SMART PARKING

PHASE3:

DEVELOPMENT PART 1:

Building an IoT sensor system with Raspberry Pi integration to hit upon parking area occupancy the use of ultrasonic sensors entails both hardware and software program additives. Here's a step-by means of-step guide to help you get started:

Hardware additives:

Raspberry Pi (any model with GPIO pins)

Ultrasonic distance sensors (e.g, HC-SR04)

Jumper wires

Breadboard (non-compulsory)

Energy supply for Raspberry Pi

Net connectivity (wireless or Ethernet)

Software additives:

Raspberry Pi OS (e.g Raspbian)

Python (pre-installed on Raspberry Pi)

RPi.GPIO library (for controlling GPIO pins)

Google Sheets (for storing and visualizing facts)

Google Sheets API (for programmatic get admission to to Google Sheets)

Setup Raspberry Pi:

Installation Raspberry Pi OS to your Raspberry Pi.

Connect it to the internet, both via wi-fi or Ethernet.

Connect Ultrasonic Sensor:

Connect the VCC pin of the ultrasonic sensor to a 5V pin at the Raspberry Pi.

Connect the GND pin to a floor pin at the Raspberry Pi.

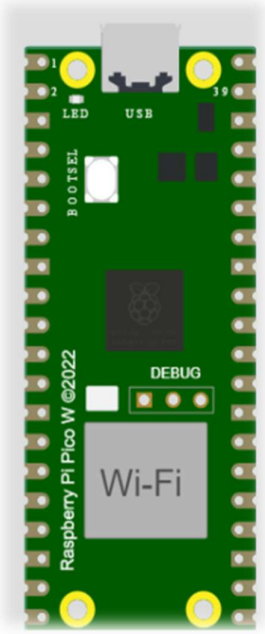
Join the TRIG pin to a GPIO pin (e.G., GPIO 17).

Connect the ECHO pin to some other GPIO pin (e.G., GPIO 18).

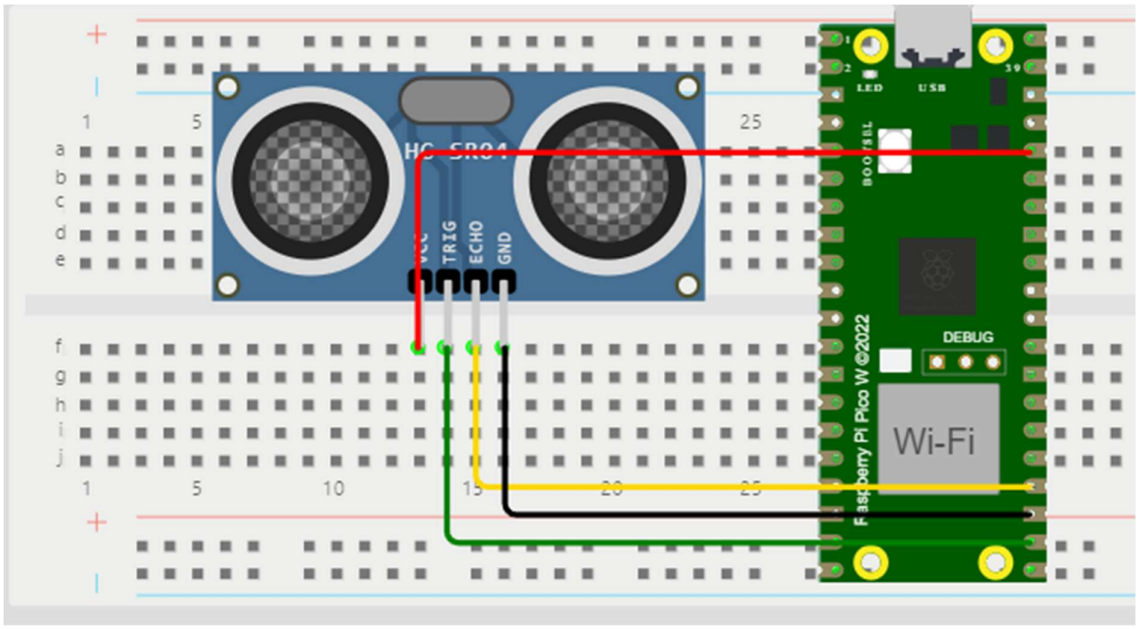
You may use a breadboard to make the connections.

Datagram:

For raspberry pi w:



Simulate module:



Python scripts for raspberry module:

```
import RPi.GPIO as GPIO

import time

import gspread

from oauth2client.service_account import ServiceAccountCredentials


# Set up Ultrasonic Sensor GPIO pins

TRIG_PIN = 17

ECHO_PIN = 18


# Initialize GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG_PIN, GPIO.OUT)
```

```
GPIO.setup(ECHO_PIN, GPIO.IN)


# Define Google Sheets-related variables
credentials_file = 'smart-parkingiot-095b74826f5a'  # Replace with your credentials file
google_sheets_name = 'parkingdata'  # Replace with your Google Sheets document name
sheet_name = 'Sheet1'  # Replace with your sheet name


# Initialize Google Sheets
scope = ['https://www.googleapis.com/auth/spreadsheets']
credentials = ServiceAccountCredentials.from_json_keyfile_name(credentials_file, scope)
gc = gspread.authorize(credentials)
worksheet = gc.open(google_sheets_name).worksheet(sheet_name)


try:
    while True:
        # Measure distance using ultrasonic sensor
        GPIO.output(TRIG_PIN, False)
        time.sleep(2)

        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 (343 m/s)

    # Get current timestamp
    timestamp = time.strftime("%Y-%m-%d %H:%M:%S")

    # Determine occupancy status based on distance threshold
    distance_threshold = 50
    occupancy_status = "Occupied" if distance < distance_threshold else "Vacant"

    # Log data to Google Sheets
    worksheet.append_row([timestamp, distance, occupancy_status])

    print(f"Timestamp: {timestamp}, Distance: {distance} cm, Occupancy: {occupancy_status}")
    time.sleep(5)  # Adjust the time interval as needed
```

```
except KeyboardInterrupt:
    GPIO.cleanup()
```

Web java script code:

```
// JavaScript code to process CSV data and update the website
Papa.parse('parkingdata.csv', {
    download: true,
    header: true,
    skipEmptyLines: true,
    dynamicTyping: true,
    complete: function(results) {
        const data = results.data;

        // Initialize counts
        let vacantCount = 0;
        let occupiedCount = 0;

        // Process the data and calculate counts
        data.forEach((entry) => {
            const status = entry['status'];

            if (status === 'vacant') {
                vacantCount++;
            } else if (status === 'occupied') {
                occupiedCount++;
```

```
        }
    });


    // Update the HTML with the counts
    document.getElementById('vacantCount').textContent = vacantCount;
    document.getElementById('occupiedCount').textContent =
occupiedCount;
  }
});
```

Parking datasheet:

Parkingdata.csv

S.NO	Timestamp	Distance(cm)	status
1	1/1/2023 11:00	50	Occupied
2	1/1/2023 11:10	233	Vacant
3	1/1/2023 11:30	315	Vacant
4	1/1/2023 12:00	233	Vacant
5	1/1/2023 12:00	108	Vacant
6	1/1/2023 12:00	50	Occupied
7	1/1/2023 12:00	344	Vacant
8	1/1/2023 12:00	400	Vacant
9	1/1/2023 12:00	112	Vacant
10	1/1/2023 12:00	50	Occupied
11	1/1/2023 12:00	50	Occupied
12	1/1/2023 12:00	124	Vacant
13	1/1/2023 12:00	50	Occupied
14	1/1/2023 12:00	222	Vacant
15	1/1/2023 12:00	50	Occupied
16	1/1/2023 12:00	312	Vacant
17	1/1/2023 12:00	50	Occupied
18	1/1/2023 12:00	200	Vacant
19	1/1/2023 12:00	50	Occupied
20	1/1/2023 12:00	213	Vacant
21	1/1/2023 12:00	50	Occupied
22	1/1/2023 12:00	200	Vacant
23	1/1/2023 12:00	400	Vacant
24	1/1/2023 12:00	253	Vacant
25	1/1/2023 12:00	50	Occupied
26	1/1/2023 12:00	123	Vacant
27	1/1/2023 12:00	50	Occupied
28	1/1/2023 12:00	105	Vacant
29	1/1/2023 12:00	50	Occupied
30	1/1/2023 12:00	100	Vacant

Tracking and Visualization:

Get right of entry to your Google Sheets document to reveal parking area occupancy.

Use Google Sheets capabilities for statistics analysis and visualization.

Hardware Setup:

Securely mount the ultrasonic sensors inside the parking place.

Make sure they have got an unobstructed view of the parking spaces.

Scaling:

For large parking areas, you can want a couple of Raspberry Pis and sensors.

Every Raspberry Pi can log information to a vital Google Sheets record or a separate record for its region.

Preservation and power:

Ensure the Raspberry Pi has a reliable energy supply.

Periodically check and preserve the sensors and connections.

This setup permits you to create a smart parking system that constantly video display units parking space occupancy and logs the statistics in a Google Sheets report for real-time tracking and evaluation.

THANK YOU!