.setAdapter(myadapter);

    childEventListener= new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            Booking addbooking = dataSnapshot.getValue(Booking.class);
            if(addbooking.getUserid().equals(FirebaseAuth.getInstance().getCurrentUser().getUid())) {
                mybookings.add(addbooking);
                myadapter.notifyDataSetChanged();
            }
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {
            Booking deletebooking= dataSnapshot.getValue(Booking.class);
            mybookings.remove(deletebooking);
            for(int i=0;i<mybookings.size();i++){
                if(mybookings.get(i).getBookingkey().equals(deletebooking.getBookingkey())){
                    mybookings.remove(i);
                }
            }
            myadapter.notifyDataSetChanged();
        }
```

```
            @Override
            public void onChildMoved(DataSnapshot dataSnapshot, String s) {

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        };
        reference.addChildEventListener(childEventListener);


        return view;
    }

}
```

Validation.java

```
//parkingEazeTeam
                package eaze.parking;

                import java.util.Calendar;
                import java.util.Date;
                import java.util.TimeZone;
                import java.util.regex.Matcher;
                import java.util.regex.Pattern;

                public class Validation {

                    private static final Pattern VALID_EMAIL_ADDRESS =
                            Pattern.compile("^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$",
                Pattern.CASE_INSENSITIVE);

                    private static final Pattern valid_password=
                            Pattern.compile("((?=.*\\\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%=:\\\\?]).{6,12})\n",
                Pattern.CASE_INSENSITIVE);
```

```
    public static boolean checkEmailForValidity(String email) {

        email = email.trim();

        Matcher matcher = VALID_EMAIL_ADDRESS.matcher(email);
        return matcher.find();

    }

    public static String emailStringChecker(String email) {
        return "";
    }

    public static boolean checkpasswordForValidity(String password) {

        password = password.trim();

        Matcher matcher2 = valid_password.matcher(password);
        return matcher2.find();

    }

    public static boolean validatePassword(String password) {
        return true;
    }

}
```

Parking1.java

```
//parkingEazeTea
m
package eaze.parking;

import android.graphics.Color;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.text.TextUtils;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;


import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

import eaze.parking.Booking;
import eaze.parking.R;

public class ParkingArea1 extends Fragment {
    private String location = "ParkingArea1";
    private TextView Heading;
    private EditText dd;
    private EditText mm;
    private EditText yyyy;
    private Spinner startTime;
    private Spinner startmins;
    private Spinner duration;
    private String start;
    private String durationhrs;
    private String minstart;
    private Button slot1;
    private Button slot2;
    private Button slot3;
    private SimpleDateFormat mdformat;
```

```
private Button slot4;
private String enterstartdate;
private Button slot5;
private Date enteredstart=null;
private Date enteredend=null;
private Button slot6;
private Button showslots;
private boolean slot1flag = false;
private boolean slot2flag = false;
private boolean slot3flag = false;
private boolean slot4flag = false;
private boolean slot5flag = false;
private boolean slot6flag = false;
private int d;
private int month;
private int year;
private int currenthours;
private int currentmins;
private FirebaseDatabase firebaseDatabase;
private DatabaseReference bookingReference;
Calendar calendar;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    if (container != null) {
        container.removeAllViews();
    }
    View view = inflater.inflate(R.layout.fragment_parkingarea, container, false);
   // if (getArguments() != null) {
        // location = getArguments().getString("location");
  //  }

          calendar = Calendar.getInstance();

  //  firebaseDatabase = FirebaseDatabase.getInstance();
  //  bookingReference = firebaseDatabase.getReference().child("Bookings");

    slot1 = (Button) view.findViewById(R.id.ParkingArea1_slot1);
    slot2 = (Button) view.findViewById(R.id.ParkingArea1_slot2);
    slot3 = (Button) view.findViewById(R.id.ParkingArea1_slot3);
    slot4 = (Button) view.findViewById(R.id.ParkingArea1_slot4);
```

```
        slot5 = (Button) view.findViewById(R.id.ParkingArea1_slot5);
        slot6 = (Button) view.findViewById(R.id.ParkingArea1_slot6);
        showslots = (Button) view.findViewById(R.id.show_slots);
        dd = (EditText) view.findViewById(R.id.ParkingArea1_start_dd);
        mm = (EditText) view.findViewById(R.id.ParkingArea1_start_mm);
        yyyy = (EditText) view.findViewById(R.id.ParkingArea1_start_yyyy);
        showslots.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (TextUtils.isEmpty(dd.getText().toString())) {
                    dd.setError("Enter date");
                    return;
                }
                if (TextUtils.isEmpty(mm.getText().toString())) {
                    mm.setError("Enter month");
                    return;
                }
                if (TextUtils.isEmpty(yyyy.getText().toString())) {
                    yyyy.setError("Enter year");
                    return;
                } else {
                    mdformat = new SimpleDateFormat("yyyy/MM/dd HH:mm");
                    Date currentdate=null;
                    try {
                        currentdate = mdformat.parse(mdformat.format(calendar.getTime()));
                    }
                    catch (Exception c){
                        c.printStackTrace();
                    }

enterstartdate=yyyy.getText().toString()+"/"+mm.getText().toString()+"/"+dd.getText().toStri
ng()
                            +" "+start+":"+minstart;


                    try {
                        enteredstart = mdformat.parse(enterstartdate);
                        Calendar cal = Calendar.getInstance(); // creates calendar
                        cal.setTime(enteredstart); // sets calendar time/date
                        cal.set(Calendar.HOUR_OF_DAY,
cal.get(Calendar.HOUR_OF_DAY)+Integer.parseInt(durationhrs));
                        System.out.println("checkff"+ mdformat.format(cal.getTime()));
                        enteredend=cal.getTime();
```

```
                }
                catch (Exception c){
                    c.printStackTrace();
                }
                System.out.println("checkff value="+currentdate.compareTo(enteredstart));
                if (currentdate.compareTo(enteredstart)==1) {
                    Toast.makeText(getActivity(), "Entered date is less than current date!",
Toast.LENGTH_SHORT).show();
                    dd.setText("");
                    mm.setText("");
                    yyyy.setText("");
                    return;
                }
            }
            slot1.setVisibility(View.VISIBLE);
            slot2.setVisibility(View.VISIBLE);
            slot3.setVisibility(View.VISIBLE);
            slot4.setVisibility(View.VISIBLE);
            slot5.setVisibility(View.VISIBLE);
            slot6.setVisibility(View.VISIBLE);
            slot6.setBackgroundColor(Color.GREEN);
            slot2.setBackgroundColor(Color.GREEN);
            slot3.setBackgroundColor(Color.GREEN);
            slot4.setBackgroundColor(Color.GREEN);
            slot5.setBackgroundColor(Color.GREEN);
            slot1.setBackgroundColor(Color.GREEN);
            slot1flag = false;
            slot2flag = false;
            slot3flag = false;
            slot4flag = false;
            slot5flag = false;
            slot6flag = false;
           // checkBookedSlots();
        }

    });
    slot1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //checkBookedSlots();
            if (slot1flag == false) {
                // String key= bookingReference.push().getKey();
```

```
                //  Booking booking = new
Booking(FirebaseAuth.getInstance().getCurrentUser().getUid(), "slot1",
location,enterstartdate,mdformat.format(enteredend),key);
                //  bookingReference.child(key).setValue(booking);
                  Toast.makeText(getActivity(),"Slot1
booked!",Toast.LENGTH_SHORT).show();
                  slot1.setBackgroundColor(Color.RED);
                }
                else {
                  Toast.makeText(getActivity(),"Slot1 is already
booked",Toast.LENGTH_SHORT).show();
                }
              }
            });
            slot2.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
               //   checkBookedSlots();
                if (slot2flag == false) {
                  //String key= bookingReference.push().getKey();
                 //   Booking booking = new
Booking(FirebaseAuth.getInstance().getCurrentUser().getUid(), "slot2",
location,enterstartdate,mdformat.format(enteredend),key);
                 //   bookingReference.child(key).setValue(booking);
                  Toast.makeText(getActivity(),"Slot2
booked!",Toast.LENGTH_SHORT).show();
                  slot2.setBackgroundColor(Color.RED);
                }
                else {
                  Toast.makeText(getActivity(),"Slot2 is already
booked",Toast.LENGTH_SHORT).show();
                }
              }
            });
            slot3.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
                // checkBookedSlots();
                if (slot3flag == false) {
                  //  String key= bookingReference.push().getKey();
                 //   Booking booking = new
Booking(FirebaseAuth.getInstance().getCurrentUser().getUid(), "slot3",
location,enterstartdate,mdformat.format(enteredend),key);
```

```
                //    bookingReference.child(key).setValue(booking);
                    Toast.makeText(getActivity(),"Slot3 booked",Toast.LENGTH_SHORT).show();
                    slot2.setBackgroundColor(Color.RED);
                }
                else {
                    Toast.makeText(getActivity(),"Slot3 is already
booked",Toast.LENGTH_SHORT).show();
                }
            }
        });
        slot4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
              // checkBookedSlots();
                if (slot4flag == false) {
//                    String key= bookingReference.push().getKey();
//                    Booking booking = new
Booking(FirebaseAuth.getInstance().getCurrentUser().getUid(), "slot4",
location,enterstartdate,mdformat.format(enteredend),key);
//                    bookingReference.child(key).setValue(booking);
                    Toast.makeText(getActivity(),"Slot4 booked",Toast.LENGTH_SHORT).show();
                    slot4.setBackgroundColor(Color.RED);
                }
                else {
                    Toast.makeText(getActivity(),"Slot4 is already
booked",Toast.LENGTH_SHORT).show();
                }
            }
        });
        slot5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
              // checkBookedSlots();
                if (slot5flag == false) {
//                    String key= bookingReference.push().getKey();
//                    Booking booking = new
Booking(FirebaseAuth.getInstance().getCurrentUser().getUid(), "slot5",
location,enterstartdate,mdformat.format(enteredend),key);
//                    bookingReference.child(key).setValue(booking);
                    Toast.makeText(getActivity(),"Slot5 booked",Toast.LENGTH_SHORT).show();
                    slot5.setBackgroundColor(Color.RED);
                }
                else {
```

```
                Toast.makeText(getActivity(),"Slot5 is already
booked",Toast.LENGTH_SHORT).show();
            }
        }
    });
    slot6.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //checkBookedSlots();
            if (slot6flag == false) {
//                String key= bookingReference.push().getKey();
//                Booking booking = new
Booking(FirebaseAuth.getInstance().getCurrentUser().getUid(), "slot6",
location,enterstartdate,mdformat.format(enteredend),key);
//                bookingReference.child(key).setValue(booking);
                Toast.makeText(getActivity(),"Slot6 booked",Toast.LENGTH_SHORT).show();
                slot6.setBackgroundColor(Color.RED);
            }
            else {
                Toast.makeText(getActivity(),"Slot6 is already
booked",Toast.LENGTH_SHORT).show();
            }
        }
    });
    startTime = (Spinner) view.findViewById(R.id.ParkingArea1_start_time);
    startmins = (Spinner) view.findViewById(R.id.ParkingArea1_start_mins);
    duration = (Spinner) view.findViewById(R.id.duration);
    ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(getActivity(), R.array.time,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    ArrayAdapter<CharSequence> adaptermins =
ArrayAdapter.createFromResource(getActivity(), R.array.min,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    ArrayAdapter<CharSequence> adapterduration =
ArrayAdapter.createFromResource(getActivity(), R.array.durationhrs,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    startmins.setAdapter(adaptermins);
```

```
        startTime.setAdapter(adapter);
        duration.setAdapter(adapterduration);
        start = startTime.getSelectedItem().toString();
        minstart = startmins.getSelectedItem().toString();
        durationhrs = duration.getSelectedItem().toString();
        startTime.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                start = startTime.getItemAtPosition(i).toString();
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });

        startmins.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                minstart = startmins.getItemAtPosition(i).toString();
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });
        duration.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                durationhrs = duration.getItemAtPosition(i).toString();
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });
        return view;
    }
    public void checkBookedSlots(){
        Query query = bookingReference;
```

```
        query.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Booking checkbook = snapshot.getValue(Booking.class);
                    if (checkbook.getLocation().equals(location)) {
                        checkSlots(checkbook);
                    }
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
    public void checkSlots(Booking dbBook) {
        Date dbstartdate=null;
        Date dbenddate=null;
        try {
            dbstartdate = mdformat.parse(dbBook.getStartdate());
            dbenddate=mdformat.parse(dbBook.getEnddate());
        }
        catch (Exception e){
            e.printStackTrace();
        }
////       int dbstart = Integer.parseInt(dbBook.getStarttime());
////        int dbend= Integer.parseInt(dbBook.getDuration())+dbstart;
////        int end = Integer.parseInt(durationhrs)+Integer.parseInt(start);
////        if (dbBook.getDate().equals(dd.getText().toString()) &&
dbBook.getMonth().equals(mm.getText().toString()) &&
dbBook.getYear().equals(yyyy.getText().toString())) {
////            System.out.println("checkff dbstart="+dbstart+" dbend=" +dbend+"
start="+Integer.parseInt(start)+" end="+end);
////            if (dbstart > Integer.parseInt(start) && dbstart < end) {
////                System.out.println("checkff slot="+dbBook.getSlotnum());
////               changeSlotColor(dbBook.getSlotnum());
////            }
////            else if(dbstart<Integer.parseInt(start) && dbend>Integer.parseInt(start)){
////                changeSlotColor(dbBook.getSlotnum());
////            }
////            else if(dbstart>Integer.parseInt(start) && dbend<end){
```

```
////                changeSlotColor(dbBook.getSlotnum());
////            }
////            else if(dbstart<Integer.parseInt(start) && dbend>end){
////                changeSlotColor(dbBook.getSlotnum());
////            }
////            else if(dbstart==Integer.parseInt(start)){
////                changeSlotColor(dbBook.getSlotnum());
////            }
////            else if(dbend==Integer.parseInt(start)){
////                if(Integer.parseInt(dbBook.getStartmins())>Integer.parseInt(minstart)){
////                    changeSlotColor(dbBook.getSlotnum());
////                }
////            }
////            else if(dbstart==end){
////              if(Integer.parseInt(minstart)>Integer.parseInt(dbBook.getStartmins())){
////                    changeSlotColor(dbBook.getSlotnum());
////                }
////            }
////        }
        if(dbstartdate.compareTo(enteredstart)==0){
            changeSlotColor(dbBook.getSlotnum());
        }
        else if(dbstartdate.compareTo(enteredstart)==1 &&
dbstartdate.compareTo(enteredend)==-1){
            changeSlotColor(dbBook.getSlotnum());
        }
        else if(dbstartdate.compareTo(enteredstart)==-1 &&
dbenddate.compareTo(enteredstart)==1){
            changeSlotColor(dbBook.getSlotnum());
        }
        else if(dbstartdate.compareTo(enteredstart)==1 &&
dbenddate.compareTo(enteredend)==-1){
            changeSlotColor(dbBook.getSlotnum());
        }
}
    public void changeSlotColor(String slotnum){
        if (slotnum.equals("slot1")) {
            slot1.setBackgroundColor(Color.RED);
            slot1flag = true;
        }
        else if (slotnum.equals("slot2")) {
            slot2.setBackgroundColor(Color.RED);
```

```
            slot2flag = true;
        }
        else if (slotnum.equals("slot3")) {
            slot3.setBackgroundColor(Color.RED);
            slot3flag = true;
        }
        else if (slotnum.equals("slot4")) {
            slot4.setBackgroundColor(Color.RED);
            slot4flag = true;

        }
        else if (slotnum.equals("slot5")) {
            slot5.setBackgroundColor(Color.RED);
            slot5flag = true;
        }
        else if (slotnum.equals("slot6")) {
            slot6.setBackgroundColor(Color.RED);
            slot6flag = true;
        }
        else {
            slot6.setBackgroundColor(Color.GREEN);
            slot2.setBackgroundColor(Color.GREEN);
            slot3.setBackgroundColor(Color.GREEN);
            slot4.setBackgroundColor(Color.GREEN);
            slot5.setBackgroundColor(Color.GREEN);
            slot1.setBackgroundColor(Color.GREEN);

        }
    }
}
```

Link to code files in the repository

https://github.com/NavkiranKaur/ParkingEaze/tree/master/code%20files

Grading for this milestone:

/1 Completed: Survey regarding substantial completion of this term

/1 Participated: Online session

/2 Login activity

/2 Data visualization activity

/2 Action control activity

/1 Modified Code Files in Appendix

/1 Link to Complete Code in Repository

Login Activity

The activity that requires the user to login with the username and the password but before that when the user click on the homescreen icon, it will be followed by splash screen.

Splash screen

When the user clicks on the homescreen of the application, the splash screen will appear for the duration of 3 seconds followed by a login screen that will ask user to login.

Figure 1 login screen view of the application

Figure 2 splash screen view of the application

Then comes the login screen where the user will have ability to enter the username and the password in order to login the application. It also has the option to register which will let the user for registration if the user is not registered yet.

Action Control Activity

After the user is logged in the mobile application, the location for the parking the vehicle are to be chosen from three options which are Humber Parking1, Humber Parking2 and Humber Parking3(Figure 3). The dialog box is displayed which shows these choices to user. When the user clicks on one of the options then a new screen appears which asks the user to choose an empty parking spot(Figure4). The slot turns green to red (Figure5)after the user booked it and therefore the spot to park the vehicle has been booked by the customer through the android application. Also, when the user opens the mobile application, it displays a runtime permission asking the user to provide the location access to the android application.

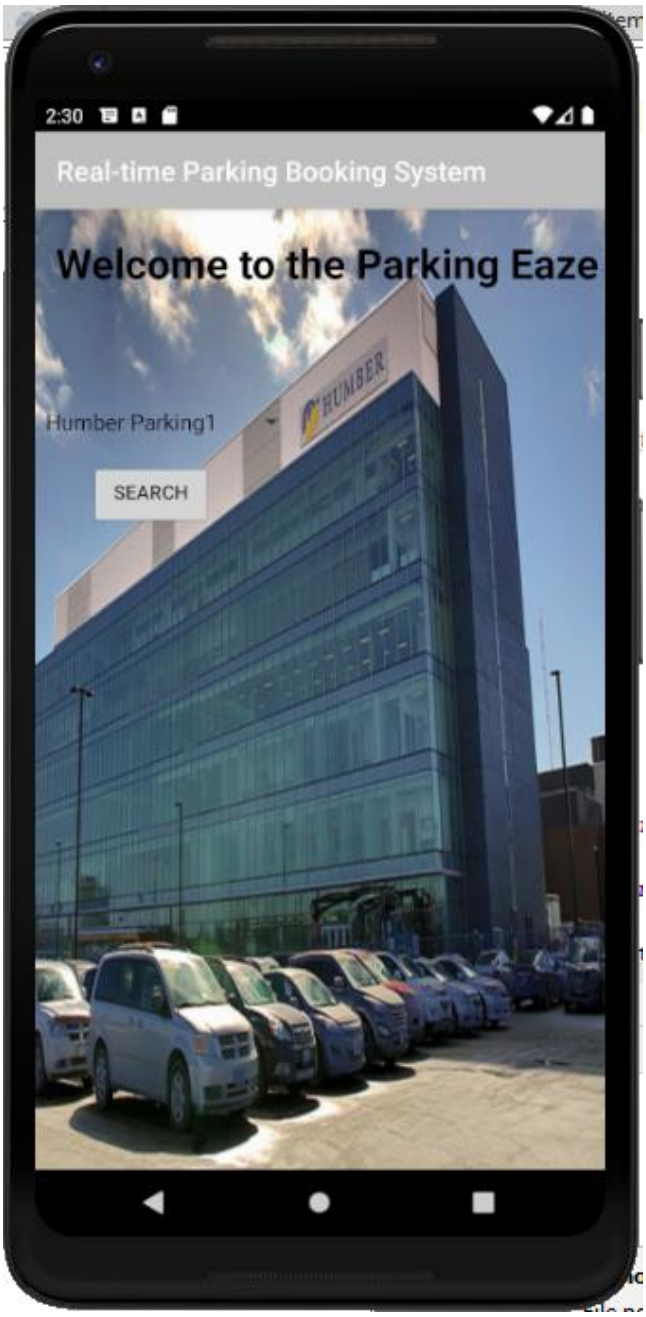


Figure 3 ;user asked to choose the parking location

Data visualization activity

The user enters the date, time and for how long the vehicle is to be parked.There are 6 slots for the user to choose from.Once,the slot is reserved,it turns red from green .

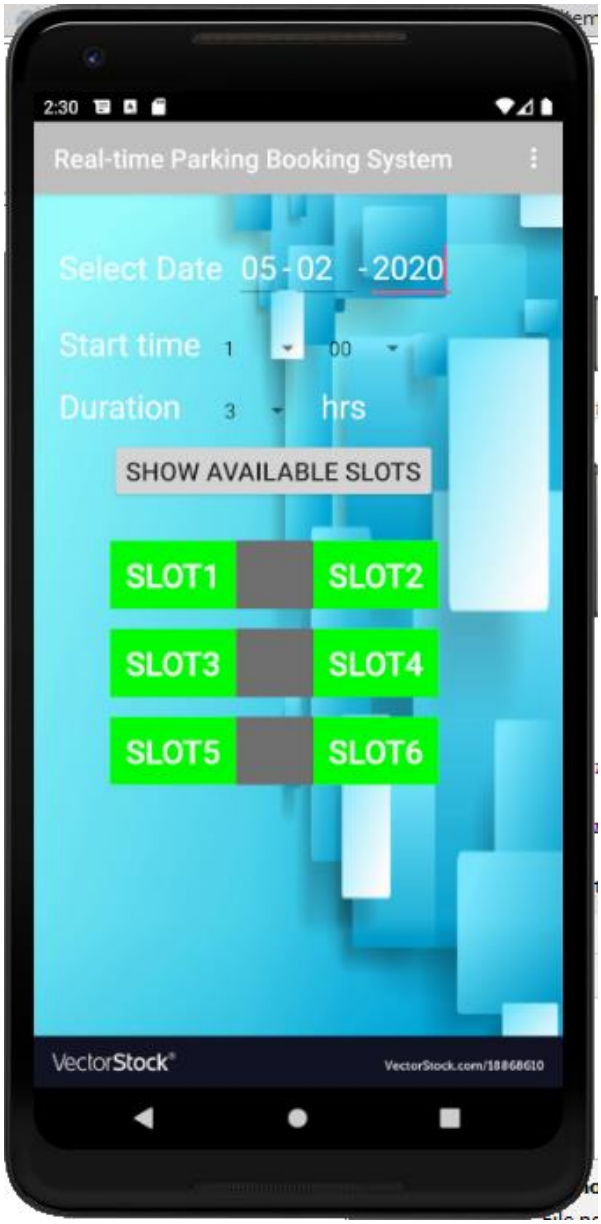


Figure 4 the parking slots are shown which are empty

Figure 5 the user booked the parking slot which turns red

It displays a ‘slot1booked’ message for around two or three seconds such as when slot1 is booked at the bottom of the screen. This is shown using the toast activity in android studio.

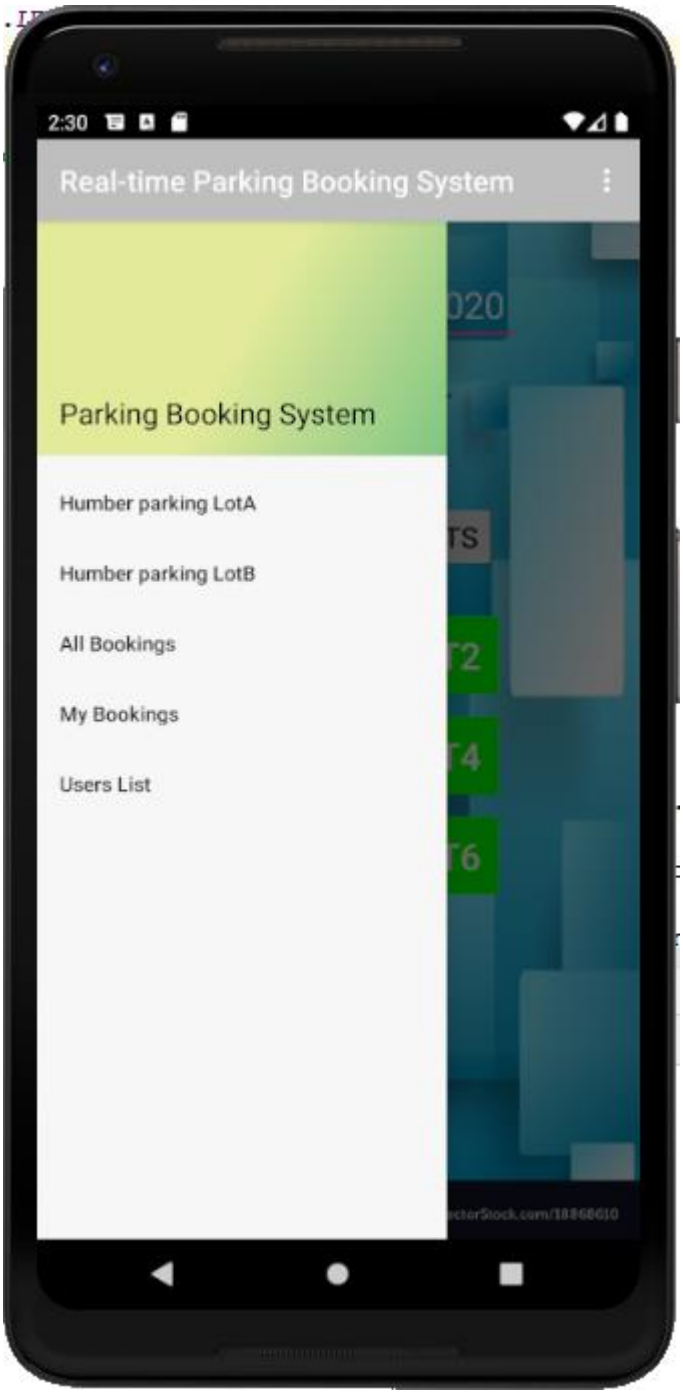


Figure 39 the navigation drawer layout has been implemented

The above figure shows the navigation drawer layout feature used to display options for the user to choose from.

Link to complete code files in repository.

https://github.com/NavkiranKaur/ParkingEaze/tree/master/code%20files