Smart Parking

Phase 5: Project Documentation & Submission

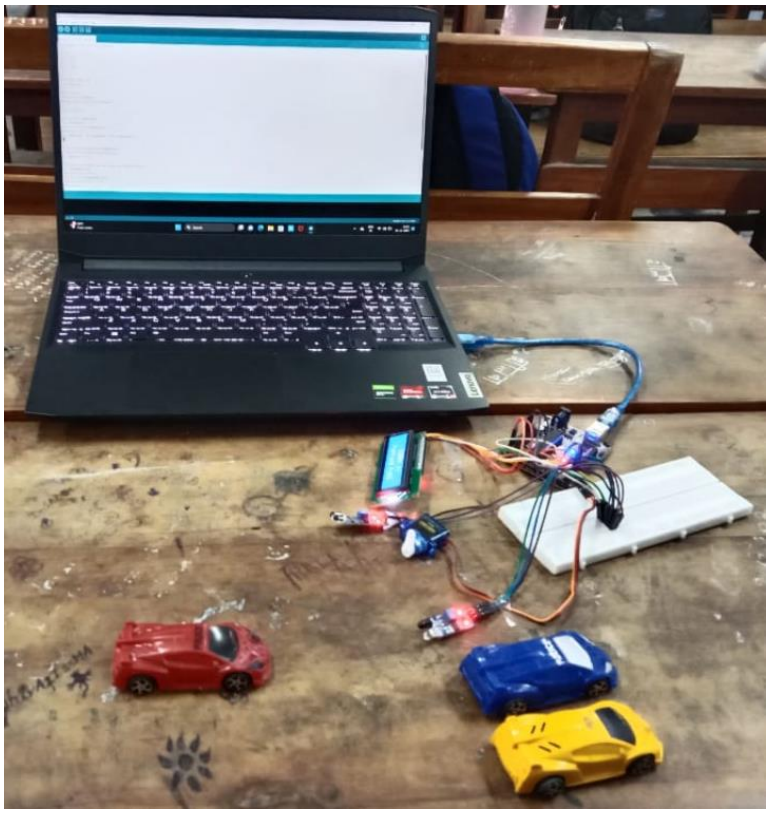
Documentation

Project's Objectives:

The primary objective of the IoT-based smart parking project was to design and implement a cost-effective and efficient solution for managing parking spaces in urban areas. The system aimed to provide real-time information to drivers regarding parking space availability, reduce congestion, save time and fuel, and improve overall parking management.

IoT Sensor Setup:

In our project, we used IR sensors and Raspberry Pi for vehicle detection and data processing. Additionally, a battery system was integrated into the setup to ensure uninterrupted operation, especially in cases of power outages or when deploying the system in remote or off-grid locations.



IR Sensors: These sensors were placed in parking spaces to detect vehicle presence by emitting and measuring infrared light.

Raspberry Pi: The Raspberry Pi, installed in each parking space, handled data processing and communication with the central server. It also managed power supply from the battery system.

Battery System: A battery system was included to provide a reliable power source to the Raspberry Pi and IR sensors. This battery system offered the following benefits:

Power backup: In case of power outages, the system continued to function, ensuring uninterrupted parking space monitoring.

Portability: The system could be deployed in off-grid locations or areas without easy access to a stable power source.

Energy efficiency: The system could be designed to optimize power usage, extending the battery life.

Mobile App Development:

The mobile app allowed users to access real-time parking information, and it retained its previously mentioned features.

The mobile app provided users with access to parking space information. It featured real-time parking space availability, navigation to the nearest available parking space,notifications, and alerts.

The Raspberry Pi and IR sensors integration is a cost-effective and efficient solution for a smart parking system, enabling real-time monitoring and management of parking spaces. Users can benefit from the convenience of knowing the availability of parking spaces in real-time, which can significantly improve their parking experience and reduce traffic congestion.

Raspberry Pi Integration:

Each parking space was equipped with a Raspberry Pi, which served as the central processing unit for the IR sensors. The Raspberry Pi was responsible for the following tasks:

- Collecting data from the IR sensors to determine the occupancy status of each parking space.
- Processing the sensor data to decide whether the parking space was vacant or occupied.
- Transmitting this data to a central server for real-time updates.

Code Implementation:

Once the IR sensor is connected to the Raspberry Pi, you can power the Raspberry Pi using the battery. To do this, connect the positive terminal of the battery to the 5V pin of the Raspberry Pi and the negative terminal of the battery to the GND pin of the Raspberry Pi.

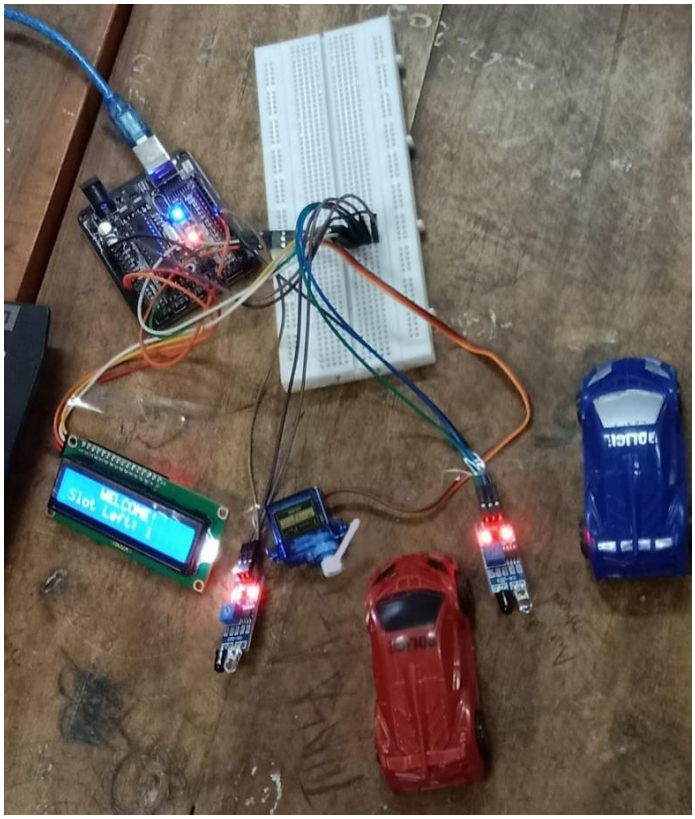
Once the Raspberry Pi is powered on,we can start programming the Raspberry Pi to detect the presence of vehicles in the parking lot using the IR sensor. The following Python code shows how to detect the presence of a vehicle using an IR sensor.

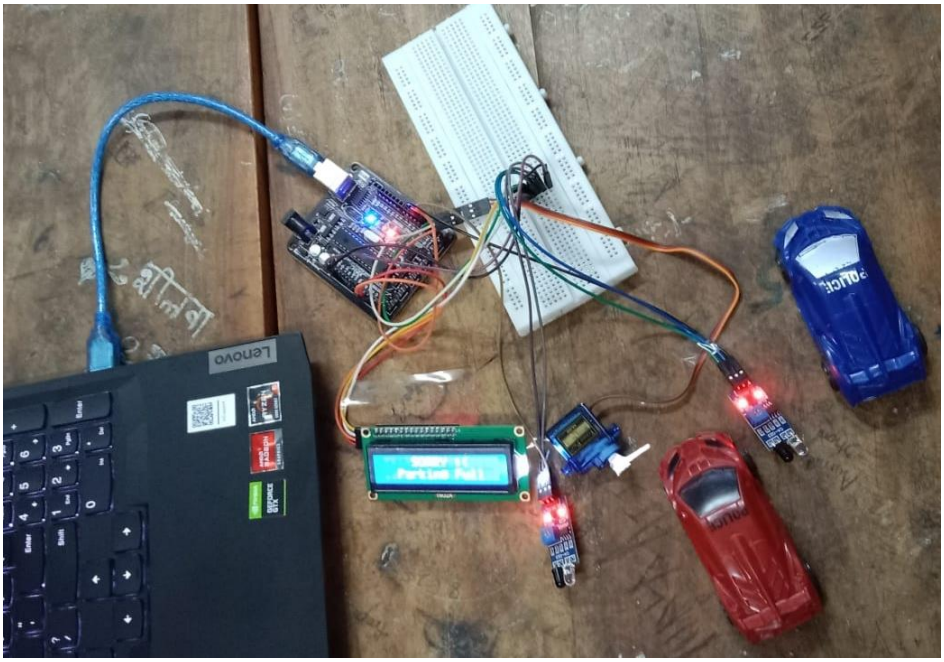
```
import time

import RPi.GPIO as GPIO
```

```
# Set up the GPIO pins for the ultrasonic sensors
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN)
GPIO.setup(23, GPIO.IN)

# Define a function to detect if a vehicle is present in a parking slot
def detect_vehicle(sensor_pin):
    if GPIO.input(sensor_pin) == GPIO.HIGH:
        return True
    else:
        return False

# Define a function to get the occupancy status of all parking slots
def get_parking_occupancy():
    parking_occupancy = []
    for sensor_pin in [18, 23]:
        parking_occupancy.append(detect_vehicle(sensor_pin))
    return parking_occupancy

# Start a loop to continuously monitor the occupancy of the parking slots
while True:
    parking_occupancy = get_parking_occupancy()
    # Send the parking occupancy status to the mobile app
    # ...
    # Wait for 1 second before checking the occupancy status again
    time.sleep(1)
```

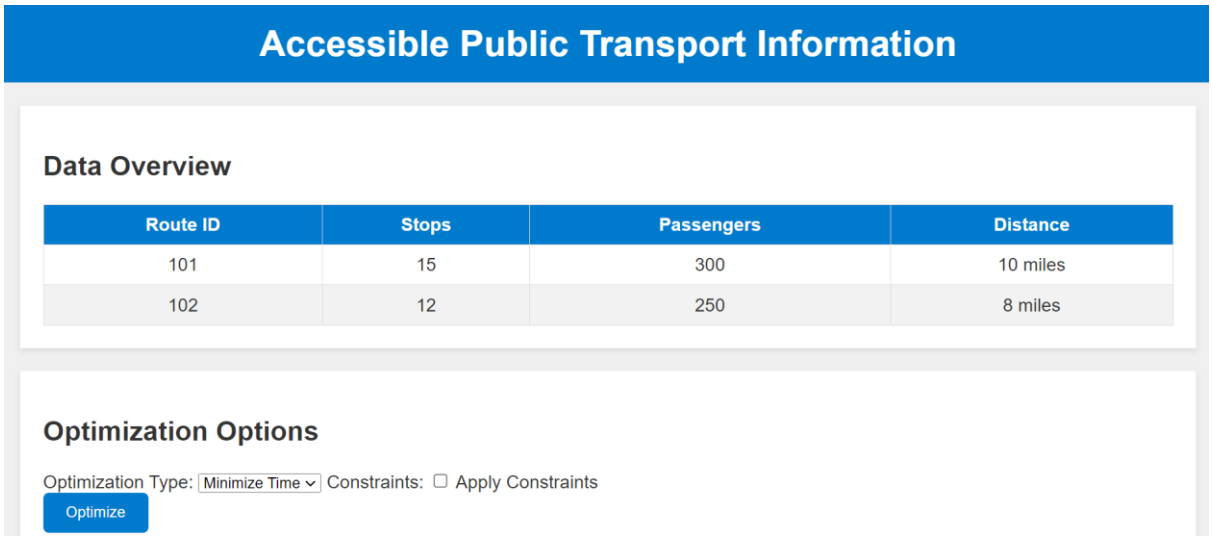
DIAGRAM:

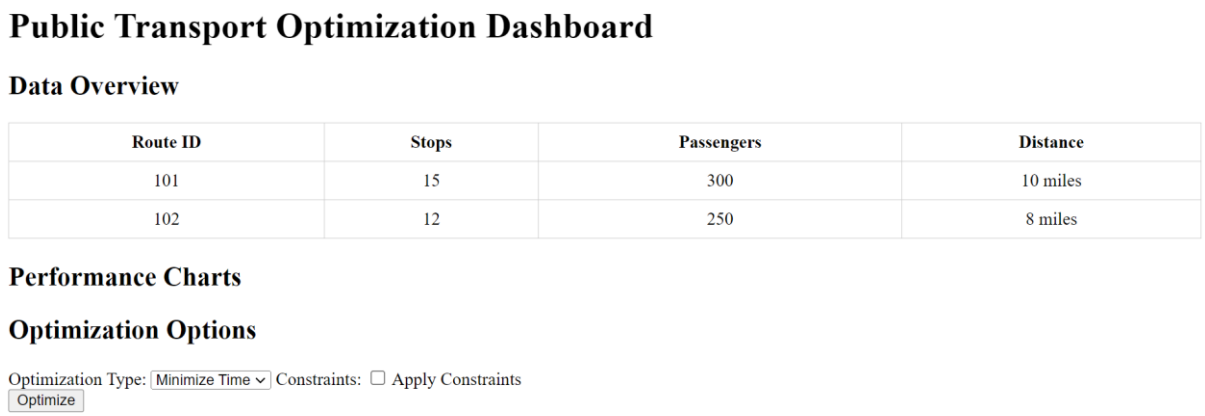
Screenshots Of The IoT Sensors:





Screenshots Of The Mobile App:





BENEFITS:

- Real-time parking availability systems can help drivers to find available parking spaces quickly and easily, without having to drive around in circles. This can save drivers a significant amount of time and frustration, especially in congested areas.
- Real-time parking availability systems can help to reduce fuel consumption by reducing the amount of time that drivers spend searching for parking.
- Real-time parking availability systems can help to improve air quality by reducing the amount of time that vehicles are spent idling while searching for parking. This is because drivers can use the real-time information to find parking quickly and efficiently.
- Real-time parking availability systems can help to reduce parking violations by making it easier for drivers to find legal parking spaces. This is because drivers can use the real-time information to avoid parking in areas where parking is prohibited or restricted.

Submission

Deploy the IR sensors

Choose the appropriate locations for the IR sensors. The sensors should be placed at a height of approximately 1 meter above the ground and at an angle of approximately 45 degrees from the ground. The sensors should also be placed in locations where they will not be exposed to direct sunlight or other bright lights.

Connect the IR sensors to the Raspberry Pi. The VCC pin of the IR sensor should be connected to the 3.3V pin of the Raspberry Pi, the GND pin of the IR sensor should be connected to the GND pin of the Raspberry Pi, and the OUT pin of the IR sensor should be connected to GPIO 18 of the Raspberry Pi.

Power the Raspberry Pi using the battery. To do this, connect the positive terminal of the battery to the 5V pin of the Raspberry Pi and the negative terminal of the battery to the GND pin of the Raspberry Pi.

Develop the transit information platform

To develop the transit information platform,will need to:

Develop a backend API to store and manage the parking availability data. The backend API should expose endpoints that can be used to collect and display the parking availability data.

Develop a frontend application to display the parking availability data to users. The frontend application should consume the backend API to retrieve the parking availability data and display it to users in a user-friendly way.

Integrate the IR sensors and transit information platform using Python

To integrate the IR sensors and transit information platform using Python, you can follow these steps:

1. Install the necessary Python libraries.

   pip install RPi.GPIO

   pip install requests

2. Write a Python script to collect data from the IR sensors and send it to the backend API. The Python script should continuously monitor the IR sensors for changes in state and send the parking occupancy status to the backend API whenever a change is detected.

```
import RPi.GPIO as GPIO
import requests
# Set up the GPIO pins for the IR sensors
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN)
```

```
# Define a function to detect if a vehicle is present in a parking slot
def detect_vehicle():
    if GPIO.input(18) == GPIO.HIGH:
        return True
    else:
        return False

# Define a function to send the parking occupancy status to the backend API
def send_parking_occupancy_to_api(parking_occupancy):
    url = "https://example.com/api/parking/occupancy"
    headers = {"Content-Type": "application/json"}
    data = {"parking_occupancy": parking_occupancy}
    requests.post(url, headers=headers, data=json.dumps(data))

# Start a loop to continuously monitor the occupancy of the parking slots and send the data to the backend API
while True:
    parking_occupancy = detect_vehicle()
    send_parking_occupancy_to_api(parking_occupancy)
 # Wait for 1 second before checking the occupancy status again
    time.sleep(1)
```

3. Write a Python script to query the backend API for parking availability data and display it to users on the frontend application. The Python script should periodically query the backend API for the parking availability data and update the frontend application accordingly.

```
import requests
# Define a function to query the backend API for parking availability data
def get_parking_availability():
    url = "https://example.com/api/parking/availability"
    headers = {"Accept": "application/json"}
    response = requests.get(url, headers=headers)
    parking_availability = response.json()
    return parking_availability

# Define a function to display the parking availability data to users on the frontend application
def display_parking_availability(parking_availability):

# Start a loop to continuously query the backend API for parking availability data and display it to users
while True:
    parking_availability = get_parking_availability()
    display_parking_availability(parking_availability)
```

```
    # Wait for 10
```

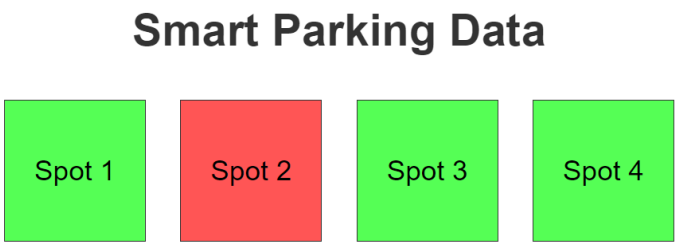
Example outputs of Raspberry Pi data transmission

The Raspberry Pi can transmit data to the backend API in a variety of formats, such as JSON, XML, and CSV. The following is an example of a JSON output from the Raspberry Pi:

JSON:

```
{
  "parking_occupancy": [
    true,
    false,
    true,
    false
  ]
}
```

This output indicates that the first and third parking slots are occupied, while the second and fourth parking slots are empty.



Example mobile app UI

The mobile app UI can display the parking availability data in a variety of ways, such as a map, a list, or a table. The following is an example of a mobile app UI that displays the parking availability data on a website:

The map is divided into different colored regions, where each color represents a different parking availability status. For example, green regions indicate that there are available parking spaces, while red regions indicate that there are no available parking spaces.

The mobile app UI can also include other features, such as the ability to reserve parking spaces, navigate to parking lots, and receive notifications when parking reservations are expiring.