PHASE 3: Development Part 1

INTRODUCTION :

Smart parking IoT systems employ sensors to detect parking space occupancy, transmitting this data to a central server or cloud platform via wireless communication. This real-time data is then accessible to users through mobile apps or digital signs, making it effortless to find available parking spaces and even reserve them in advance. These systems not only enhance user convenience but also contribute to reduced traffic congestion, lower carbon emissions, and optimized space utilization, benefiting both drivers and parking facility operators. By combining IoT sensors, data analytics, and user-friendly interfaces, smart parking systems are reshaping urban mobility and city planning, offering a more efficient and sustainable approach to managing parking resources.

STEPS:

Setting up Raspberry Pi:

- Get a Raspberry Pi and install the necessary operating system (e.g., Raspberry Pi OS).
- Connect the Raspberry Pi to the internet via Ethernet or Wi-Fi.

Wokwi Configuration:

- Wokwi is a simulator for electronics and IoT projects. You can use it to simulate the Raspberry Pi and sensor integration before deploying it to real hardware. Follow Wokwi's setup instructions to create a virtual environment for your project.

Ultrasonic Sensor Integration:

- Connect the ultrasonic sensor to the Raspberry Pi using jumper wires. The HC-SR04 typically has four pins: VCC, Trig, Echo, and GND.
- Connect VCC to a 5V pin on the Raspberry Pi.
- Connect Trig and Echo to GPIO pins of the Raspberry Pi (e.g., GPIO 17 for Trig and GPIO 27 for Echo).
- Connect GND to a ground pin on the Raspberry Pi.

Python Script for Ultrasonic Sensor:

- Write a Python script to interface with the ultrasonic sensor. You can use the GPIO library to control the GPIO pins.
- Use the script to trigger the sensor, measure the distance, and detect if a parking space is occupied based on the distance measured.
- The script can periodically run and send the occupancy data to the cloud server.

Cloud Integration:

- Choose a cloud service provider (e.g., AWS, Google Cloud, or Azure) for IoT and set up an IoT Core or IoT Hub.
- Create a Thing or Device in the IoT service to represent your Raspberry Pi.
- Generate security certificates and keys for the device.
- Modify your Python script to establish a secure connection to the cloud using MQTT or HTTP, depending on your cloud provider.
- Send data (e.g., occupancy status) to the cloud whenever a parking space's occupancy changes.

Mobile App Server:

- Set up a mobile app server to receive and process data from the cloud.
- You can use a cloud-based server or build one using a framework like Flask.
- Create RESTful APIs to receive occupancy data from the cloud and store it in a database.

Mobile App:

- Develop a mobile app that communicates with the mobile app server to display real-time parking space occupancy status.

Testing and Deployment:

- Test your system thoroughly in the Wokwi simulator or on real hardware.
- Once everything is working correctly, deploy the system to your target environment (e.g., a parking lot).

Monitoring and Maintenance:

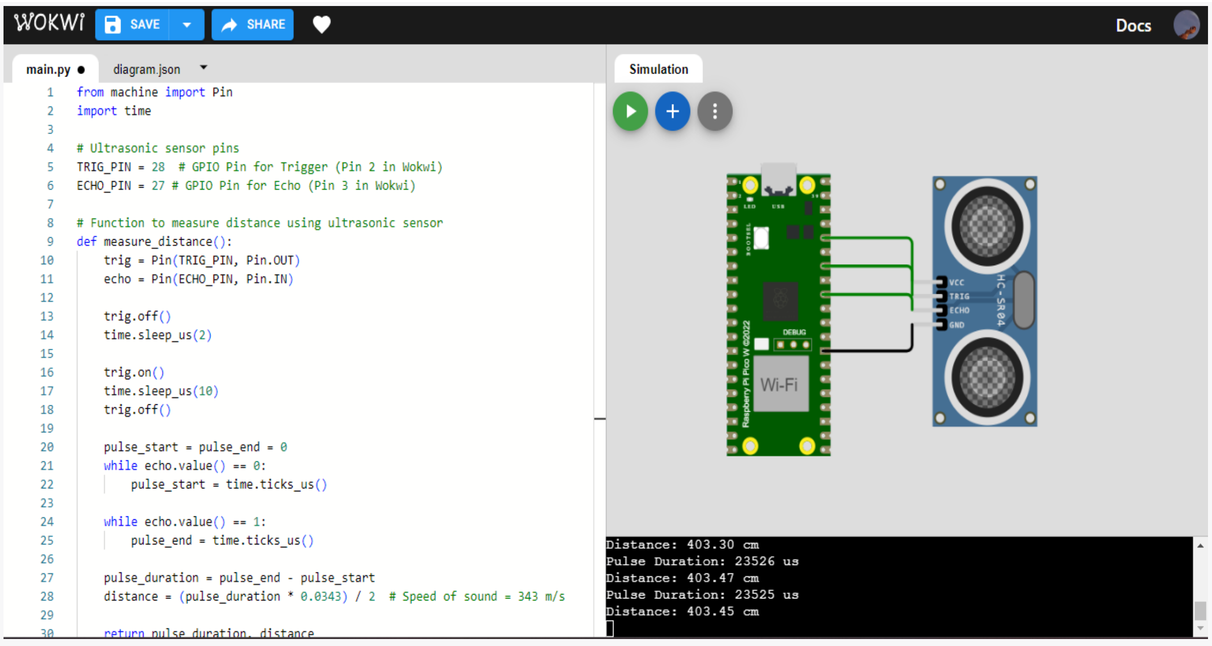
- Implement monitoring and alerting to ensure the system's reliability.
- Regularly update and maintain the system, including firmware updates for the Raspberry Pi.

Connection of IoT sensors to detect parking space occupancy.

CODE:

```
 from machine import Pin
import time

# Ultrasonic sensor pins
TRIG_PIN = 28  # GPIO Pin for Trigger (Pin 2 in Wokwi)
ECHO_PIN = 27 # GPIO Pin for Echo (Pin 3 in Wokwi)

# Function to measure distance using ultrasonic sensor
def measure_distance():
    trig = Pin(TRIG_PIN, Pin.OUT)
    echo = Pin(ECHO_PIN, Pin.IN)

    trig.off()
    time.sleep_us(2)

    trig.on()
    time.sleep_us(10)
    trig.off()

    pulse_start = pulse_end = 0
    while echo.value() == 0:
        pulse_start = time.ticks_us()

    while echo.value() == 1:
        pulse_end = time.ticks_us()

    pulse_duration = pulse_end - pulse_start
    distance = (pulse_duration * 0.0343) / 2  # Speed of sound = 343
m/s

    return pulse_duration, distance

# Main function to collect and print data
def main():
    while True:
        try:
            pulse_duration, distance = measure_distance()
            print("Pulse Duration: {} us".format(pulse_duration))
```

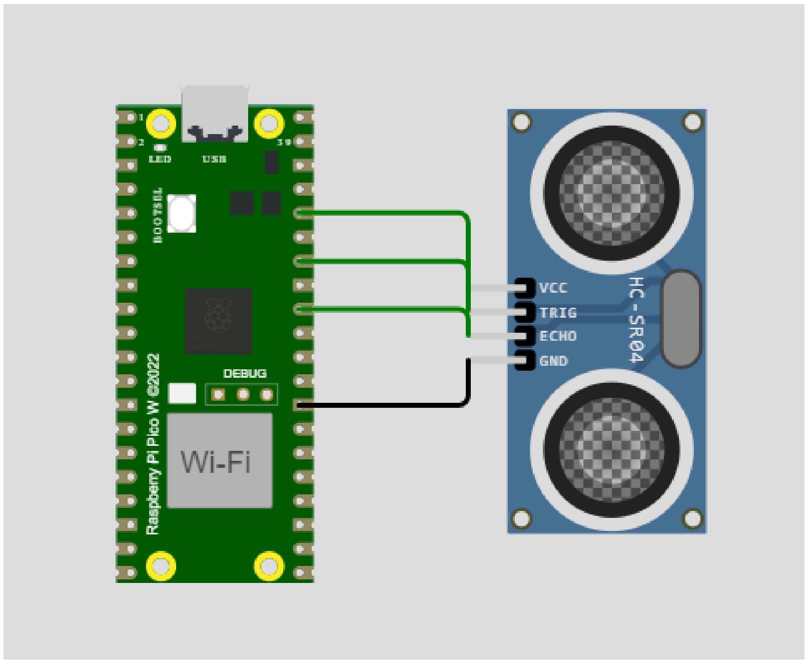
```python
            print("Distance: {:.2f} cm".format(distance))

            time.sleep(1)  # Wait for a second before the next
measurement

        except Exception as e:
            print("Error:", e)

if __name__ == "__main__":
    main()
```



```python
from machine import Pin
import time

# Ultrasonic sensor pins
TRIG_PIN = 28  # GPIO Pin for Trigger (Pin 2 in Wokwi)
ECHO_PIN = 27 # GPIO Pin for Echo (Pin 3 in Wokwi)

# Function to measure distance using ultrasonic sensor
def measure_distance():
    trig = Pin(TRIG_PIN, Pin.OUT)
    echo = Pin(ECHO_PIN, Pin.IN)

    trig.off()
    time.sleep_us(2)

    trig.on()
    time.sleep_us(10)
    trig.off()

    pulse_start = pulse_end = 0
    while echo.value() == 0:
        pulse_start = time.ticks_us()

    while echo.value() == 1:
        pulse_end = time.ticks_us()

    pulse_duration = pulse_end - pulse_start
    distance = (pulse_duration * 0.0343) / 2  # Speed of sound = 343 m/s

    return pulse_duration, distance
```

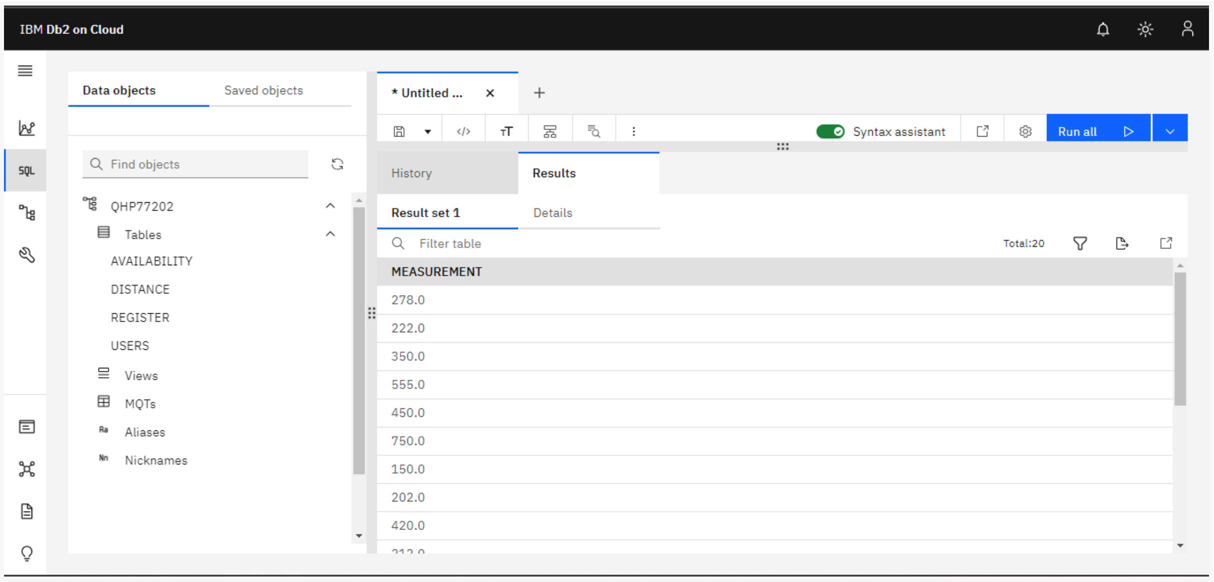
```
Distance: 403.30 cm
Pulse Duration: 23526 us
Distance: 403.47 cm
Pulse Duration: 23525 us
Distance: 403.45 cm
```

CIRCUIT DIAGRAM:



"Since Wokwi's MicroPython Raspberry Pi does not support IBM Cloud DB2 modules, simulated values are added to the table."

CLOUD CONNECTION:



MEASUREMENT
278.0
222.0
350.0
555.0
450.0
750.0
150.0
202.0
420.0

CONCLUSION:

The sensors of raspberry pi in wokwi are connected. The simulated values are successfully added to IBM Cloud DB2 database.