**Bachelor of Technology (B. Tech.)**

**Computer and Communication Engineering (CCE)**

**Amrita School of Engineering**

**Coimbatore Campus (India)**

**Academic Year – 2022 - 23**

**Team 08**

| S. No | Name | Roll Number |
|---|---|---|
| 1 | B Ambareesh | CB.EN.U4CCE20006 |
| 2 | Narendran S | CB.EN.U4CCE20036 |
| 3 | Pabbathi Greeshma | CB.EN.U4CCE20040 |
| 4 | Santosh | CB.EN.U4CCE20053 |

**Fifth Semester**

**19CCE301 Internet of Things (IoT) Project**

# IoT-Based Smart Parking System Using Raspberry Pi

**Faculty In-charge – Peeyush K P Sir**

## ABSTRACT

Raspberry Pi is a miniature computer which has the adaptability with sensors. The automated car parking system uses the Raspberry Pi. This work proposes an automated car parking system that uses infrared sensors. The infrared sensor senses the movement of the vehicle and transmits the signal to the Pi. LED display is installed at the main entrance of the parking area to provide enough information about slot availability. In a real-time scenario, the proposed work can be enhanced further for multiple levels containing n number of slots with multiple Raspberry Pi.

**Keywords – Raspberry Pi, IR Sensor, LED Display.**

## INTRODUCTION

One of the most common problems today is the saturation of parking spaces. Vehicles continue to outnumber existing parking spaces, thus clogging roads. Incidences of violence over occupancy, deformed cars due to a space crunch, and overcharging for parking are some problems that result.

Most cities propose increasing parking spaces to combat the problem. Parks and vacant plots are used as potential parking spaces and multi-level facilities are being built, irrespective of the limited land space and resources.

But there exists a silly problem. People enter the parking and then come to know that it's full. Shouldn't it automate? Don't you think, we should already know if the parking has space for us or not? Yeah, isn't it a good thought? This project will help us show the availability of car slots to park the vehicle.

Hence space is utilized in the best efficient way i.e., maximum cars are accommodated in minimum space. The parking system built traditionally does not employ any intelligent monitoring system. Rather, human beings are employed to serve the purpose.

All vehicles after entering the parking area are found wasting time searching for the parking slots. Sometimes, a blockage is created. Since not even a single manpower is involved, maintenance and operation costs are low. This idea can be applied to the entrance of public places like malls, amusement parks, tech parks for visitors etc.

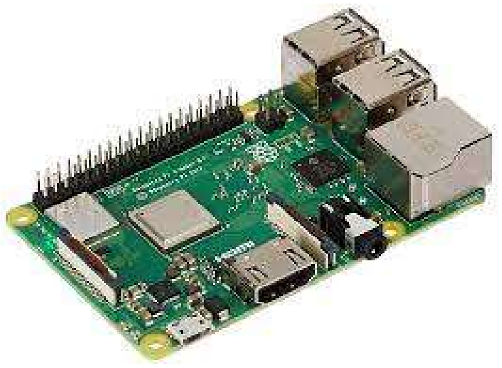
## COMPONENTS

| Hardware | Software Applications & Online Services | Hand Tools & Fabrication Machines |
|---|---|---|
| Raspberry Pi 3 – Adapter and HDMI Cable | Thonny Integrated Development Environment | Soldering Iron (Generic) |
| Two IR Photoelectric Sensor Module | Python 3 | Solder Wire, Lead-Free |
| Standard LCD Display – 16 X 2 Alphanumeric | Adafruit IO Cloud Service | Solder Flux, Soldering |
| Servo Motor | VNC Viewer (Optional) | |

|  |  |  |
|---|---|---|
| Bread Board & Connecting Jumper Wires |  |  |

**Raspberry Pi 3 – Adapter and HDMI Cable**

Raspberry Pi is a single-board computer. Raspberry Pi is a small-scale computer a size little bigger than a credit card, it has enough power to support games, a word processor like open office, image editor like Gimp.



It showcases a Broadcom system on a chip (SOC). It consists of an ARM-compatible CPU and a graphic processing unit (GPU). The speed of the CPU ranges from 700MHz to 1.2GHz for pi3. The onboard memory varies from 256MB to 1GB RAM. OS is stored in digital SD cards.

**IR Photoelectric Sensor Module**

An IR proximity sensor works by applying a voltage to a pair of IR light-emitting diodes (LEDs) which in turn, emit infrared light. This light propagates through the air and once it hits an object it is reflected towards the sensor. If the object is close, the reflected light will be stronger than if the object is further away.

The sensing unit, in the form of an integrated circuit (IC), detects the reflected infrared light, and if its intensity is strong enough, the circuit becomes active. When the sensing unit becomes active, it sends a corresponding signal to the output terminal which can then be used to activate any number of devices.

**Standard LCD Display – 16 X 2 Alphanumeric**

An LCD (Liquid Crystal Display) screen is an electronic display module and has a wide range of applications. A 16x2 LCD display is a very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in a 5x7 pixel matrix.

### Servo Motor

A 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.

## WORKING

At the entrance, the IR sensor detects the presence of a car. Based on the availability of the slots, an appropriate message is displayed in the LED display. Otherwise "no space available" message is displayed. After displaying the slot numbers, the automatic boom barrier allows the car to enter.

The car checks for the allotted slot and it is sensed by an IR sensor. This sensor checks the status of the slots and sends the availability details to the main control unit, at the entrance which is connected to a single network.

When the IR Sensor detects a car entering or leaving the parking area, the number plate of the car is stored in the database online and the same is displayed on the LCD module for the public. The data is also captured online, and statistics are advertised via Adafruit IO.

Additionally, a slot number for parking is provided to the car automatically and periodically updated when a car enters or leaves the parking area. Further, the car details along with the time can be captured using a Pi camera to calculate parking costs.

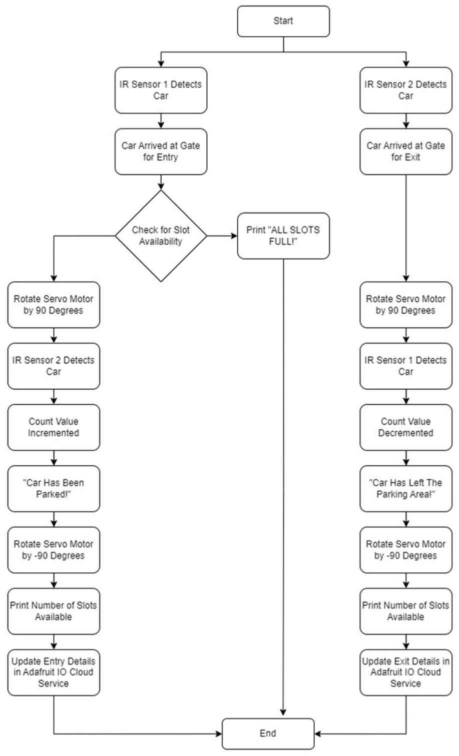

Start
IR Sensor 1 Detects Car
IR Sensor 2 Detects Car
Car Arrived at Gate for Entry
Car Arrived at Gate for Exit
Check for Slot Availability
Print "ALL SLOTS FULL!"
Rotate Servo Motor by 90 Degrees
IR Sensor 2 Detects Car
Count Value Incremented
"Car Has Been Parked!"
Rotate Servo Motor by -90 Degrees
Print Number of Slots Available
Update Entry Details in Adafruit IO Cloud Service
Rotate Servo Motor by 90 Degrees
IR Sensor 1 Detects Car
Count Value Decremented
"Car Has Left The Parking Area!"
Rotate Servo Motor by -90 Degrees
Print Number of Slots Available
Update Exit Details in Adafruit IO Cloud Service
End

Figure 1 – Algorithmic Flow Chart

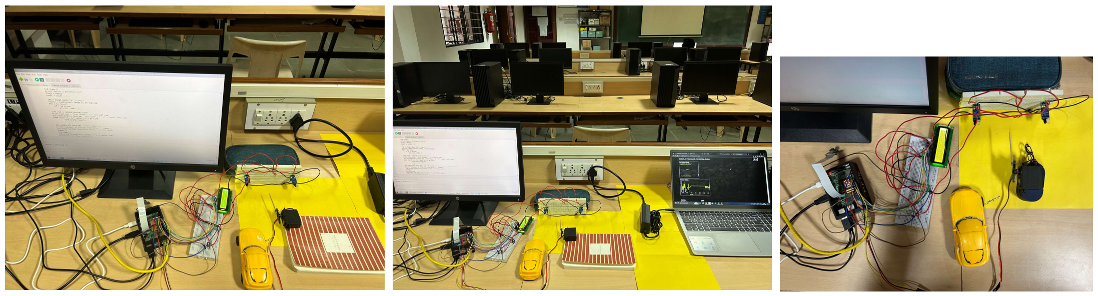


Figure 2 – Real Time Working

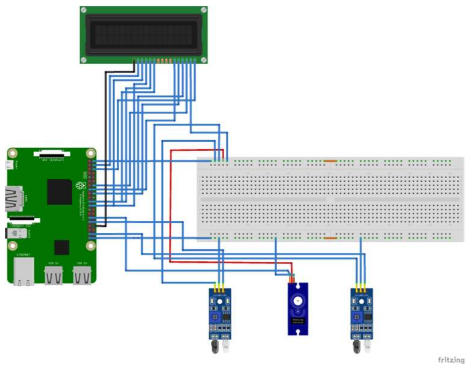


Figure 3 – Circuit Diagram 3D Representation

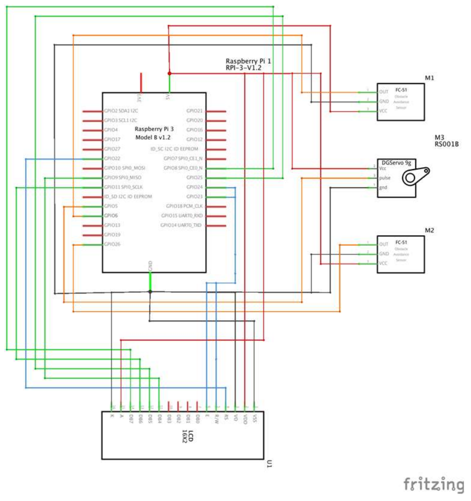


Figure 4 – Schematic Pin Diagram

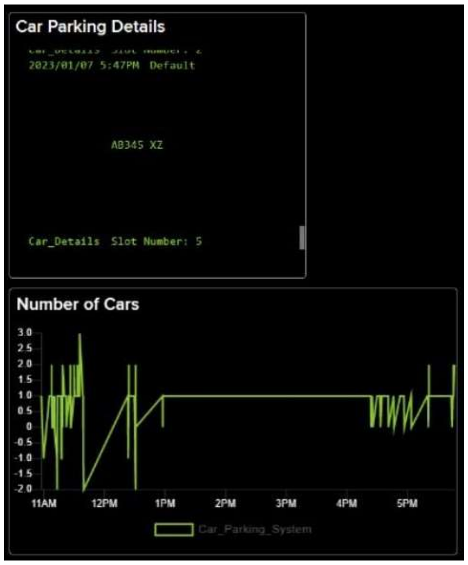


Figure 5 – Adafruit IO Dashboard

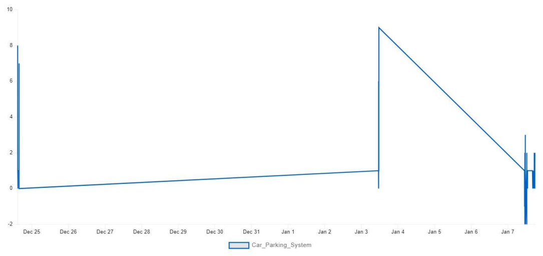
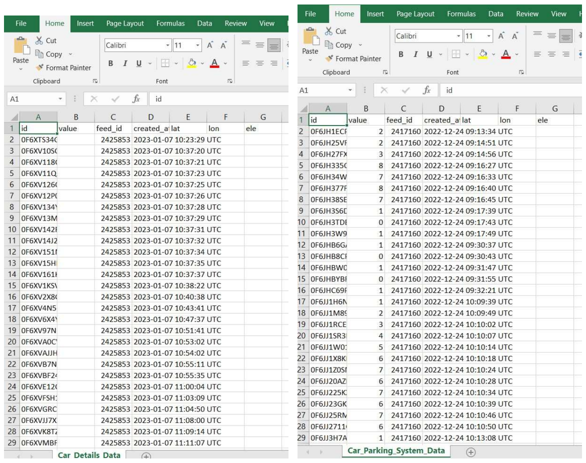


Figure 6 – Car Parking System Feed

Figure 7 – Sample Number Plate



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | id | value | feed_id | created_a | lat | lon | ele |
| 2 | 0F6XTS34( | | 2425853 | 2023-01-07 10:23:29 UTC | | | |
| 3 | 0F6XV10S( | | 2425853 | 2023-01-07 10:37:20 UTC | | | |
| 4 | 0F6XV118( | | 2425853 | 2023-01-07 10:37:21 UTC | | | |
| 5 | 0F6XV11Q. | | 2425853 | 2023-01-07 10:37:23 UTC | | | |
| 6 | 0F6XV126( | | 2425853 | 2023-01-07 10:37:25 UTC | | | |
| 7 | 0F6XV12P( | | 2425853 | 2023-01-07 10:37:26 UTC | | | |
| 8 | 0F6XV134\ | | 2425853 | 2023-01-07 10:37:28 UTC | | | |
| 9 | 0F6XV13M | | 2425853 | 2023-01-07 10:37:29 UTC | | | |
| 10 | 0F6XV142I | | 2425853 | 2023-01-07 10:37:31 UTC | | | |
| 11 | 0F6XV14J2 | | 2425853 | 2023-01-07 10:37:32 UTC | | | |
| 12 | 0F6XV151I | | 2425853 | 2023-01-07 10:37:34 UTC | | | |
| 13 | 0F6XV15HI | | 2425853 | 2023-01-07 10:37:35 UTC | | | |
| 14 | 0F6XV161I | | 2425853 | 2023-01-07 10:37:37 UTC | | | |
| 15 | 0F6XV1KS\ | | 2425853 | 2023-01-07 10:38:22 UTC | | | |
| 16 | 0F6XV2X8( | | 2425853 | 2023-01-07 10:40:38 UTC | | | |
| 17 | 0F6XV4N5 | | 2425853 | 2023-01-07 10:43:41 UTC | | | |
| 18 | 0F6XV6X4\ | | 2425853 | 2023-01-07 10:47:37 UTC | | | |
| 19 | 0F6XV97N | | 2425853 | 2023-01-07 10:51:41 UTC | | | |
| 20 | 0F6XVA0C\ | | 2425853 | 2023-01-07 10:53:02 UTC | | | |
| 21 | 0F6XVAJJH | | 2425853 | 2023-01-07 10:54:02 UTC | | | |
| 22 | 0F6XVB7N | | 2425853 | 2023-01-07 10:55:11 UTC | | | |
| 23 | 0F6XVBF24 | | 2425853 | 2023-01-07 10:55:35 UTC | | | |
| 24 | 0F6XVE12( | | 2425853 | 2023-01-07 11:00:04 UTC | | | |
| 25 | 0F6XVFSH: | | 2425853 | 2023-01-07 11:03:09 UTC | | | |
| 26 | 0F6XVGRC | | 2425853 | 2023-01-07 11:04:50 UTC | | | |
| 27 | 0F6XVJJ7X | | 2425853 | 2023-01-07 11:08:00 UTC | | | |
| 28 | 0F6XVK8T: | | 2425853 | 2023-01-07 11:09:14 UTC | | | |
| 29 | 0F6XVMBI | | 2425853 | 2023-01-07 11:11:07 UTC | | | |

Car Details Data

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | id | value | feed_id | created_a | lat | lon | ele |
| 2 | 0F6JH1ECI | 2 | 2417160 | 2022-12-24 09:13:34 UTC | | | |
| 3 | 0F6JH25VI | 2 | 2417160 | 2022-12-24 09:14:51 UTC | | | |
| 4 | 0F6JH27FX | 3 | 2417160 | 2022-12-24 09:14:56 UTC | | | |
| 5 | 0F6JH335( | 8 | 2417160 | 2022-12-24 09:16:27 UTC | | | |
| 6 | 0F6JH34W | 7 | 2417160 | 2022-12-24 09:16:33 UTC | | | |
| 7 | 0F6JH377I | 8 | 2417160 | 2022-12-24 09:16:40 UTC | | | |
| 8 | 0F6JH38SE | 7 | 2417160 | 2022-12-24 09:16:45 UTC | | | |
| 9 | 0F6JH3S6I | 1 | 2417160 | 2022-12-24 09:17:39 UTC | | | |
| 10 | 0F6JH3TDI | 0 | 2417160 | 2022-12-24 09:17:43 UTC | | | |
| 11 | 0F6JH3W9 | 1 | 2417160 | 2022-12-24 09:17:49 UTC | | | |
| 12 | 0F6JHB6G/ | 1 | 2417160 | 2022-12-24 09:30:37 UTC | | | |
| 13 | 0F6JHB8CI | 0 | 2417160 | 2022-12-24 09:30:43 UTC | | | |
| 14 | 0F6JHBW( | 1 | 2417160 | 2022-12-24 09:31:47 UTC | | | |
| 15 | 0F6JHBYBI | 0 | 2417160 | 2022-12-24 09:31:55 UTC | | | |
| 16 | 0F6JHC69I | 1 | 2417160 | 2022-12-24 09:32:21 UTC | | | |
| 17 | 0F6JJ1H6N | 1 | 2417160 | 2022-12-24 10:09:39 UTC | | | |
| 18 | 0F6JJ1M8( | 2 | 2417160 | 2022-12-24 10:09:49 UTC | | | |
| 19 | 0F6JJ1RCE | 3 | 2417160 | 2022-12-24 10:10:02 UTC | | | |
| 20 | 0F6JJ1SR3I | 4 | 2417160 | 2022-12-24 10:10:07 UTC | | | |
| 21 | 0F6JJ1W0: | 5 | 2417160 | 2022-12-24 10:10:14 UTC | | | |
| 22 | 0F6JJ1X8KI | 6 | 2417160 | 2022-12-24 10:10:18 UTC | | | |
| 23 | 0F6JJ1Z0SI | 7 | 2417160 | 2022-12-24 10:10:24 UTC | | | |
| 24 | 0F6JJ20AZI | 6 | 2417160 | 2022-12-24 10:10:28 UTC | | | |
| 25 | 0F6JJ225K: | 7 | 2417160 | 2022-12-24 10:10:34 UTC | | | |
| 26 | 0F6JJ23GK | 6 | 2417160 | 2022-12-24 10:10:39 UTC | | | |
| 27 | 0F6JJ25RM | 7 | 2417160 | 2022-12-24 10:10:46 UTC | | | |
| 28 | 0F6JJ2711( | 6 | 2417160 | 2022-12-24 10:10:50 UTC | | | |
| 29 | 0F6JJ3H7A | 1 | 2417160 | 2022-12-24 10:13:08 UTC | | | |

Car_Parking_System_Data

Figure 8 – CSV Data from Adafruit

**PYTHON CODE**

```
# GPIO and Associated Peripherals: -
# # import board # Implements a general-purpose board structure which has the
functionality needed for a range of purposes
import RPi.GPIO as GPIO # Module to control the GPIO on a Raspberry Pi
```

```python
from RPLCD.gpio import CharLCD # A Raspberry Pi LCD library

# # Pi Camera Interfacing: -
# import picamera # Provides a pure Python interface to the Raspberry Pi
camera module
# picam = picamera.PiCamera()

from PIL import Image # Provides a class with the same name which is used to
represent a PIL image
import pytesseract # A python wrapper for Google's Tesseract-OCR (Optical
Character Recognition)
import random # Implements pseudo-random number generators for various
distributions

# # import time # Provides various time-related functions
from time import sleep # Suspends execution of the current thread for a given
number of seconds

# Import library and create an instance of REST client:-
from Adafruit_IO import Client
# from Adafruit_IO import Feed
ADAFRUIT_IO_USERNAME = 'XXXXXXXXXX' # Username
ADAFRUIT_IO_KEY = '.....XXXXXXXXXX.....' # Active Key

# 'XXXXX' is IO Feed created in Adafruit: -
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
cars_feed = aio.feeds('XXXXX')
details_feed = aio.feeds('XXXXX')

sensor1 = 6 # Pin to be connected to the IR sensor 1
sensor2 = 26 # Pin to be connected to the IR sensor 2
motor_pin = 5 # Pin to be connected to the motor

GPIO.setwarnings(False) # Disable warnings
GPIO.setup(sensor1, GPIO.IN) # Selected pin is configured as input
GPIO.setup(sensor2, GPIO.IN) # Selected pin is configured as input

# Direction Control of Servo Motor using Pulse Width Modulation (PWM): -
GPIO.setup(motor_pin, GPIO.OUT) # Selected pin is configured as output
servo = GPIO.PWM(motor_pin, 50) # Pin number, frequency in Hz
servo.start(0) # Start PWM signal generation with 0% duty cycle

# LCD Interfacing: -
lcd = CharLCD(pin_rs = 22, pin_rw=24, pin_e=23, pins_data= [9,25,11,8],
numbering_mode = GPIO.BCM, cols=16, rows=2, dotsize=8)
lcd.clear() # Clear the LCD
lcd.backlight = True # Turn backlight ON
```

```
# Necessary Variable Initializations: -
slots= [1,2,3,4,5,6,7,8,9,10]
alnum = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
count = 10; flag1 = flag2 = False; data = {}

while True:

    if (GPIO.input(sensor1) == False and flag2 == False and count > 0):
        print("Sensor 1 Detected (1)")
        flag1 = True
        sleep(0.5)

    if flag1 == True:
        servo.ChangeDutyCycle(7.5) # 90-degree turn

#         # Capture Number Plate Image using Pi Camera: -
#         picam.rotation = 180 # Retrieves or sets the current rotation of the
camera’s image
#         picam.start_preview(alpha = 200) # Retrieves or sets the opacity of
the renderer
#         time.sleep(5) # Suspends execution for the given number of seconds
#
#         picam.capture("Rotated_Image.png") # Capture an image from the
camera, storing it in the output
#         picam.stop_preview() # Hides the preview overlay
#         picam.close() # Finalizes the state of the camera

        if (GPIO.input(sensor2) == False):

            lcd.clear()
            print("Sensor 2 Detected (2)")
            flag1 = False
            count = count - 1

            img = Image.open("Sample_Number_Plate.jpg") # Opens and identifies
the given image file
            result = pytesseract.image_to_string(img) # Returns unmodified
output as string from Tesseract OCR processing

            # Print only necessary characters from the number plate: -
            number_plate = []
            for i in result:
                if i in alnum:
                    print(i, end='')
                    number_plate.append(i) # Place new items in the available
space
```

```
            str_number_plate = ''.join([str(elem) for elem in number_plate]) #
List comprehension
            slot_num = random.choice(slots) # Return a random element from the
non-empty sequence
            print_slot_num = "Slot Number: " + str(slot_num)

            # Send feed to Adafruit IO: -
            aio.send(details_feed.key, str(str_number_plate) + "\n" +
print_slot_num)
            aio.send(cars_feed.key, str(10-count))

            # Data Logging for Car IN: -
            with open('abc.txt', mode ='w') as file:
                     file.write(str_number_plate)
                     # print(result)

            # Write the specified Unicode string to the display: -
            print("\nNumber of Slot(s) Available: ", count)
            lcd.write_string("Available Slots: " + str(count))
            sleep(3); lcd.clear()
            lcd.write_string("Slot Number: " + str(slot_num))

            print("Car has been parked!")
            slots.remove(slot_num) # Removes the first occurrence of the
element with the specified value
            data[slot_num] = result
            servo.ChangeDutyCycle(10) # Neutral
            sleep(0.5)

    if (GPIO.input(sensor2) == False and flag1 == False and count<=10):
        print("Sensor 2 Detected (3)")
        flag2 = True
        sleep(0.5)

    if flag2 == True:
        servo.ChangeDutyCycle(7.5) # 90-degree turn

#          # Capture Number Plate Image using Pi Camera: -
#          picam.rotation = 180 # Retrieves or sets the current rotation of the
camera’s image
#          picam.start_preview(alpha = 200) # Retrieves or sets the opacity of
the renderer
#          time.sleep(5) # Suspends execution for the given number of seconds
#
#          picam.capture("Rotated_Image.png") # Capture an image from the
camera, storing it in the output
#          picam.stop_preview() # Hides the preview overlay
#          picam.close() # Finalizes the state of the camera
```

```
        img = Image.open("Sample_Number_Plate.jpg") # Opens and identifies the
given image file
        result = pytesseract.image_to_string(img) # Returns unmodified output
as string from Tesseract OCR processing

        if (GPIO.input(sensor1) == False):

            lcd.clear()
            print("\nSensor 1 Detected (4)")
            flag2 = False
            count = count + 1

            # Print only necessary characters from the number plate: -
            number_plate = []
            for i in result:
                if i in alnum:
                    print(i, end='')
                    number_plate.append(i) # Place new items in the available
space
            str_number_plate = ''.join([str(elem) for elem in number_plate]) #
List comprehension

            # Data Logging for Car OUT: -
            with open('abc.txt', mode ='w') as file:
                     file.write(str_number_plate)
                     # print(result)

            aio.send(cars_feed.key, str(10-count)) # Send feed to Adafruit IO
            print("\nNumber of Slot(s) Available: " + str(count) +"\nBYE....")
            lcd.write_string("Visit Us Again!"); sleep(3); lcd.clear()
            lcd.write_string("Available Slots: " + str(count)) # Write the
specified Unicode string to the display

            print("Car has left the parking area!")
            slots.append(slot_num) # Place new items in the available space
            data.pop(slot_num) # Removes the last value from the
list
            servo.ChangeDutyCycle(10) # Neutral
            sleep(0.5)
```

## CONCLUSION

This work deals with an automated car parking system which makes utilization of Raspberry Pi with Adafruit innovation. Which comprises a model that grants vehicle drivers to effectively find unfilled stopping openings. The proposed model comprises IR sensors which sense the nearness of the vehicle and further transmit signals to Raspberry Pi.

## REFERENCES

1. Automated Multilevel Car Parking System using Raspberry Pi with Zigbee, Dr G. G Sivasankari, Tanya Pallavi, Shiva Kumar B R, Vishwas J, Shital Verma, International Journal of Engineering Research & Technology (IJERT), ICACT – 2016 Conference Proceedings, ISSN: 2278-0181
2. IoT-based Smart Parking System Using Raspberry Pi – https://www.hackster.io/ajinkyagokhale/iot-based-smart-parking-system-using-raspberry-pi-c7b2dc
3. Smart Parking Lot Using Raspberry Pi – https://www.instructables.com/Smart-Parking-Lot-Using-Raspberry-Pi/
4. Smart Parking System Using Raspberry Pi – https://blog.adafruit.com/2022/07/01/smart-parking-system-using-raspberry-pi-raspberry_pi-piday-raspberrypi/
5. Car Parking System Using Raspberry Pi – https://www.youtube.com/watch?v=w-9ddDrOlew&ab_channel=RahulJadhav