Smart Parking using IoT

**Description:**

Smart parking systems that leverage IoT (Internet of Things) technology, combined with the compact and efficient Raspberry Pi Pico W microcontroller, offer innovative solutions to urban parking challenges. Raspberry Pi Pico W, with its wireless capabilities, connects parking sensors and cameras to the internet, allowing real-time monitoring and management of parking spaces. These systems detect and relay information on parking space availability to a centralized platform or mobile app, enabling drivers to quickly locate and reserve parking spots, reducing congestion and pollution. By utilizing Raspberry Pi Pico W, smart parking solutions enhance convenience, reduce traffic congestion, and improve urban mobility while optimizing parking resource utilization.

**Components:**

* **Raspberry Pi Pico W:** The Pico W is a microcontroller board with Wi-Fi and Bluetooth capabilities, making it suitable for wireless communication and data transfer.
* **Ultrasonic Sensors:** Ultrasonic sensors can be installed at each parking space to detect the presence of vehicles.
* **LED Displays:** LED displays can be placed at the entrance of the parking lot to indicate the number of available spaces.
* **Server/Cloud:** A server or cloud platform is used to collect, store, and analyse data from the sensors.
* **Mobile App/Web Interface:** Users can access information about parking space availability via a mobile app or web interface

**Working Process:**

* **Sensor Data Collection:** Ultrasonic sensors are placed at each parking space to detect the presence of a vehicle. When a vehicle enters or leaves a parking space, the sensor sends data to the Raspberry Pi Pico W.
* **Data Processing:** The Pico W processes the sensor data and sends it to the server or cloud platform using Wi-Fi.
* **Data Analysis:** The server or cloud platform analyses the data to determine parking space availability.
* **User Interface:** Users can access real-time information about available parking spaces via a mobile app or a web interface.

**Features:**

* **Real-Time Monitoring:** Users can check the availability of parking spaces in real-time.
* **Alerts:** The system can send alerts to users when a parking space becomes available.
* **Reservation:** Some systems allow users to reserve parking spaces in advance.
* **Payment Integration:** Payment systems can be integrated into the app for convenient transactions.
* **Analytics:** Data collected can be used to analyse parking patterns and optimize space utilization.

**Benefits:**

* **Reduced Congestion:** Smart parking systems help reduce congestion by directing drivers to available parking spaces.
* **Time and Fuel Savings:** Drivers save time and fuel by avoiding the search for parking.
* **Environmentally Friendly:** Reduced traffic congestion leads to lower emissions and a smaller carbon footprint.
* **Revenue Generation:** Parking operators can optimize space utilization and generate more revenue.

**Implementation:**

* Deploy ultrasonic sensors at each parking space.
* Connect the sensors to the Raspberry Pi Pico W.
* Write code to handle data collection, processing, and communication.
* Set up a server or cloud platform to store and analyse data.
* Develop a mobile app or web interface for users.

**Considerations:**

* **Security:** Ensure data security and privacy in the system.
* **Scalability:** Design the system to handle a large number of parking spaces.
* **Reliability:** The system should be robust and reliable for real-world applications.

**Coding:**

**import machine**

**import time**

**import urequests**

**import network**

**# WiFi credentials**

**WIFI\_SSID = "Wokwi-GUEST"**

**WIFI\_PASSWORD = ""**

**# ThingSpeak API settings**

**THINGSPEAK\_API\_KEY = "J9U2H9SP2QM95J1G"**

**THINGSPEAK\_CHANNEL\_ID = 2319272**

**# Define the GPIO pins for the sensor**

**TRIG\_PIN = machine.Pin(2, machine.Pin.OUT)**

**ECHO\_PIN = machine.Pin(3, machine.Pin.IN)**

**# Set the occupancy threshold in centimeters**

**OCCUPANCY\_THRESHOLD = 30.0 # Adjust this value as needed**

**# Connect to WiFi**

**wifi = network.WLAN(network.STA\_IF)**

**if not wifi.isconnected():**

**print("Connecting to WiFi...")**

**wifi.active(True)**

**wifi.connect(WIFI\_SSID, WIFI\_PASSWORD)**

**while not wifi.isconnected():**

**pass**

**print("Connected to WiFi")**

**def send\_to\_thingspeak(occupancy):**

**url = "https://api.thingspeak.com/update?api\_key={}&field1={}".format(THINGSPEAK\_API\_KEY, occupancy)**

**response = urequests.get(url)**

**response.close()**

**try:**

**while True:**

**# Trigger the ultrasonic sensor**

**TRIG\_PIN.on()**

**time.sleep\_us(10)**

**TRIG\_PIN.off()**

**pulse\_start = time.ticks\_us()**

**pulse\_end = time.ticks\_us()**

**# Wait for the echo signal**

**while ECHO\_PIN.value() == 0:**

**pulse\_start = time.ticks\_us()**

**while ECHO\_PIN.value() == 1:**

**pulse\_end = time.ticks\_us()**

**# Calculate the distance**

**pulse\_duration = time.ticks\_diff(pulse\_end, pulse\_start)**

**distance = (pulse\_duration / 1000000) \* 17150 # Speed of sound in cm/s**

**# Check for occupancy**

**if distance < OCCUPANCY\_THRESHOLD:**

**indhu = "Occupied"**

**print(indhu)**

**send\_to\_thingspeak(1) # Send 1 to indicate occupancy to ThingSpeak**

**else:**

**indhu1 = "Not Occupied"**

**print(indhu1)**

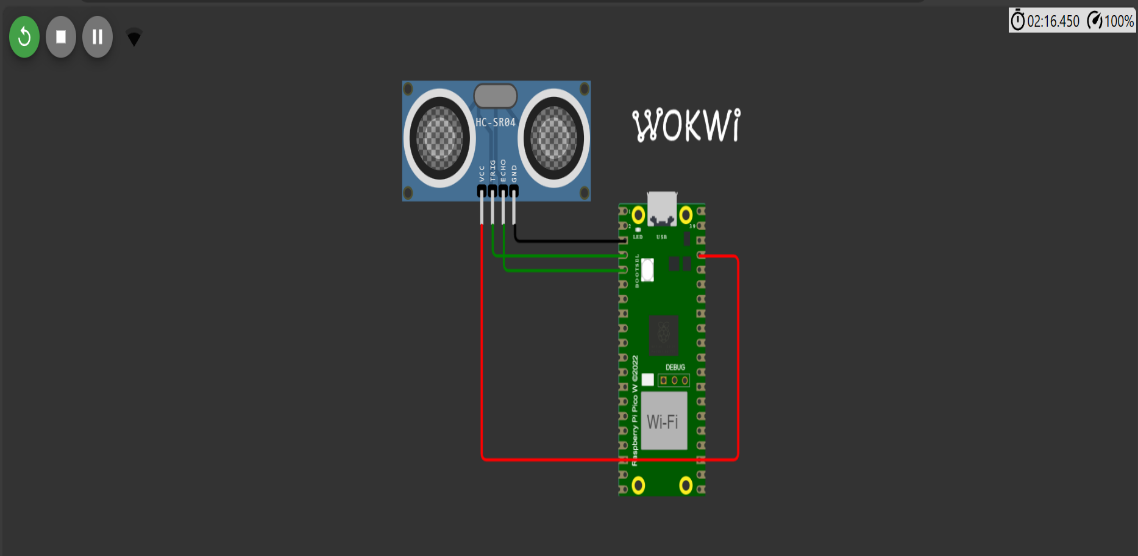
**send\_to\_thingspeak(0) # Send 0 to indicate no occupancy to ThingSpeak**

**time.sleep(5) # Delay for 15 seconds (ThingSpeak allows up to 15-second updates)**

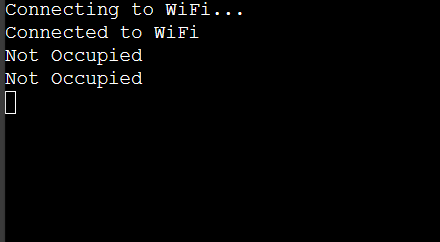
**except KeyboardInterrupt:**

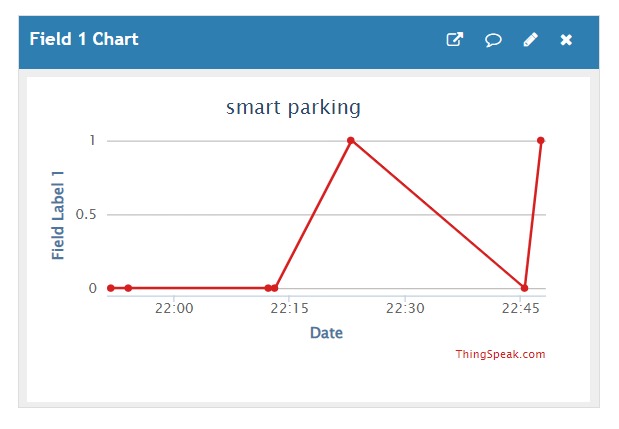
**Pass**

**Simulation Circuit:**



**Output:**

**In WOKWI:**

 **In ThingSpeak(cloud platform):**

**Conclusion:**

* This document provides an overview of the Ultrasonic Sensor Data Logger program, its connections, sensor details, ThingSpeak configuration, and how to run it. Adjust the Wi-Fi credentials, ThingSpeak details, and occupancy threshold as needed for your specific setup.