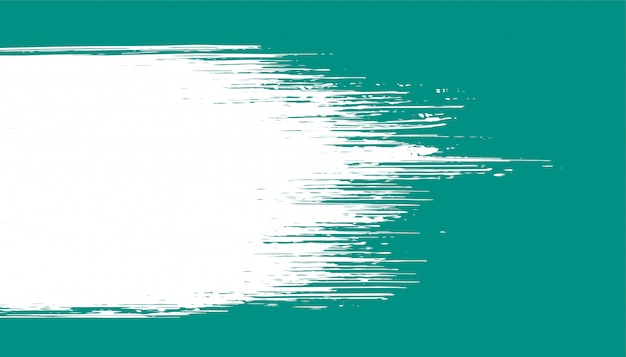
PHASE-3

TOPIC:

Start building the IoT sensor system and Raspberry Pi integration.



Submitted by:

Derengula Haresh Babu

Reg no:723921244015

Certainly! Setting up an IoT sensor system with Raspberry Pi integration involves several steps, including configuring the sensors, writing Python scripts, and setting up communication with the cloud or a mobile app server. Here's a step-by-step guide to help you get started:

**1. Gather Components:**

* Raspberry Pi (with Wi-Fi capabilities)
* IoT sensors (e.g., ultrasonic sensors for parking space detection)
* Breadboard and jumper wires
* Power supply for Raspberry Pi and sensors

**2. Connect Ultrasonic Sensors:**

Connect the ultrasonic sensors to the Raspberry Pi using jumper wires. Typically, ultrasonic sensors have four pins: VCC, GND, Trig (trigger), and Echo. Connect them as follows:

* VCC to 5V pin on Raspberry Pi
* GND to GND pin on Raspberry Pi
* Trig to a GPIO pin (e.g., GPIO17)
* Echo to another GPIO pin (e.g., GPIO18)

**3. Write Python Script for Sensor Data Collection:**

Write a Python script on the Raspberry Pi to collect data from the ultrasonic sensors. Use the GPIO library to interface with the sensors. Here's a basic example script for a single sensor:

import RPi.GPIO as GPIO

import time

# Set GPIO pin numbers

TRIG\_PIN = 17

ECHO\_PIN = 18

# Setup GPIO mode and pins

GPIO.setmode(GPIO.BCM)

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

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

def measure\_distance():

# Trigger ultrasonic sensor

GPIO.output(TRIG\_PIN, True)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, False)

# Wait for echo to be high (start of pulse)

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

pulse\_start = time.time()

# Wait for echo to be low again (end of pulse)

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

pulse\_end = time. time()

# Calculate distance using speed of sound (343m/s)

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150 # in centimeters

return distance

try:

while True:

# Measure distance

dist = measure\_distance()

print("Distance: {:.2f} cm".format(dist))

time.sleep(1) # Delay for 1 second between measurements

except KeyboardInterrupt:

print("Measurement stopped by user")

GPIO.cleanup()

**4. Set Up Cloud/Mobile App Communication:**

You can use platforms like AWS IoT, Google Cloud IoT, or MQTT for cloud communication. If you want to send data to a mobile app, you might need to set up a REST API or use WebSocket communication. The method depends on the specific cloud service or mobile app platform you're using.

**5. Modify Script for Cloud/Mobile Communication:**

Modify your Python script to send data to the cloud or mobile app server. For example, if you're using MQTT for communication, you can use the **paho-mqtt** library. Install it using **pip install paho-mqtt**.

Remember to replace **mqtt.example.com** with your MQTT broker address and adjust the **mqtt\_topic** according to your preference.

import paho.mqtt.client as mqtt

# MQTT broker settings

mqtt\_broker = "mqtt.example.com"

mqtt\_port = 1883

mqtt\_topic = "parking\_space"

client = mqtt.Client()

try:

client.connect(mqtt\_broker, mqtt\_port, 60)

while True:

# Measure distance

dist = measure\_distance()

print("Distance: {:.2f} cm".format(dist))

# Publish data to MQTT topic

client.publish(mqtt\_topic, str(dist))

time.sleep(1) # Delay for 1 second between measurements

except KeyboardInterrupt:

print("Measurement stopped by user")

client.disconnect()

GPIO.cleanup()

**6. Run the Script:**

Save the modified script on your Raspberry Pi and run it. You should see the sensor data being sent to the specified destination (cloud or mobile app server).

Please note that this is a basic example, and depending on your specific project requirements and cloud/mobile app setup, you might need to implement additional security measures and error handling in your Python script.

7.Reference:

These information can be done by using IBM, Online skill up and internet.