IBM NAAN MUDHALVAN INTERNET OF THINGS

Phase 3:

Development part 1

Topic:

Smart parking

Team members:

M. Sabitha jones(922121106075)

A.Santhi (922121106079)

B.Sathyadevi(922121106084)

K. Varnigadevi (922121106102)

M. Varsha (922121106103)

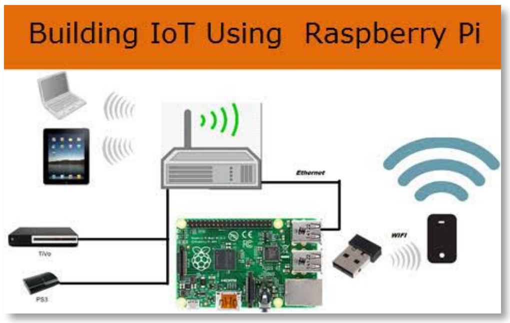
College name:

SSM Institute Of Engineering and Technology.

College code:

9221

Start building the IoT sensor system and Raspberry Pi integration.



SYSTEM ARCHITECTURE

In many cities, people would appreciate their luck if they could find a parking slot smoothly. People keep roaming around in search of vacant parking slots, and after a lot of struggle, they find one. Due to lack of a proper mechanism to identify free parking slots, they move randomly in search Of parking space wasting a lot of time. This problem can be solved if the drivers could check the availability of parking spaces in And around their intended destination.

Parking Sensors:

For our parking system we have made use of sensors like Infrared, Passive Infrared (PIR) and Ultrasonic sensors. The work of these sensors is the same i.e. to sense the parking area and determine whether a parking spot is vacant or Not. In this case we are using ultrasonic sensors to detect the presence of a car. The ultrasonic sensors are wirelessly connected to Raspberry pi using the ESP8266 chip. The sensors are Connected to a 5V supply either from raspberry pi or an external source.External source being more Preferable.

INTEGRATING RASPBERRI PI WITH ULTRASONIC SENSOR

HSC-SR04 is the commonly used ultrasonic distance Sensor which operates to find the distance Between 2 to 400 cm distances. This ultrasonic distance sensor module Contains four pins such as

- VCC -Input Power of 5V
- TRIG – Trigger Input
- ECHO – Echo Output
- GND –Ground Upon trigger

The TRIG pin with minimum Required High signal of at least 10 S duration, it will Transmit eight 40 KHz Ultrasonic burst through the sender. The ECHO pin will get high voltage for the duration of Time taken for Sending and receiving ultrasonic sound Signals. Based on the distance of obstacle the pulse width Will vary Entire operation of ultrasonic control such as to Initiate the sender to trigger sound waves, listening. The Receiver to calculate the distance will be controlled Through Raspberry Pi and Python scripts.

INTEGRATING RASPBERRI PI WITH CAMERA

The Raspberry Pi Camera Module is specifically designed Extension hardware to work in connection with Raspberry Pi. Raspberry Pi Camera connects with the Raspberry Pi device through the dedicated CSI interface bus which is capable of Transmitting high volume of pixel data. The transmission of Data to the camera and Pi device will happen through BCM2835 processor embedded to the Camera board. The

Camera board is a tiny, at around 25mm x 20mm x 9mm and Weights just 3g making it portable of mobile like applications and other digital devices where size matters in specific. The camera board connects to Raspberry Pi by the way of a short Ribbon cable Once the camera module is integrated then the Raspberry pi should be installed with camera module to access The camera by any of the high level programming languages. The prototype establishes here uses Python Script for Controlling the camera, that is to take to take the capture of Number plate details of car being parked. The camera module Code will be activated once confirmed that the vehicle is in Specific distance from the sensor and this will be intimated by The ultrasonic sensor to the camera module to activate.

CHALLENGES OF SMART PARKING SYSTEM

The challenges for the proposed system are wide from Protecting the system from environmental conditions and Harsh weather so as to making it operational in all weather. Integration of the devices amongst various hardware and Software modules Protection can be provided by using a Proper insulator case for the hardware module which will not affect the functionality of the device and also provide Durability, resistance against external weather and mechanical Forces. The major challenge in Parking Systems is of system Integration due to wide variety of hardware and software Platforms involved and hence possess a great concern to the system scalability. The technology platform supporting P&E, PARC and PUCRS systems comprises of Dynamic messaging systems, a myriad of hardware sensors and traffic control devices, wireless and wire line. Telecommunications systems, computer clients, servers and Hardware drivers and application interfaces. Enabling all these Devices from thousands of different vendors to communicate With each other and tying them together into one platform is the greatest challenge in reducing the complexity and cost of Smart parking. The variety of infrastructure hardware and Software systems that need to be integrated is enormous and an add-on to it the conventional older hardware making investment in Smart Parking solution. It is highly risky and fragmented. Another major pain point comes from the electronic payment vendors. These payment processors provide permit based electronic payment, typically for a Convenience fee. Scalability is the key to many of these hosted solutions which is the ability of the transaction processor to support over wide geographical market and Service areas with minimal cost. Income based Market evaluation.

Configure IoT sensors to detect parking space occupancy

Configuring IoT sensors like ultrasonic sensors to detect parking space occupancy involves setting up the hardware and writing code to interpret sensor data. Here's how you can do it:

Hardware setup:

1. *Ultrasonic Sensors*:

Connect ultrasonic sensors to your Raspberry Pi's GPIO pins. Ultrasonic Sensors typically have four pins: VCC (power), GND (ground), TRIG (trigger), and ECHO (echo).Connect VCC to 5V on the Raspberry Pi.Connect GND to a ground pin on the Raspberry Pi.Connect TRIG to a GPIO Pin for triggering the sensor (e.g., GPIO23).Connect ECHO to another GPIO pin for receiving the echo (e.g., GPIO24).

2. Power Supply:

Make sure the Raspberry Pi has a stable power supply.

Python Script to Detect Occupancy:

You can use a Python script to read data from the ultrasonic sensor and determine parking space Occupancy based on the measured distance.

```
Import RPi.GPIO as GPIO

Import time

# Set up GPIO pins for ultrasonic sensor

TRIG_PIN = 23

ECHO_PIN = 24

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG_PIN, GPIO.OUT)

GPIO.setup(ECHO_PIN, GPIO.IN)

Def measure_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 * 17150 # Speed of sound in cm/s

return distance
```

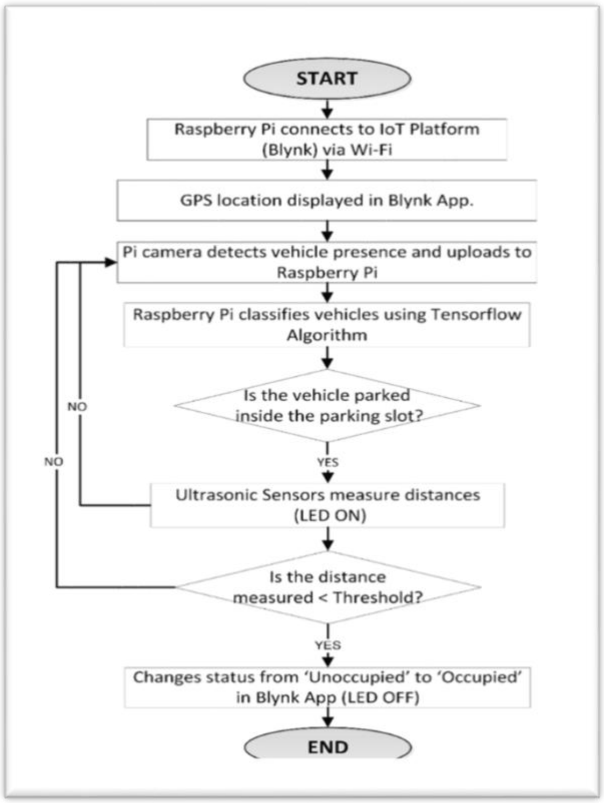
```
try:
      while True
       distance = measure_distance()
 if distance < 50: # Adjust this threshold based on your parking space
 print("Parking space occupied")
 else:
      print("Parking space available")
except Keyboard Interrupt:
 GPIO.cleanup()
```

This script sets up the ultrasonic sensor, triggers it, measures the time it takes for the echo to return, and Calculates the distance. If the distance is less than a certain threshold (e.g., 50 cm), it considers the Parking space as occupied. You can customize the threshold value and further integrate this script into the IoT system to send occupancy information to the cloud or mobile app server, as described in the previous response.

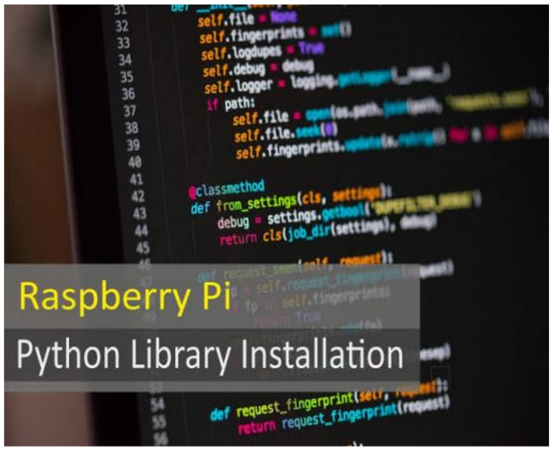
System Workflow

The general workflow of the developed smart parking System is described in the flowchart.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. The detection reading by the ultrasonic sensor sometimes fluctuates due to surrounding presence. The Pi camera will then monitor the parking slot to detect any physical presence that appeared in the parking slot. The GPS module updates the location of the parking area to the cloud system. Users can run the Blynk App. On their smartphones and get updates about the outdoor parking lot status. The notification can be seen whether the parking slot is occupied or still vacant. When the parking slot status meets the condition from the ultrasonic sensor and Pi camera, the result condition is uploaded to the cloud server to be displayed into the server application. The user can also view the Parking area's real-time condition through server application.

Flowchart of the system:


START
Raspberry Pi connects to IoT Platform (Blynk) via Wi-Fi
GPS location displayed in Blynk App.
Pi camera detects vehicle presence and uploads to Raspberry Pi
Raspberry Pi classifies vehicles using Tensorflow Algorithm
Is the vehicle parked inside the parking slot?
NO
YES
Ultrasonic Sensors measure distances (LED ON)
Is the distance measured < Threshold?
NO
YES
Changes status from 'Unoccupied' to 'Occupied' in Blynk App (LED OFF)
END

Write Python scripts on Raspberry Pi to collect data from sensors and send it to the cloud or mobile app server.



To collect data from sensors (ultrasonic sensors) and send it to a cloud or mobile app server using a Raspberry Pi, you can follow these general steps and create a Python script. In this example, I'll use the MQTT protocol to send data to a cloud-based MQTT broker. You'll need to adapt this code according to Your specific cloud or mobile app server setup.

Here's a basic Python script:

```
Import RPi.GPIO as GPIO

Import time

Import paho.mqtt.client as mqtt
```

```
# Set up GPIO pins for ultrasonic sensor

TRIG_PIN = 23

ECHO_PIN = 24

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG_PIN, GPIO.OUT)

GPIO.setup(ECHO_PIN, GPIO.IN)

# MQTT Broker Details

MQTT_BROKER = “mqtt.example.com” # Replace with your MQTT
broker address

MQTT_PORT = 1883

MQTT_TOPIC = “parking/occupancy”

# Initialize MQTT client

Client = mqtt.Client(“Parking Sensor”)

Client.connect(MQTT_BROKER, MQTT_PORT, 60)

try:

       while True:

 # Trigger the ultrasonic sensor

      GPIO.output(TRIG_PIN, True)

       Time.sleep(0.00001)

          GPIO.output(TRIG_PIN, False)

 # Measure time until the ECHO pin goes high

 while GPIO.input(ECHO_PIN) == 0:
        Pulse_start = time.time()
 while GPIO.input(ECHO_PIN) == 1:
        Pulse_end = time.time()
# Calculate distance from the time elapsed
 Pulse_duration = pulse_end – pulse_start
 Distance = pulse_duration * 17150 # Speed of sound in cm/s
 # Send the distance data to the MQTT broker
 Client.publish(MQTT_TOPIC, f"Distance: {distance:.2f} cm")
Time.sleep(1) # Adjust the interval as needed
Except KeyboardInterrupt:
 GPIO.cleanup()
```

This script sets up an MQTT client, reads data from an ultrasonic sensor, and publishes the distance measurement to the specified MQTT topic. You need to replace the MQTT broker details with the information of your cloud-based MQTT broker or mobile app server. Make sure you have the necessary Libraries installed. You can install the paho-mqtt library using:

```
Pip install paho-mqtt
```

Adjust the GPIO pins, MQTT details, and data format as per your specific setup and requirements. additionally, ensure that your Raspberry Pi is connected to the internet and can reach the MQTT broker or server.