A Mini Project Report

on

SMART CAR PARKING USING NODE MCU

by

PRAJAKTA MARKAD (T223042)
AKANKSHA MATE (T223043)
NITIN MORE (T223044)

Under the guidance of

MR. A. P. KSHIRSAGAR

Department of Computer Engineering,

ZEAL College of Engineering and Research, SAVITRIBAI PHULE PUNE UNIVERSITY

2019-2020

Date:

CERTIFICATE

This is to certify that,

PRAJAKTA MARKAD (T223042)
AKANKSHA MATE (T223043)
NITIN MORE (T223044)

of class T.E(C)COMP; have successfully completed their mini project work on "SMART CAR PARKING'' at **ZEAL College of Engineering and Research, Pune** the partial fulfillment of the Graduate Degree course in T.E at the department of **Computer Engineering**, in the academic Year 2019-2020 Semester –VI as prescribed by the **Savitribai Phule Pune University.**

MR. A. P. KSHIRSAGAR
Guide

Dr. SUNIL SANGVE
Head of the Department
(Department of Computer Engineering)

Acknowledgment

Completing a task is never a one-man task. It offers the results of valuable contribution of a number of individuals in a direct or indirect manner that helps in shaping and achieving an objective. We extend our sincere gratitude to all those who extend their fullest co-operation in formulating this project.

We express a deep sense of gratitude to our project guide Mr. A. P. Kshirsagar for his efforts in giving start and keen interest, criticism. His valuable guidance was indeed a source of inspiration for us. We are thankful to him for lending his precious time and patient listening he gave each time.

We are also very thankful to our head of computer engineering department Dr. Sangve sir. He always motivated all of us for development of innovative ideas of us. He managed to provide us with knowledge about new technology in market and motivated us to use the same.

We have deep feeling of gratitude for our principle Dr. Kate sir and our whole management. He has best knowledge of mindset of students and today's cutting-edge technology. He always provided us new technology like fast internet for information.

We always have special place for our Lab Assistants who are very friendly and always helped us in any system problems and hazards. They provided us knowledge about correct way of using systems in laboratories. They always tracked laboratories for faulty machines and got repaired for students. Big thanks for them.

And lastly, thanks to all our friends who contributed in this project work and made possible to work seamlessly. Everyone who worked for this project day-night so this project my help many people in future after some integration. Without friends and our family's contributions, we couldn't have made this far.

PRAJAKTA MARKAD (T223042)
AKANKSHA MATE (T223043)
NITIN MORE (T223044)

Contents

Abstract

Parking has become a major problem in metro cities like Chennai, Mumbai, and other big cities, especially for the parking spaces for hotels, restaurants, and movie theatres.
The aim of the project is to design an intelligent system that keeps a track of vacant parking spaces and notify whether the parking is full or not to avoid wastage of time and fuel to find an empty spot in a parking lot.

The person responsible for handling the system will receive messages in an application to keep him informed about the status of the parking lots.

This would help to save time, man-power, confusion and it would make the management of parking easier, as the person responsible would not have to manually check for the slots.

Chapter 1

Introduction

Parking has become a major problem in metro cities like Pune, Delhi , especially for the parking spaces for hotels, restaurants and movie theatres. The number of cars to be managed per person in a public parking system is increasing regularly.

A report tells that 45% of the total traffic is generated by vehicles searching for parking spaces.

Due to rapid increase in the vehicles there exists a problem for parking of vehicles, and the resources for parking are also limited. It leads to traffic congestion and also pollution. So, we have a need to maintain the vehicle park management in order to reduce the wastage of time.

If we see in the larger cities when we visit the shopping malls or tourist places or any other commercial areas there arises a problem for parking of our vehicle. We have so many methods of parking systems such as using WSN, RFID methods. But the major drawback of those systems is they help us to find the available spaces for parking but not the exact location of those spaces. It can be overcome by using Automatic Car Parking System using IR Sensors.

The cars are considered as object by the IR sensors, and the first detection of the car prompts the servo motor to open the boom-barrier.

The parking manager gets a message on his app about the empty parking spaces and then he can direct the car to an available position.

The IR sensor sends infra-red light through the IR-LEDs, which is reflected by an object in front of it. At the disturbance, the LED lights up and a signal is sent to the ESP8266 Wi Fi-module board and an action is taken accordingly.

When the vehicle is exiting the parking space, the IR sensor again sends a signal, which then prompts the Servo motor to open the gate and after the gate is open and the car is leaving, the manager is informed of the empty parking space.

If the parking is full, consequent message is also sent to the manager.

If a slot is full, that message is also sent through the app.

Chapter 2
Literature Review

The proposed study involves the designing and developing of a partially-automatic parking system based on electronic devices only, which will be highly helpful for developing an efficient, cost-effective and easy to manage car parking system.

With the advancement in the car parking systems the problem of finding vacant parking spaces and all the hassle is going to deprive.
In this project, we have identified the need for an electronically working, moderate to long range obstacle sensor which when triggered, a barrier to allow entry for the vehicle into the car park is lifted.

We have also identified the need for method to keep the car park manager up – to – date with the current status of the parking slots.

The circuit is activated when the IR sensor detects a disturbance and keeps working till all sensors are once again in an inactive state, that is, there is no disturbance.
Once a disturbance is detected the working of the circuit starts and ends when there are no disturbances.

One major limitation of manual parking is the inability of a person to remember the places which are full and places where a car can be parked. So, the use of messages to keep track of the parking spaces overcomes this limitation that is imposed by the manual work.

Each car is detected by a sensor, and they are placed strategically such that other ting objects will not be considered as obstacles. We present the general architecture of the circuit with description of the sensors, their operation as well as the implementation. We create the messaging system such that a message of the status of the lot is sent after a certain interval of time.

Chapter 3
System Analysis

We have designed an automated parking system prototype as a project to minimize human efforts and make the system efficient and fast.
The requirements for this are:

- *Hardware:*
  - IR sensors
  - Arduino Uno
  - Minimum 2 GB RAM
  - Micro Servo motor
  - Node MCU (ESP8266)
  - Breadboard
  - Jumper Wires

- *Software:*
  - Windows 7 and above
  - Arduino IDE

EXPLAINATION:

NODE-MCU (ESP8266):

The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal

external circuitry, including the front-end module, is designed to occupy minimal PCB area.

The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts. The applications of ESP8266 are Smart power plugs, Home automation, Wi-Fi location-aware devices, Industrial wireless control, Security ID tags.

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core.

A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE".
This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

NodeMCU provides access to the GPIO (General Purpose Input/Output) and a pin mapping table is part of the API documentation.

GPIO pins:

I/O index	ESP8266 pin
0 [*]	GPIO16
1	GPIO5
2	GPIO4

3	GPIO0
4	GPIO2
5	GPIO14
6	GPIO12
7	GPIO13
8	GPIO15
9	GPIO3
10	GPIO1
11	GPIO9
12	GPIO10

Image:

- **IR SENSOR:**

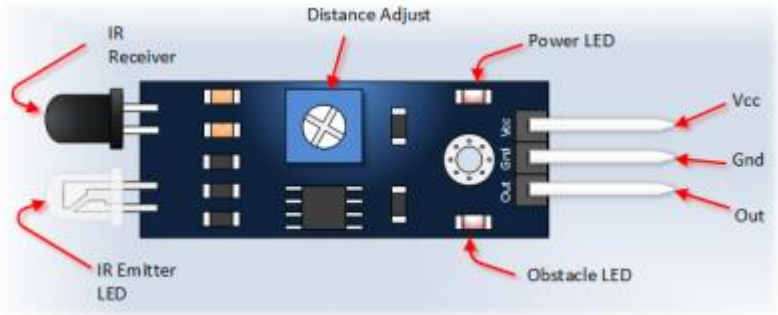
Infrared Obstacle Sensor Module has built-in IR transmitter and IR receiver that sends out IR energy and looks for reflected IR energy to detect presence of any obstacle in front of the sensor module.
The module has on board potentiometer that lets user adjust detection range. The sensor has very good and stable response even in ambient light or in complete darkness.

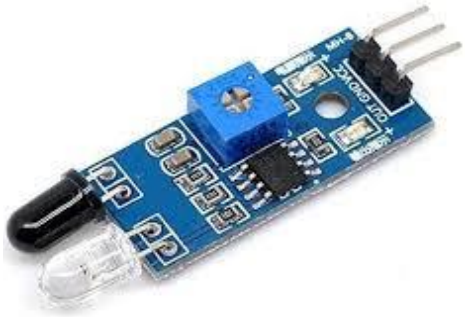
An IR sensor consists of an IR LED and an IR Photodiode; together they are called as Photo–Coupler or Opto–Coupler. As said before, the Infrared Obstacle Sensor has built-in IR transmitter and IR receiver. Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations. Hence, they are called IR LED's.

Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human eye. Infrared receivers are also called as infrared sensors as they detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors. Infrared Photodiodes are different from normal photo diodes as they detect only infrared radiation.

When the IR transmitter emits radiation, it reaches the object and some of the radiation reflects back to the IR receiver. Based on the intensity of the reception by the IR receiver, the output of the sensor is defined.

Image:





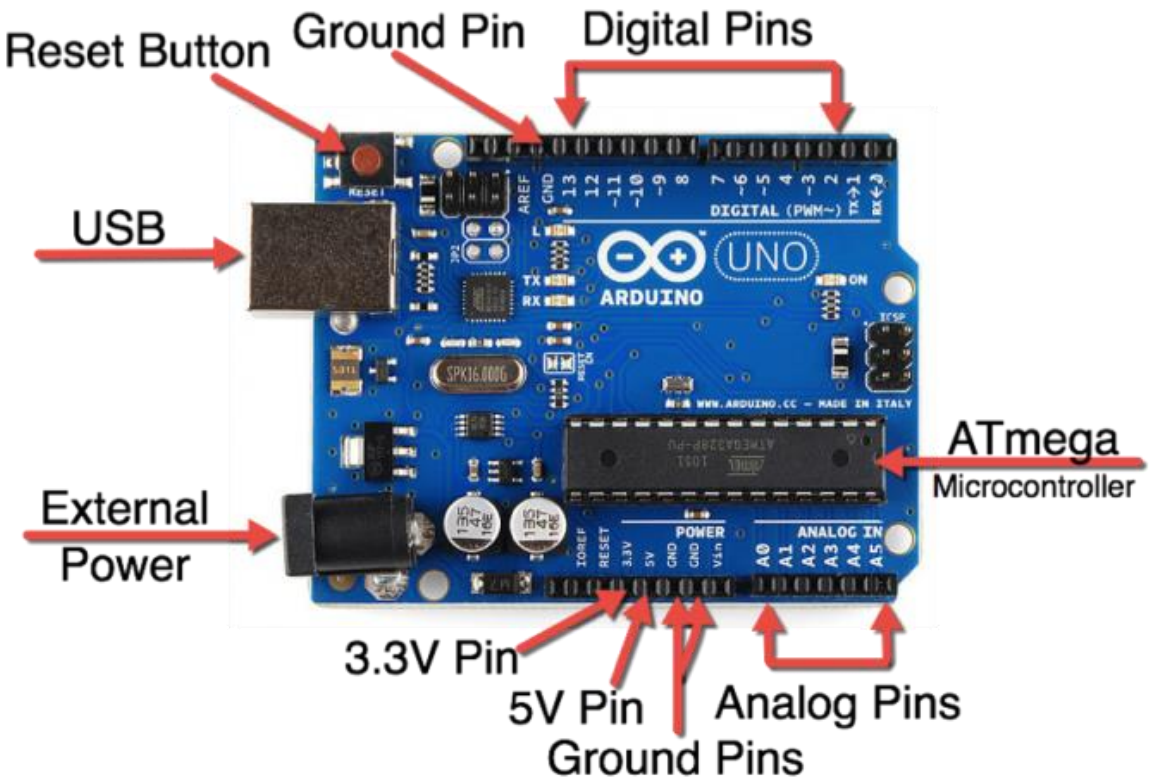
- **Arduino Uno:**

The Arduino Uno is the most common version of Arduino family. The Arduino Uno is a micro controller board based on the ATmega328.It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Uno is great choice for beginners. It contains everything needed to support the micro controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get

started. The Arduino Uno is a good choice for beginners since it is easy to start with.

Image:



- **USB:** The USB port is used to power the board from the computer's USB port and also to transfer the program code from computer into the Arduino microcontroller.

- **External Power:** It is used to power the board if the USB connector is not used.An AC adapter(9 volts,2.1mm barrel tip,center positive) could be used for providing external power.If there is no power at the power socket,then the Arduino will use power form the USB socket.But it is safe to have power at both the power socket and USB socket.

- **Digital Pins(I/O):** The Arduino Uno has 14 digital pins(0 to 13) of which the 6 are PWM(~).This pins can be either inputs or outputs.But we need to mention it in the Arduino sketch(Arduino programming).The PWM(Pulse Width Modulated) pins acts as normal digital pins and also used to control some functions. Say for example, control the dimming of LED and control the

direction of servo motor. Both digital inputs and digital outputs can read one of the two values either HIGH or LOW.

- **Analog Pins:** The Analog pins(0 to 5) acts as inputs which is used to read the voltage in analog sensors such as temperature sensor,gas sensor,etc.Unlike digital pins which can only read one of the two values(HIGH or LOW),the analog inputs can measure 1024 different voltage levels.

- **ATmega Microcontroller:** The Arduino uses ATmega328 microcontroller. It is a single chip microcontroller created by Atmel. This chip works well with Arduino IDE.If damaged,this controller can be easily replaced.The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader).It has also 2 KB of SRAM and 1 KB of EEPROM.

- **3.3V Pin:** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **5V Pin:** The regulated power supply used to power the microcontroller and other components on the board. This can come either from an on-board regulator, or be supplied by USB or another regulated 5V supply.

- **Reset Button:** It is used to reset the microcontroller. Pushing this button will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino.

Arduino IDE:

A sketch is the name that Arduino uses for a program.It is the unit of code that is uploaded to and run on an Arduino board to execute the function like blinking a Led. To write a sketch,we need to install the Arduino Software known as Integrated Development Environment(IDE).

➢ Micro Servo Motor SG90:

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a

relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor, although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system.

Servomotors are used in applications such as robotics, CNC Machinery or automated manufacturing.

The micro servo motor SG90 is tiny and lightweight with high output power. This servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. It comes with a 3 horns (arms) and hardware.

To fully understand how the servo works, you need to take a look under the hood. Inside there is a pretty simple set-up: a small DC motor, potentiometer, and a control circuit. The motor is attached by gears to the control wheel.

As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction.

When the shaft of the motor is at the desired position, power supplied to the motor is stopped. If not, the motor is turned in the appropriate direction. The desired position is sent via electrical pulses through the signal wire.

The motor's speed is proportional to the difference between its actual position and desired position. So if the motor is near the desired position, it will turn slowly, otherwise it will turn fast. This is called proportional control.

Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement.

The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the motor determines position of the shaft, and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired 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.

For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counter clockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position.

Image:

- **Blynk Libraries:**

Blynk is the most popular Internet of Things platform for connecting any hardware to the cloud, designing apps to control them, and managing your deployed products at scale.

- With Blynk Library you can connect over 400 hardware models (including ESP8266, ESP32, Node MCU, all Arduinos, Raspberry Pi, Particle, Texas Instruments, etc.) to the Blynk Cloud.
- With Blynk apps for iOS and Android apps you can easily drag-n-drop graphic interfaces for any DIY or commercial project. It's a pure WYSIWG experience: no coding on iOS or Android required.
- Hardware can connect to Blynk Cloud (open-source server) over the Internet using hardware connectivity available on your board (like ESP32), or with the use of various shields (Ethernet, WiFi, GSM, LTE, etc). Blynk Cloud is available for every user of Blynk for free. Direct connection over Bluetooth is also possible.

Chapter 4
Design

The circuit uses Node MCU ESP8266 Wi-Fi, micro Servo Motor, IR sensors, and the Blynk application for notification.

When the car is first detected by the IR sensor, a Servo motor is set into motion. The servo motor moves the arm on the rotor which act as a boom barrier.

When the car moves ahead of the barrier, the Servo motor moves again and the barrier is closed.

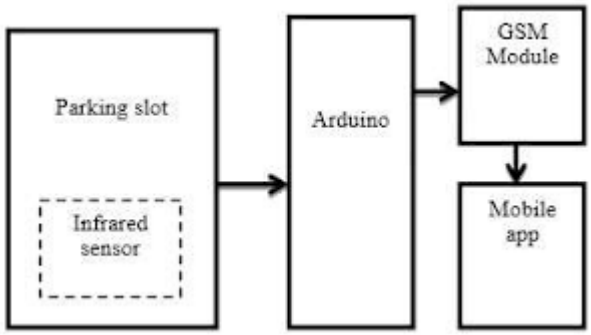
The messages for empty slots are sent to the Parking Manager via the Blynk app. So, on entering into the parking, the manager can tell the person where to park the car, without having to mover from his place at the check-post.

When the car enters the slot, again a message is sent to the manager that the particular slot is full.

The IR sensor first sends the command after detection to the Node MCU and then Node-MCU sends a command to the Blynk app.

When the car is leaving the parking lot, again the Manager gets a message that that slot is empty.

The Manager also gets a message informing him when the parking is full. He may close the Parking when all slots are empty, that is, when he gets the message Parking Empty.



Chapter 5
Code

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SoftwareSerial.h>
#include <SimpleTimer.h>
#include<Servo.h>

Servo myservo;
char auth[] = "ac173b0527c94a91a6cde0dcdfe6bdef";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "ZONG MBB-E8231-6E63";
char pass[] = "08659650";

SimpleTimer timer;

String myString;
 // complete message from Arduino, which consists of sensors data

char rdata; // received characters

int firstVal, secondVal,thirdVal; // sensors
int led1,led2,led3,led4,led5,led6;
// This function sends Arduino's up time every second to Virtual Pin (1).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.

Myservo.attach(D6);
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V1, millis() / 1000);

}
```

```
void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);

    timer.setInterval(1000L,sensorvalue1);
    timer.setInterval(1000L,sensorvalue2);
    timer.setInterval(1000L,sensorvalue3);
    timer.setInterval(1000L,sensorvalue4);
    timer.setInterval(1000L,sensorvalue5);
    timer.setInterval(1000L,sensorvalue6);


}

void loop()
{
   if (Serial.available() == 0 )
    {
  Blynk.run();
  timer.run(); // Initiates BlynkTimer
    }

  if (Serial.available() > 0 )
  {
    rdata = Serial.read();
    myString = myString+ rdata;
   // Serial.print(rdata);
    if( rdata == '\n')
    {
     Serial.println(myString);
  // Serial.println("Praj");
// new code
String l = getValue(myString, ',', 0);
String m = getValue(myString, ',', 1);
String n = getValue(myString, ',', 2);
String o = getValue(myString, ',', 3);
String p = getValue(myString, ',', 4);
String q = getValue(myString, ',', 5);
```

```
// these leds represents the leds used in Blynk application
led1 = l.toInt();
led2 = m.toInt();
led3 = n.toInt();
led4 = o.toInt();
led5 = p.toInt();
led6 = q.toInt();

  myString = "";
// end new code
    }
  }

}

void sensorvalue1()
{
int sdata = led1;
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V10, sdata);

}
void sensorvalue2()
{
int sdata = led2;
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V11, sdata);

}

void sensorvalue3()
{
int sdata = led3;
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V12, sdata);

}
```

```
void sensorvalue4()
{
int sdata = led4;
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V13, sdata);

}

void sensorvalue5()
{
int sdata = led5;
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V14, sdata);

}

void sensorvalue6()
{
int sdata = led6;
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V15, sdata);

}
String getValue(String data, char separator, int index)
{
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

Chapter 6
Conclusion

The design of this project is a representation of an innovative approach to reduce manual labour and work cost in the parking system.

All sensors used are available easily in the market and the app is easy to use.

Thus, the Smart Car Parking is a good approach to reduce manual effort, increase efficiency of work and also consequently reduce the risk of traffic jams.

Chapter 7
Future Scope

We can further improve it by making the system fully automated. The person can check for themselves using Cloud Service or Maps or through apps which parkings are relatively empty.

After that, they can choose and book the lot online. After using the parking space, online payment facility which works on Timer can be used.

References

https://www.youtube.com/watch?v=p06NNRq5NTU

https://www.youtube.com/watch?v=BfUtpScdQ9Y

https://components101.com/sensors/ir-sensor-module

https://electronicshobbyists.com/servo-motor-interfacing-arduino-arduino-servo-control/