**SMART PARKING**

* **Abstract:**

Parking slots have become a widespread problem in urban development. In this context, the growth of vehicles inside the university’s campus is rapidly outpacing the available parking spots for students and staff as well. This issue can be mitigated by the introduction of parking management for the smart campus which targets to assist individually match drivers to vacant parking slots, saving time, enhance parking space utilization, decrease management costs, and alleviate traffic congestion. This paper develops an IoT Raspberry Pi-based parking management system (IoT-PiPMS) to help staff/students to easily find available parking spots with real-time vision and GPS coordinates, all by means of a smart phone application.

* **Introduction:**

The growing number of automobiles is rapidly outpacing the parking lots availability in urban areas; thus, parking has become a widespread challenge in metropolises development [1]. This problem can be observed clearly in cities, especially centers, shopping malls, open markets, government offices, hospitals, schools, and universities [2]. There are two main types of parking lots, indoor and outdoor. In conventional parking management systems, the information about parking spots cannot be efficiently collected and updated to the management platform [3, 4]. In such systems, many parking spaces exist as local data silos, but their parking spot information cannot be shared remotely with drivers [5]. Additionally, data of isolated parking cannot be transmitted to a unified platform for city parking management

**IoT-based smart parking system deployment requires integrating various devices, sensors, and microcontrollers. For example, it can be a microcontroller transmitting data to the cloud environment or a Bluetooth beacon. With its help, consumers can control parking locally.**

****

**Smart Parking**

* **Hardware Setup:**

1. Raspberry Pi Setup
2. Ultrasonic Sensors
3. Power Supply

**1. Raspberry Pi Setup**:

* Get a Raspberry Pi board (e.g., Raspberry Pi 4).
* Install an operating system (e.g., Raspberry Pi OS) on the Pi.
* Make sure your Pi is connected to the internet, either via Wi-Fi or Ethernet.

**2. Ultrasonic Sensors:**

* Connect your ultrasonic sensors to the Raspberry Pi using GPIO pins.
* Ensure you have a ground (GND), echo, and trigger pin connection for each sensor.

**3. Power Supply:**

* Power the Raspberry Pi and sensors using an appropriate power supply.
* **Software Setup:** .

1.Raspberry Pi Python Code

2. Cloud or Mobile App Server

3. Data Storage

* **Raspberry Pi Python Code:**
* You will need the RPi.GPIO library for controlling GPIO pins on the Raspberry Pi. Install it if it&#39; is not

already installed.

**System Workflow:**

The general workflow of the developed smart parking system is described in the flowchart of Figure 12. The Raspberry Pi should be connected to the Wi-Fi network when the power of the microcontroller is turned on. The Raspberry Pi’s Wi-Fi connection can be checked through the system monitor or cloud system network. The user must access the Internet through a smart portable device or laptop to discover and access the vacant parking slot in the smart parking system. The system starts working by checking and updating the detection of the ultrasonic sensor, Pi camera, and GPS module

**Program:**

pip install RPi.GPIO

PYTHON SCRIPT:

import RPi.GPIO as GPIO

import time

import requests

# Configure GPIO pins

TRIG\_PIN = 18

ECHO\_PIN = 24

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG\_PIN, GPIO.OUT)

GPIO.setup(ECHO\_PIN, GPIO.IN)

def get\_distance():

GPIO.output(TRIG\_PIN, True)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, False)

while GPIO.input(ECHO\_PIN) == 0:

pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = (pulse\_duration \* 34300) / 2 # Speed of sound = 343 m/s

return distance

try:

while True:

distance = get\_distance()

print(f&quot;Distance: {distance} cm&quot;)

# Send data to a cloud server or mobile app server

# Replace &#39;YOUR\_SERVER\_URL&#39; with the actual server URL

server\_url = &#39;YOUR\_SERVER\_URL&#39;

data = {&#39;distance&#39;: distance}

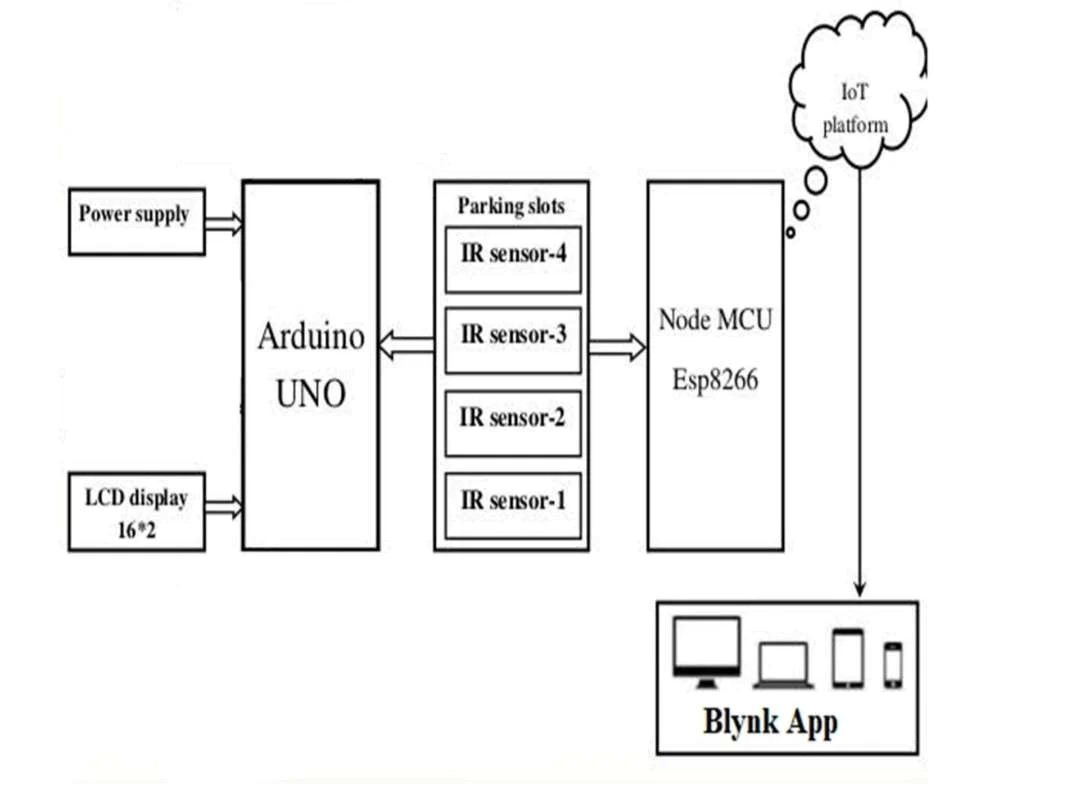
response = requests.post(server\_url, json=data)

time.sleep(1) # Adjust the sampling interval as needed

except KeyboardInterrupt:

GPIO.cleanup()

* **Cloud or Mobile App Server:**
* Set up a cloud server or a mobile app server to receive and store the data from the Raspberry Pi.
* You can use platforms like AWS, Google Cloud, or a self-hosted server.
* **Data Storage:**
* Decide how you want to store and analyze the data.
* You can use databases like MySQL, PostgreSQL, or NoSQL databases like MongoDB.
* **Mobile App :**
* If you want to build a mobile app to display parking space occupancy, you&#39;ll need to develop the app and configure it to communicate with the server.
* **Flow chart:**

****

* **Notifications :**
* Implement notifications to inform users of parking space availability in real-time.
* **Security:**
* Ensure the security of your IoT system, especially when handling data transmission and storage.
* **Testing:**
* Thoroughly test your setup to ensure it works as expected.
* **NEO-6M GPS Module:**
* NEO-6M GPS Module is a GPS module capable of supporting microcontrollers. It is user-friendly and uses up only a little space on board. It is a very useful device for locating device location.
* This module is chosen to allow the positioning of the parking lot location.
* **Green LEDs:**
* LEDs are very efficient in energy usage; it consumes only 10% of the power of what a normal incandescent bulb would normally consume. This, in turn, reduces the power costs due to low operating power. The LED is used in this project to indicate whether a parking spot is occupied or vacant by switching its light: ON green for vacant and OFF for occupied. One LED is used for each parking slot
* **Conclusion:**

An IoT-based smart parking system using Raspberry Pi 4 B+ has been designed and fabricated by utilizing ultrasonic sensors, Pi camera, GPS module, LEDs, and the Blynk IoT platform. The developed system has been tested and validated to be used in the smart campus environment or similar outdoor parking lot. The system’s reliability has been improved by using multiple sensors to detect the existence of vehicles. Staff/students/visitors can easily monitor the parking lot around the campus via a GUI over the Blynk App. by accessing the system dashboard on their smartphones.

The GPS sensor allows drivers to easily access the parking lot. The developed IoT-PiPMS provides accessibility, intelligence, comfortable, and improves the driver user experience. For practical implementation, our system can be extended to include multiple RPi and Pi cameras. The parking lot can be divided into several sections; each can be covered by one RPi and one Pi camera. The number of ultrasonic sensors will be increased to cover the entire parking area. The camera captions and video streaming function can be specified for management only rather than users to reduce data usage and improve system privacy and safety.