IBM PROJECT

Project name: Smart parking

Phase 3

```
import BlynkLib
import BlynkTimer
import RPi.GPIO as GPIO

# Initialize Blynk
BLYNK_AUTH = 'your_blynk_auth_token'
blynk = BlynkLib.Blynk(BLYNK_AUTH)

# Initialize GPIO pins for sensors
sensor_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN)

# Initialize timer
timer = BlynkTimer.Timer()

# Function to check parking status
def check_parking_status():
```

```
    if GPIO.input(sensor_pin) == GPIO.HIGH:
        blynk.virtual_write(0, 255)  # Set V0 to 255 if parking spot is occupied
    else:
        blynk.virtual_write(0, 0)  # Set V0 to 0 if parking spot is vacant

# Register virtual pin handler
@blynk.handle_event('read V0')
def read_virtual_pin_handler(pin):
    check_parking_status()

# Start Blynk
blynk.run()

# Start timer to periodically check parking status
timer.set_interval(3000, check_parking_status)  # Check every 3 seconds
```

Explanation:

Include necessary libraries, like ESP8266WiFi and BlynkSimpleEsp8266, to work with the ESP8266-based device and Blynk service.

Define your Blynk authentication token, Wi-Fi credentials, and the pins used for the ultrasonic sensor.

Initialize the Blynk connection with your credentials.

In the setup function, set the trigger and echo pins as OUTPUT and INPUT, respectively, and initialize serial communication for debugging.

In the loop function, trigger an ultrasonic sensor reading by sending a pulse from the trigger pin and measuring the time it takes for the echo pin to receive the return pulse.

Calculate the distance based on the measured duration and send this value to the Blynk app using Blynk.virtualWrite.

Print the distance to the serial monitor for debugging.

You can add your own logic to determine parking space availability based on the distance measured and notify users through the Blynk app or other means.

Delay for a set period before taking the next sensor reading and ensure that Blynk is running with Blynk.run().

This code is a basic example of how to integrate an ultrasonic

sensor with Blynk to create a smart parking system. Depending on your specific requirements, you may need to extend the code to include additional features like notifications, user interfaces, and parking space management.