Date : 31-10-2023

Project Id:Proj_223339_Team_3

Project Title: Smart parking

PHASE-5

IOT

Documentation:

Objectives:

The goal of parking management is to ensure that parking spaces are available and accessible to those who need them while also preventing overcrowding and promoting safety. Parking management can encompass a wide range of activities, including: Design and construction of parking facilities.

I)IOT device setup

Select a Communication Protocol : Decide on a communication protocol (e.g., Wi-Fi, Bluetooth, NB-IOT) based on your specific requirements, such as range, power consumption, and data transmission speed.

Power Supply :Ensure the device has a reliable power source. Depending on the device type, this could be a battery, solar panel, or wired power supply.

Device Placement :Install the device in a strategic location within the parking space. For sensors, this is typically on the ground. Cameras or other visual devices may require mounting at an elevated position for a clear view.

Connectivity Setup: Connect the device to the chosen communication network. This might involve configuring Wi-Fi credentials, setting up a LORAWAN gateway, or establishing a cellular connection.

Configure Device Parameters: Depending on the specific device, configure parameters such as sensitivity, reporting intervals, and threshold values. This ensures accurate data collection.

Integration with Centralized System: Integrate the IOT device with the centralized smart parking system. This involves registering the device on the platform and ensuring it can send and receive data.

Remember, specific setup steps may vary depending on the type and brand of the IOT device you're using. Always refer to the manufacturer's guidelines and documentation for detailed instructions.

ii) Platform Development:

Developing a smart parking platform involves integrating various technologies to optimize parking spaces, enhance user experience, and improve overall efficiency. Here are key components and considerations for smart parking platform development:

Sensors and IOT Devices: Utilize sensors and IOT devices to detect occupancy in parking spaces. These devices can include cameras, ultrasonic sensors, and magnetic sensors, which provide real-time data about available parking spots.

Data Collection and Analysis: Gather data from sensors and other sources to analyze parking patterns, peak usage times, and user behavior. Machine learning algorithms can be employed to predict parking availability and optimize resource allocation.

Mobile Apps and User Interfaces: Develop user-friendly mobile applications that allow users to check parking availability, reserve spots, and make payments. Implement intuitive interfaces and smooth navigation to enhance user experience.

Data Privacy and Compliance: Adhere to data privacy regulations and ensure that user data is securely stored and processed. Implement encryption and other security measures to protect sensitive information.

Feedback Mechanism: Implement a feedback mechanism within the app for users to report issues, provide suggestions, or rate their parking experience. Use this feedback to continually improve the system.

By addressing these aspects in your smart parking platform development, you can create a seamless and efficient parking experience for users while optimizing resource utilization and contributing to smart city initiatives.

iii)Code implementation

Creating a complete smart parking system involves various technologies, including sensors, microcontrollers, databases, and sometimes mobile apps or web interfaces for user interaction. Below is an example of a simple Python-based implementation for a smart parking system using Raspberry Pi, ultrasonic sensors, and a database (SQLite in this case). This script assumes you have installed the necessary libraries and set up your Raspberry Pi with the sensors.

Python Copy code

```
import RPI. GPIO as GPIO
import  time
```

```
import sqlite3
# Initialize GPIO and database
GPIO. Set mode(GPIO.BOARD)
conn = sqlite3.connect('parking. Db ‘)
cursor = conn. .cursor()
# Create table if not exists
Cursor .execute('''CREATE TABLE IF NOT EXISTS parking_ status (id INTEGER PRIMARY KEY
AUTOINCREMENT, spot_ number INT, status INT)''') conn. commit()
# Set up ultrasonic sensor pins
TRIG = 11
ECHO = 13
GPIO. .setup(TRIG, GPIO.OUT)
GPIO. setup(ECHO, GPIO.IN)
Def get_ distance():
    # Function to measure distance from the ultrasonic sensor
    GPIO. output(TRIG ,True)
    Time .sleep(0.00001)
    GPIO. output(TRIG ,False)

    Pulse _start=time. time()
    Pulse _end=time. .time()

  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
distance = round(distance, 2)
```

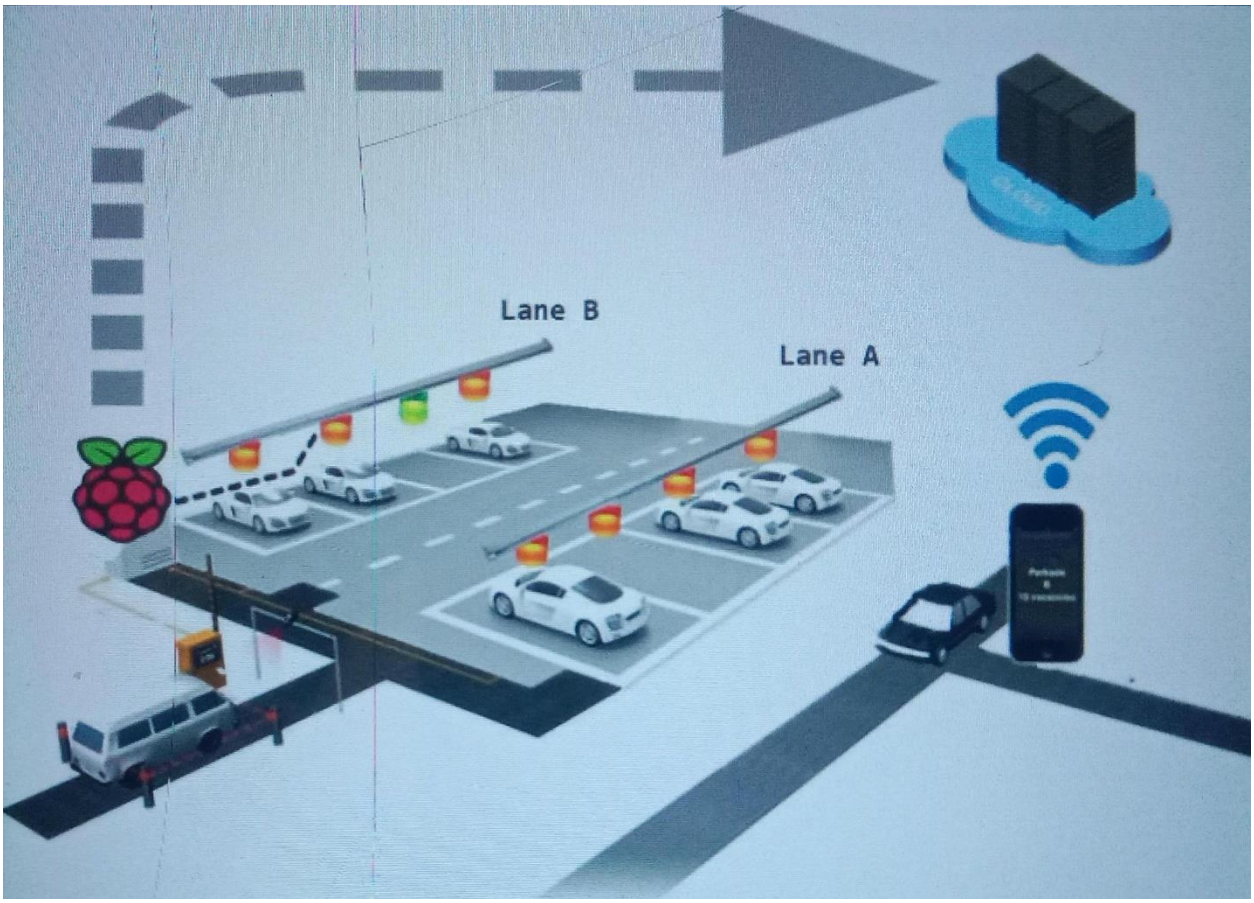
```
return distance

try:
    while True:
        distance = get_ instance()
        print("Distance: {} cm  “.format(distance))


        if distance < 30:
            # Parking spot occupied
            cursor. execute("INSERT INTO parking _status (spot_ number, status) VALUES (?, ?)", (1, 1))
            conn. commit()
 else:
     # Parking spot vacant
     Cursor .execute("INSERT INTO parking. _status (spot_ number, status) VALUES (?, ?)", (1, 0))
    conn. commit()
time. sleep(5) # Check parking status every 5 seconds
 except Keyboard interrupt:
      # Clean up GPIO and close the database connection
     GPIO .cleanup()
     Conn .close()
```

In this script, the ultrasonic sensor measures the distance. If the distance is less than 30 cm, it considers the parking spot occupied (status = 1) and stores this information in the database. Otherwise, it marks the spot as vacant (status = 0). You can modify and expand this code to handle multiple parking spots, integrate it with a web interface, or use IoT platforms for real-time monitoring. Additionally, make sure to handle exceptions and errors for a more robust implementation.

iv)Diagram



V) schematics

Certainly! Smart parking systems use various technologies like sensors, IoT, and mobile apps to optimize parking space usage. Here's a basic schematic representation:

Plaintext Copy code

Smart Parking System Schematic: +---------------------------------------+ | Parking Area | | | | +---------------------------+ | | | Parking Spaces | | | | | | | +------+ +------+ +------+ | | | | | 01 | | 02 | | 03 | | | | | +------+ +------+ +------+ | | | | | | | +---------------------+ | | | +-------------------------------------+ +---------------------------------------+ | Sensor Nodes | | +------+ +------+ +------+ | | | S1 | | S2 | | S3 | | | +------+ +------+ +------+ +------+ | +-------------------------------------+ +---------------------------------------+ | Control Unit | | | | +-----------------------------+ | | | Data Processing | | | | and

Decision Making | | | +------------------------------+ | | | | | | +----+ +----+ | | | | | | +--------+ +--------+ | | | LED | | Mobile | | | | Display| | App | | | +--------+ +--------+ | | | +--------------------------------------+

In this schematic:

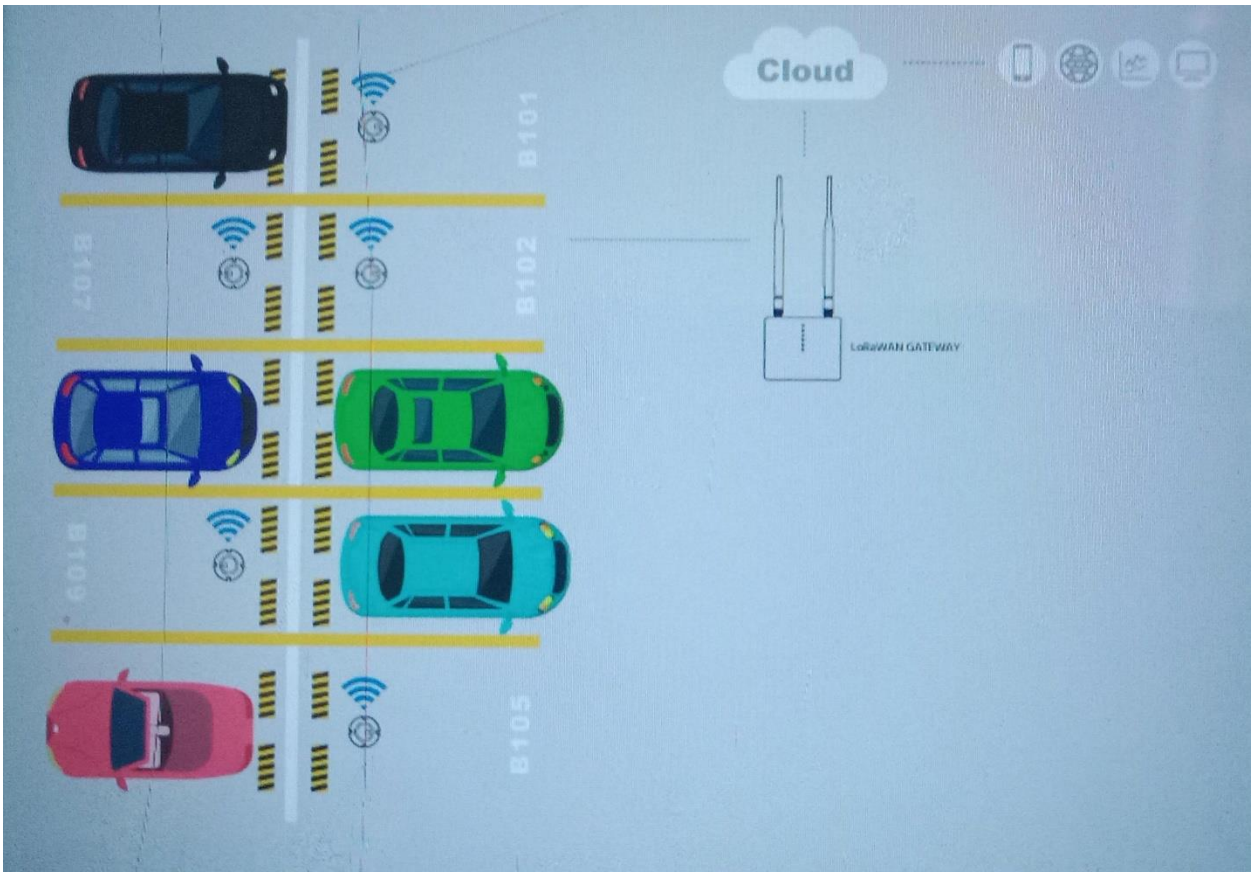
Parking Area: Represents the physical parking spaces where vehicles are parked.

Sensor Nodes (S1, S2, S3, etc.): These are sensors placed in each parking space to detect the presence of a vehicle. They send data to the control unit.

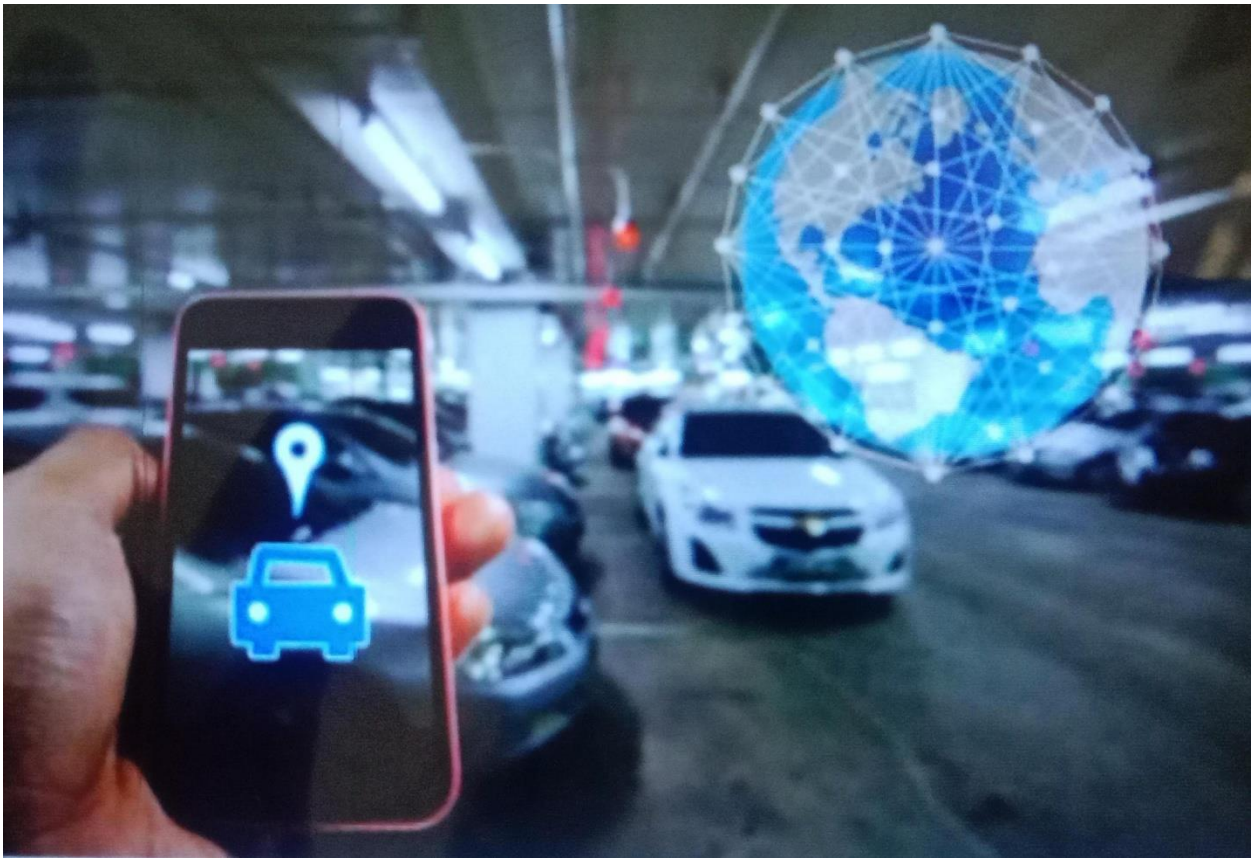
Control Unit: Processes data from sensor nodes, makes decisions (like whether a space is occupied or vacant), and communicates with LED displays or a mobile app to inform drivers about parking space availability.

This schematic illustrates a basic setup. Real-world implementations might include additional features like payment systems, security cameras, or integration with navigation apps for finding available parking spaces.

Vi)Screen shot sharing IOT device:



Vii)Data sharing:



Viii)Title details:

Overhead sensors scan the vicinity and identify the position of a parked car or an empty parking space while also measuring the length of an available parking space and registering whether vehicles have been in an illegal space (e.g. in front of a fire hydrant or other no-parking zone)

Conclusion

As a conclusion, this project will help in reducing the amount of time a driver has to spend around the parking just to find an available spot, reducing the amount of traffic around the parking and also reducing the bad parking around the parking space.