PROJECT REPORT

on

IOT BASED CAR PARKING MANAGEMENT SYSTEM

Submitted by

ALLEN JOHN (20318015)

in partial fulfillment of requirement for the award of the degree

of

BACHELOR OF TECHNOLOGY
in
ELECTRONICS AND COMMUNICATION

DIVISION OF ELECTRONICS ENGINEERING
SCHOOL OF ENGINEERING
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI-682022
MAY 2022

DIVISION OF ELECTRONICS ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI-682022

CERTIFICATE

Certified that the report entitled ***"IoT Based Car Parking Management System"*** *is a Bonafide work of*

ALLEN JOHN (20318015)

towards the partial fulfilment for the award of the degree of B.Tech in Electronics and Communication of Cochin University of Science and Technology, Kochi-682022.

Project Guide **Head of the Division**

Dr. Mythili P **Dr. Anju Pradeep**

ABSTRACT

This project titled “IoT based car parking management system” is a modest effort to solve the challenges that are present in traditional parking lots by applying technology and by proposing improvements to the system. The project is a functioning prototype of a parking lot which is technology enabled and upon implementing it to a real-life system, has the potential to resolve many of the common driver’s obstacles when it comes to parking their vehicle in a public compound.

The main motivation of this project is to reduce the traffic congestion inside a parking area, ease the process of parking, facilitate the parking fee transaction, assist the parking lot administrator with daily traffic information and revenue generated, reduce carjacking and to save the environment.

The primary technology used for this project is the Internet of Things due to its low cost, efficiency and interconnectivity which ensures a flawless operation. By using Internet of Things, the system communicates with the administrator of the parking lot as well as the driver thereby making the parking lot “smart”.

This project enables a car driver to know the free slots available in a parking lot before entering the driveway of the parking space using a smartphone application. Once the driver is in the driveway, a boom barrier operated by the system controls the flow of the vehicles and their entry is determined by the free slots available inside the space as well as the congestion level. The driver additionally requires an account to gain entry and their entry can be denied if their balance is too low. The process of transaction is done without any contact between the driver. Once inside, the driver is informed and made aware of the nearest free spot available using a messaging app along with the directions to the nearest slot. When attempting to leave, the system also lets the driver know that the driveway is blocked/congested. A bill is produced to the driver informing them of their time spent inside and their total balance remaining using the same app. The information regarding the entry and exit of each driver is recorded and stored using PHP and MySQL.

The overall attempt has been predicted to increase the efficiency of a parking lot by saving time, fuel, paper and money. It has successfully automated the transaction process thereby reducing the labour cost to the company and security of the vehicle. The implementation of this project in real is expected to bring a big improvement in user well-being as well as to the environment.

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	Internet of Things
Gate	Boom barrier / Boom gate
MCU	Node MCU ESP8266
Motor	Servo Motor
Arduino	Arduino MEGA 2560
LCD	Liquid Crystal Display
IR	Infra-Red
RFID	Radio Frequency Identification
UID	Unique Identifier
AI	Artificial Intelligence
PHP	Hypertext Preprocessor
SQL	Structured Query Language
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
IP	Internet Protocol
COVID-19	Coronavirus Disease 19
TX / RX	Transmitting / Receiving Pins

LIST OF TABLES

CONTENTS

1. INTRODUCTION

1.1.OVERVIEW

As the human population is increasing rapidly, so is the case with the number of vehicles on the road. The rise in per capita income along with the prestige of owning a car has caused the necessity of owning a vehicle especially in the cities to rise. The Civic governments across the country have realised this in the past, thus have spent considerable amounts of funds for road infrastructure and expanding highways. However, no care was given to the parking spaces which are as important as the roads, which the vehicles use. The roadside has been used as temporary parking space by millions of drivers as they are easy to park and they can easily walk to their destination. However, this diminishes the area for other vehicles to pass by as well for the pedestrians and increases the chance of accidents at the same time. This has led to a growing number of institutions and commercial establishments to introduce a system of Pay and Park, where each driver will drive their vehicle into a compounded area owned and maintained by the establishment.

The Pay and Park offers many benefits over parking at the roadside such as the vehicles being parked in an organized manner, providing security, reduced chance of accidents and usually ensuring guaranteed parking once inside a Pay and Park. The system is more beneficial to the owner of the establishment as it provides an additional revenue and removes the vehicular traffic near their compound, allowing smooth traffic to flow through the road, thus ensuring a healthy footfall at their establishment. This system has since become mandatory for every commercial establishment in a city and the situation has led to car parking difficulties because the space of parking is not sufficient with the growth in the number of vehicles.

Parking lots are the systems which have received minuscule improvements and advancements in their operation over the years. In contrast to buildings and city developments which are being modernised and upgraded every period, the parking lots are still operated and maintained in conventional manner.

One of the biggest issues with the traditional parking lots is the manner in which parking fees are collected. A person is employed by the owner to collect fare from the driver and to hand over the ticket. The person has to note down the licence plate number of each vehicle, has to note down the time of entry and has to calculate the amount the driver has

to pay while exiting. The driver also has to spend time to do the transaction and there can be instances where the driver doesn't have change. All of this is time consuming. There is also an added risk of the driver or the employee to catch infectious disease such as COVID-19.

Parking space is of considerable importance, especially in large commercial areas such as malls, hospitals, city centres and residential societies that demand extensive parking space. When a driver enters the drive-way of a parking lot, they are worried about the availability of parking space and the congestion inside. They need to drive from one level to another level to see the available parking space, especially at the peak hours or during weekends and public holidays. As a result, drivers may spend a lot of time and waste energy by turning around in the car park without guidance. This is a cause of concern especially for the users who have come to work in offices or for shopping in malls having to begin their day in the parking lot getting annoyed by loud car honks and breathing polluted air. Repetition of such inconvenience leads to reduced quality of life and increase in their stress levels and frustrations. The rate of vehicles entering and exiting keeps on fluctuating, causing few of the parking spots to be left entirely vacant. Multi-storied parking spaces require commodities such as illumination, air ventilation and manpower for handling the traffic at all times. This raises the overall cost of electricity. Since the current parking system offers details only at the entrance of the parking ground, people are still unable to find a parking space.

The traditional parking lots which run unprofitably and inefficiently thus begs for automation.

1.2. INTERNET OF THINGS

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A thing in the internet of things can be a person that interacts with the system or a group of sensors that send their results to the system over the network. Organizations in a variety of industries are adopting IoT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business.

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analysed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. By means of low-cost computing, the cloud, big data, analytics, AI, Machine Learning and mobile technologies, physical things can share and collect data with minimal human intervention. In this hyperconnected world, digital systems can record, monitor, and adjust each interaction between connected things. The physical world meets the digital world—and they cooperate.

The internet of things helps people live and work smarter, as well as gain complete control over their lives. IoT is essential to business as it provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations.

IoT enables companies to automate processes and reduce labour costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver goods, as well as offering transparency into customer transactions.

As such, IoT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive.

The internet of things offers several benefits to organizations. Some benefits are industry-specific, and some are applicable across multiple industries. Some of the common benefits of IoT enable businesses to:

- monitor their overall business processes;
- improve the customer experience (CX);
- save time and money;
- enhance employee productivity;
- integrate and adapt business models;
- make better business decisions; and
- generate more revenue.

IoT encourages companies to rethink the ways they approach their businesses and gives them the tools to improve their business strategies.

Generally, IoT is most abundant in manufacturing, transportation and utility organizations, making use of sensors and other IoT devices; however, it has also found use cases for organizations within the agriculture, infrastructure and home automation industries, leading some organizations toward digital transformation.

The ability to monitor operations surrounding infrastructure is also a factor that IoT can help with. Sensors, for example, could be used to monitor events or changes within structural buildings, bridges and other infrastructure. This brings benefits with it, such as cost saving, saved time, quality-of-life workflow changes and paperless workflow.

A home automation business can utilize IoT to monitor and manipulate mechanical and electrical systems in a building. On a broader scale, smart cities can help citizens reduce waste and energy consumption.

IoT touches every industry, including business within healthcare, finance, retail and manufacturing.

Some of the advantages of IoT include the following:

- ability to access information from anywhere at any time on any device;
- improved communication between connected electronic devices;
- transferring data packets over a connected network saving time and money; and
- automating tasks helping to improve the quality of a business's services and reducing the need for human intervention. [1]

1.3. OBJECTIVE

By extrapolating the benefits offered by the Internet of Things, we can automate the traditional parking lots to a smarter, efficient and profitable system with greater security and accountability. The shortcomings of a traditional parking lot thus need to be analysed and their solutions which can be made possible by means of IoT are to be discussed. A comparison between issues of a traditional parking lot and the possible solutions are elucidated below.

ISSUES	SOLUTION
The driver is anxious if the parking lot has free slots.	By providing an app which shows the number of free slots available at the time.
The driver wants to know the free parking slots after reaching the driveway. (The driver was driving and the smartphone was not in use / The smartphone is unavailable at the time of operation)	By providing an LCD display at the entrance of the driveway that shows the number of free slots available.
The driver wants to know the layout of the parking lot along with the vacancy information of each slot.	By including a layout feature in the app that displays all the slot information.
The driver wants to know the layout of the parking lot along with the vacancy information of each slot after reaching the driveway without a smartphone.	By providing an LCD display at the entrance of the driveway that shows the layout of the vacancy information available slots.
The driver is made to stop and wait at the entrance for the employee to note its licence plate number.	An RFID card embedded inside the car that can collect the Unique ID of the car in order to identify it.
The driver is handed a bill which is made of paper and ink thus causing a toll on the environment.	The driver instead gets a message on their messaging app, welcoming them with appropriate information instead.

ISSUES	SOLUTIONS
The driver can drive inside the parking lot discreetly bypassing the requirements needed for the entry.	A boom barrier is installed that prevents the driver from entering unless they have marked their presence in the system and satisfies the requirements.
The driver, once inside, is stuck as all the parking slots are taken.	The boom barrier will open only if there is a vacant slot for the driver.
The driver, once inside, is stuck as there are vehicles not parked properly and/or numerous vehicles are trying to enter/exit at the same time, thus creating a chaotic atmosphere.	The boom barrier will open only if the congestion levels inside have diminished.
The driver is stuck inside as they aren't sure where to park.	The app-based messaging service details the directions for the nearest available slot with word-based directions.
Once the vehicle is parked, the fare is calculated by means of hour spent, hence the charge is inexact and thus more money is charged from the driver.	When the system is computerized, the charge calculated is by means of minutes, thus increasing the savings for the user due to calculation based on minutes being more accurate.
Once the driver is exiting, they are made to do transactions by lifting their windows, exchange bank notes, deal with shortage of changes, receive a bill and potentially catch infectious disease if the employee has it.	The RFID scanner detects the car by scanning its embedded RFID card, makes changes to the database of the driver and calculates the fare based on time spent inside. Their account balance is deducted automatically. A bill is produced and sent to their messaging application.
The driver is uninformed about the time they spent inside and the entry and exit time.	The messaging app sends the time spent inside along with the bill. Drivers can

	easily know the entry and exit time by checking the time of messages.
The owner of the parking lot has to pay regular salary to the employee thus the parking lot results in extremely unprofitable along with inoperability of the lot when the employee is unavailable under working hours.	The automated parking lot is highly frugal during operation and requires only cheap internet connection and very less power.
Difficulty faced by the administrator to determine the number of cars entering the parking lot per day and the revenue generated.	Providing a neat website developed using PHP that stores the details of entry and exit of each driver and the revenue generated in total.
The driver wants to track their vehicle if it got carjacked.	Upon implementing the system to every parking lot in the city, they collect information of the car, its entry and exit times thus easing the process of investigation.

Thus, a smart parking lot is expected to deliver the following solutions to the traditional parking lot.

1.4. DESCRIPTION

The project is a prototype of a traditional parking lot with all the features and technologies loaded in so that upon practical implementation, it could resolve all the issues discussed in the objectives.

The project consists of a big drawing board which is decorated and modelled on a real-life parking lot. The board is layered with chart papers with the pathways, slots and area for the hardware clearly marked. The project consists of 6 toy cars that prototypes real cars for the purpose of demonstration. They are fitted with RFID cards at their undercarriage. The board has 6 slots, meant for the parking of the 6 toy cars. There is a broad driveway for the car to navigate inside the parking lot. The wires of all the things are concealed using the chart paper.

The slots are fitted with IR sensors which sense the presence of the car. There are 2 IR sensors placed in-front and at the back of the boom gate as well. The boom gate or boom barrier is a bar/pole pivoted onto the shaft of the servo motor. The front of the drive-way consists of a large LCD display. The RFID sensor is placed halfway across the pole in such a way that it can scan an approaching car in either direction. There is space dedicated for the hardware that include Arduino Mega, Node MCU, Arduino UNO and 4 female headers that power all the sensors, motor and display. The project requires the internet in order to function, hence an active internet connection provider (Smartphone with mobile hotspot capability, Wi-Fi Modem) is placed close to the project. A Computer that is connected to the same Wi-Fi is also required in-order to host the XAMPP server for the functioning of the database.

The project requires unlimited supply of power during runtime since a power interruption can cause the parking lot to not work. Hence the Hardwares are powered using a power-bank which is always kept at charging.

1.5. LAYOUT

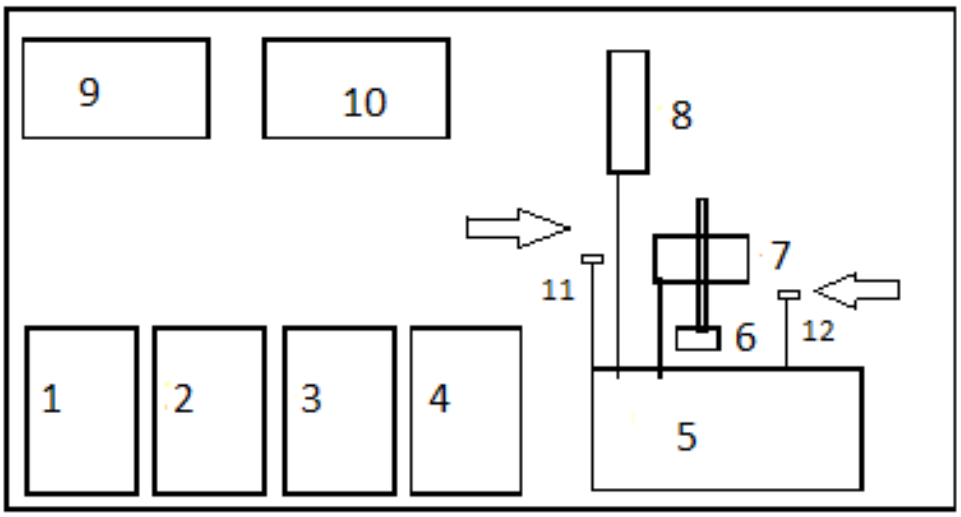


Figure 1

1. SLOT 4
2. SLOT 3
3. SLOT 2
4. SLOT 1
5. HARDWARES (ARDUINO MEGA + UNO + NODE-MCU + WIRING BOARD)
6. SERVO MOTOR
7. RFID SCANNER
8. LCD DISPLAY
9. SLOT 5
10. SLOT 6
11. EXIT IR SENSOR
12. ENTRY IR SENSOR

The slots 1,2,3,4,5,6 are further fitted with their respective IR sensors. The arrows represent the direction of vehicle movements towards and away from the driveway.

2. SURVEY OF TRADITIONAL PARKING LOTS

In-order to ensure effectiveness and profitability of the implementation of this project, proper survey of the pre-existing parking lots is to be done. Hence a survey was conducted to analyse the parameters required for profit generation and utilisation of space and resources.

Area

For this survey, a small commercial parking ground in a downtown area of Kochi was selected. The parking area is 3.5 cents (141 sq.-m). An average car is estimated to require 60 to 80 sq.-feet (5.5 to 7.5 sq.-m) surface area for convenient parking. The parking lot had sufficient space to house 8 cars at a time. The parking lot also contained a small room for the employee which had an area of 4*3 meters (~13 sq.-m).

The space required for navigation was 68 sq.-m. A minimum space is left out for the vehicle to navigate across the parking lot and to enter/exit the parking lot. This value depends on the number of slots and in the case of 8 slots, a minimum space of 60 sq.-m was left for navigation.

From the observations, it was concluded that by the elimination of the room for the employee, 13 sq.-m of space can be saved by which 2 additional slots can be added to the existing lot, thereby increasing the number of slots to 10.

Cost saved by removing the room for the employee

Cost of land per sq.-feet in the area = 2984/sq.ft

Converting it into sq.-meter = ₹31928/sq.-m

Land area used for the example = 141 sq.-m

Cost of the equivalent land = 45,01,848 ₹

After removing the room for the employee, total area = 128 sq.-m

Cost after the deduction = 40,89,784 ₹

Cost saved = 4,12,064 ₹

Thus, the cost savings become more profound with increasing land prices.

Monthly expenditure

Salary of a part-time employee per month (morning shift) = 13000 ₹

Salary of a part-time employee per month (night shift) = 14000 ₹

Cost of materials (i.e., paper, pen, billing machine) ~ 1000

Expenditure for a month = 28000 ₹

Expenditure of the smart parking lot

Initial investment and setup cost ~ 20000 ₹ (approx.)

Monthly expenditure = 500 ₹ (For Wi-Fi)

600 ₹ (For Server)

375 ₹ (For power consumption ~ 100Watts and 5₹ per unit)

~1475₹

Hence, the practical implementation of this project can prove to be highly cost effective and can boost the revenue from the parking lot and at the same time, reduce the fare for the drivers thereby benefiting both the owner and the drivers at the same time.

3. LIST OF COMPONENTS

List of each component and their costs are aggregated here

Sl. No.	Component name	Number	Cost in ₹
1	Arduino Mega	1	1250
2	Node MCU ESP 8266	1	600
3	Arduino UNO	1	500
4	RFID Sensor	1	250
5	LCD Display	1	900
6	IR Sensors	8	720
7	RFID Cards	7	420
8	Servo Motor	1	270
9	I2C Module	1	220
8	Toy Cars	6	480
9	Board	1	200
10	Wires, Headers & LEDs	60	180

Table 2

Total cost incurred = 5990 ₹

The components used in the project are described briefly

3. Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on the computer, used to write and upload computer code to the physical board.

The Arduino platform has been popular since it doesn't require a separate piece of hardware (programmer) in order to load new code onto the board like other electronic programmable boards, rather uses a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package. [2]

3.1. Arduino Mega 2560

Figure 2

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. [3]

SPECIFICATIONS

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16 DC
Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM 8 KB EEPROM	4 KB
Clock Speed	16 MHz

3.2.Node MCU ESP8266

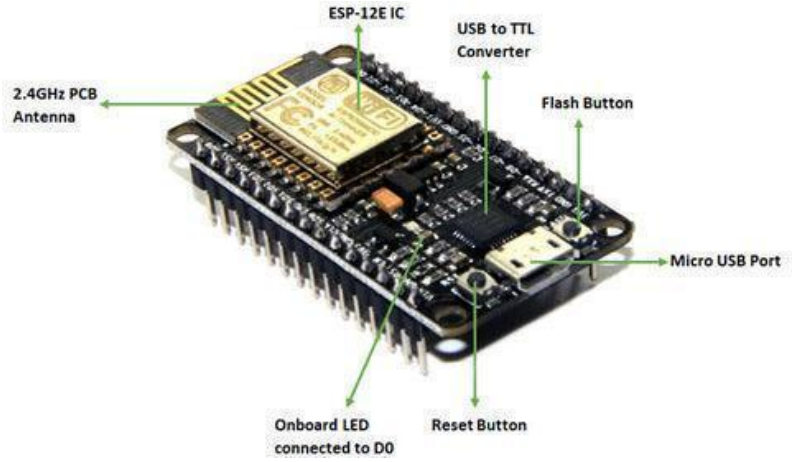


Figure 3

NodeMCU ESP8266 is an open-source Lua based firmware and development board specially targeted for IoT based applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems and hardware which is based on the ESP-12 module, and like this, it can also be programmed using Arduino IDE and can act as both Wi-Fi Hotspot or can connect to one. It has one Analog Input Pin, 16 Digital I/O pins along with the capability to connect with serial communication protocols like SPI, UART, and I2C. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. Its applications include prototyping for IoT devices, low powered battery-operated applications, and projects requiring I/O interface with Bluetooth and Wi-Fi capabilities. [4]

SPECIFICATIONS

Microcontroller:	Tensilica 32-bit RISC CPU Xtensa LX106
Operating Voltage:	3.3V
Input Voltage:	7-12V
Digital I/O Pins (DIO):	16
Analog Input Pins (ADC):	1
UARTs:	1
SPIs:	1
I2Cs:	1
Flash Memory:	4 MB
SRAM:	64 KB
Clock Speed:	80 MHz

3.3.RFID READER

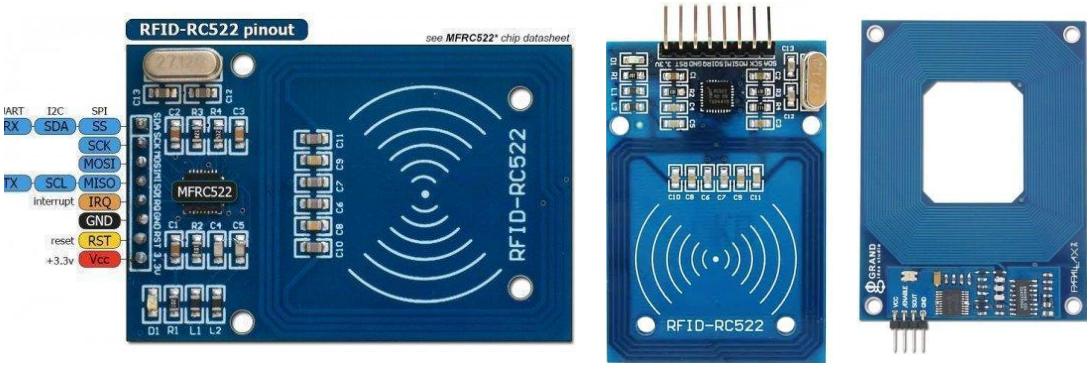


Figure 4

A radio frequency identification reader (RFID reader) is a device used to gather information from an RFID tag, which is used to track individual objects. Radio waves are used to transfer data from the tag to a reader.

RFID is a technology similar in theory to bar codes. However, the RFID tag does not have to be scanned directly, nor does it require line-of-sight to a reader. The RFID tag must be within the range of an RFID reader, which ranges from 3 to 300 feet, in order to be read. RFID technology allows several items to be quickly scanned and enables fast identification of a particular product, even when it is surrounded by several other items.

RFID technology uses digital data in an RFID tag, which is made up of integrated circuits containing a tiny antenna for transferring information to an RFID transceiver. The majority of RFID tags contain at least an integrated circuit for modulating and demodulating radio frequency and an antenna for transmitting and receiving signals. Frequency ranges vary from low frequencies of 125 to 134 kHz and 140 to 148.5 kHz, and high frequencies of 850 to 950 MHz and 2.4 to 2.5 GHz. Wavelengths in the 2.4 GHz range are limited because they can be absorbed by water. [5]

3.4.20*4 LCD DISPLAY

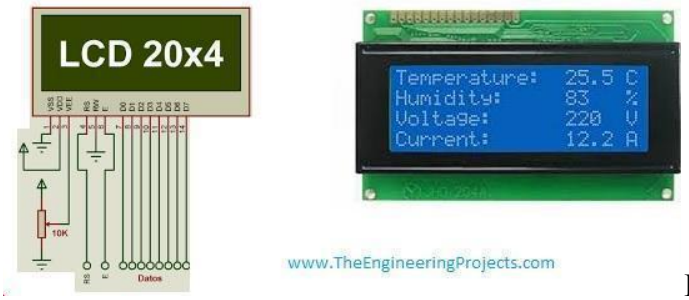


Figure 5

A 20x4 LCD display is a very basic module and is commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 20x4 LCD means it can display 20 characters per line and there are 4 such lines. In this LCD each character is displayed in a 5x7 pixel matrix. The LCD has two registers, namely, Command and Data. This is a standard HD44780 controller LCD. [6]

20 Characters x 4 Lines

Built-in HD44780 Equivalent LCD Controller

Works directly with ATMEGA, ARDUINO, PIC and many other microcontroller/kits.

4- or 8-bit data I/O interface

Low power consumption

3.5. IR SENSORS

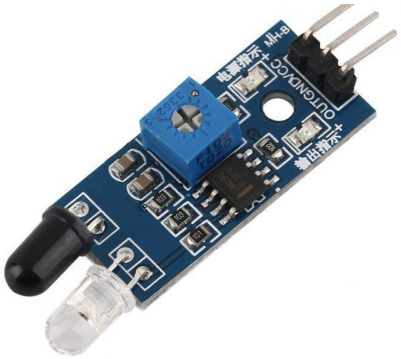


Figure 6

IR sensor is an electronic device that emits light in order to sense some object of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. An IR sensor consists of an IR LED and an IR Photodiode, together they are called Photo Coupler or Optocoupler. [7]

3.6. RFID CARDS

RFID technology was adopted instead of alternative methods such as number plate reading via image recognition, QR code scanning, fingerprint scanning, Bluetooth, NFC since it's relatively inexpensive, more accurate, quicker, consumes lesser power and eliminates the need for the driver to halt the vehicle or roll down the window for authentication.

Figure 7

Radio Frequency Identification describes a technology that is based on an automatic, contactless identification of objects by means of radio waves. RFID cards are thus laminated plastic cards made of PVC, which are equipped with a radio frequency antenna or smart card inlay inside and can be written and read contactless by an RFID reader.

The key advantages compared to PVC cards with barcodes or magnetic stripes are their contactless identification without direct visual contact. Since the RFID transponder is located inside the card, it is virtually wear-free. This means that the chip cards retain their functionality even when wet, polluted or subjected to mechanical influences. In addition, cards with passive transponders do not require their own energy source and are therefore maintenance-free. The reading distances range from 1 cm to several meters, depending on the RFID technology and reader used. The most important application areas of RFID smart cards are ID cards for time tracking, access control and ticketing applications.[8]

3.7.SERVO MOTOR

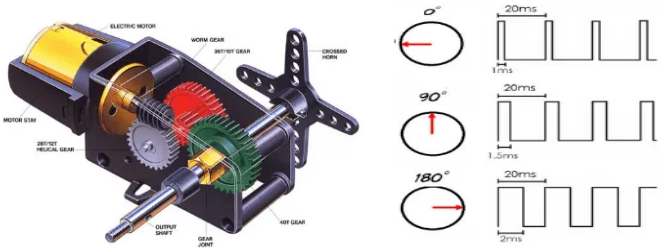


Figure 8

Servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a servo mechanism.

Servo motor is controlled by PWM (Pulse width Modulation) which is provided by the control wires. Its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically, a servo motor is made up of a DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of the DC motor is converted into torque by Gears.

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degrees from either direction from its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. [9]

3.8.I2C MODULE

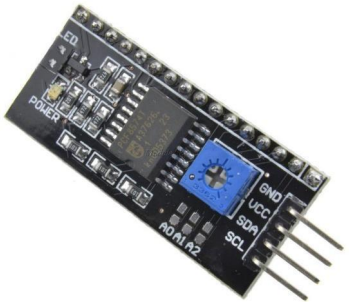


Figure 9

The I2C Module has an inbuilt PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display. These modules are currently supplied with a default I2C address of either 0x27 or 0x3F. If there 3 sets of pads labelled A0, A1, & A2 then the default address will be 0x3F. If there are no pads the default address will be 0x27. The module has a contrast adjustment pot on the underside of the display. This may require adjusting for the screen to display text correctly. [10]

Operating Voltage: 5V
Backlight and Contrast is adjusted by potentiometer
Serial I2C control of LCD display using PCF8574
Come with 2 IIC interface, which can be connected by Dupont Line or IIC dedicated cable
Compatible for 16x2 LCD
This is another great IIC/I2C/TWI/SPI Serial Interface
With this I2C interface module the user is able to realize data display using 2 wires.

4. LIST OF TOOLS AND SERVICES

4.1. BLYNK APPLICATION

Blynk is an IoT platform that allows users to quickly build interfaces for controlling and monitoring hardware projects from iOS and Android devices. After downloading the Blynk app, one can create a project dashboard and arrange buttons, sliders, graphs, and other widgets onto the screen. Using the widgets, the user can turn pins on and off or display data from sensors. [11]

There are three major components in the platform:

Blynk App - allows users to create interfaces for their projects using various widgets provided.

Blynk Server - responsible for all the communications between the smartphone and hardware. The user can use Blynk Cloud or run a private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a NodeMCU.

Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands. Blynk allows users to create local Blynk Server, allowing everything to be kept inside the home network.

4.2. DATABASE & RDBMS

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.[12]

Relational database management systems (RDBMS) are used to store and manage huge volumes of data. All the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A Relational Database Management System (RDBMS) is a software that –

- Enables one to implement a database with tables, columns and indexes.

- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

4.3. SQL

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc. This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDBMS. One can easily create and manipulate the database, access and modify the table rows and columns, etc.[13]

4.4. PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web-based software applications (an application that executes on the server and generates the dynamic page.).[14]

PHP is a recursive acronym for "PHP: Hypertext Preprocessor". It is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking and even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

PHP is pleasingly fast in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.

4.5. MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and an easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

MySQL uses a standard form of the well-known SQL data language. It works very quickly and works well even with large data sets. is very friendly to PHP, the most appreciated language for web development. It supports large databases, up to 50 million rows or more in a table. [15]

4.6. PHPMyAdmin

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement. It supports

- Intuitive web interface
- Import data from CSV and SQL
- Export data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, LATEX and others
- Administering multiple servers
- Creating graphics of your database layout in various formats
- Creating complex queries using Query-by-example (QBE)
- Searching globally in a database or a subset of it. [16]

4.7. Telegram

Telegram is a freeware, cross-platform, cloud-based instant messaging (IM) service. The service also provides end-to-end encrypted video calling, VoIP, file sharing and several other features. It first rolled out on iOS and Android in late 2013, and now has an estimated 550 million monthly users. Telegram's user base tends to increase whenever a privacy scandal hits one of its larger competitors.

What makes Telegram unique is its focus on privacy, encryption, and an open-source API. There are countless unofficial clients to go along with the official Telegram apps and web interface. It also allows multiple devices to use the same account (verified by SMS), and multiple accounts on the same device.

Telegram application was adopted for the project as it's popularity is growing tremendously hence most of the subscribers of the system can easily receive notifications by means of telegram. [17]

4.8. Telegram Bot

Bots are third-party applications that run inside Telegram. Users can interact with bots by sending them messages, commands and inline requests. One canTelegram is an instant messaging service that allows for the creation of bots. control their bots using HTTPS requests to Telegram's Bot API. At the core, Telegram Bots are special accounts that do not require an additional phone number to set up.

Bots can be configured to send and receive messages. This is useful for Arduino projects as one can receive updates from your project or issue its commands via their Telegram app from anywhere.

Each library only supports a single type of Arduino and has different features implemented. The only thing that needs to be different for each board is the actual sending of requests to Telegram. [18]

4.9. XAMPP

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. [19]

5. BLOCK DIAGRAM

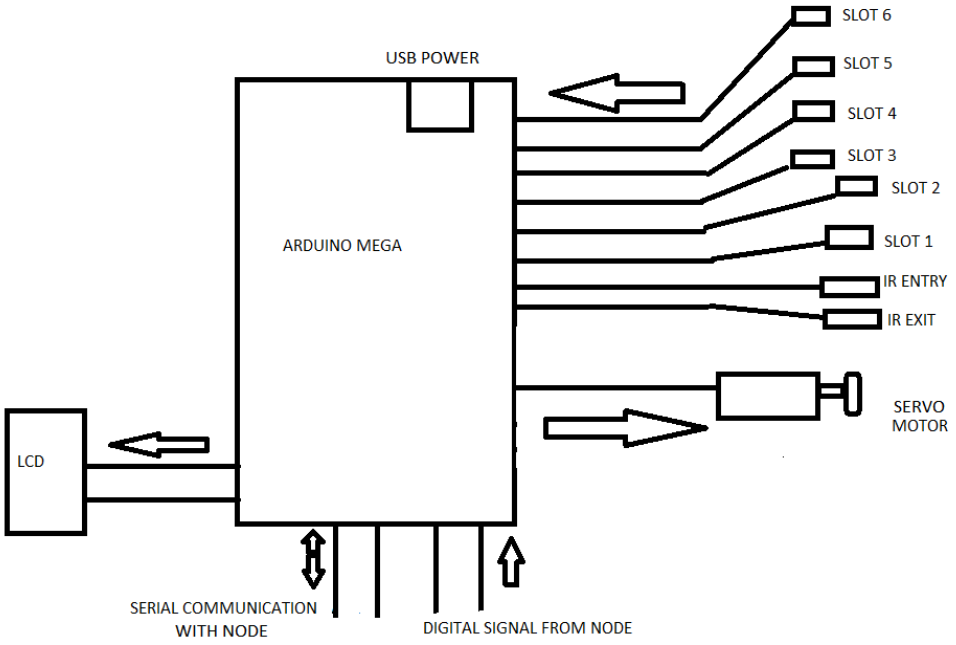


Figure 10 BLOCK DIAGRAM OF ARDUINO

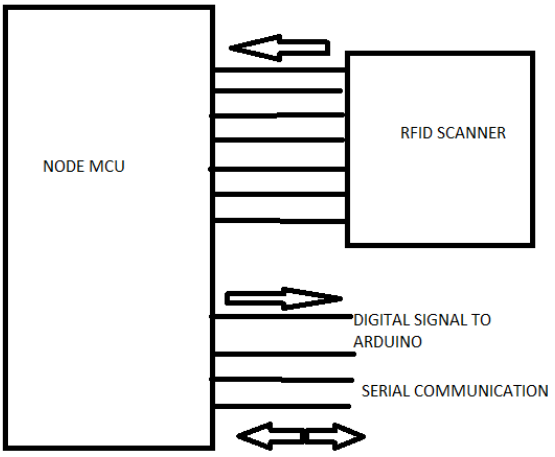


Figure 11 BLOCK DIAGRAM OF NODEMCU

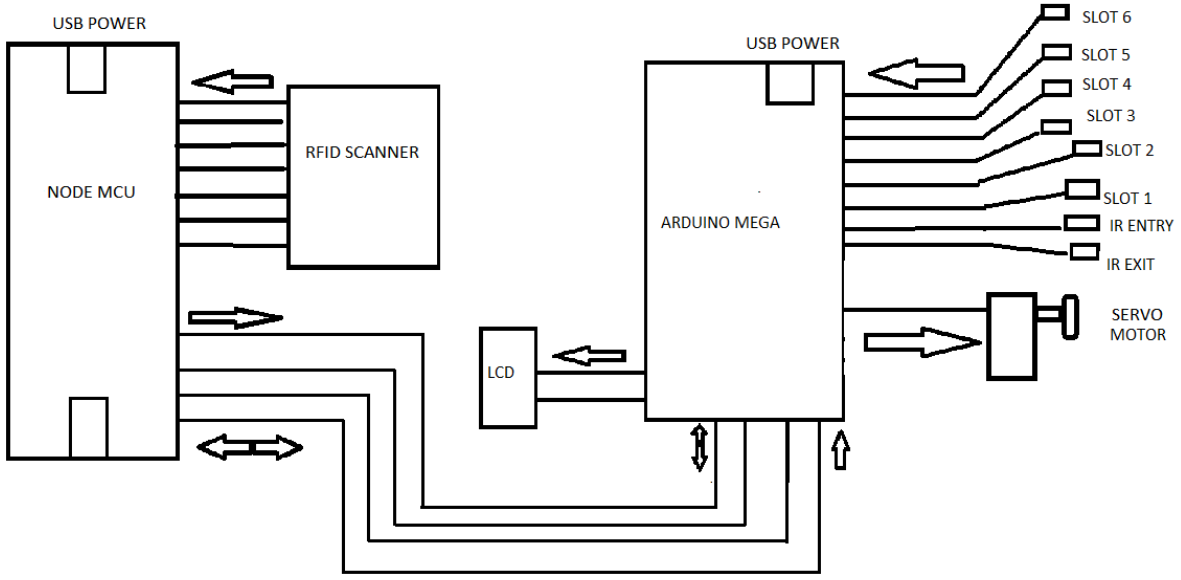


Figure 12 BLOCK DIAGRAM OF THE HARDWARE

6. SYNOPSIS OF WORKING

6.1. INITIALISATION

Configuration

The working of the Car Parking Management System starts with the administrator configuring the MCU. Firstly, the admin installs a software called XAMPP Control Panel on their PC in order to turn on the PHP Webserver and further turns on the Apache and MySQL modes located in the application, ON. The admin then opens the webapp and enters their email ID and password in-order to gain entry to the webapp interface. Then, they enter the devices tab and generate a unique device ID which is used for the web app to recognise the MCU. The device is thus created after a name is submitted for the device.

This ID and IP address is then compiled into the MCU. In the device tab, there are 2 modes: - Enrollment and Attendance. Enrollment mode is used to add new users into the database whereas attendance mode is when the web-app is under normal operation. When the MCU is first put to use, each car has to be registered into the database. For that, the admin selects the Enrolment mode.

Enrolment

Here the RFID scanner is connected to the MCU and MCU can obtain the values from the RFID scanner whenever an RFID card is brought next to it. Once a card is brought close to the scanner, it reads the value and sends the value to the web-app by sending a URL which consists of the IP address, UID and device ID which was procured earlier. When a web-app receives the UID, it proceeds to the next step, enrolling the users. Here, the website consists of entry fields where admin is asked to enter the details of the user: - Name, Gender, Vehicle registration number, Telegram credentials (Token and chat ID) and initial balance. There are options to modify and delete the user entries too. Once all the car UIDs are registered by providing details to the database, the enrollment is done and the admin goes back to the devices tab to switch the mode to attendance.

Telegram BOT

This is the step to register each driver to a telegram server. By using a Bot named BotFather, the token is created and chat ID is created using another bot named IDBot. These values are submitted to the website database during the enrolment.

6.2. OPERATION

Arduino

Once the enrollment is done, the normal operation is commenced. At the end of the driveway, 2 IR sensors are found with a boom barrier separating them.

Figure 13

The 2 IR sensors are placed to detect the presence of the vehicle from either side. When a car is brought from the driveway and approaches the first IR sensor, it sends a signal to Arduino which then checks the condition of the sensors inside of the parking lot.

Inside, there are 6 slots, each equipped with IR sensors. Arduino checks if there are enough slots free and if there is no congestion. Once the requirement is set, the Arduino asks the driver to bring the car forward to the RFID scanner which is located on the road, close to the boom gate. When the car is near the scanner, it reads the UID of the card which is attached to the undercarriage of the car.

Attendance

Once the scanner detects the presence of a card, it scans the card and retrieves an 8-bit hexadecimal value and sends it to MCU. MCU extracts it and generates a URL, composed of IP address, UID and the Device ID which is to be sent to the web-app. Once the URL reaches the web-app, it extracts the UID from the URL and checks the presence of UID in the database. If it is not found, it sends a response back to MCU and MCU sends a signal back to Arduino intending that the user's ID is unavailable. Arduino then informs the user that their card is not registered with the database thus denying their entry. If the user is registered with the database, the web-app then checks the records of the user. If the remaining balance of the user is too low, it sends a response to the MCU regarding it and MCU sends a signal back to Arduino intending that the user has low balance. Arduino then informs the user that their balance is low and to recharge to gain entry.

Admission into the parking lot

When the user has sufficient balance in their account, the web-app sends a response back to the MCU, which includes the following keywords about the driver 1. Log info (The word login in this case) 2. Username 3. Remaining balance 4. Telegram credentials.

The MCU upon receiving the payload, it extracts the log info, name and balance. It then sends a signal to Arduino to admit the user. Arduino after analysing the signal, opens the gate for the car to enter. This is done by signalling the motor to rotate so as to keep the pole perpendicular to the ground. Once the car enters, it passes through the exit IR sensor, signalling the Arduino that the car has entered the lot and to close the gate. Arduino then signals the motor to close the gate. A variable used to store the number of free slots is updated by decrementing it.

Telegram Entry Message

MCU after retrieving the log info, name, time, telegram token and chat ID, it generates a string to be sent to the user via telegram. If the log info denotes that the user has entered the lot, it generates a string welcoming the user by their name and revealing their remaining balance. At the same time, MCU also has the information regarding the nearest slot available in the parking lot. From this data, MCU generates directions to head to the nearest slot and includes it in the string. It initiates the telegram server and creates the address of the user using the telegram token and chat ID and sends the message to the user.

Nearest Slot (Arduino)

Once the driver parks their car at a slot, the Arduino collects all the slot change information via the IR sensors. The presence of the car in front of the IR sensor causes it to send a constant signal to Arduino, which analyses it and makes changes to the slot variables. These variables are used to display the slot information at the LCD display located at the front of the driveway. When a driver has entered the gate, the display flashes the message denoting the nearest available slot for the driver to consider. This nearest value is computed based on the priority-based computation of the slot variables.

Nearest Slot (MCU)

The Arduino and MCU are connected by means of serial communication. Whenever there is a change in slot information, Arduino generates a string by combining all the slot variables. It then sends it to MCU. Once the signal reaches MCU, it extracts all the slot information from the string to the slot variables. Using the same formula used by Arduino to compute the nearest vacant slot, MCU uses it and stores the value to a variable. Depending on this value, a string is generated which conveys directions to that nearest slot. This string is then merged with the telegram message before it is sent to the respective driver.

Blynk

When the MCU is started, it has established connection with the Blynk server using the appropriate credentials. Whenever a slot change is observed by the Arduino, it serially transmits the slot data to MCU which processes it and stores them in individual variables. Each driver is given a Blynk app which is connected to the server using the same ID as that of the MCU. The app consists of LEDs arranged in a manner that corresponds to the individual slots in the parking lot. When an LED is turned on, the slot corresponding to that LED is occupied. Once that's done, the value of the variable is used to turn on or off the LEDs in the Blynk app server. The LEDs are controlled using virtual pins and appropriate commands are sent by the MCU depending on the value of each variable in real time. The MCU also sends the nearest slot number to Blynk.

Exiting from the parking lot.

The following events occur when the driver leaves. When the car is moved away from the sensor, the Arduino identifies a vacant slot and updates the LCD display regarding the new slot information and sends the data to MCU as well. MCU uses this data to update the Blynk server and updates the nearest slot variable depending on the slot value. When the car touches the exit IR sensor, the car need not wait for the Arduino's next instruction (about moving closer to the gate). When the scanner scans the card, it extracts the UID and sends it to MCU which retransmits back to the web-app. Web-app uses the UID to check if the user exists in the database or not. If the user is not registered, it sends the subsequent response to MCU which then signals back to Arduino, which blocks the driver from exiting. When the car is registered, the web-app goes through the database and extracts the following information about the driver; - 1. Log info (The word logout in this case), 2. User-name, 3. Time spent inside the parking lot (exit time – entry time), 4. Fare of the user calculated from the time spent inside, 5. Remaining balance of the user, 6. Telegram token and chat ID.

From this string of data, the information is extracted and stored in variables and since the log info is logout, it generates a corresponding string for the telegram.

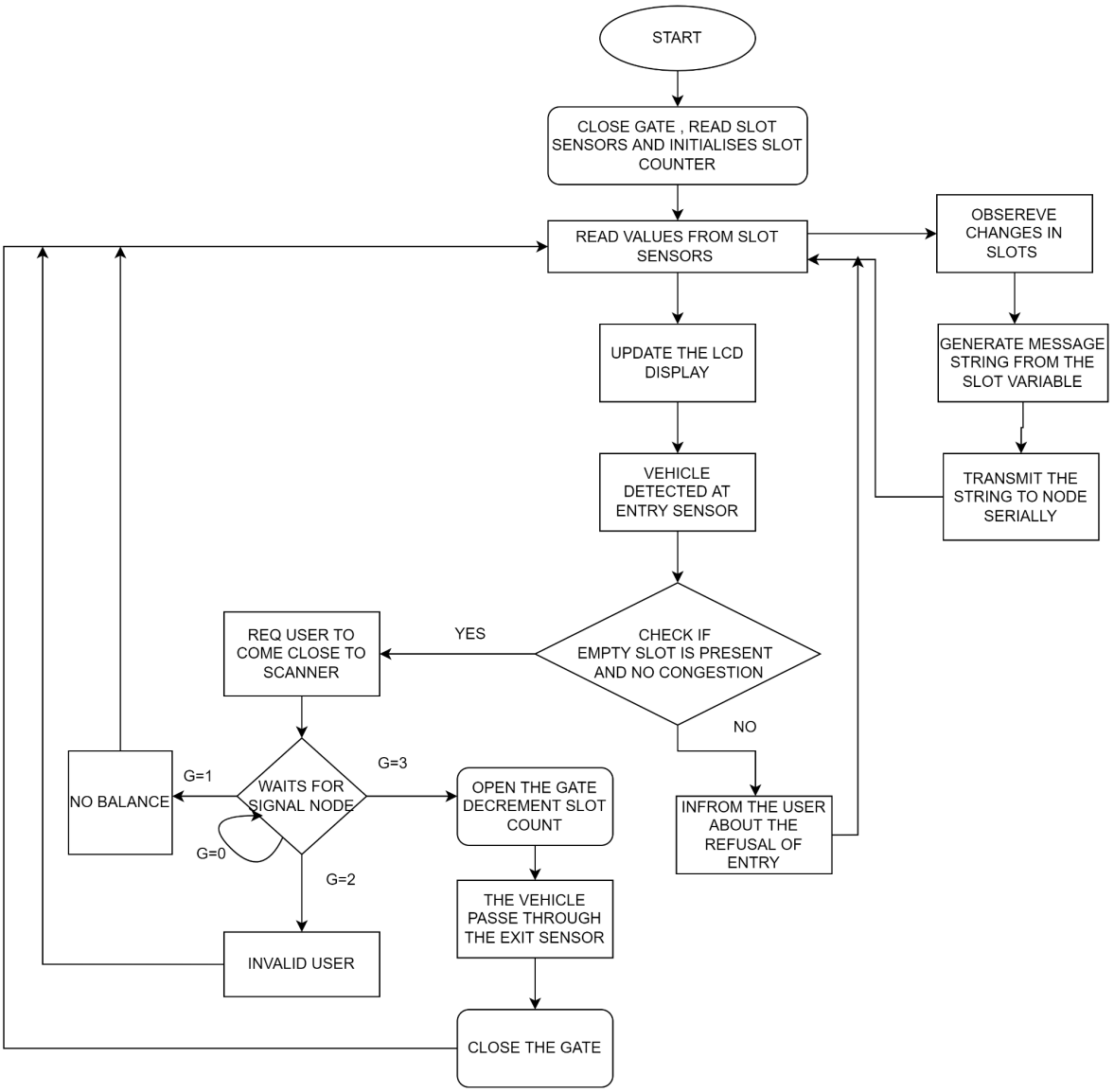
The MCU signals the Arduino that the card is valid and Arduino raises the gate by signalling the motor. Once the car passes through the entry sensor, the Arduino closes the gate. The variable used to store the number of free slots is updated by incrementing it.

Telegram exit message

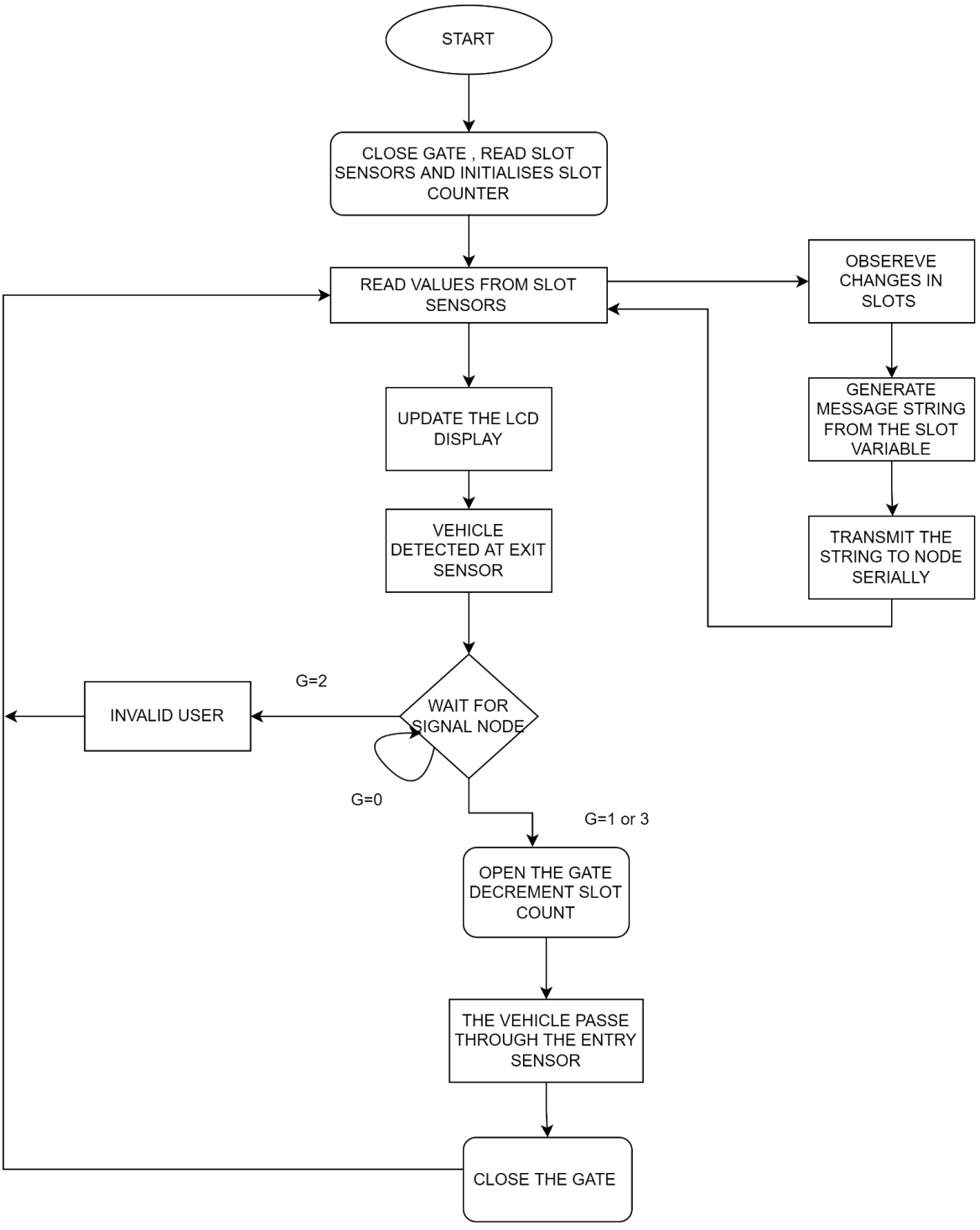
The message to be sent to the driver's telegram is generated once the gate is lifted. The message depends on the remaining balance of the driver. When the remaining balance is positive (i.e., there is an amount left in their account), the MCU produces a farewell message along with their name and all the other relevant information. However, when the balance is negative (The driver has spent too much time inside and has depleted their amount such that their remaining balance is less than 0), The MCU produces a warning message for paying back the dues and informs that their admission will be denied during their next arrival to the parking lot. The message also includes the relevant information for the user.

7. FLOW DIAGRAM

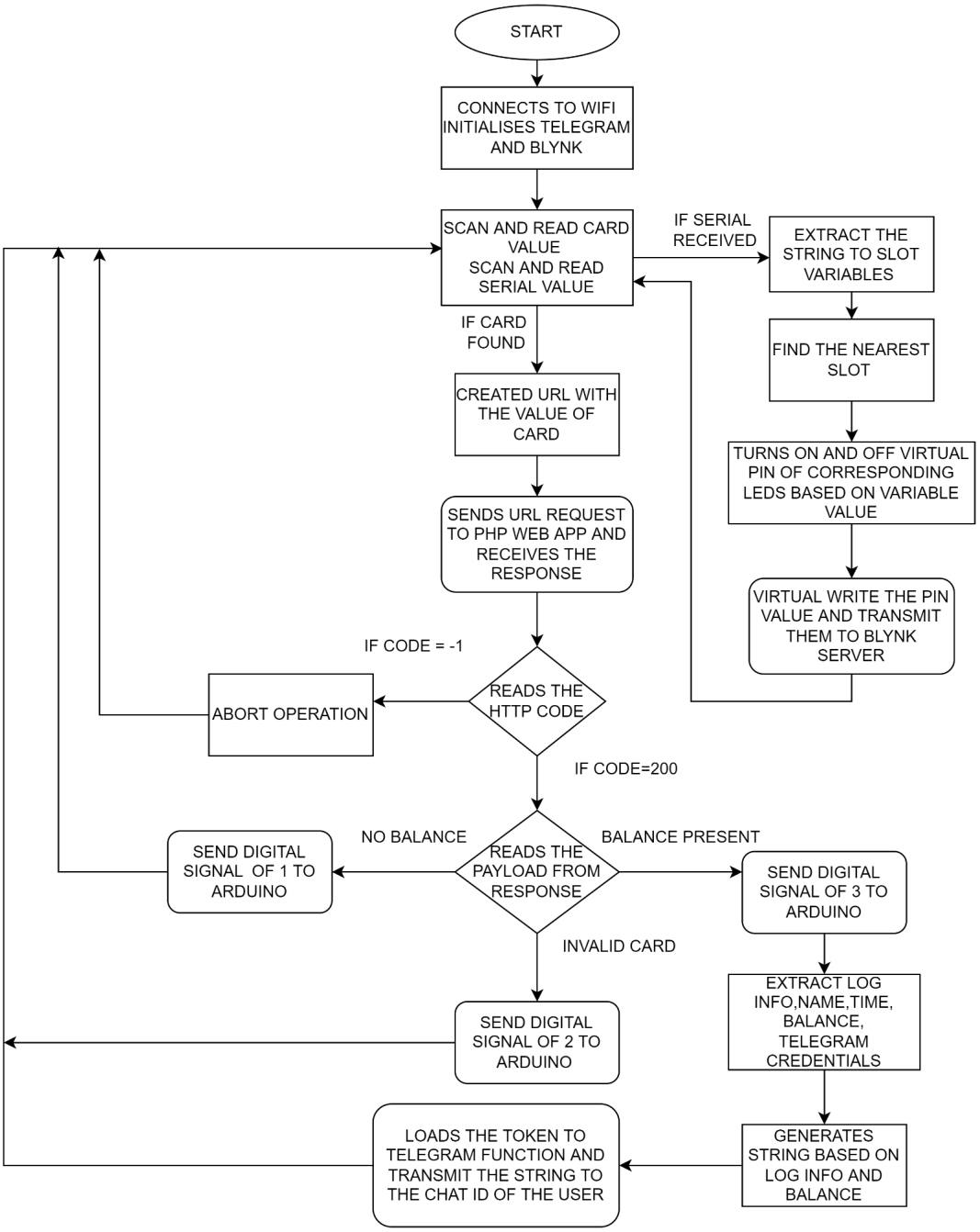
7.1. Flow diagram for Arduino when the car is entering



7.2 Flow diagram for Arduino when the car is leaving



7.3 Flow diagram for Node MCU at all times



8. CIRCUIT DIAGRAM

8.1. Arduino only

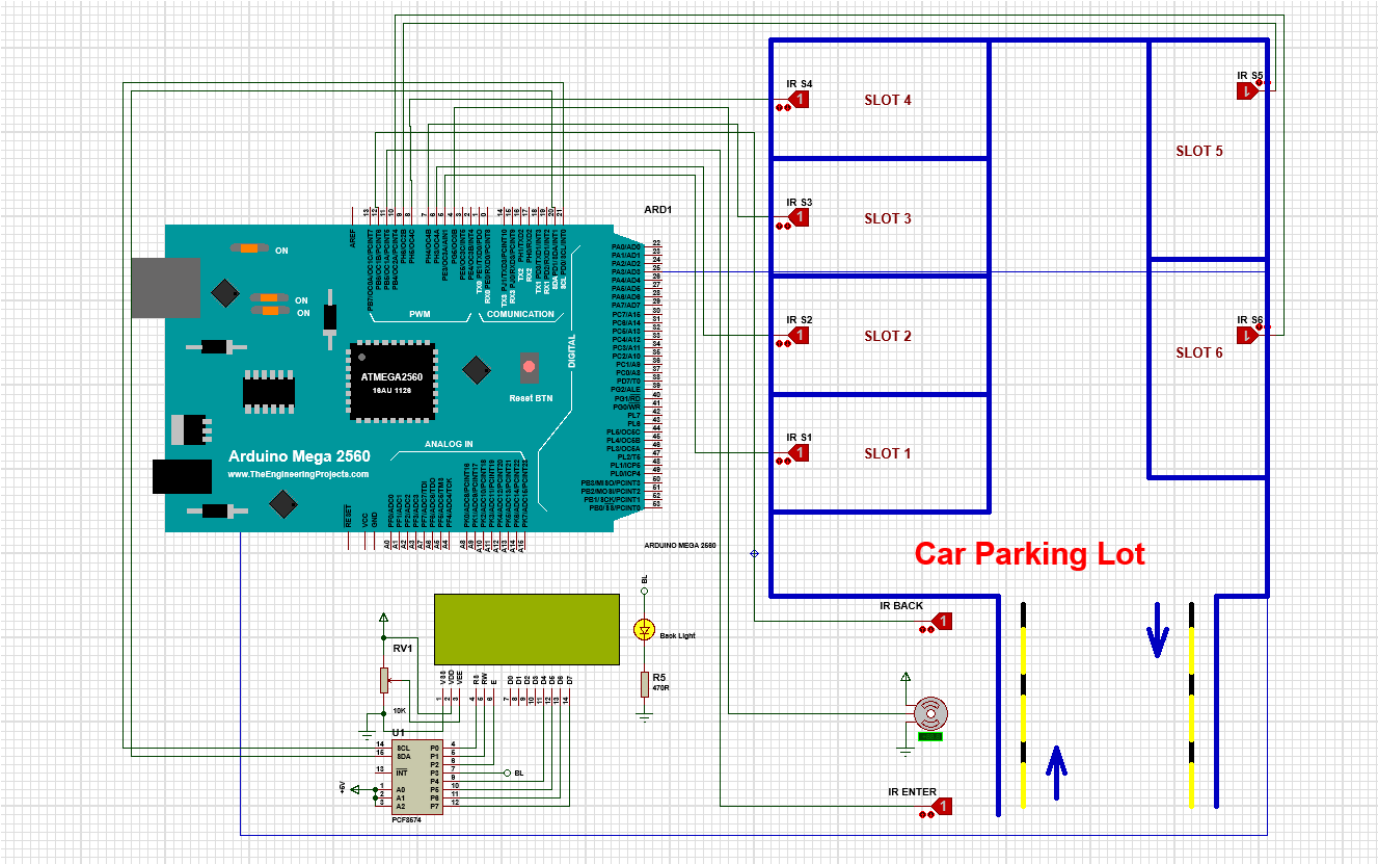
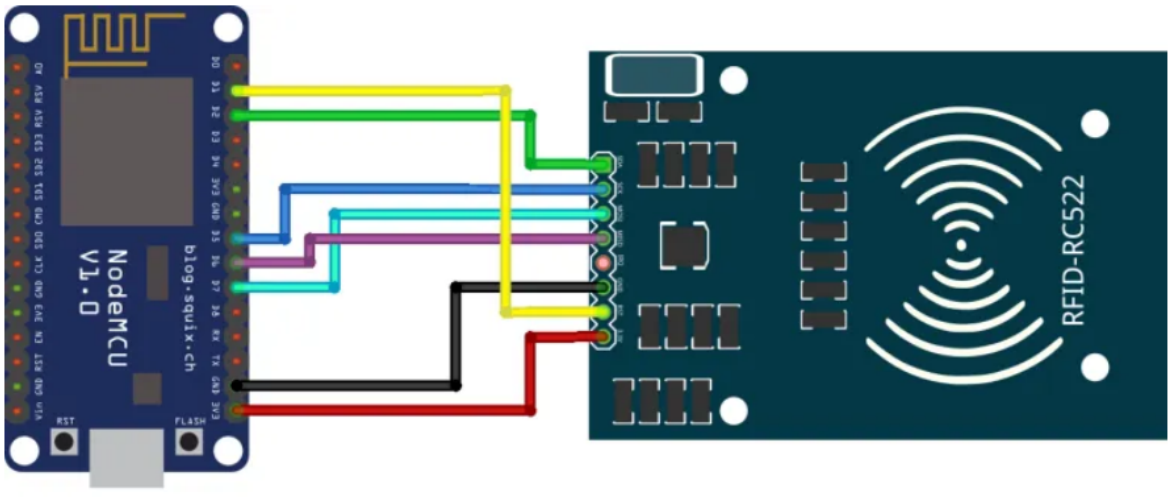
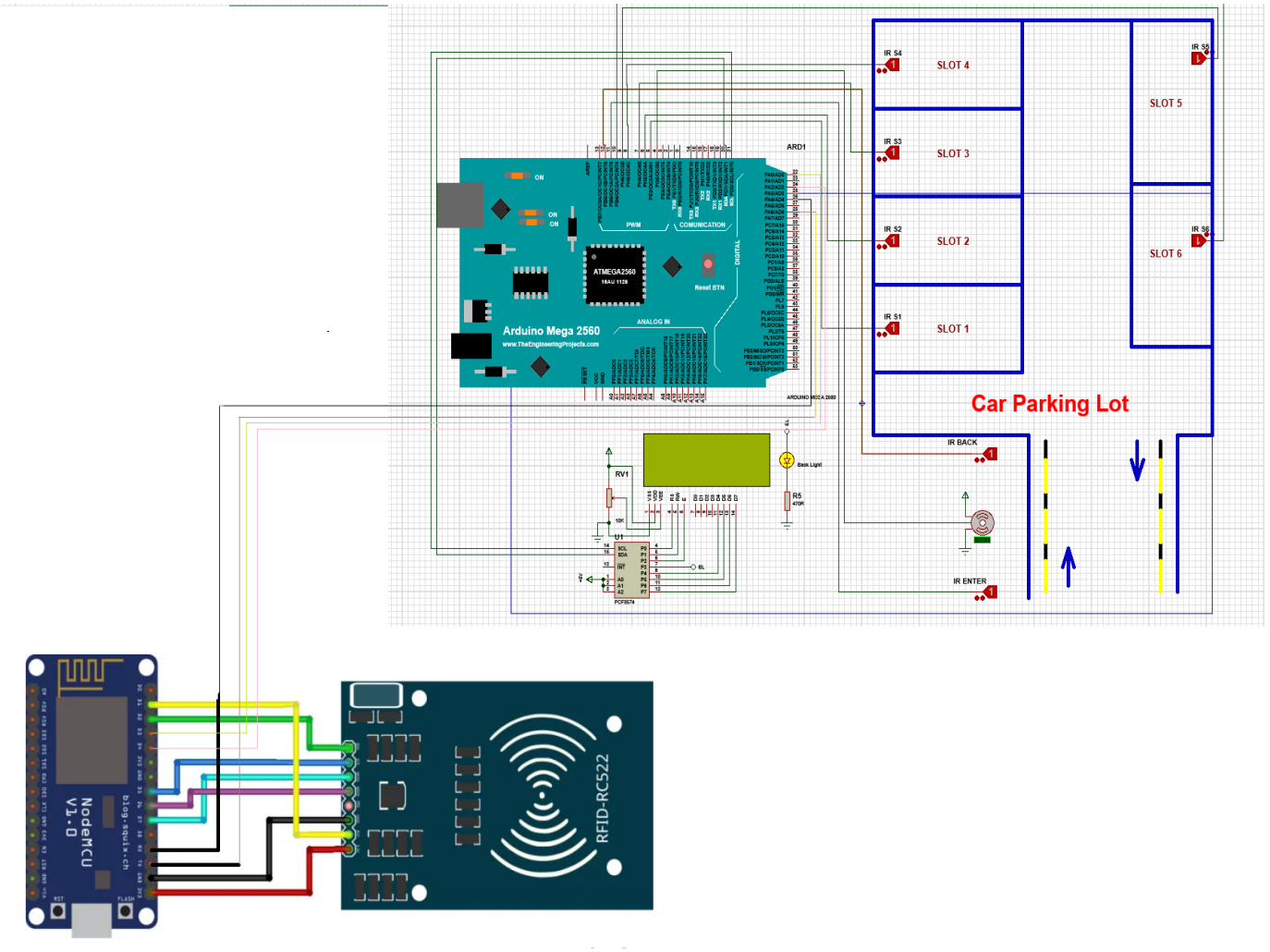


Figure 14

8.2. Node Only



8.3. Arduino and Node MCU Connected and Combined



Connections from Node MCU to Arduino:

D3 -> 22

D4 -> 24

Rx -> 26

Tx -> 28

9. PROGRAM CODE AND PRINCIPLE OF OPERATION

The program code is divided into 3 parts. The principle of operation is provided in comments (//) after each line of code.

9.1. ARDUINO CODE

```
//--------Start of the Arduino Program code---------
//Header Files
#include <Servo.h> // for interfacing motor
#include <Wire.h> //for the servo motor
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h> //For the I2C module that controls the LCD display.
//Library Files
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); //initialising the pins for lcd.
SoftwareSerial espSerial(26, 28);//Initiliasing pins for receiving and transmitting. The indentifier used to transmit to Arduino.
Servo myservo; //initialising identifier for Servo motor.


//Initialising Global variables
#define ir_enter 11 //Entry IR Sensor. For detecting presence of car at the entrance.
#define ir_back  12 //Exit IR Sensor. For detecting presence of car at the exit.

#define ir_car1 5 //IR sensor variable for each car slot.
#define ir_car2 6
#define ir_car3 7
#define ir_car4 8
#define ir_car5 9
#define ir_car6 10

int S1=0, S2=0, S3=0, S4=0, S5=0, S6=0; //The variables hold free state(0) or occupied for each car slot(1).
int c; //Holds the value rerieved from NODEMCU : 1/2/3. Used to control the gate and for appropriate comment for the LCD display.
bool flag1=0, flag2=0; // Flags for holding the IR sensor value to keep the gate open.
int slot = 6; //The number of free slots remaining.
String msg1,msg2; //msg1 for previous Blynk value string. msg2 for current blynk value string.


void setup(){ //Setting up the components, variables and basic functions.
  Serial.begin(115200); //Baud rate of Arduino.
  espSerial.begin(115200);  //Baud rate for Serial communiation with NODE.
  pinMode(ir_car1, INPUT);  //Setting the pin mode for each IR pins.
  pinMode(ir_car2, INPUT);
  pinMode(ir_car3, INPUT);
  pinMode(ir_car4, INPUT);
  pinMode(ir_car5, INPUT);
  pinMode(ir_car6, INPUT);

  pinMode(ir_enter, INPUT);
  pinMode(ir_back, INPUT);
  pinMode(22,INPUT); //Pins 3 and 4 to recieve the signal from NodeMCU.
  pinMode(24,INPUT);

  myservo.attach(4);  //Attach servo variable to pin 4.
  myservo.write(90); //Keep the gate in closed position (90 degrees)..

  lcd.begin(20, 4);  //Displaying the intro message onto the LCD.
  lcd.setCursor (0,1);
  lcd.print("    Car  parking  ");  lcd.setCursor (0,2);  lcd.print("       System     ");
  delay(1000);
  lcd.clear();

  Read_Sensor(); //Reads value from IR sensors for the first time. Initialises the variable slot.
  msg1 = String ( S6 * 100000 + S5 * 10000 + S4 * 1000 + S3 * 100 + S2 * 10 + S1 );
  espSerial.println(msg1);  //Converts the slot variables to string and trasmit it to NODE.
  int total = S1+S2+S3+S4+S5+S6; //To find the number of free slots at the begining.
  slot = 6 - total; //Initialises the free slots variable once the Arduino is starting up.
}
```

```
283 void Read_Sensor(){ //Reads value from the IR sensors and updates the slot variables.
284 S1=0, S2=0, S3=0, S4=0, S5=0, S6=0;
285
286 if(digitalRead(ir_car1) == 0){S1=1;}
287 if(digitalRead(ir_car2) == 0){S2=1;}
288 if(digitalRead(ir_car3) == 0){S3=1;}
289 if(digitalRead(ir_car4) == 0){S4=1;}
290 if(digitalRead(ir_car5) == 0){S5=1;}
291 if(digitalRead(ir_car6) == 0){S6=1;}
292 }
```

```
103 void loop(){
104
105  Read_Sensor(); //Read the values from the IR sensors in real time. Keeps the slot information updated.
106  int space = S1+S2+S3+S4+S5+S6; // For determining congestion.
107  Serial.println(slot);
108  msg2 = String ( S6 * 100000 + S5 * 10000 + S4 * 1000 + S3 * 100 + S2 * 10 + S1 );
109  //Transforms the slot variables to string by arranging in order.
110  if(msg2 != msg1)         //Checks if the slot string produced is the same as that of the last cycle.
111  { //Eliminates sending the same string repetedily and msg is trasmitted only if a change is observed in the string.
112      espSerial.println(msg2);    //Sends the serial data to NodeMCU to be processed and make the Blynk application work.
113      msg1 = msg2;      //Old message string is stored with new message and thus compared in real time.
114  }
```

```
116 //Displays slot information and free slots in the LCD display.
117 lcd.setCursor (0,0);
118 lcd.print("  Free Slots: ");
119 lcd.print(slot);
120 lcd.print("    ");
121
122 lcd.setCursor (0,1);
123 if(S1==1){lcd.print("S1:Taken ");}
124      else{lcd.print("S1:Empty");}
125
126 lcd.setCursor (10,1);
127 if(S2==1){lcd.print("S2:Taken ");}
128      else{lcd.print("S2:Empty");}
129
130 lcd.setCursor (0,2);
131 if(S3==1){lcd.print("S3:Taken ");}
132      else{lcd.print("S3:Empty");}
133
134 lcd.setCursor (10,2);
135 if(S4==1){lcd.print("S4:Taken ");}
136      else{lcd.print("S4:Empty");}
137
138  lcd.setCursor (0,3);
139 if(S5==1){lcd.print("S5:Taken ");}
140      else{lcd.print("S5:Empty");}
141
142 lcd.setCursor (10,3);
143 if(S6==1){lcd.print("S6:Taken ");}
144      else{lcd.print("S6:Empty");}
```

```
146 if(digitalRead (ir_enter) == 0 && flag1==LOW) //When the entry sensor detects car and.
147 {   // the car is attempting to enter.
148   if(slot>0 && space+slot>4)  //To allow the car inside only if there is
149   { // vacanat slot and no congestion.
150       flag1=HIGH; //to prevent this 'if statement' to reitereate. Stores car presence in flag1.
151 
152     if(flag2==LOW)  //When no presence of car was detected in exit sensor (Car is entering).
153     {
154       c = 0;  //Data from NodeMCU.
155       Serial.println("Show your card");
156       lcd.clear();
157       lcd.setCursor(0,1);
158       lcd.print("Kindly drive closer");
159       lcd.setCursor(0,2);
160       lcd.print("  to the Gate!");
161       check();  //Calls the checking function to accept value from NodeMCU
162       lcd.clear();
163       Serial.println(c);
164     if(c == 3)  //To allow the car to enter the parking lot.
165     {
166         delay(100);
167         Serial.println("ACCESS");
168         myservo.write(180);  //Opens the gate my signaling the motor to rotate 180D.
169         slot = slot-1;  c = 0; // Decrements the slot count by 1.
170         Near(); //Calls Near function to find and display nearest slot
171      }
```

```
172     else
173     if(c == 1)  //Insufficient balance detected.
174     {flag1 = LOW; //Removes the car data from the flag to continue sensing for new cars.
175       delay(100);
176       lcd.clear();
177       c = 0;
178       lcd.setCursor(0,1);
179       lcd.print("Insufficient balance!");
180       delay(1000);
181       lcd.clear();
182       delay(1000);
183     }else
184     if(c == 2)  //Invalid car detected.
185     {flag1 = LOW; //Removes the car data from the flag to continue sensing for new cars.
186       delay(100);
187       c = 0;
188       lcd.clear();
189       lcd.setCursor(0,1);
190       lcd.print("Invalid Card!");
191       delay(1000);
192       lcd.clear();
193       Serial.print("Invalid Card!");
194       delay(100);
195     }
196     }
```

```
    else    //The car is leaving. This is implied since flag2 is NOT LOW.
      {   // ie: the car has touched the exit sensor FIRST.
        lcd.clear();
        lcd.setCursor(0,1);
        lcd.print("Goodbye!");
        lcd.setCursor(0,2);
        lcd.print("See you soon.");
        delay(5000);
        lcd.clear();
        Serial.println("Vehicle exits");
      }
  }else if (slot==0)  //If all the slots are full.
  {
  lcd.setCursor (0,0);
  lcd.print(" Sorry Parking Full ");
  delay(1500);
  }
  else if(space+slot <5)
  {
    lcd.clear();
    lcd.setCursor (1,1);
    lcd.print("Parking lot crowded");
    lcd.setCursor (4,2);
    lcd.print("Please Wait!");
    delay(1500);
    lcd.clear();
  }
}
```

```
if(digitalRead (ir_back) == 0 && flag2==LOW)   //When the exit sensor detects car and car is attempting to leave.
{
  flag2=HIGH; //to prevent this 'if statement' to reitereate. Stores car presence in flag2.
  if(flag1==LOW)  //When no presence of car was detected in entry sensor (Car is leaving).
  {
      c = 0;
      Serial.println("Show your card");
      lcd.clear();
      lcd.setCursor(0,1);
      lcd.print(" Kindly place your");
      lcd.setCursor(0,2);
      lcd.print(" Card on the sensor!");
      check();
      lcd.clear();
      Serial.println(c);
      if(c == 3 || c == 1)  //To allow the car to leave the parking lot. Ignore the balance of the driver.
      {
        myservo.write(180); Serial.println("Gate Opens ");slot = slot+1;  //Opens the gate my signaling the motor
        c = 0;
      }

      else if(c==2)  //Invalid car detected.
      {
        flag2 = LOW;  //Removes the car data from the flag to continue sensing for new cars.
        c = 0;
        lcd.clear();
        lcd.setCursor(0,1);
        lcd.print("Invalid Card!");
        delay(1000);
        lcd.clear();
        Serial.print("Invalid Card!");
        delay(100);
      }
  }else //The car is entering. This is implied since flag1 is NOT LOW.
    {   // ie: the car has touched the entry sensor FIRST.
      lcd.clear();
      lcd.setCursor(0,1);
      lcd.print("Welcome!");
      delay(5000);
      lcd.clear();
      delay(100);
    }
}
```

```
if(flag1==HIGH && flag2==HIGH)  //To close the gate.
{ //If the car has passed both the sensors ie. flag1 = high and flag2 = high, close the gate.
  if(digitalRead(ir_enter) == 1 && digitalRead(ir_back) == 1)
  {
  delay (1000);
  myservo.write(90);  //Closes the gate by signaling the motor to rotate 90D..
  Serial.println("Gate Closes ");
  flag1=LOW, flag2=LOW; //Reinitialises the flag information for new car.
  }
}

delay(1);
}
```

```
bool g,h;
void check() // To convert the 2-bit data from NodeMCU to a decimal value.
{
  delay(100);
  g = digitalRead(22);
  h = digitalRead(24);
  if(g == LOW && h == LOW) //No response.
  {
    if(digitalRead(ir_enter)==0 || digitalRead(ir_back)==0) //To recurse only if the car is still waiting in the queue.
    {
      check(); //Recursion (To keep the arduino waiting for the response from NodeMCU).
    }
    else
    {
      flag1 = LOW;flag2 = LOW;
    }
  }else
  if(g == LOW && h == HIGH)
  {
    c = 1; //Insufficient balance from the card.
  }else if(g == HIGH && h == LOW)
  {
    c = 2; //Card not recognized.
  }else if(g == HIGH && h == HIGH)
  {
    c = 3; //Allow the entry of the user.
  }
}

```

```
void Near() //To find the nearest parking slot.
{
  int le; //Variable to store the least number.
  //Compares with empty slot order in ascending order and stores the value accordingly.
  if(S1 == 0){le = 1;}else if(S2 == 0){le = 2;}else if(S3 == 0){le = 3;}
  else if(S4 == 0){le = 4;}else if(S5 == 0){le = 5;}else if(S6 == 0){le = 5;}
   lcd.clear();
      lcd.setCursor(0,1);
      lcd.print("Kindly Park your car");
      lcd.setCursor(0,2);
      lcd.print("at Slot ");lcd.print(le);    //Asks the driver to park their car in nearest slot.
      delay(5000);
      lcd.clear();
}
```

9.2. NODE MCU CODE

```
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>

//NodeMCU---------------------------
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <BlynkSimpleEsp8266.h>

#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

#define BLYNK_TEMPLATE_ID "ID"
#define BLYNK_DEVICE_NAME "Name"
#define BLYNK_AUTH_TOKEN "Token"
#define BLYNK_PRINT Serial


char auth[] = BLYNK_AUTH_TOKEN;


//*****************************************************************************

#define SS_PIN  D2
#define RST_PIN D1
//*****************************************************************************
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
//*****************************************************************************
/* Set these to your desired credentials. */
const char *ssid = "WiFi SSID";
const char *password = "Wifi Password";
const char* device_token  = "Device token";

BlynkTimer timer;
//*****************************************************************************


X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure secured_client;


//*****************************************************************************
String URL = "http://192.168.43.159/rfidattendance/getdata.php"; //computer IP or the server domain
String getData, Link;
String OldCardID = "";
unsigned long previousMillis = 0;
int n;
int p1,p2,p3,p4,p5,p6;
int le = 6;
String msg; //To extract serial message from Arduino
//*****************************************************************************

void setup() {

  pinMode(D3,OUTPUT);   pinMode(D4,OUTPUT);
  digitalWrite(D3,LOW);  digitalWrite(D4,LOW);
  delay(1000);
  Serial.begin(115200);
  Blynk.begin(auth, ssid, password);
  SPI.begin();  // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card
  //----------------------------------------------
  connectToWiFi();
}
```

```
336 void connectToWiFi(){
337     WiFi.mode(WIFI_OFF);        //Prevents reconnection issue (taking too long to connect)
338     delay(1000);
339     WiFi.mode(WIFI_STA);
340     Serial.print("Connecting to ");
341     Serial.println(ssid);
342     WiFi.begin(ssid, password);
343 
344     while (WiFi.status() != WL_CONNECTED) {
345       delay(500);
346       Serial.print(".");
347     }
348     Serial.println("");
349     Serial.println("Connected");
350     secured_client.setTrustAnchors(&cert); // Add root certificate for api.telegram.org
351 
352     Serial.print("IP address: ");
353     Serial.println(WiFi.localIP());  //IP address assigned to your ESP
354     configTime(0, 0, "pool.ntp.org"); // get UTC time via NTP
355   time_t now = time(nullptr);
356 
357     delay(1000);
358 }
```

```
 68 void loop() {
 69   Blynk.run(); //Starts Blynk
 70   //check if there's a connection to Wi-Fi or not
 71   if(!WiFi.isConnected()){
 72     connectToWiFi();    //Retry to connect to Wi-Fi
 73   }
 74   if (Serial.available()) { //If Node observes presence of message.
 75         Blynk_function();
 76   }
 77   //-----------------------------------------------
 78   if (millis() - previousMillis >= 15000) {
 79     previousMillis = millis();
 80     OldCardID="";
 81   }
 82   delay(50);
 83   //-----------------------------------------------
 84   //look for new card
 85   if ( ! mfrc522.PICC_IsNewCardPresent()) {
 86     return;//got to start of loop if there is no card present
 87   }
 88   // Select one of the cards
 89   if ( ! mfrc522.PICC_ReadCardSerial()) {
 90     return;//if read card serial(0) returns 1, the uid struct contians the ID of the read card.
 91   }
 92   String CardID ="";
 93   for (byte i = 0; i < mfrc522.uid.size; i++) {
 94     CardID += mfrc522.uid.uidByte[i];
 95   }
 
 99   String CardID ="";
100   for (byte i = 0; i < mfrc522.uid.size; i++) {
101     CardID += mfrc522.uid.uidByte[i];
102   }
103   //-----------------------------------------------
104   if( CardID == OldCardID ){
105     return;
106   }
107   else{
108     OldCardID = CardID;
109   }
110   //-----------------------------------------------
111   Serial.println(CardID);
112   SendCardID(CardID);
113   delay(1000);
114 }
```

```
void SendCardID( String Card_uid ){
  Serial.println("Sending the Card ID");
  if(WiFi.isConnected()){
    HTTPClient http;
//Declare object of class HTTPClient
    //GET Data
    getData = "?card_uid=" + String(Card_uid) + "&device_token=" + String(device_token); // Add the Carc
    //GET methode
    Link = URL + getData;
    http.begin( Link); //initiate HTTP request   //Specify content-type header

    int httpCode = http.GET();   //Send the request
    String payload = http.getString();    //Get the response payload

    Serial.println(Link);   //Print HTTP return code
    Serial.println(httpCode);   //Print HTTP return code
    Serial.println(Card_uid);     //Print Card ID
    Serial.println(payload);    //Print request response payload

    if(payload == "Not found!") //If it is invalid card, Sends the data 01 to Arduino.
    {  digitalWrite(D3,HIGH);
       delay(1000);
       digitalWrite(D3,LOW);
    }else
    if(payload == "Low Balance")  //If there is no balance in User's card, Sends the data 10 to Arduino
    {  digitalWrite(D4,HIGH);
       delay(1000);
       digitalWrite(D4,LOW);
    }else
```

```
   if (httpCode == 200) {  //httpCode = 200 => connection is successful and NodeMCU can receive desired data from the MyPHPAdmin.
                           //httpCode =  -1 => error in connection.
    if (payload == "succesful") {  //During edit stage, if card is new and read succesfully.
       delay(100);
     http.end();
     }
     else if (payload == "available") {  //During edit stage, if card was already saved and swiped again.
     delay(100);
     http.end();
     }else   // NOT in edit stage ie. Under normal operation. Extracting useful information from the SQL response.
     {
       int y = 0;  //Initialising y = 0. y is used to measure size of each line.
       while(payload[y]!='{'){  //When payload reaches '{' symbol, measures the number of decimal places in amount.
         y++;
       }int z= y;
       while(payload[z]!='/'){  //When payload reaches '/' symbol, measures the number of decimal places in time remaining.
         z++;
       }int a= z;
       while(payload[a]!='}'){ //When payload reaches '}' symbol, measures the number of decimal places in telegram token.
         a++;
       }int b= a;
       while(payload[b]!='l'){  //When payload reaches 'l' symbol, measures the number of decimal places in telegram chat ID.
         b++;
       }

       int j = payload.substring(0,y).toInt(); //Extracts and converts the amount to integer variable j.
       int ti = payload.substring(y+1,z).toInt();  //Extracts and converts the time remaining to integer variable ti.
       String token = payload.substring(z+1,a);  //Extracts  the token to String variable token.
       String chat = payload.substring(a+1,b);   //Extracts  the tchat ID to String variable chat.
       Serial.println(j);
       Serial.println(ti);
       Serial.println(token);
       Serial.println(chat);
       if (payload.substring(b, b+5) == "login") { //When there's enough balance and valid card, allows user to enter.

        digitalWrite(D3,HIGH); digitalWrite(D4,HIGH); //Sending signal 11 to Arduino
        delay(1000);
        digitalWrite(D3,LOW);  digitalWrite(D4,LOW);

        String user_name = payload.substring(b+5); //Extracting the name of user from payload and storing in user_name.
        Serial.println(user_name);
        telegram(Card_uid,user_name,j,ti,"login",token,chat);  //Calling telegram function
         // and sending all the extracted information to telegram function
       }
```

```
        else if (payload.substring(b, b+6) == "logout") { //When the user is exiting.
          digitalWrite(D3,HIGH); digitalWrite(D4,HIGH); //Sending signal 11 to Arduino
          delay(1000);
          digitalWrite(D3,LOW);digitalWrite(D4,LOW);
          String user_name = payload.substring(b+6);  //Extracting the name of user from payload and storing in user_name.
          Serial.println(user_name);
          telegram(Card_uid,user_name,j,ti,"logout",token,chat);  //Calling telegram function
      }  `//and sending all the extracted information to telegram function.
    }
      delay(100);
      http.end();  //Close connection
    }
  }
}
```

```
void telegram(String CI, String un, int amount, int time_, String adm, String token, String chat)
{
    if(adm == "login")  //If the user is entering the lot.
    {
      String y;
      if(le == 1 || le == 2)
      {
        y = "turn left and park your car";
      }else if(le == 3 || le == 4)
      {
        y = "head straight, turn left and park your car";
      }
      else if(le == 5 || le == 6)
      {
        y = "turn right and park your car along the wall";
      }
      String n = "Welcome " + un + ". Your account balance is : " + String(amount) + ". Your nearest slot is "
      + String(le) + ". Kindly " + String(y); //Produces the welcome message to be sent to their telegram.
      Serial.println(n);
      delay(100);
      mesg(n,token,chat); //Calls the sending function.
    }
```

```
    if(adm == "logout") //If the user is entering the lot.
    {//Produces the exit message to be sent to their telegram.
      String n;
      if(amount >= 0)
      {
        n = "Dear " + un + ". You have spent " + String(time_)
        + " seconds inside the parking lot. Your remaining balance is : " + String(amount);
      }else
      {
        amount = amount + 2 * amount + 10;
        n = "Dear " + un + ". You have spent " + String(time_)
        + " seconds inside the parking lot. You have a pending debt of " + String(amount)
        + ". Kindly clear your dues before your next arrival.";
      }
      Serial.println(n);
      delay(100);
      mesg(n,token,chat);//Calls the sending function.
    }
}
```

```
void mesg(String y, String token, String chat)
{
  UniversalTelegramBot bot(token, secured_client);
  //Starts Telegram bot using the Token of each user's telegram.
  bot.sendMessage(chat, y, "");
  //Sends the message obtained from the calling function 'y'. It uses chat ID to identify the user.
}
```

```
void Blynk_function()
{
      msg = Serial.readStringUntil('\r'); //extracts the entire string
        if(msg!="\n") //To read till entire line
        {                     //Here, the data from arduino is converted from string to integer.
                              //Then it each of its digit is stored in individual integer variables
            n = msg.toInt();
            p1 = n/100000;    //For eg. msg = 110010. p1 = 1 p2 = 1 p3 = 0 p4 = 0 p5 = 1 p6 = 0
            n = n - p1*100000;
            p2 = n/10000;
            n = n - p2*10000;
            p3 = n/1000;
            n = n - p3*1000;
            p4 = n/100;
            n = n - p4*100;
            p5 = n/10;
            n = n - p5*10;
            p6 = n;
            String ss;
            ss = String(p1)+String(p2)+String(p3)+String(p4)+String(p5)+String(p6);
            Serial.println(ss);
            Blynk.virtualWrite(V7, 6-p1-p2-p3-p4-p5-p6);
            //Sends remaining slot information to blynk's 7th virtual pin.
            le = 6;
```

```
          if(p1==1)//Here it turns on/off the Blynk LED depending on the value of p which was extracted before.
          {
            Blynk.virtualWrite(V1, HIGH);
          }else
          {
            Blynk.virtualWrite(V1, LOW);le = 1;
          }
          if(p2==1)
          {
            Blynk.virtualWrite(V2, HIGH);
          }else
          {
            Blynk.virtualWrite(V2, LOW);if(le>1){le = 2;}
          }
          if(p3==1)
          {
            Blynk.virtualWrite(V3, HIGH);
          }else
          {
            Blynk.virtualWrite(V3, LOW);if(le>2){le = 3;}
          }
          if(p4==1)
          {
            Blynk.virtualWrite(V4, HIGH);
          }else
          {
            Blynk.virtualWrite(V4, LOW);if(le>3){le = 4;}
          }
```

```
            if(p5==1)
            {
              Blynk.virtualWrite(V5, HIGH);
            }else
            {
              Blynk.virtualWrite(V5, LOW);if(le>4){le = 5;}
            }
            if(p6==1)
            {
              Blynk.virtualWrite(V6, HIGH);
            }else
            {
              Blynk.virtualWrite(V6, LOW);if(le>5){le = 6;}
            }
      }
}
```

9.3. PHP GETID CODE

```php
<?php
//Connect to database
require 'connectDB.php';
date_default_timezone_set('Asia/Kolkata'); //Definiing timezone(For india)
$d = date("Y-m-d");
$t = date("H:i:s");

if (isset($_GET['card_uid']) && isset($_GET['device_token'])) { //When web-app recieves the UID and device token

    $card_uid = $_GET['card_uid'];  //Extracts UID and token to variable
    $device_uid = $_GET['device_token'];


    $sql = "SELECT * FROM devices WHERE device_uid=?";  //Checks validity of device token
    $result = mysqli_stmt_init($conn);
    if (!mysqli_stmt_prepare($result, $sql)) {
        echo "SQL_Error_Select_device";
        exit();
    }
    else{
        mysqli_stmt_bind_param($result, "s", $device_uid);
        mysqli_stmt_execute($result);
        $resultl = mysqli_stmt_get_result($result);
        if ($row = mysqli_fetch_assoc($resultl)){
            $device_mode = $row['device_mode'];
            $device_dep = $row['device_dep'];
            if ($device_mode == 1) {
                $sql = "SELECT * FROM users WHERE card_uid=?";  //Checks existence of UID in database
                $result = mysqli_stmt_init($conn);
                if (!mysqli_stmt_prepare($result, $sql)) {
                    echo "SQL_Error_Select_card";
                    exit();
                }

else{
    mysqli_stmt_bind_param($result, "s", $card_uid);
    mysqli_stmt_execute($result);
    $resultl = mysqli_stmt_get_result($result);
    if ($row = mysqli_fetch_assoc($resultl)){
        //****************************************************
        //An existed Card has been detected for Login or Logout
        if ($row['add_card'] == 1){
        if ($row['device_uid'] == $device_uid || $row['device_uid'] == 0){
                $Uname = $row['username'];  //Extracts the name, serial number, User account balance and Telegram token and chat ID.
                $Number = $row['serialnumber'];
                $Amount = $row['amount'];
                $Token = $row['tokenid'];
                if ($Amount <= 10)  //Denies entry for the user if the balance is less than 10.
                {
                    echo "Low Balance"; //The response is sent back to NODE
                    exit();
                }
                $sql = "SELECT * FROM users_logs WHERE card_uid=? AND checkindate=? AND card_out=0";
                $result = mysqli_stmt_init($conn);
                if (!mysqli_stmt_prepare($result, $sql)) {
                    echo "SQL_Error_Select_logs";
                    exit();
                }
                else{
                    mysqli_stmt_bind_param($result, "ss", $card_uid, $d);
                    mysqli_stmt_execute($result);
                    $resultl = mysqli_stmt_get_result($result);
                    //****************************************************
```

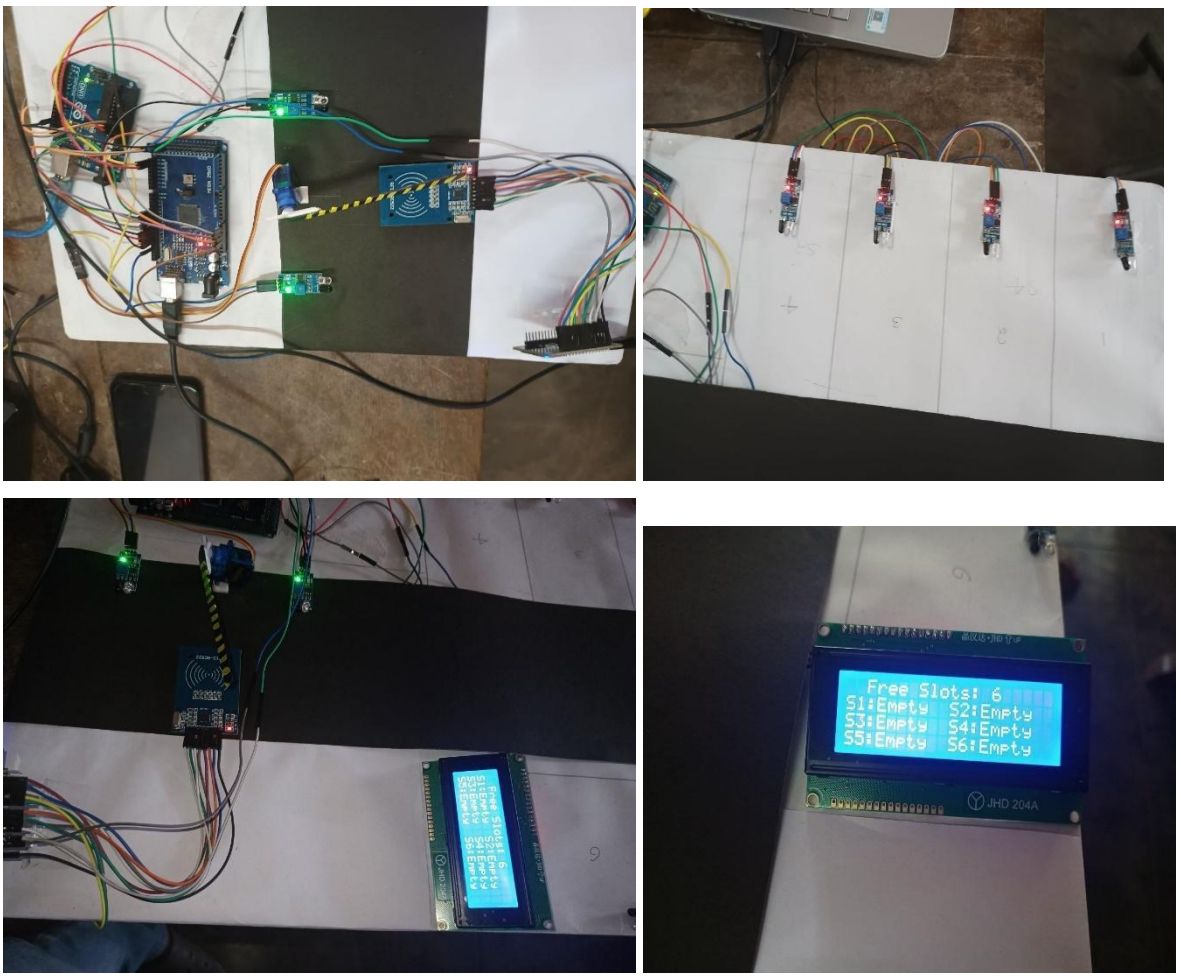
```php
//Login
if (!$row = mysqli_fetch_assoc($resultl)){  //When user is entering the parking lot
    $sql = "INSERT INTO users_logs (username, serialnumber, amount, card_uid, device_uid, device_dep, checkindate, timein, timeout)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
    $result = mysqli_stmt_init($conn);
    if (!mysqli_stmt_prepare($result, $sql)) {
        echo "SQL_Error_Select_login1";
        exit();
    }
    else{
        $timeout = "00:00:00";  //The timeout field is left 0 when user is entring the parking lot.
        mysqli_stmt_bind_param($result, "sdissssss", $Uname, $Number, $Amount, $card_uid, $device_uid, $device_dep, $d, $t, $timeout);
        mysqli_stmt_execute($result);   //Updates the userlog database with all the details of the user when admitting.

        echo $Amount."{0"."/".$Token."login".$Uname;
        exit();
    }
}
//*****************************************************

//Logout
else{   //When user is leaving the parking lot
    $sql = "SELECT timein FROM users_logs WHERE id=?";
    $result = mysqli_stmt_init($conn);
    $r = $row['timein'];    //Copies the entry time of user
    $minute1 = date("i",strtotime($r)); //Extracts minute
    $second1 = date("s",strtotime($r)); //Extracts time
    $sql="UPDATE users_logs SET amount=?,timeout=?, card_out=1  WHERE card_uid=? AND checkindate=? AND card_out=0";
    $result = mysqli_stmt_init($conn);  //Updates the previous record of the user in userlog which was made when user entered the lot.
    if (!mysqli_stmt_prepare($result, $sql)) {  //Done by updating the timeout field and the account balance field.
        echo "SQL_Error_insert_logout1";
        exit();
    }
    else{
        $a = $minute1*60 + $second1;    //converts the entry time minute and second into integer value
        $minute2 = date("i",strtotime($t)); //Extracts minutes and seconds of the current time (Time of exit)
        $second2 = date("s",strtotime($t));
        $b = $minute2*60 + $second2;    //Converts the exit time minute and second into integer value.
        $z = $b - $a;       //Finds the difference of the 2 time to extract the time spent inside the lot.
        if($z > 300)        //To find the charge based on time spent inside.
        {
            $c = $Amount - intdiv(($z-300),3) - 50; //When driver spends more than 5 minutes, for every 3 seconds, 1Rs is charged.
        }else
        if($z > 10) //When driver spends more than 10 seconds inside, for every 6 seconds, 1Rs is charged
        {
            $c = $Amount - intdiv($z,6);
        }
        else{          //Refuses to charge when the driver spends less than 10 seconds.
                $c = $Amount;
            }

        mysqli_stmt_bind_param($result, "isss", $c, $t, $card_uid, $d);
        mysqli_stmt_execute($result);

        echo $c."{".$z."/".$Token."logout".$Uname;
        $sql="UPDATE users SET amount=? WHERE card_uid=?";
        $result = mysqli_stmt_init($conn);
        if (!mysqli_stmt_prepare($result, $sql)) {
            exit();
        }
        else{
            mysqli_stmt_bind_param($result, "is", $c, $card_uid);   //Updates the time and balance of the previous record of the driver.
            mysqli_stmt_execute($result);

            exit();
        }

        exit();
    }
}
```

```
                    }
                }
            }
            else {
                echo "Not Allowed!";
                exit();
            }
        }
        else if ($row['add_card'] == 0){
            echo "Not registerd!";
            exit();
        }
    }
    else{
        echo "Not found!";  //When the card is not registered.
        exit();
    }
}
}
```

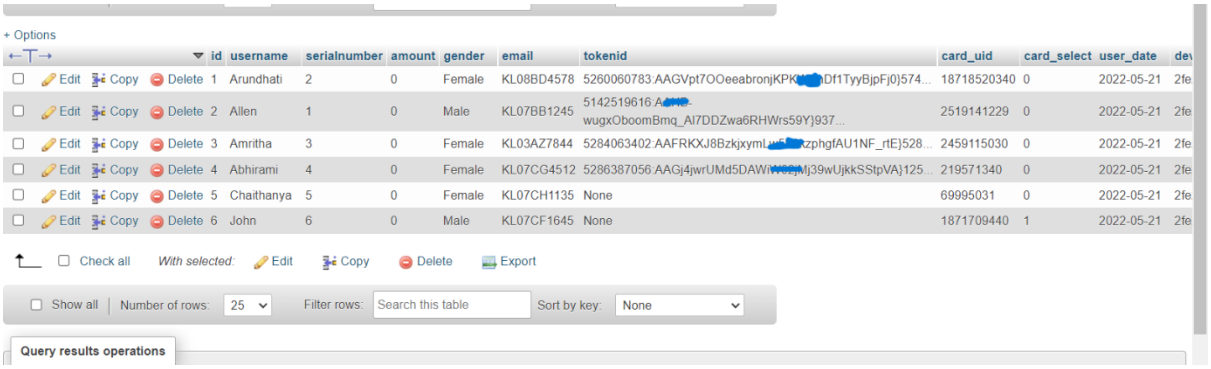
9.4 SQL CODE

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `username` varchar(30) NOT NULL DEFAULT 'None',
  `serialnumber` double NOT NULL DEFAULT '0',
  `amount` int(11) NOT NULL DEFAULT '0',
  `gender` varchar(10) NOT NULL DEFAULT 'None',
  `email` varchar(50) NOT NULL DEFAULT 'None',
  `tokenid` varchar(70) NOT NULL DEFAULT 'None',
  `card_uid` varchar(30) NOT NULL,
  `card_select` tinyint(1) NOT NULL DEFAULT '0',
  `user_date` date NOT NULL,
  `device_uid` varchar(20) NOT NULL DEFAULT '0',
  `device_dep` varchar(20) NOT NULL DEFAULT '0',
  `add_card` tinyint(1) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

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

--
-- Table structure for table `users_logs`
--

CREATE TABLE `users_logs` (
  `id` int(11) NOT NULL,
  `username` varchar(100) NOT NULL,
  `amount` int(11) NOT NULL,
  `serialnumber` double NOT NULL,
  `card_uid` varchar(30) NOT NULL,
  `device_uid` varchar(20) NOT NULL,
  `device_dep` varchar(20) NOT NULL,
  `checkindate` date NOT NULL,
  `timein` time NOT NULL,
  `timeout` time NOT NULL,
  `card_out` tinyint(1) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

10.GALLERY

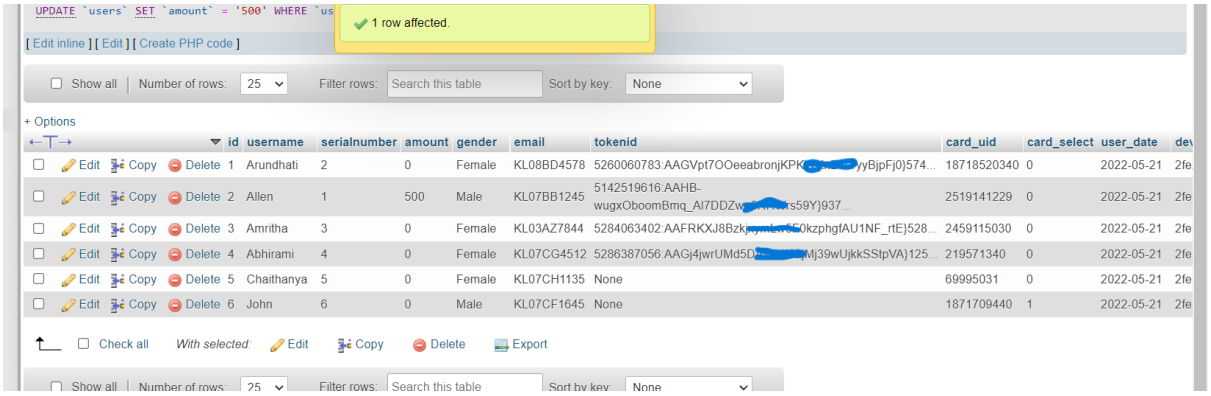


Driveway, Car slots, LCD Display

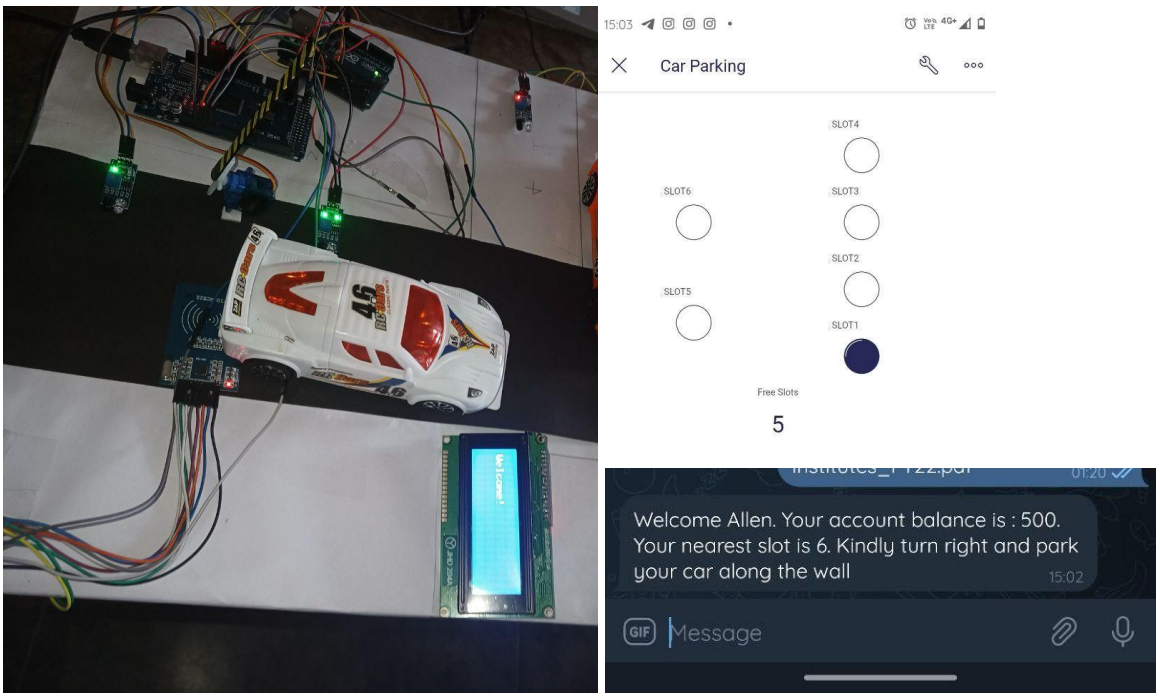


All 6 users Registered initially in the MySQL Database in phpMyAdmin interface. 0 balance for each user initially.

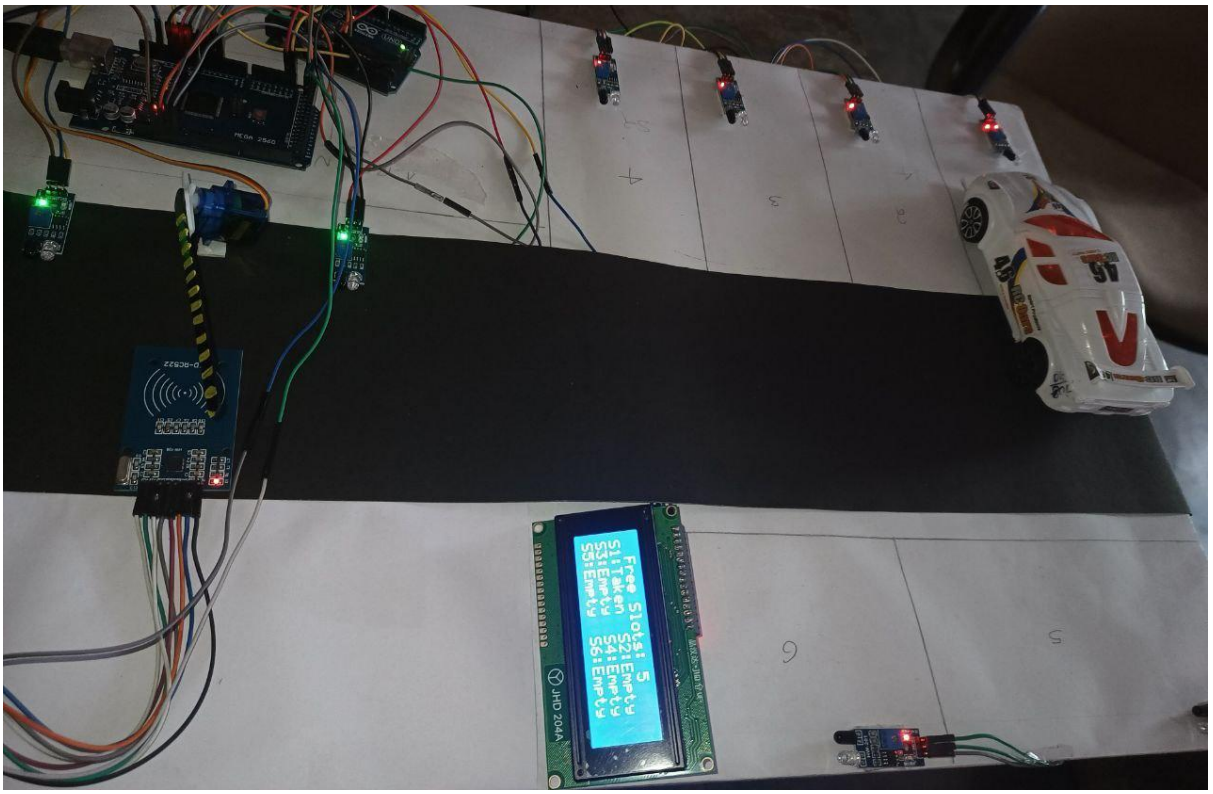
Entry denied



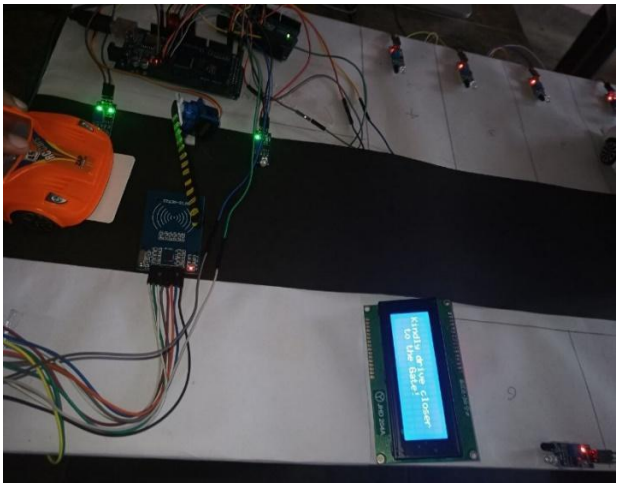
The same user is recharged with 500 credits.

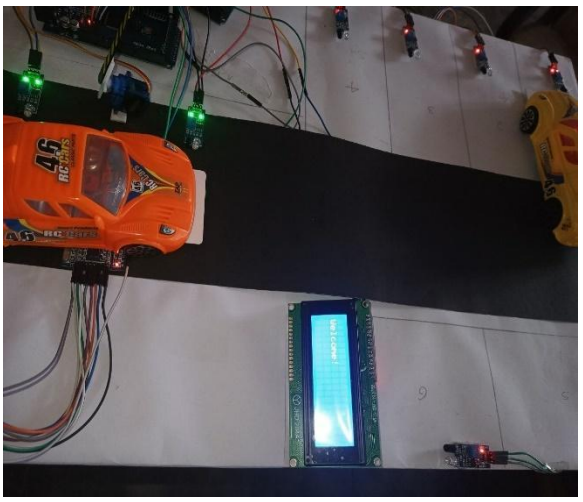


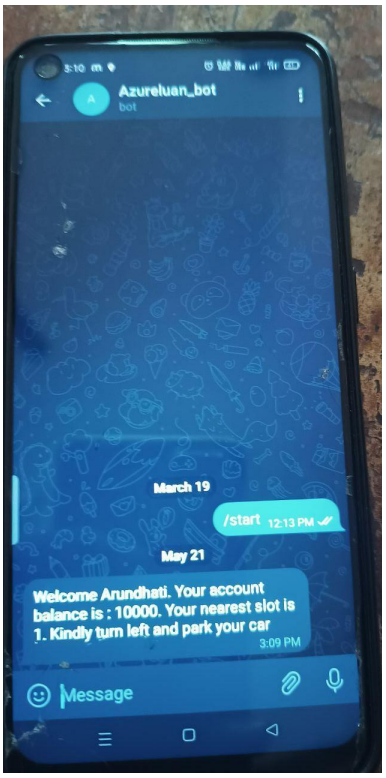
1. Gate opens and the user gains entry. 2. Blynk app 3. Received telegram message for the same.

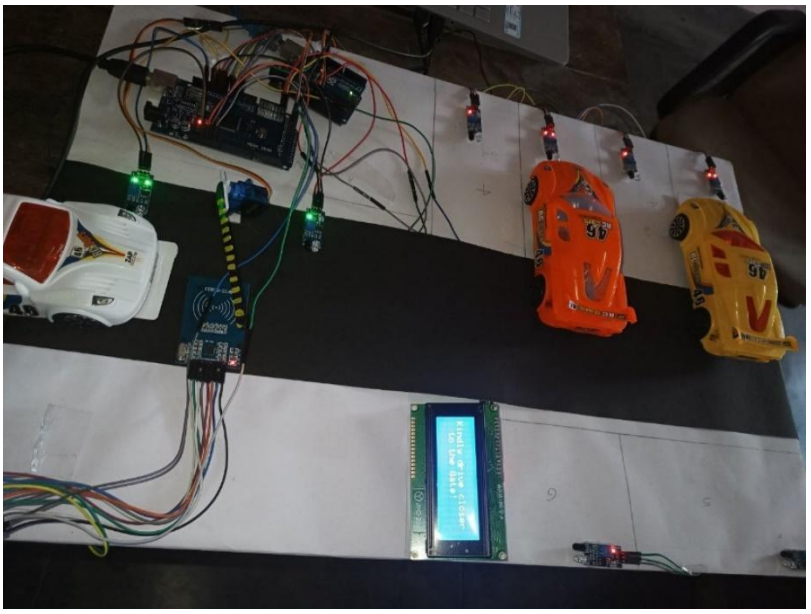


4.Car parked in a slot. 5. LCD Display updated.

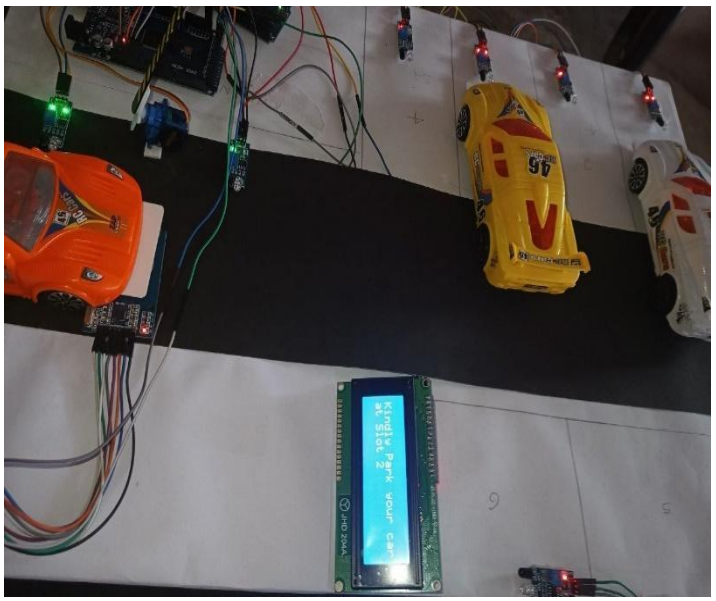


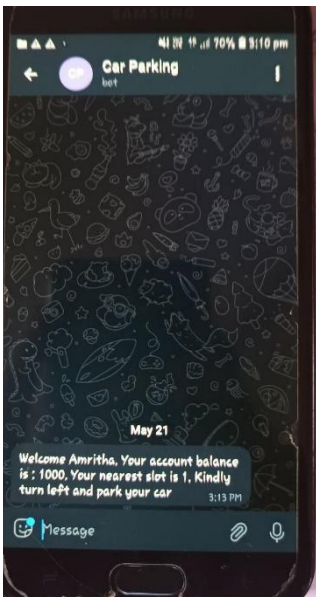


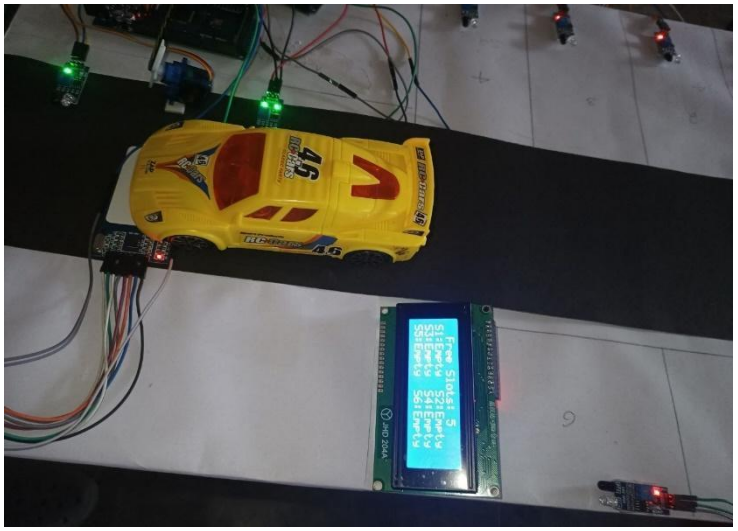




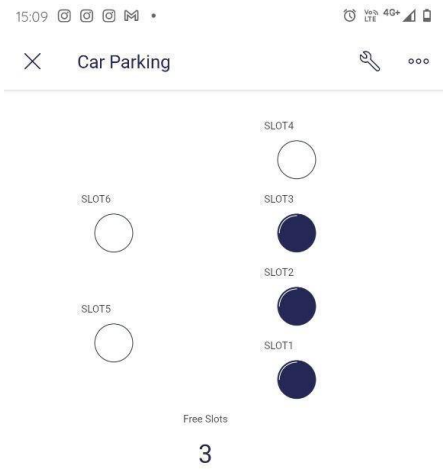
Entry of the next user.

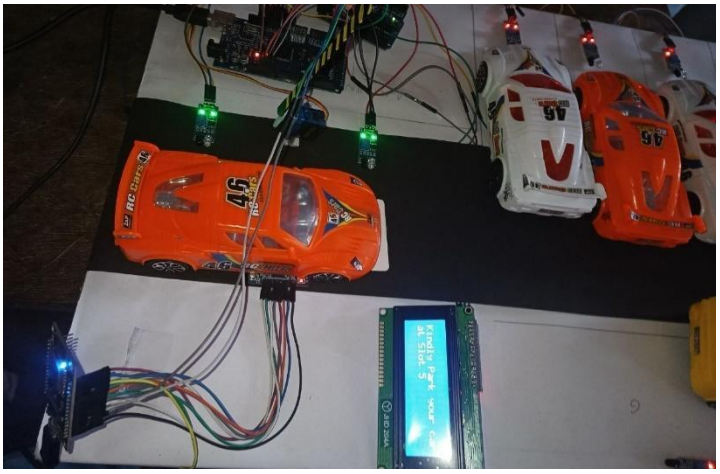

Kindly Park your car at Slot 2

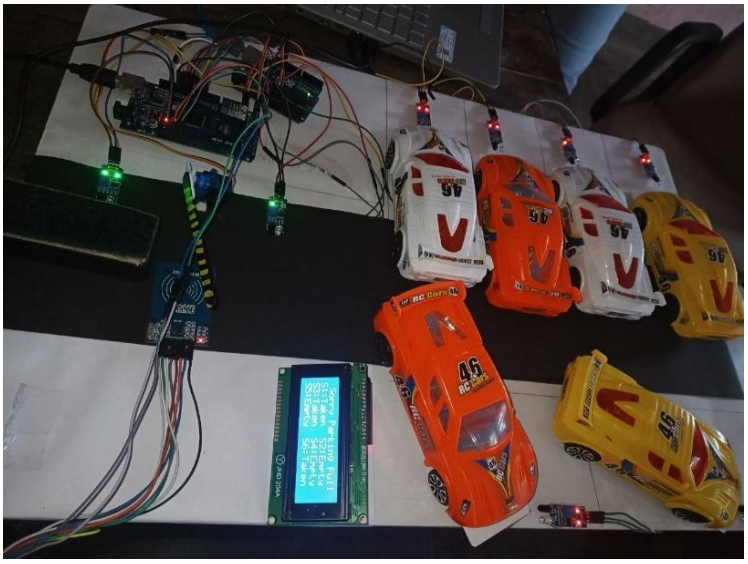

Car Parking
May 21
Welcome Amritha. Your account balance is : 1000. Your nearest slot is 1. Kindly turn left and park your car
3:13 PM
Message


Kindly Park your car at Slot 3

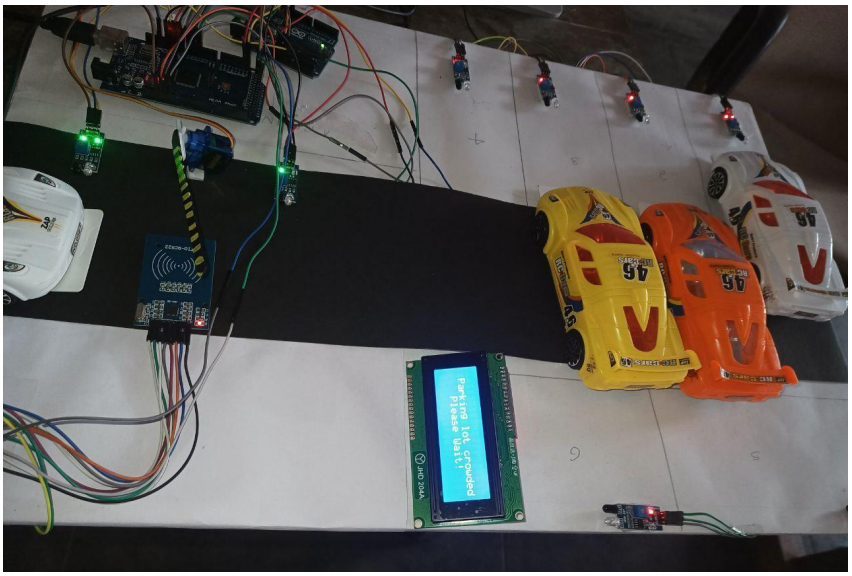

Car Parking
May 21
Welcome Amritha. Your account balance is : 1000. Your nearest slot is 1. Kindly turn left and park your car
3:13 PM
Dear Amritha. You have spent 119 seconds inside the parking lot. Your remaining balance is : 871
3:15 PM
Message


15:09
Car Parking
SLOT4
SLOT6
SLOT3
SLOT2
SLOT5
SLOT1
Free Slots
3

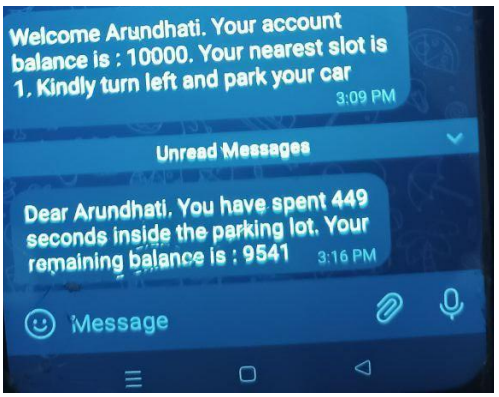

Kindly Park your car at Slot 5

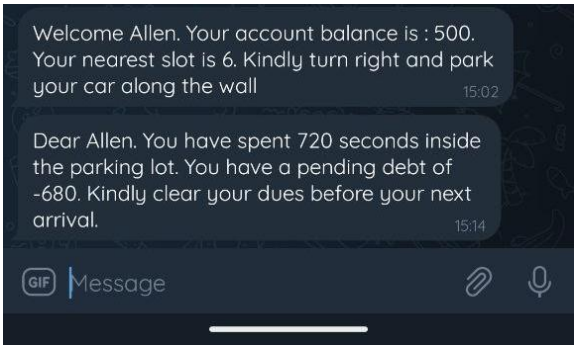


Parking FULL.

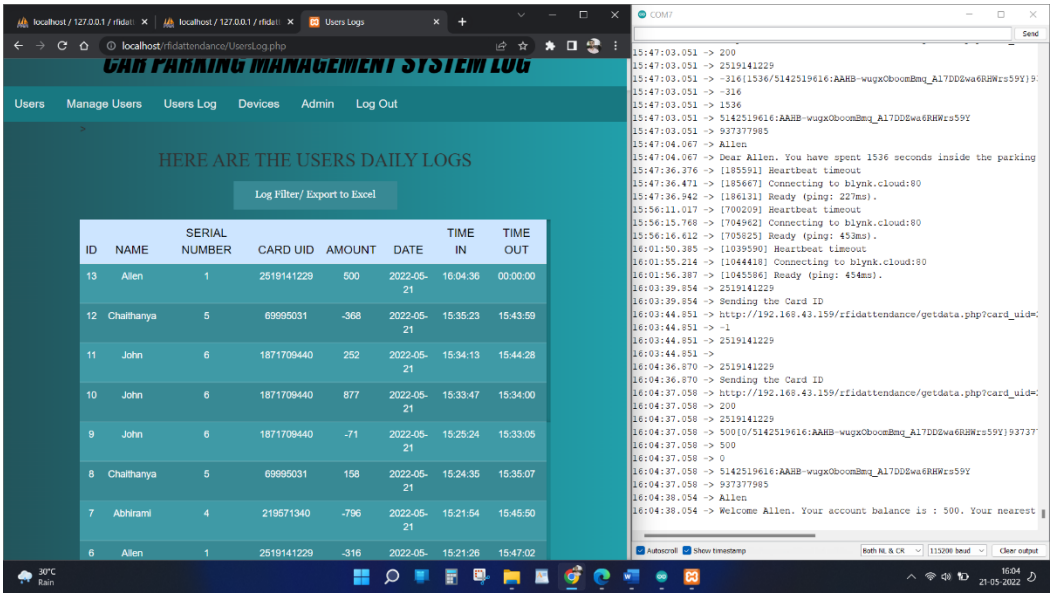


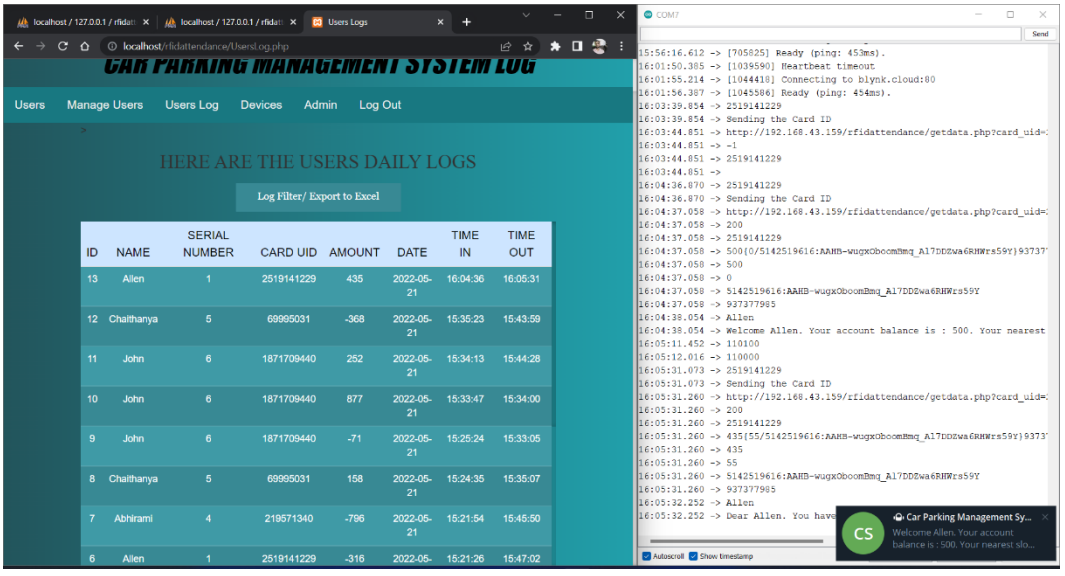
2 Cars are blocking the way => the parking lot is congested.



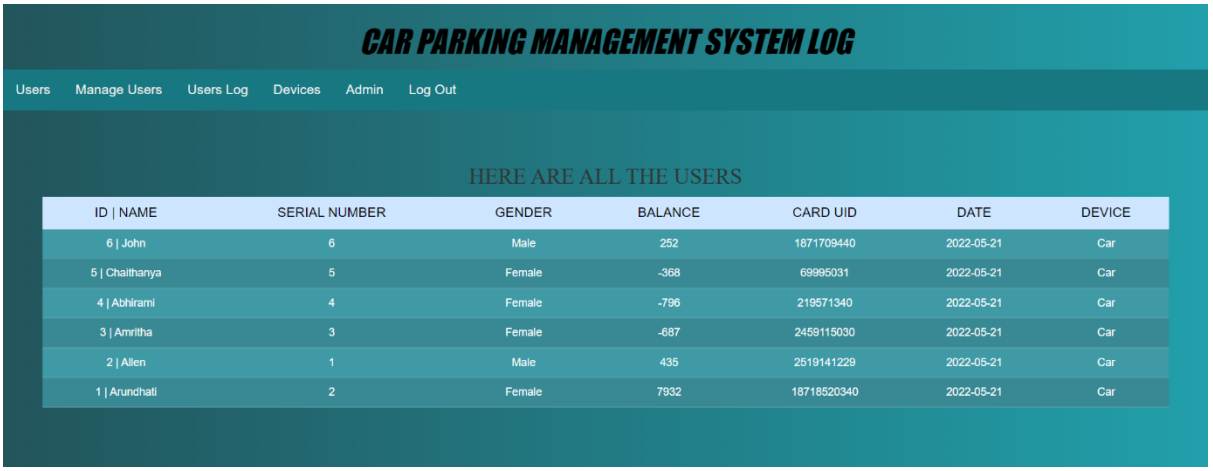


Bill received when 1. There is enough balance 2. There are dues.

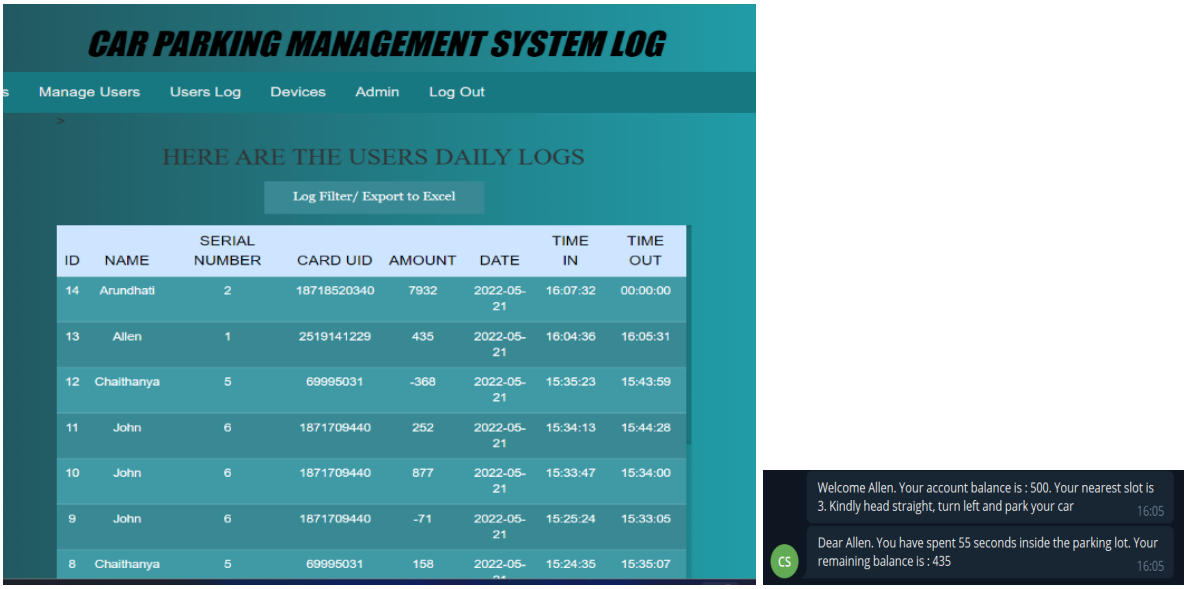


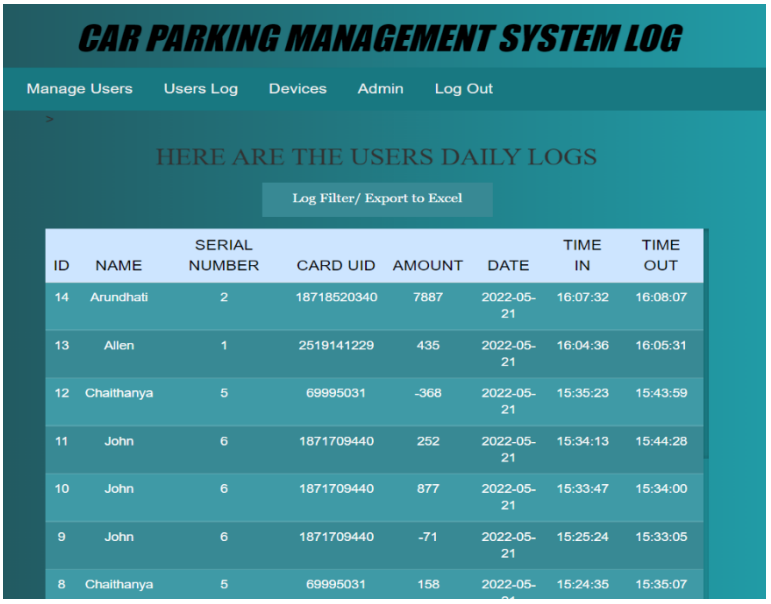


PHP web-app interface. 1. When User- Allen has entered. 2. When User Allen has left the parking lot. Real time user entry and exit with details as shown along with the serial monitor of NODE in Arduino IDE.



List of every user.





When the next user has entered and left the parking lot.

11. FUTURE SCOPE

- By implementing a reservation system, each user can reserve their slot before entering the parking lot. This can reduce the driver's anxiety even further as they don't have to face a situation where no parking lots are free upon reaching the parking space. The driver will be charged a certain sum and the reservation will have a time limit.
- By using a larger and more improved display, more information about the parking space can be displayed (Such as an OLED display which can show more information per sq.-inch).
- The server database can be converted to cloud-based database for better data security and data processing
- By installing small cameras (Such as CCTV cameras) into the system, the parking space can be converted to a more secure solution. This can however raise the price of the system.
- In general, parking lots contain more than 100 slots. This certainly wouldn't be supported in small MCUs like Arduino MEGA due to the shortage of pins. Hence, specialised MCUs are to be developed that can handle/manage the growth.
- As discussed earlier, working of the system requires uninterrupted power supply and internet connectivity. Efficient mobile power solutions are needed to be applied which can store energy and discharge to the system during power outages. In the case of the internet, the system should have provision for GSM.

12.CONCLUSION

The project has been tested numerous times and has worked successfully in most of the attempts. It has been anticipated that once the prototype is proved to be sustainable and reliable, it can be developed into a practical model with realistic scale components. This would however require higher power with higher capacities. The challenges that would arise in this case are to be discussed and resolved.

Thus, the project has been deemed to work in experimental conditions and can be used for illustrative purposes to show how a smart parking management system can be developed.

Furthermore, it has been observed that the principles and methods used for this project can be reproduced for a wide variety of applications in the real world such as theatres, retails shops, restaurants, hotels, libraries etc. In all these instances, the user's entry can be determined based on the crowd inside, as well as letting the user know of the condition beforehand by means of smartphone apps, digitising the payment receipt, communicating with the users via messaging apps and collecting data about user activity thus ensuring proper revenue for the establishment.

13.REFERENCES

13.1. https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT

13.2. https://learn.sparkfun.com/tutorials/what-is-an-arduino/

13.3. http://store.arduino.cc/products/arduino-mega-2560-rev3

13.4. https://quartzcomponents.com/products/nodemcu-development-board

13.5. https://www.techopedia.com/definition/26992/radio-frequency-identification-reader-rfid-reader

13.6. https://www.sunrom.com/p/20x4-lcd-white-on-blue

13.7. https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/

13.8. https://www.allaboutcards.de/en/plasticcards/rfid-media/rfid-cards

13.9. https://circuitdigest.com/article/servo-motor-working-and-basics

13.10. https://www.electronicscomp.com/i2c-module-16x2-lcd-india

13.11. https://docs.blynk.cc/

13.12. https://www.tutorialspoint.com/mysql/mysql-introduction.htm

13.13. https://www.techtarget.com/searchdatamanagement/definition/SQL

13.14. https://www.javatpoint.com/php-tutorial

13.15. https://www.tutorialspoint.com/mysql/mysql-introduction.htm

13.16. https://www.phpmyadmin.net/

13.17. https://dbpedia.org/page/Telegram_(software)

13.18. https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot/blob/master README.md

13.19. https://en.wikipedia.org/wiki/XAMPP