Name:I.Mohamed Nasir

NM ID:AU952621106010

College Name:S.Veerasamy Chettiar College of Engineering and Technology

SMART PARKING

PHASE 4: Development Part 2

To build the IOT Smart parking system

Requirements:

Wokwi

ThingSpeak

ESP32

Relay

GND

VCC

Resistor

Ultrasonic sensor

LED

Wokwi:

Wokwi is an embedded systems and IoT simulator supporting ESP32, Arduino, and the Raspberry Pi Pico. Your code never leaves your computer - Wokwi runs the simulation inside VS Code, using the firmware binaries from your project.

ThingSpeak:

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak™ from your devices, create instant visualizations of live data, and send alerts using web services like Twitter® and Twilio®.

Procedure:

Step 1:

Connect the ESP32 microcontroller with the ultrasonic sensor.

Step 2:

Connect the LED's and the relay with the GPIO pins of ESP32.

Step 3:

Setup each step in the code and connect Wi-Fi to the ESP32.

Step 4:

Declare and initialize each component of the smart water management. For consumption the water level "Flow sensor" is not available in the wokwi so, instead of this we used "Ultrasonic sensor".

Step 5:

The condition for glowing LED's and motor is based on the level of the water using the ultrasonic sensor.

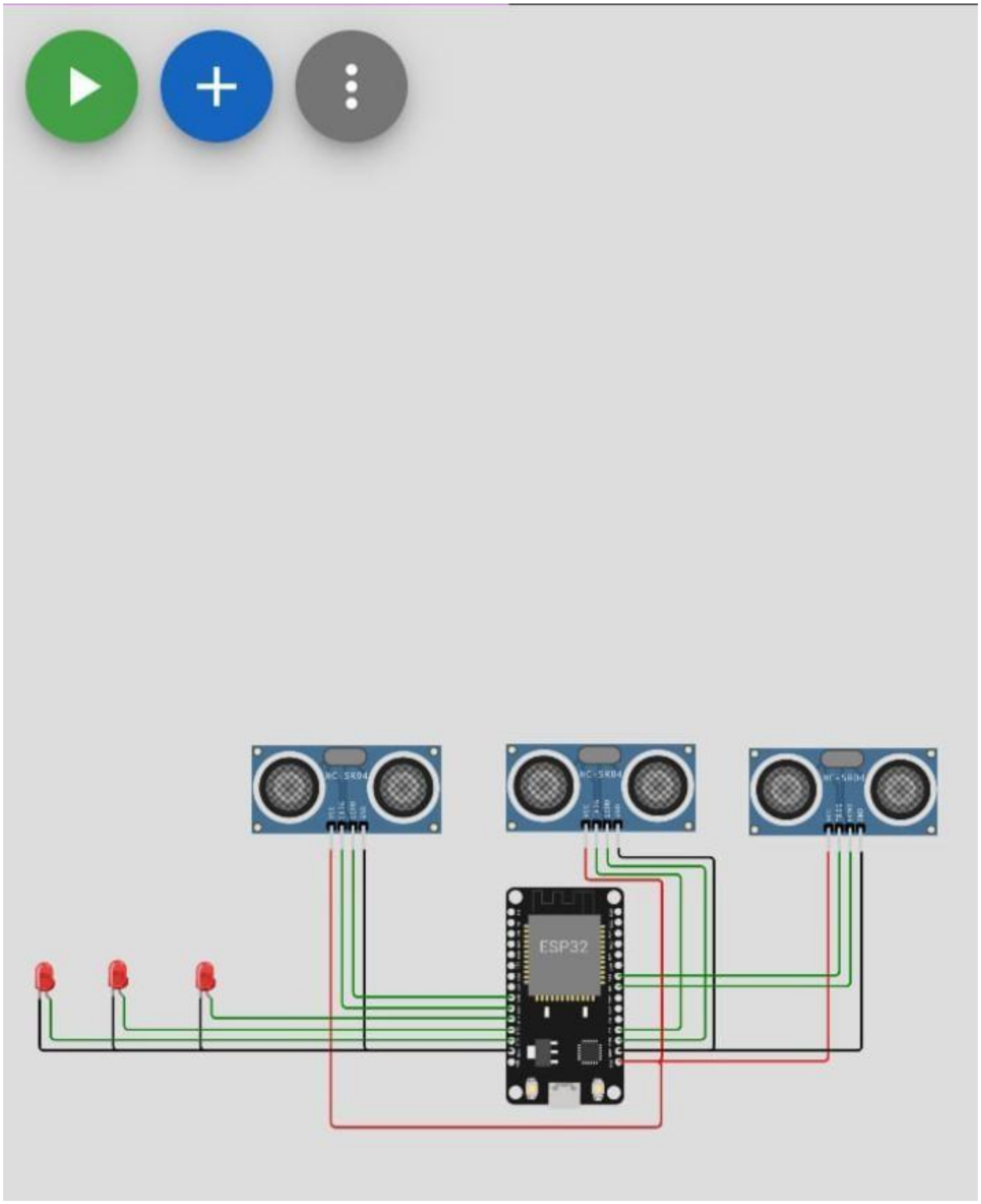
Step 6:

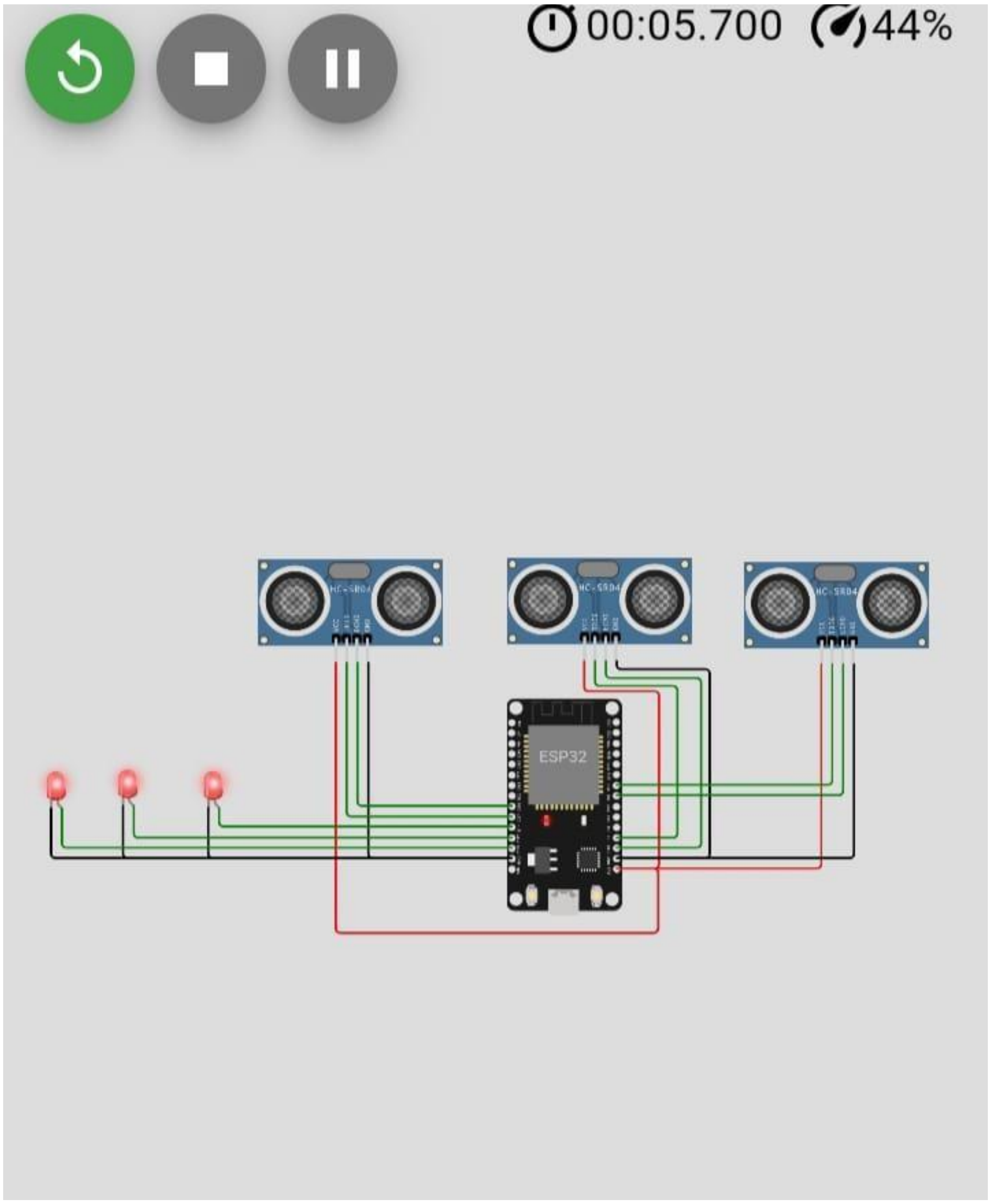
Run the simulation and check the distance of the water level through LED's glow, Motor run and push up messages.

Step 7:

View the real-time data transmitting in the sensor through ThingSpeak.

SIMULATION:



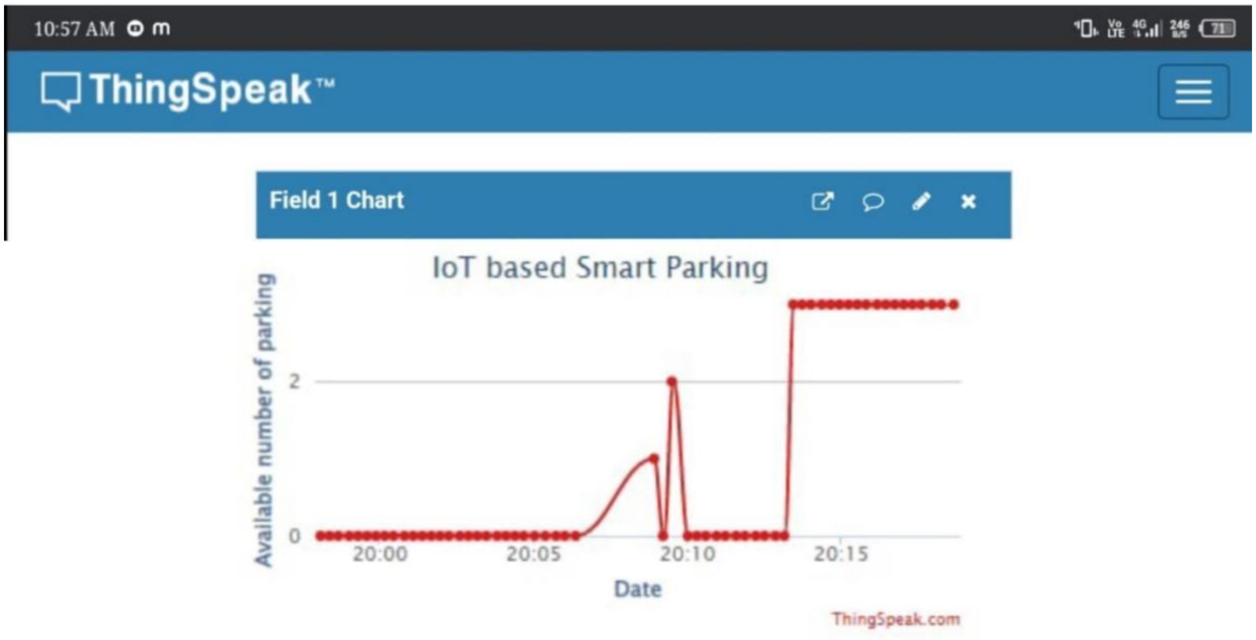

00:05.700
44%
HC-SR04
VCC
TRIG
ECHO
GND
HC-SR04
HC-SR04
ESP32

SOURCE CODE :

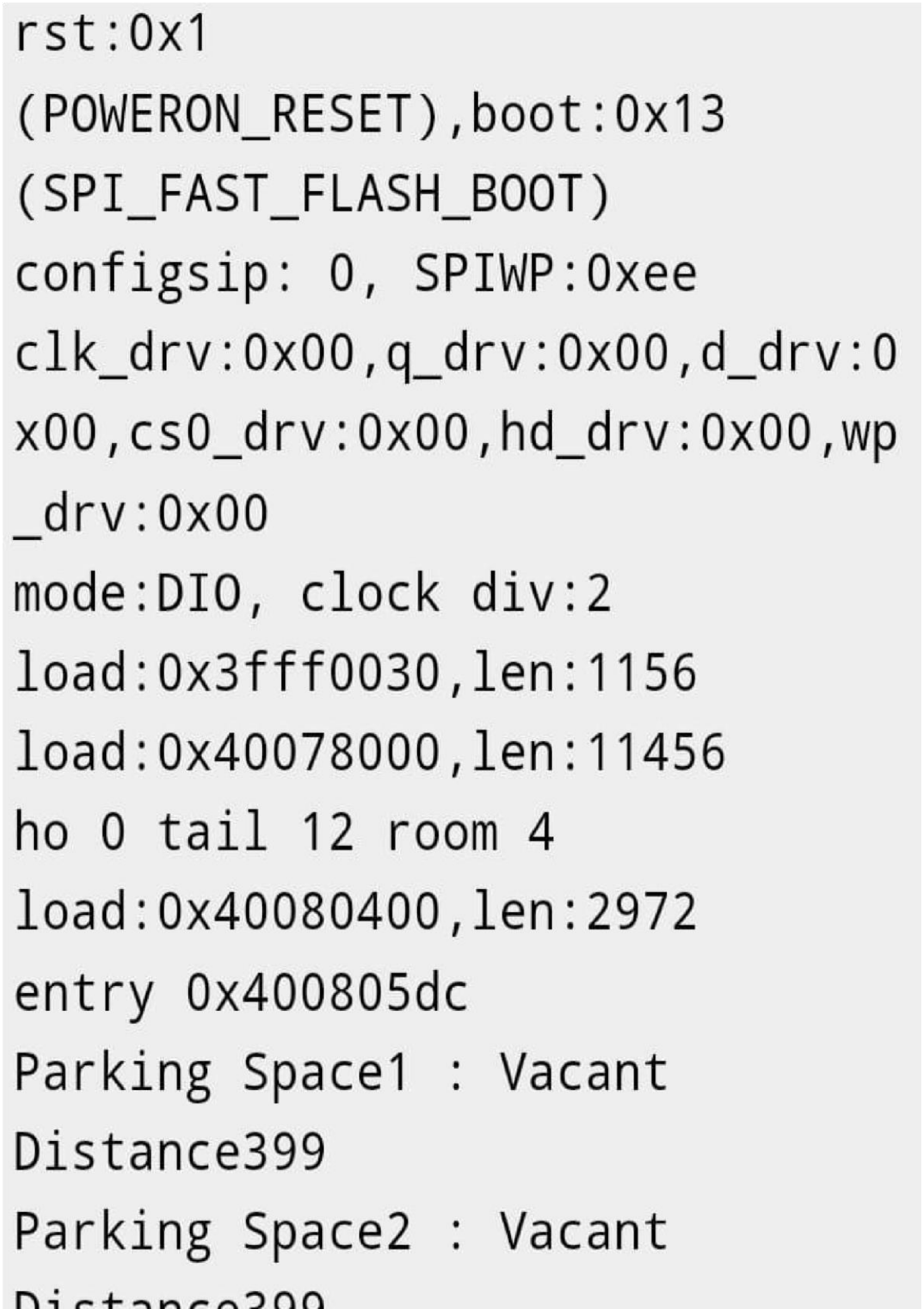
```
Import time
From machine import Pin, Signal, UART
Import network
Import urequests

# Wi-Fi Configuration
Ssid = “your_wifi_ssid”
Password = “your_wifi_password”
thingSpeakApiKey =
“your_thingspeak_api_key”

# Pins Configuration
trigPin1 = Pin(27, Pin.OUT)
echoPin1 = Pin(26, Pin.IN)
ledPin1 = Signal(Pin(13, Pin.OUT))
trigPin2 = Pin(2, Pin.OUT)
echoPin2 = Pin(15, Pin.IN)
ledPin2 = Signal(Pin(12, Pin.OUT))
trigPin3 = Pin(18, Pin.OUT)
echoPin3 = Pin(5, Pin.IN)
ledPin3 = Signal(Pin(14, Pin.OUT))

uart = UART(0, 115200)

# Connect to Wi-Fi
Sta_if = network.WLAN(network.STA_IF)
Sta_if.active(True)
Sta_if.connect(ssid, password)

While not sta_if.isconnected():
   Time.sleep(1)
   Print(“Connecting to WiFi…”)

Print(“Connected to WiFi”)
```

```
Def send_data_to_thingspeak(distance,
field2_value):
   Base_url =
https://api.thingspeak.com/update?
   Api_key_param = “api_key=” +
thingSpeakApiKey
   Field1_param = “field1=” + str(distance)
   Field2_param = “field2=” + str(field2_value)
   url = base_url + api_key_param + “&” +
field1_param + “&” + field2_param
   response = urequests.get(url)

   if response.status_code == 200:
      print(“Data sent to ThingSpeak
successfully”)
   else:
      print(“Failed to send data to ThingSpeak,
HTTP error code:”, response.status_code)

def measure_distance(trig_pin, echo_pin):
   trig_pin.value(0)
   time.sleep_us(2)
   trig_pin.value(1)
   time.sleep_us(10)
   trig_pin.value(0)
   duration =
machine.time_pulse_us(echo_pin, 1, 30000)  #
Timeout of 30ms
   if duration > 0:
      distance = duration * 0.034 / 2
   else:
      distance = 0
   return distance

while True:
   distance1 = measure_distance(trigPin1,
echoPin1)
   if distance1 < 200:
      ledPin1.on()
      print(“Parking Space 1: Occupied”)
      send_data_to_thingspeak(distance1, 1)
   else:
      ledPin1.off()
      print(“Parking Space 1: Vacant”)
      send_data_to_thingspeak(distance1, 0)

   time.sleep(1)

   distance2 = measure_distance(trigPin2,
echoPin2)
   if distance2 < 200:
      ledPin2.on()
      print(“Parking Space 2: Occupied”)
      send_data_to_thingspeak(distance2, 1)
   else:
      ledPin2.off()
      print(“Parking Space 2: Vacant”)
      send_data_to_thingspeak(distance2, 0)

   time.sleep(1)

   distance3 = measure_distance(trigPin3,
echoPin3)
   if distance3 < 200:
      ledPin3.on()
      print(“Parking Space 3: Occupied”)
      send_data_to_thingspeak(distance3, 1)
   else:
      ledPin3.off()
      print(“Parking Space 3: Vacant”)
      send_data_to_thingspeak(distance3, 0)

   time.sleep(1
```

Real time output:



SOURCE CODE OUTPUT:



```
rst:0x1
(POWERON_RESET),boot:0x13
(SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0
x00,cs0_drv:0x00,hd_drv:0x00,wp
_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Parking Space1 : Vacant
Distance399
Parking Space2 : Vacant
Distance399
```

Code for Blynk App:

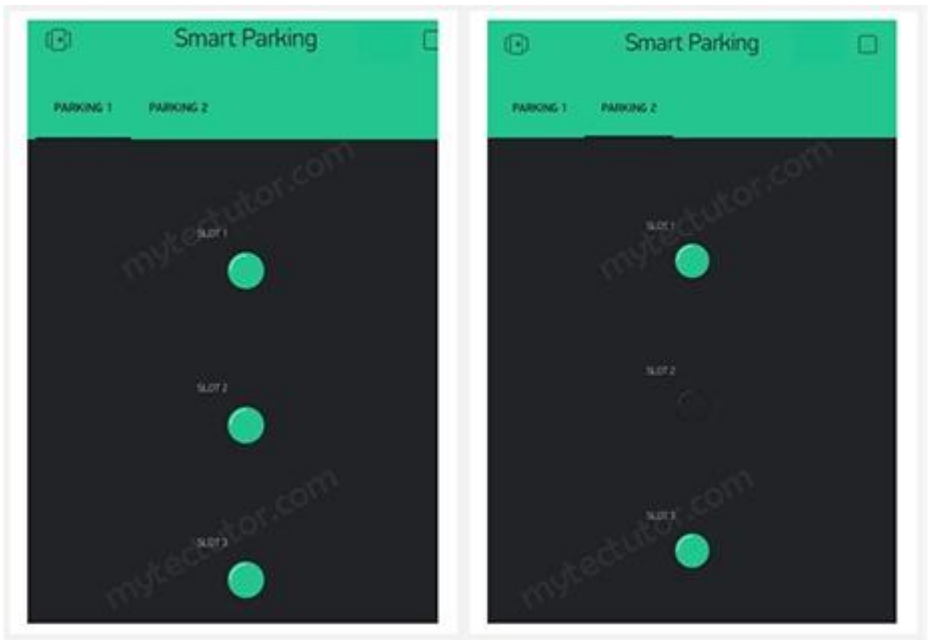
```
import BlynkLib

import network

import time

from machine import Pin, Signal, UART

import urequests


# Wi-Fi Configuration

WIFI_SSID = "your_wifi_ssid"  # Replace with your Wi-Fi SSID

WIFI_PASSWORD = "your_wifi_password"  # Replace with your Wi-Fi password

THING_SPEAK_API_KEY = "your_thingspeak_api_key"  # Replace with your ThingSpeak API key


# Pins Configuration

trigPin1 = Pin(27, Pin.OUT)

echoPin1 = Pin(26, Pin.IN)

ledPin1 = Signal(Pin(13, Pin.OUT))

trigPin2 = Pin(2, Pin.OUT)

echoPin2 = Pin(15, Pin.IN)

ledPin2 = Signal(Pin(12, Pin.OUT))

trigPin3 = Pin(18, Pin.OUT)

echoPin3 = Pin(5, Pin.IN)

ledPin3 = Signal(Pin(14, Pin.OUT))

uart = UART(0, 115200)


# Initialize Blynk

BLYNK_AUTH = 'YOUR_BLYNK_AUTH_TOKEN'  # Replace with your Blynk authentication token

blynk = BlynkLib.Blynk(BLYNK_AUTH)
```

```
# Connect to Wi-Fi
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect(WIFI_SSID, WIFI_PASSWORD)

while not sta_if.isconnected():
    time.sleep(1)
    print("Connecting to WiFi...")

print("Connected to WiFi")

# Define Blynk functions
@blynk.VIRTUAL_WRITE(1)
def v1_write_handler(value):
    # This function is called when V1 (virtual pin 1) value is updated in the Blynk app
    print('V1 Value:', value)

def send_data_to_thingspeak(distance, field2_value):
    base_url = 'https://api.thingspeak.com/update?'
    api_key_param = 'api_key=' + THING_SPEAK_API_KEY
    field1_param = 'field1=' + str(distance)
    field2_param = 'field2=' + str(field2_value)
    url = base_url + api_key_param + '&' + field1_param + '&' + field2_param
    response = urequests.get(url)
    if response.status_code == 200:
        print("Data sent to ThingSpeak successfully")
    else:
        print("Failed to send data to ThingSpeak, HTTP error code:", response.status_code)
```

```
def measure_distance(trig_pin, echo_pin):
    trig_pin.value(0)
    time.sleep_us(2)
    trig_pin.value(1)
    time.sleep_us(10)
    trig_pin.value(0)
    duration = machine.time_pulse_us(echo_pin, 1, 30000)  # Timeout of 30ms
    if duration > 0:
        distance = duration * 0.034 / 2
    else:
        distance = 0
    return distance

while True:
    distance1 = measure_distance(trigPin1, echoPin1)
    if distance1 < 200:
        ledPin1.on()
        print("Parking Space 1: Occupied")
        send_data_to_thingspeak(distance1, 1)
    else:
        ledPin1.off()
        print("Parking Space 1: Vacant")
        send_data_to_thingspeak(distance1, 0)
    time.sleep(1)

    distance2 = measure_distance(trigPin2, echoPin2)
    if distance2 < 200:
```

```
    ledPin2.on()
    print("Parking Space 2: Occupied")
    send_data_to_thingspeak(distance2, 1)
else:
    ledPin2.off()
    print("Parking Space 2: Vacant")
    send_data_to_thingspeak(distance2, 0)
time.sleep(1)

distance3 = measure_distance(trigPin3, echoPin3)
if distance3 < 200:
    ledPin3.on()
    print("Parking Space 3: Occupied")
    send_data_to_thingspeak(distance3, 1)
else:
    ledPin3.off()
    print("Parking Space 3: Vacant")
    send_data_to_thingspeak(distance3, 0)
time.sleep(1)

blynk.run()
```

Output:


Smart Parking
PARKING 1
PARKING 2
SLOT 1
SLOT 2
SLOT 3
Smart Parking
PARKING 1
PARKING 2
SLOT 1
SLOT 2
SLOT 3