TEESSIDE UNIVERSITY

SCHOOL OF COMPUTING, ENGINEERING, AND DIGITAL TECHNOLOGIES

Assignment Documentation

Proposed title: Parking Occupation Detection

Author:

Marwa Abdulqader (72046593).

Module:

Internet of Things

Module Assessor:

Dr. Santhosh Jayagopalan

Dec 2022

Teesside University

British Applied College
الكلية البريطانية التطبيقية

Abstract:

The following report is written to find practical solutions by providing smart parking that depends on electronic sensors that we have supplied, in addition to programming them to create a smart system for managing parking lots, and this is done through a series of steps and methodologies for systematic analysis and experimental research.

The study depends on the theoretical and practical sides, and the theoretical side deals with a theoretical overview of the problems that may impede the realization of the parking management system for cars, and for this, we have also clarified the causes of the problem and mentioned how to solve it if we encounter any of these obstacles. As for the practical side, it relied on selecting a type of smart parking, making a model for it, and adopting plans to allocate ranges for parking lots according to land uses. Also, a simplified model of the smart car park was manufactured to illustrate the idea of the parking lot, and computer programs were used to simulate the model.

Keywords:

IoT, Parking Occupation Detection, Ultrasonic, IR sensor, Jumpers.

Table of Contents

Introduction:

The internet of things, known as IoT, is a network of connected electronic equipment, digital and physical machinery, items, animals, or people which can exchange data across a network without needing human-to-human or human-to-computer contact.

The term "thing" refers to any natural or artificial object capable of being given an Internet Protocol address or in other word IP address and can transmit information over a network, including people with implanted heart monitors, farm animals with biochip transponders, cars with built-in pressure sensor monitors, and other examples.

Parking offers a secure and practical means to keep and access automobiles, making it a vital component of daily living. The present transportation system would suffer considerably if parking were absent, especially in cities where parking is already scarce.

Parking, although, is not without its issues. There is a severe lack of parking places in several cities, which causes crowding and congestion in locations where parking is accessible. Numerous problems, including increased air pollution, traffic noise, and congested roads, may result from this. Parking occupancy monitoring will be used to solve these problems and guarantee that parking remains a valuable component of our transportation system.

The Internet of Things, or IoT, is a technology that links real-world objects to the internet so that they can interact with one another and share information. It can be utilized in parking lots to keep an eye on and regulate how the spaces are used. For instance, sensors can be utilized to track when a car enters and exits a parking place, and this information can be used to modify each spot's charge in accordance with the request. Automatic car monitoring, payment methods, and automated ticketing are also a few additional uses for IoT in parking lots.

Problem domain and scenario:

Due to the growing rate of ownership of vehicles in public areas, parking has become a contentious and perplexing issue for many individuals. Parking issues, including airports, bus terminals, and retail malls, may happen everywhere. Inaccessible parking can harm nearby businesses and lower the standard of living for locals. Cities continuously review and assess the effectiveness of their parking initiatives since the significance of parking. Limited data regarding parking occupancy is one of the issues that individuals deal with daily.

People also waste time, money, and effort trying to find the perfect position. Additionally, the flow of traffic improves as fewer vehicles are needed to look for available parking spaces.

An Internet of Things (IoT)-based system is required for a smart parking system. This system relays information about available (and taken) parking spaces via a wired or wireless network and a web-based or mobile application. The Internet of Things gadget, which would have a controller as well as several sensors, would be dispersed throughout several separate parking spaces. Users would appreciate a real-time update of available parking spots and the ability to select the one that best suits their needs.

Top Reasons to Implement Parking Occupation detection:

Parking Management System is a prime requirement in almost all metropolitan cities of the world. Every vehicle owner must park his or her car in a secure designated parking slot available.

To escalate this problem there is a need for integrated and sophisticatedly developed parking management systems, which offer full-fledged parking services. Smart Parking involves the use of low-cost sensors, real-time data, and applications that allow users to monitor available and unavailable parking spots. The goal is to automate and decrease time spent manually searching for the optimal parking floor, spot, and even lot. Some solutions will encompass a complete suite of services such as online payments, parking time notifications, and even car searching functionalities for very large lots. A parking solution can greatly benefit both the user and the lot owner.

Here are some of the reasons to implement a parking management system:

- **Parking management optimization**
  Users should locate the best available spot, which will save time, resources, and effort and lead to the parking lot filling quickly, so commercial and corporate companies may make rational use of the available space.

- **Minimal traffic**
  Whenever fewer drivers are forced to move to find an available parking spot, traffic flow improves.

- **Enhanced User Experience**
  A smart parking solution will integrate the entire user experience into a unified action. Driver’s payment, spot identification, location search, and time notifications all seamlessly become part of the destination arrival process.

- **Integrated Payments and POS**

  Returning users can replace daily, manual cash payments with account invoicing and application payments from their phones. This could also enable customer loyalty programs and valuable user feedback.

- **Increased Safety**

  Parking lot employees and security guards contain real-time lot data that can help prevent parking violations and suspicious activity. License plate recognition cameras can gather pertinent footage.

  Also, decreased spot-searching traffic on the streets can reduce accidents caused by the distraction of searching for parking.

- **Real-Time Data and Trend Insight**

  Over time, a smart parking solution can produce data that uncover correlations and trends of users and lots. These trends can prove to be invaluable to lot owners as to how to make adjustments and improvements to drivers.

- **Decreased Management Costs**

  More automation and less manual activity save on labor costs and resource exhaustion.

- **Increased Service and Brand Image**

  A seamless experience can skyrocket a corporate or commercial entity's brand image to the user. Whether the destination is a retail store, an airport, or a corporate business office, visitors will surely be impressed with the cutting-edge technology and convenience factors.

  The main objective of the Parking Management System is to make the complex job simple. Moreover, a well-developed and planned system provides comprehensive scope to your parking needs. It essentially means that it offers a host of solutions. It not only tackles the parking citation work efficiently but also satisfies the tracking needs of the parking system.

Hardware and software requirements:

Software:

The IDE that will be used for the project will be Visual Studio, which is a source text editor, that expands automated processes, and debugging.

The programming language that will be used is Python, which is object-oriented with a dynamically organized software program language. It is mainly needed for software development as well as for procedure as a package to tie separate modules co-operatively due to its high-level made data sets.

Hardware:

The architecture of the Internet of Things:

The Internet of Things (IoT) is being used in increasingly more contexts due to the versatility of the underlying technology. The Internet of Things performs as intended in the many contexts for which it was built and developed. However, there isn't a commonly accepted, well-defined architecture for how it should be put to use. The capabilities and industry applications of IoT determine its underlying structure. Even yet, IoT is constructed after a fundamental process flow.

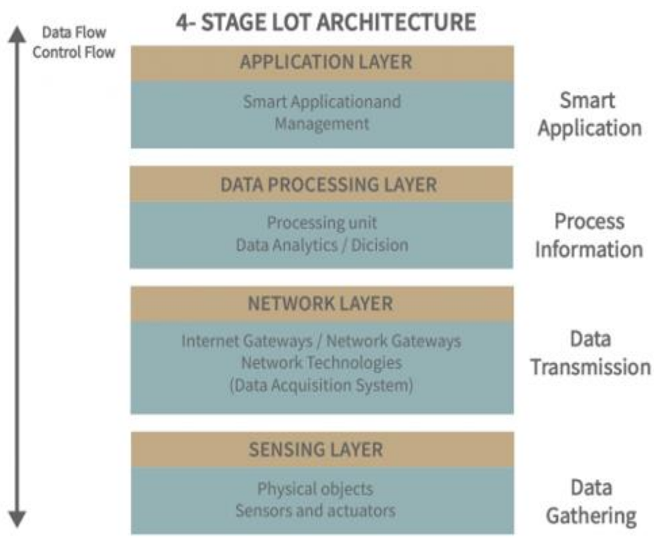


Figure1. IoT Architecture

Sensing Layer – This Sensing layer contains components such as sensors, actuators, and devices. These sensors or actuators take in input (physical or environmental factors), process that data, and then emits the processed data through a network.

Network Layer – This layer contains components such as the Data Acquisition System (DAS) and Internet/network gateways. DAS is responsible for the consolidation and transformation of data (Collecting data and aggregating data then converting analog data of sensors to digital data etc). In addition to bridging the gap between Sensor networks and the Internet, advanced gateways carry out several standard gateway tasks, such as filtering incoming data for malicious software, making decisions based on that data, providing data management services, etc.

Data processing Layer - That's the brains behind the Internet of Things. Data is evaluated and pre-processed here before being sent to a data center, where it is accessed by software applications—often referred to as business applications—that monitor and manage the data and prepare subsequent actions. That's where "Edge IT" (also known as "edge analytics") comes in.

Application Layer - It's the fourth and final layer of the IoT architecture. End-user applications in industries like agriculture, healthcare, aerospace, farming, defense, and so on all rely on data that is stored and processed in data centers or the cloud.

In this project, we will use the following components:

Raspberry Pi 4: The Raspberry Pi is a small, inexpensive device the size of a credit card that connects to a computer display and operates with a regular mouse and keyboard. With the help of this competent small gadget, individuals of all ages may learn about computing and be able to write in languages including Python and Scratch.

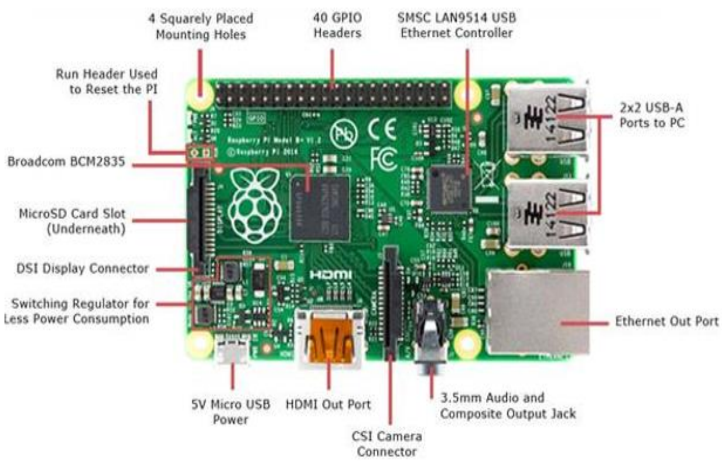


Figure2. Raspberry pi diagram

The primary command center for the parking occupancy detection system is the Raspberry Pi. It will be in charge of gathering data from the sensors and camera, evaluating the data, and notifying the user when a space becomes available.

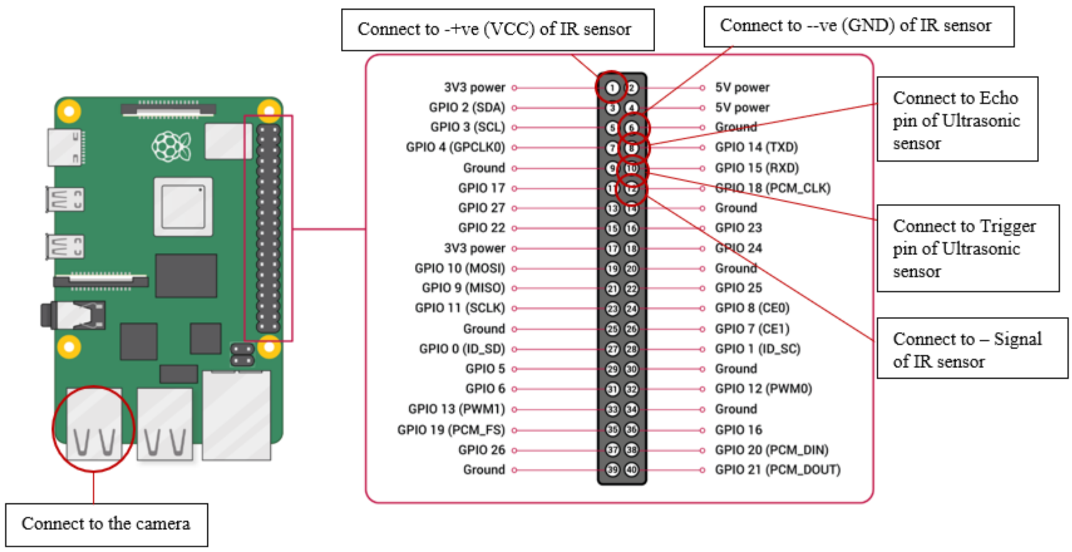


Figure3. Physical GPIO pin configuration

IR Infrared Obstacle Avoidance Sensor: To identify whether there are any vehicles in the parking lot, an IR sensor will be employed. When an automobile is nearby, the IR sensor will recognize the infrared energy it emits and notify the Raspberry Pi.

The Infrared Obstacle Avoidance Sensor is equipped with a set of infrared sensors that are both capable of transmitting and receiving infrared signals.

The infrared LED is responsible for emitting infrared signals at a specific frequency. If there is an obstruction in the path of the infrared light, the light will be reflected by the obstruction, and the receiver will be able to detect it.

The sensor will provide a low-level output signal in the OUT pin when it detects an obstruction, which will cause the LED indicator to light up. The sensor has a range of detection that goes from 2 to 30 centimeters. The potentiometer on the sensor allows for the detection distance to be set to a variety of different values.

Features:

- Detection angle: 35 °
- Voltage: DC 3-5V
- Detection distance: 2 ~ 30cm
- Dimensions: 41.3 x 14.3mm

Motion sensor: In a project to identify parking occupancy, a motion sensor will be essential as a spare sensor. It will be utilized to sense when a car is pulling into or pulling out of a parking place and to detect activity in a specified region. The motion sensor will be used to instantly notify a central system of the parking occupancy by sending an alert whenever a vehicle is spotted. The parking management system will then be updated with this data in order to maximize the parking possibilities offered to drivers.

A motion sensor, commonly known as a motion detector, is a piece of technology used to track and record movement. In addition to devices, dispensers, gaming consoles, and virtual reality headsets, motion sensors are commonly employed in residential and commercial security systems. Motion sensors are often embedded systems having three main parts: a sensor unit, an embedded computer, and hardware, in contrast to other different sensors, which can be handled and separated or the mechanical component.

Because motion sensors are able to be configured to carry out incredibly specialized tasks, these three components come in a variety of sizes and configurations. Motion sensors, for instance, can be utilized to turn on floodlights, set off audio alarms, turn on switches, and even call the police.

Motion sensors come in two varieties: active motion sensors, in which a transmitter and a receiver are both present in active sensors. This kind of sensor measures variations in the quantity of sound or radiation that is reflected back into the receiver in order to detect motion.

Additionally, there are passive motion sensors that do not have a transmitter. The sensor detects motion depending on a perceived rise in radiation in its surroundings rather than recording a steady reflection.

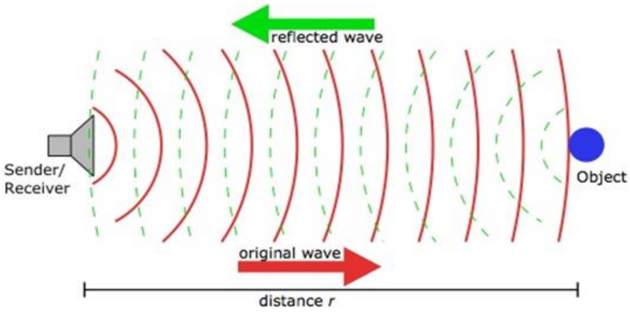


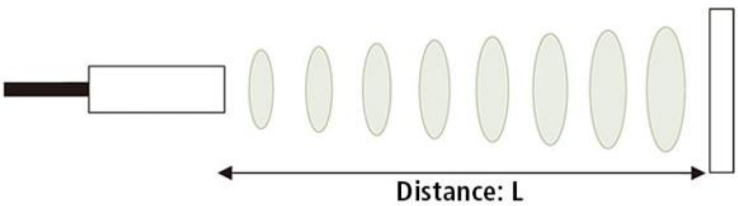
Figure4. Active ultrasonic motion sensor.

Ultrasonic Module Distance Measurement Transducer Sensor: The distance between the automobile and the parking place will be measured using the ultrasonic sensor. This will enable the Raspberry Pi to estimate the distance between the user and the parking and evaluate whether it is near enough to the available parking or not.

Ultrasonic sensors, also known as level sensors, get their name from the use of ultrasonic waves in the distance measurement process.

The sensor head is responsible for both the transmission of an ultrasonic wave and the reception of the wave after it has been reflected from the target. The distance to the target can be determined using ultrasonic or level sensors by measuring the amount of time that elapses between the signal's emission and reception.

Figure5. Ultrasonic waves



Camera sensor: The parking lot will be scanned with the camera. These pictures will be used by Raspberry Pi to locate open parking places and notify the user.

One such custom-made module for Raspberry Pi hardware is the Camera Board. It utilizes a bespoke CSI interface to connect to Raspberry Pi hardware. The sensor's native resolution for still images is 5 megapixels. It can record at a maximum of 1080p at 30 frames per second while in video mode. This camera module is perfect for portable applications due to its compact size and low weight.

Compulsory Apparatus:

You'll need the following gear to carry out this example:

- Electronics based on the Raspberry Pi
- A current-capable power source of at least 1A
- Something Like a Camera Board

Jumper Camera Board: A ribbon cable links the camera board to the Pi computer. A camera PCB is connected to the ribbon cable, and the other end is soldered onto the Raspberry Pi board. The camera won't turn on if you attach the ribbon cables backward. The camera PCB's blue backing must face away from the cable, while the Raspberry Pi hardware's blue backing.

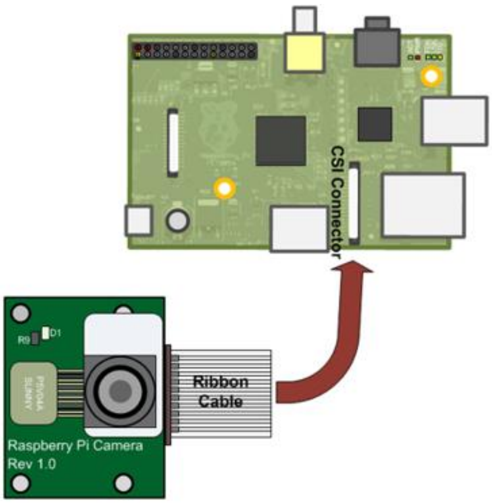


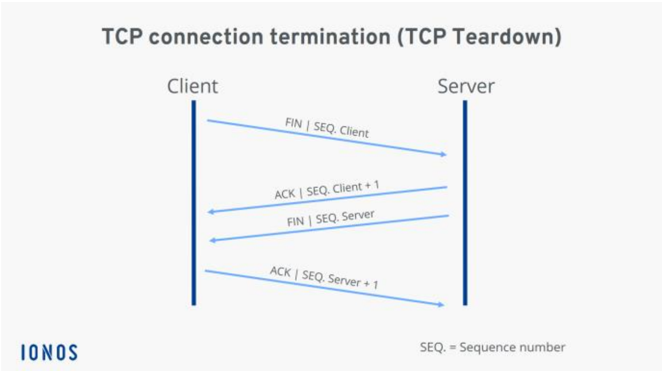
Figure6. Jumper camera board

TCP: is used to organize data so that it can be transmitted securely between the server and the client. It ensures the integrity of data delivered over the network, regardless of its size. As a result, it is used to transfer data from higher-level protocols that demand that all transmitted data arrive.

UDP (User Datagram Protocol), which runs on top of IP, is another comparable protocol (Internet Protocol). The distinction between TCP and UDP is that TCP is a connection-based protocol, whereas UDP is not. In other words, when TCP is utilized, a session is established between the hosts, and the transfer is ensured. Each data packet is sent via UDP, but there is no means to check whether it has been received or to resend it within the network layers. An application can run on top of UDP and do its own checks to ensure that each packet is received, but this is not the same as leaving it to the networking stack to do so.

TCP is commonly compared to the telephone system, while UDP is compared to the postal service. When you establish a link with the other person via the phone, you can be guaranteed that the message is received. If you were disconnected during the phone call, you would be aware of it and would be able to contact the other party again. When using the postal system, you never know for definite whether or not the mail will be delivered. The letter may be lost or damaged on its way to its destination after you have posted it. If the recipient has moved, they may never receive the letter.

At first glance, it may appear that there is no reason to prefer UDP over TCP; after all, if you can have the extra reassurance, why would you worry about UDP? The reason for this is that TCP involves a lot of overhead. A confirmation must be created for each data packet sent, and even if no data is being sent, there will frequently be some form of stay-alive signal. Whereas with less important material, you might just want to send and forget it, hoping it gets to the other end. The session might also be handled further up the networking stack (but I'm getting ahead of myself here).



VNC: is a program that allows you to remotely access your Raspberry Pi's graphical desktop. Setting up VNC is simple, but it normally only allows you to connect from another computer on the same network as your Raspberry Pi.

Implementation:

Setting up the Raspberry Pi 4 with the internet cable, power supply, mouse, keyboard, and monitor was the first step in the development process.

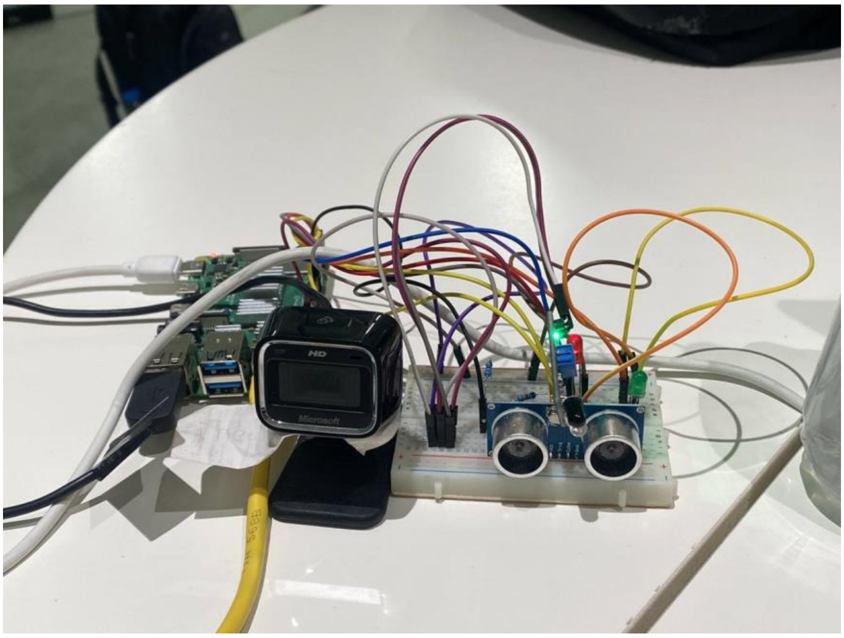


Figure8. The components

Next, we start to download the Raspberry Pi 4 operating system and download the official Raspberry Pi Imager.

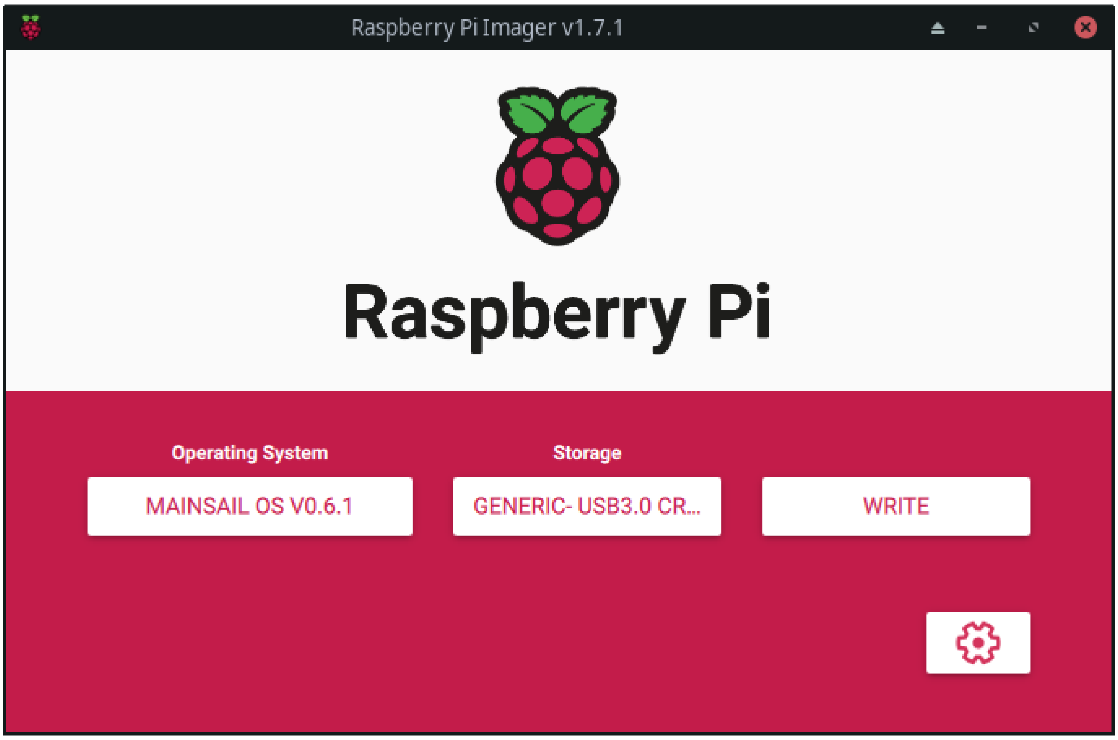


Figure9. Raspberry pi operating system

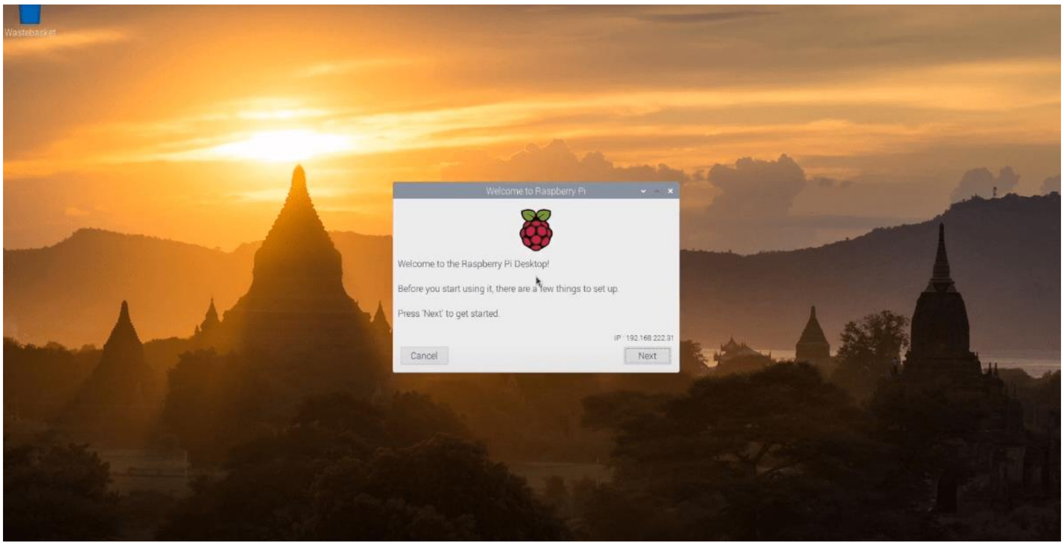


Figure10. Installing Raspberry pi

Now that the Raspberry Pi 4 is installed and running, we have downloaded python to use in this project.

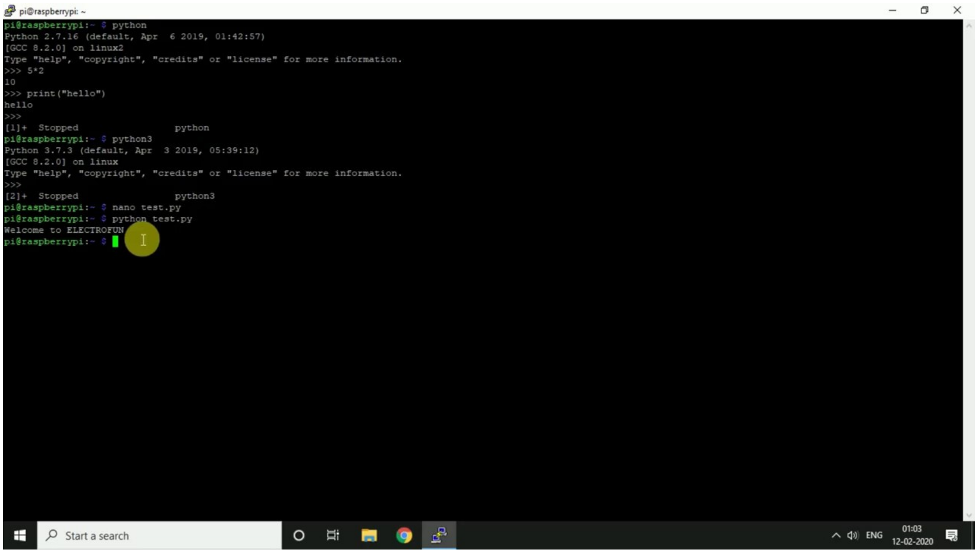


Figure10. Installing the libraries

We also downloaded the libraries we will be using in this project. The most important libraries that we will use:

1. RPi.GPIO Python Library

The General-Purpose Input/Output (GPIO) pins on a Raspberry Pi can be controlled using the Python library known as the RPi.GPIO library (RPi is short for Raspberry Pi). To operate with the Raspberry Pi and its GPIO pins, this library was created specifically. It offers a practical method to communicate with the GPIO pins, making it simple to operate hardware and read sensor data.

To control LEDs, buttons, sensors, and other electrical devices connected to the Raspberry Pi's GPIO pins, we can use the RPi.GPIO library. It's very effortless to use and can be easily incorporated into our Python code.

2. Cv2 Library

The cv2 (OpenCV) library is an effective open library that supports image and video processing, machine learning, and computer vision. It's widely used in a variety of projects, including robotics, surveillance systems, and self-driving cars.

The cv2 library on a Raspberry Pi 4 can be used to perform a variety of computer vision tasks, including image and video processing, object detection and tracking, and facial recognition. The library includes a set of functions for performing complex image processing and computer vision algorithms in Python.

Once cv2 is installed, we import it into our Python code and we can use the functions provided by the library. But in our project, we will use it to detect the license plate of the cars.

3. Pytesseract library

The Tesseract OCR (Optical Character Recognition) engine has a Python wrapper called the pytesseract library. It enables us to recognize text in images, extract text from images, and transform that text into machine-readable text using the Tesseract engine.

The pytesseract library can be used on a Raspberry Pi 4 to perform OCR on images. The library is suitable for a system like the Raspberry Pi because it is small and requires little processing power.

Installing both the library and the Tesseract OCR engine is required in order to use the pytesseract library on a Raspberry Pi 4.

Once installed, we can use pytesseract functions to perform OCR on images, text extraction from images, and text recognition in our Python code. Additionally, it supports preprocessing images before OCR and a variety of languages.

It is important to note that the quality of the image, lighting, and language of the text can all affect how accurate the OCR results are.

4. Firebase-admin library

The Firebase Admin Python Library is a package that enables programmers to communicate with Firebase services from a Python script or application, including the Firebase Realtime Database and Firebase Authentication. In order to read and write data to the Firebase Realtime Database, authenticate users, and carry out other Firebase-related tasks, the library offers a set of APIs.

The Firebase Admin Python Library, when installed on a Raspberry Pi 4, enables the device to connect to Firebase services and carry out operations from Python scripts running on the device, including reading and writing data to the database, authenticating users, and managing Firebase resources. When you want to store data in the cloud and access it from various devices as part of an IoT project, this can be helpful.

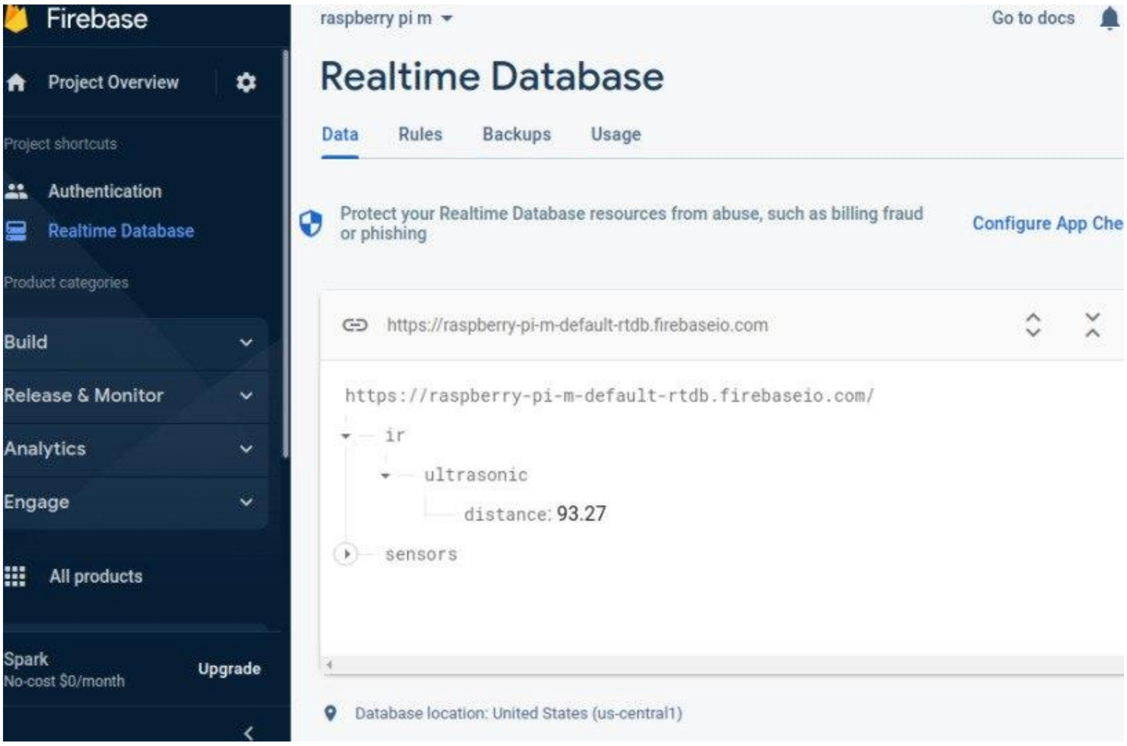


Figure11. Firebase-admin

```python
#include the necessary libraries
import RPi.GPIO as GPIO
import time
from camera import Cam
import firebase_admin
from firebase_admin import db
from firebase_admin import credentials
```

Figure12. importing the packages

```python
import RPi.GPIO as GPIO
import time
import cv2
import pytesseract
from pytesseract import Output
```

Figure13. importing the packages

Everything is in place; our Raspberry Pi is ready to start our project. The Raspberry Pi has been equipped and connected to all hardware devices and sensors, and the necessary libraries have been installed.

As we explained previously, to implement our project, we will need to connect the IR sensor with the ultrasonic sensor. The role of the IR sensor is to capture movements, and the role of the ultrasonic sensor is to measure the distance between the device and the moving object after the IR sensor detects the movement. So, they should work together.

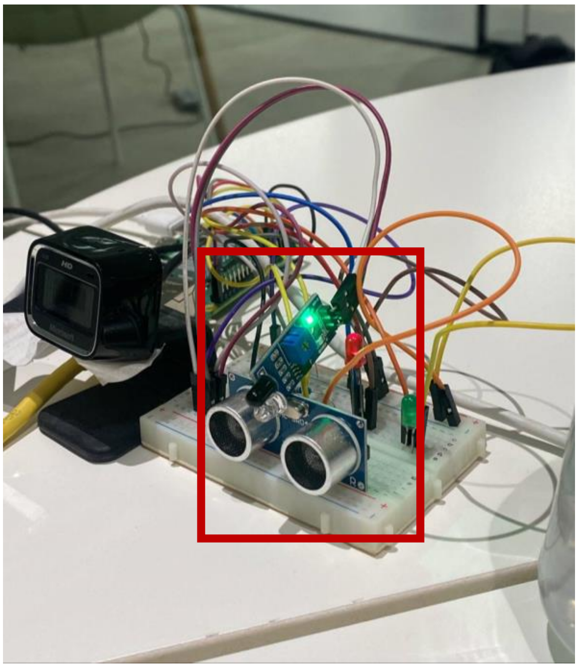


Figure14. Ultrasonic & IR sensor

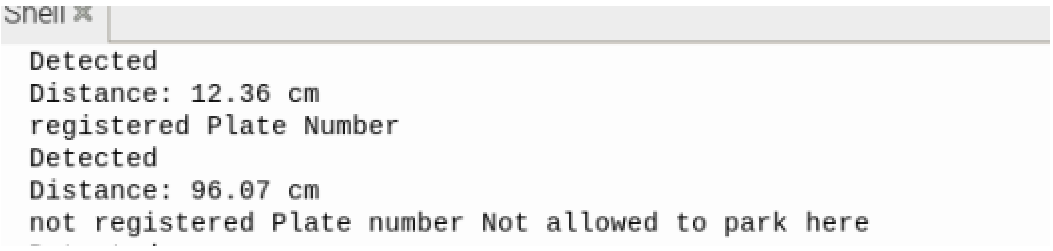
```python
# define a function to calculate the distance from the ultrasonic sensor
def measure_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

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

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

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
```

```python

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    print(f'Distance: {distance} cm')

    ref.set({'distance': distance})

    if distance < 500:
        Cam()

    return distance

```

```python
# define a function to detect if an object is detected by the IR sensor
def detect_object(ch):
    print("Detected")
    measure_distance()

```

Figure15. Ultrasonic & IR sensor code

After we made sure that both of the ultrasonic and the IR sensors are working together properly, we will write code to interface with the camera and ensure that it is functioning properly. Once the camera code is working as intended, we will integrate it with the ultrasonic sensor code. The ultrasonic sensor will be used to measure the distance to an object, and if the distance is close enough, the camera will be activated to capture an image.


Shell

```
Detected
Distance: 12.36 cm
registered Plate Number
Detected
Distance: 96.07 cm
not registered Plate number Not allowed to park here
```

Figure16. The output of Ultrasonic & IR sensor code

As previously mentioned, we will utilize the necessary libraries to analyze the images and extract the number of cars present. This information will then be compared against the data stored in our database to achieve the desired result. The desired result of this project is to determine if the cars captured in the image are registered within the system or not.

Figure17. The camera sensor

The code

```
def Cam():
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(24,GPIO.OUT)
    GPIO.setup(23,GPIO.OUT)
    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()
```

```
    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()

        d = pytesseract.image_to_data(frame, output_type=Output.DICT)
        n_boxes = len(d['text'])
        for i in range(n_boxes):
            if int(d['conf'][i]) > 60:
                (text, x, y, w, h) = (d['text'][i], d['left'][i], d['top'
                # don't show empty text
                if text and text.strip() != "":
                    frame = cv2.rectangle(frame, (x, y), (x + w, y + h),
                    frame = cv2.putText(frame, text, (x, y - 10), cv2.FON
                    if text == "61316":
```

Main code.py camera.py

```
ame, output_type=Output.DICT)



['text'][i], d['left'][i], d['top'][i], d['width'][i], d['height'][i])

) != "":
le(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
(frame, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255), 3)
```

```
if text == "61316":
    print("registered Plate Number")
    GPIO.output(23,GPIO.HIGH)
    GPIO.output(24,GPIO.LOW)
    time.sleep(2)
    return
else:
    print("not registered Plate number Not allowed to park here")
    GPIO.output(24,GPIO.HIGH)
    GPIO.output(23,GPIO.LOW)
    return
    # break
```

```python

        # Display the resulting frame
        cv2.imshow('frame', frame)


        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
     # When everything done, release the capture
    cap.release()
    cv2.destroyAllWindows()
```

Figure18. The code of the camera sensor

The output

Figure19. The output of the code of the camera sensor

The final step in the process is to connect the project to a database in order to store data. The chosen database service is Firebase, and the necessary libraries have already been installed.

```
# Fetch the service account key JSON file contents
cred = credentials.Certificate('./raspberry-pi-m-firebase-adminsdk-o4p7s-

# Initialize the app with a service account, granting admin privileges
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://raspberry-pi-m-default-rtdb.firebaseio.com/'
})
```

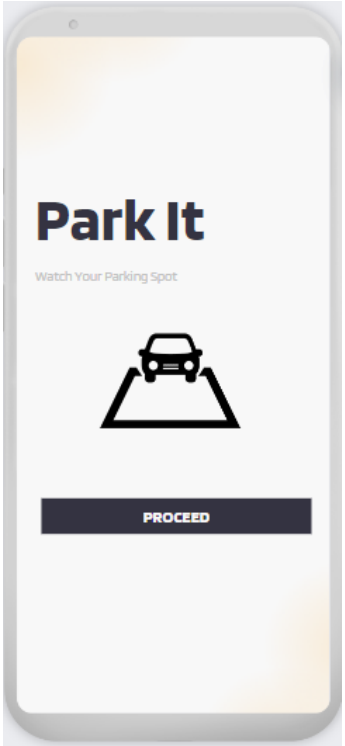
```
:e account key JSON file contents
;.Certificate('./raspberry-pi-m-firebase-adminsdk-o4p7s-8beb9e5cdf.json')

ıpp with a service account, granting admin privileges
itialize_app(cred, {
: 'https://raspberry-pi-m-default-rtdb.firebaseio.com/'


:('ir/ultrasonic')
```

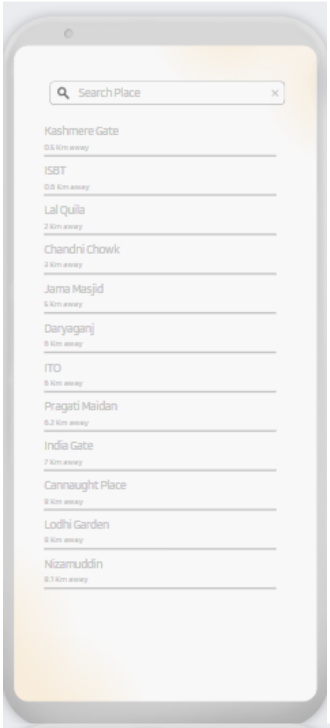
Figure20. Firebase code

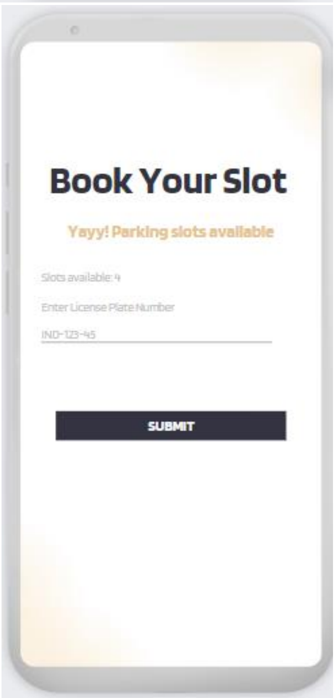
The picture shows that the data collected from the sensors is being stored in a Firebase database.

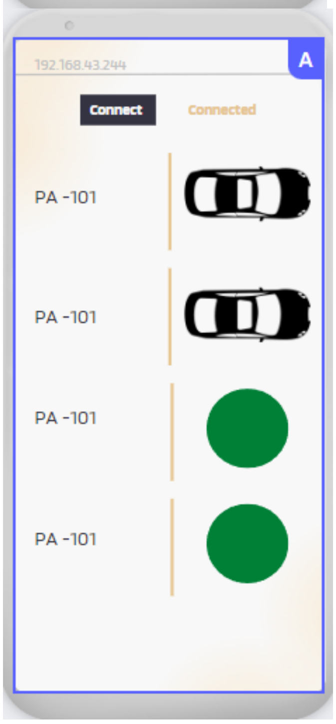
Figure21. Firebase output

The initiative to detect parking occupancy will be linked to a phone app. Wi-Fi technology will be used to link the parking sensors to the program. The application will be able to get up-to-date information about parking availability and occupancy in real time as a result. The application will also be configured to notify users when a parking space becomes available. The program will be able to tell when a user is close to a parking place and send them notice by utilizing IR and Ultrasonic sensors. The program will also be linked to a cloud-based platform, which will allow it to store and process the information gathered by the parking sensors.









Social, economic, and ethical issues

Ethical issues:

Managing the number of parking spaces occupied and the activity of cars has various benefits. In contrast, if the data on car motion and parking is unsecured, it would raise ethical and moral problems. Given that it offers thorough data on the motion of vehicles within a building or in parking, the smart car pairing system has even more ethical limitations. The application is continually aware of how the cars are moving. This application would raise serious safety concerns when utilized to observe vital parking, offices, and homes. We will consider these security and ethical concerns with the technology advancements for the smart parking management system and attempt to take the necessary actions to solve these concerns.

The following are the ethical issues that have been considered in the project:

1. Confidentiality: The parking occupancy detection system will safeguard the sensitive and personal information that may be exposed to attackers or other nefarious actors, including license plate numbers, driver identities, and payment details.
1. To prevent the loss of privacy, monitoring technology including cameras, sensors, and data-gathering tools will be created.
2. Security: The system will be trustworthy and safe to safeguard user data, thwart unwanted access, and shield users from harmful assaults.
3. Accessibility: No matter the user's age, gender, race, or capability, the system will be used by everyone.
4. Transparency: The system will be honest with participants about its services, costs, and data-collection rules. It will also be transparent in how it operates.

Social issues:

One of the unresolved issues in IoT technology that contributes to road congestion, high carbon emissions, and time waste is the issue of parking in large metropolitan areas.

Drivers are led directly to a parking space that is open. As a result, they don't waste as much time driving around in loops seeking parking. It goes without saying that drivers will spend less on gasoline if they drive more carefully when looking for a parking spot. leading to less money being spent on gasoline.

Since they are aware of where they can park, drivers are less likely to become focused while searching for a space. Accidents will be reduced and everyone's safety, including that of other drivers and pedestrians, will improve if they keep their focus on the road.

The following are the social issues that have been considered in the project:

1. Accessibility: Despite physical capability or impairment, the system will be intended to be available to all users.
2. Usability: The program will be simple to operate and comprehend.
3. Reliability: The system will be trustworthy and efficient in determining parking occupancy.
4. Environmental Effect: The system will be made to have as little of an effect on the environment as possible, including by using less energy.

Economic issues:

By improving facility use and developing new income streams, sensor data will provide insightful information about the locations with the most and least parking traffic. This aids facility owners in determining where to reduce parking and where to increase it appropriately. Simultaneously, it is simpler to identify and take action against the abuse of designated parking spaces or emergency access routes. Smart parking systems can also allow businesses to charge more for their parking spaces when they are not in use.

The following are the economic issues that have been considered in the project:

1. Parking Space Availability: The system will assist municipal managers and transportation authorities in pinpointing places where parking is in high demand thus that appropriate policy changes can be made.
2. Cost-effectiveness: The system will assist in lowering the price of managing parking equipment in addition to the price of parking services.
3. Traffic Management: The system will aid in reducing traffic on the highways and in parking spaces by delivering real-time data on parking spaces that are obtainable.
4. Revenue Generation: The system will assist in generating income from the sale of parking permits in addition to parking penalties and costs.
5. Public Safety: The system will contribute to a decrease in unlawful parking and increase street and parking space protection.

Technical issues:

Technical issues encompass any issue or challenge with a computer system's equipment or software, including difficulties with the connection, operating system, software program, or system component. System breaks, system faults, software issues, and hardware maladies are examples of typical technological problems.

The technological challenges we ran through while working on the parking occupancy detection project are mentioned below.

Sensor Malfunction was one of the technical problems that came up during the implementation of the parking occupancy detection project.

Due to new and more effective sensors that could identify parking occupancy despite adverse weather conditions, the problem of malfunctioning sensors caused by inadequate maintenance and harsh weather conditions has been resolved.

Additionally, among the technical difficulties we encountered were False Detection and inaccurate sensor readings. When a space was vacant, the detecting system incorrectly identified it as being occupied. Additionally, faulty sensors might produce false occupancy data, cause motorists to look for parking spaces that are not occupied, and fail to identify the presence of vehicles in the parking lot.

The problem was resolved by utilizing a camera sensor, which also confirmed the spots' availability and identified occupancy.

Moreover, the flawed identification algorithms. The camera sensor resolved this problem since the algorithms utilized to detect automobiles in the lot erroneously identified objects, which produced inaccurate findings.

Additionally, the network problems. The data gathered from the sensors was prevented from accessing the server when the network linking them to it was not functioning properly.

Data storage problems also occurred. A server was required to store the sensor data that was gathered. There were difficulties in handling the data when the server was unable to save the data effectively. High capacity, however, was the ideal solution.

Conclusion:

In brief, the report was created to provide parking occupancy detection that relies on the electronic sensors that we have provided, furthermore, to script them to develop an intelligent system for managing parking lots. This is accomplished through a series of stages and techniques for methodical analysis and innovative research. The investigation was divided into conceptual and practical components. The theoretical component dealt with a theoretical overview of the issues that might prevent the implementation of a parking management system for automobiles. For this, we also explained the issues' root causes and provided solutions in case we ran into them.

Regarding the practical aspect, it depended on picking a kind of smart parking, creating a model for it, and implementing strategies to assign parking lots to different ranges in accordance with land usage. To further explain the concept of the parking lot, a simpler model of the smart car park was created, and computer programs were utilized to replicate the model.

Figure22. Parking occupation detection project

Teesside University

British Applied College

الكلية البريطانية التطبيقية

References:

SGBotic Pte Ltd. (n.d.). *Infrared Obstacle Avoidance Sensor*. [online] Available at: https://www.sgbotic.com/index.php?dispatch=products.view&product_id=2527#:~:text=Description-. In-text citation: (SGBotic Pte Ltd, n.d.).

GeeksforGeeks. (2020). The architecture of the Internet of Things (IoT). [online] Available at: https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/.
In-text citation: (GeeksforGeeks, 2020).

Gillis, A. (2022). What is IoT (Internet of Things) and How Does it Work? - Definition from TechTarget.com. [online] IoT Agenda. Available at: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT. In-text citation: (Gillis, 2022).

InterviewBit. (2022). IoT Architecture - Detailed Explanation. [online] Available at: https://www.interviewbit.com/blog/iot-architecture/. In-text citation: (InterviewBit, 2022).

AN OVERVIEW OF COMMON PARKING ISSUES, PARKING MANAGEMENT OPTIONS, AND CREATIVE SOLUTIONS. (2003). [online] Available at: https://ccdcboise.com/wp-content/uploads/2016/02/Parking-Problems-and-Creative-Solutions.pdf.

In-text citation: (AN OVERVIEW OF COMMON PARKING ISSUES, PARKING MANAGEMENT OPTIONS, AND CREATIVE SOLUTIONS, 2003).

Raspberry Pi (2013). What is a Raspberry Pi? [online] Raspberry Pi. Available at: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. In-text citation: (Raspberry Pi, 2013).

Jost, D. (2019). What is a Motion Sensor? [online] FierceElectronics. Available at: https://www.fierceelectronics.com/sensors/what-a-motion-sensor. In-text citation: (Jost, 2019).