2025 TEEP Progress Report Week-8

Used Wokwi to complete the tasks for Labs 1 through 8

(Before creating the physical project, please use **Wokwi** for online simulation to complete the initial design of the circuit and code. Then, use **Thonny** together with the **ESP32** development board to build the actual project. Please **record a video of the project** in operation, upload it to YouTube, and provide the video link(LAB7) so we can better understand your learning progress.)

Lab1- On ESP32 , complete the following tasks:

A. Hardware Setup

Connect a large push-button module to GPIO XX.

Connect an RGB LED as follows: LED_R → GPIO XX,LED_Y → GPIO XX and LED_G → GPIO XX

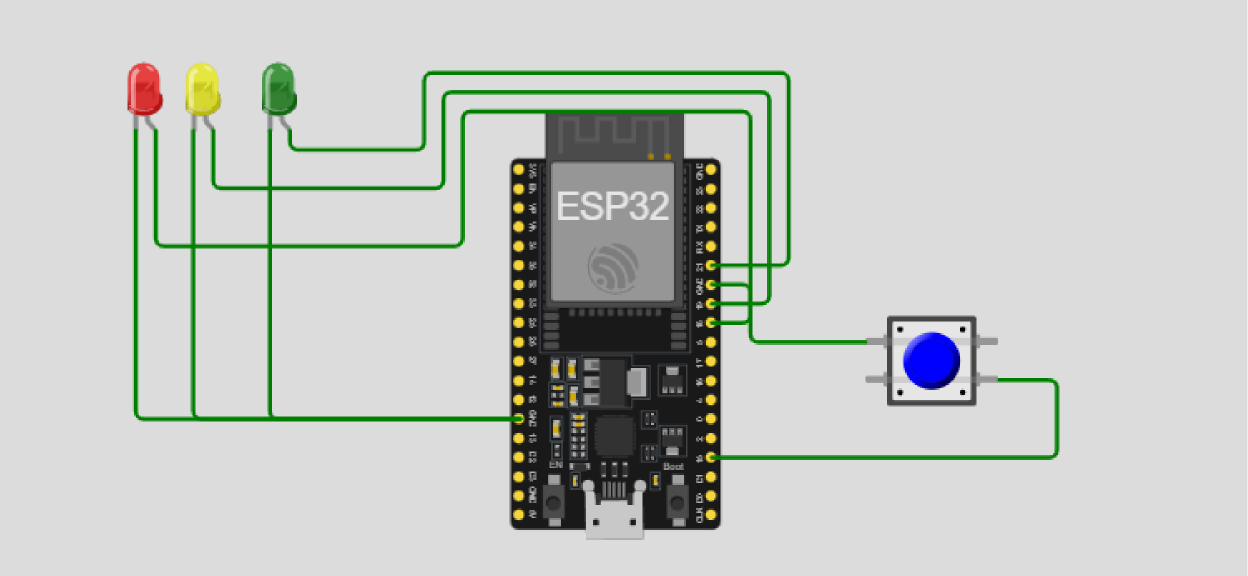
B. Program Behavior

On program start, configure an external interrupt whose source is the large push-button.Each button press advances the mode and performs the following behavior:

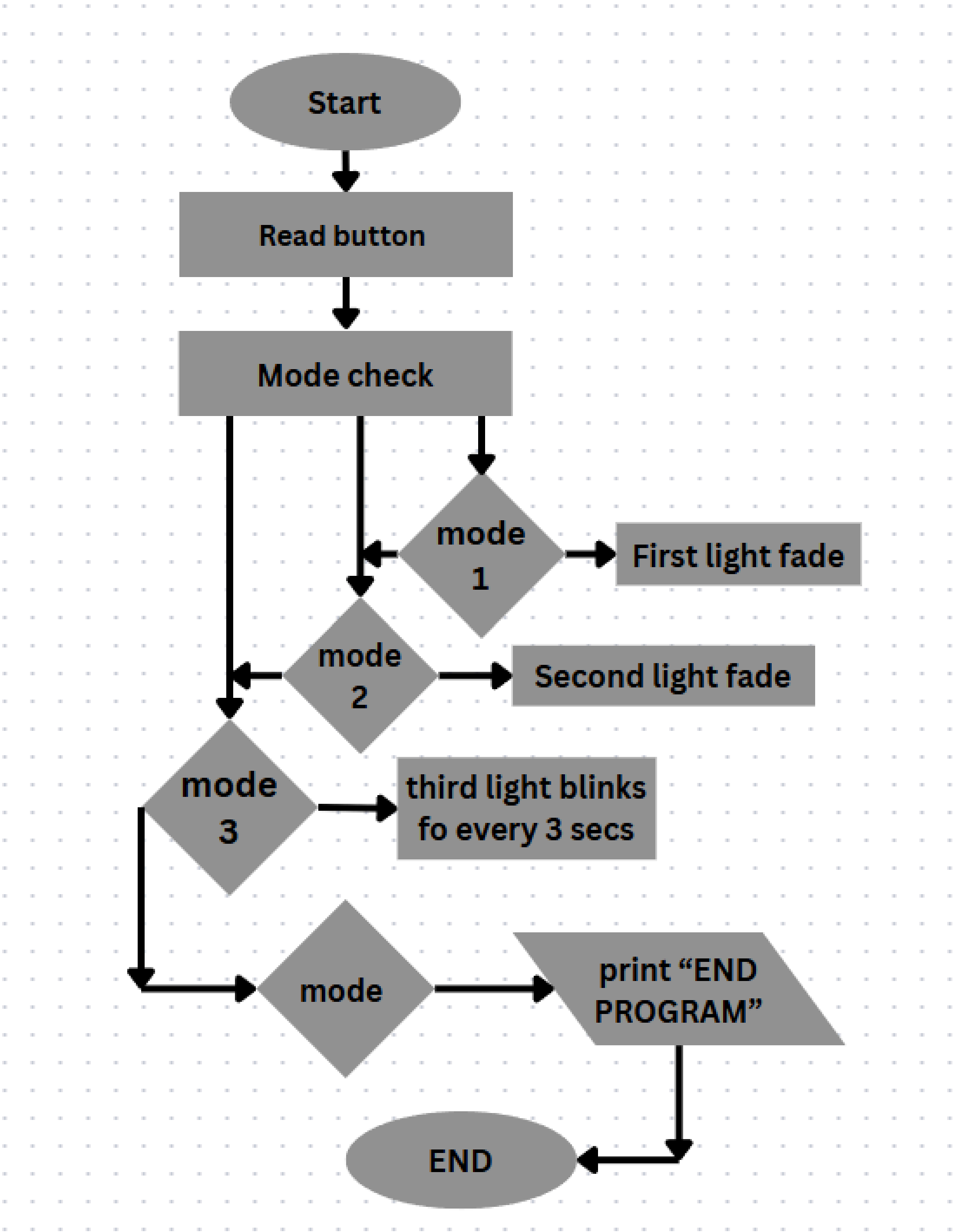
1. **Function 1:** First press: The red (R) LED quickly fades in and quickly fades out, repeating continuously until the next press.
2. **Function 2:** Second press: The yellow (Y) LED quickly fades in and quickly fades out, repeating continuously until the next press.
3. **Function 3:** Third press: The green (G) LED turns ON for 3 seconds and OFF for 1.5 seconds, repeating this cycle three times.
4. **Function 4:**Fourth press: Stop the program and print “End Program!” to the shell.

Connections :

- Button → `GPIO 15`
- LED_R → `GPIO 18`
- LED_Y → `GPIO 19`
- LED_G → `GPIO 21`

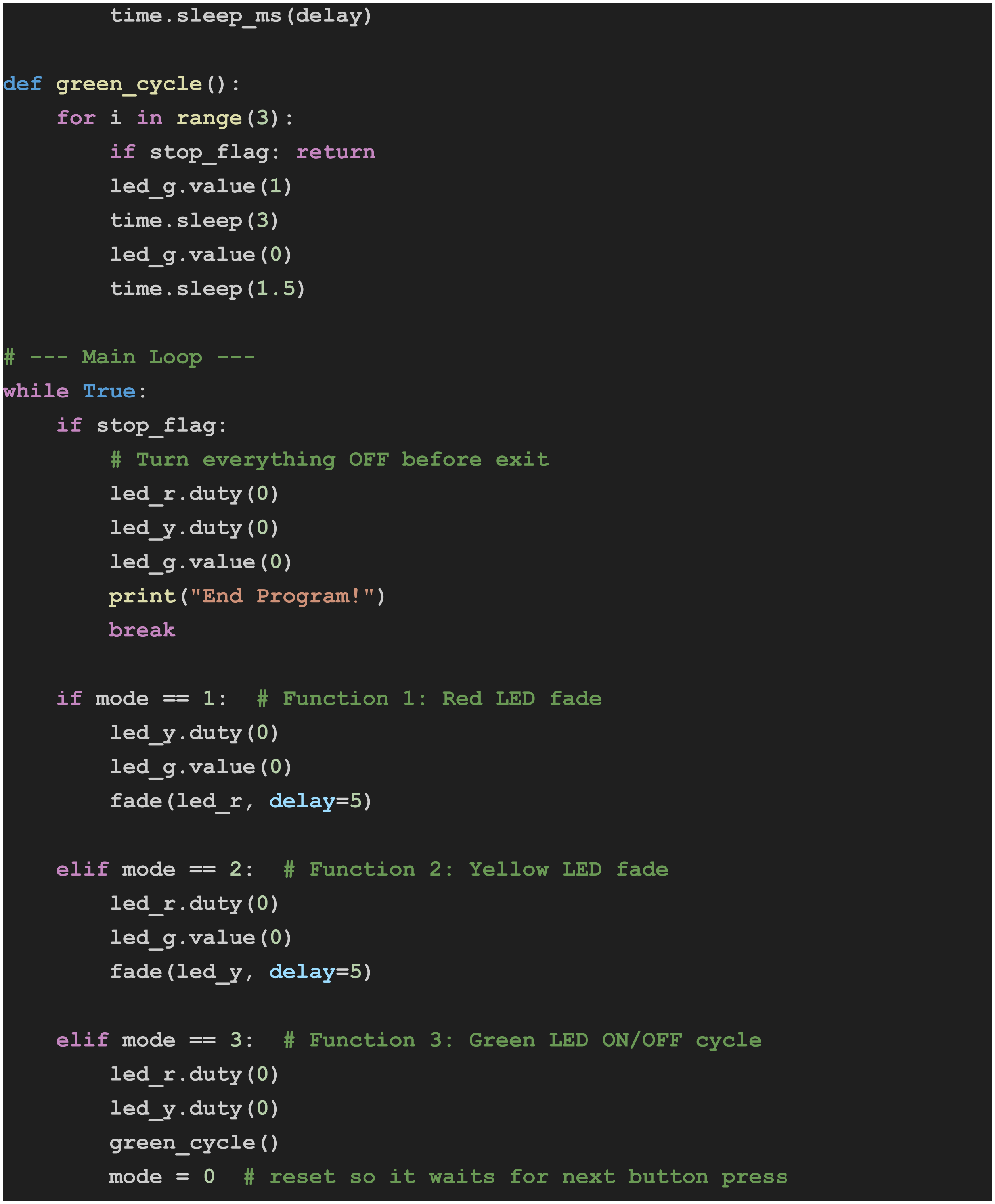


Flow chart :


Start
Read button
Mode check
mode 1
First light fade
mode 2
Second light fade
mode 3
third light blinks fo every 3 secs
mode
print “END PROGRAM”
END

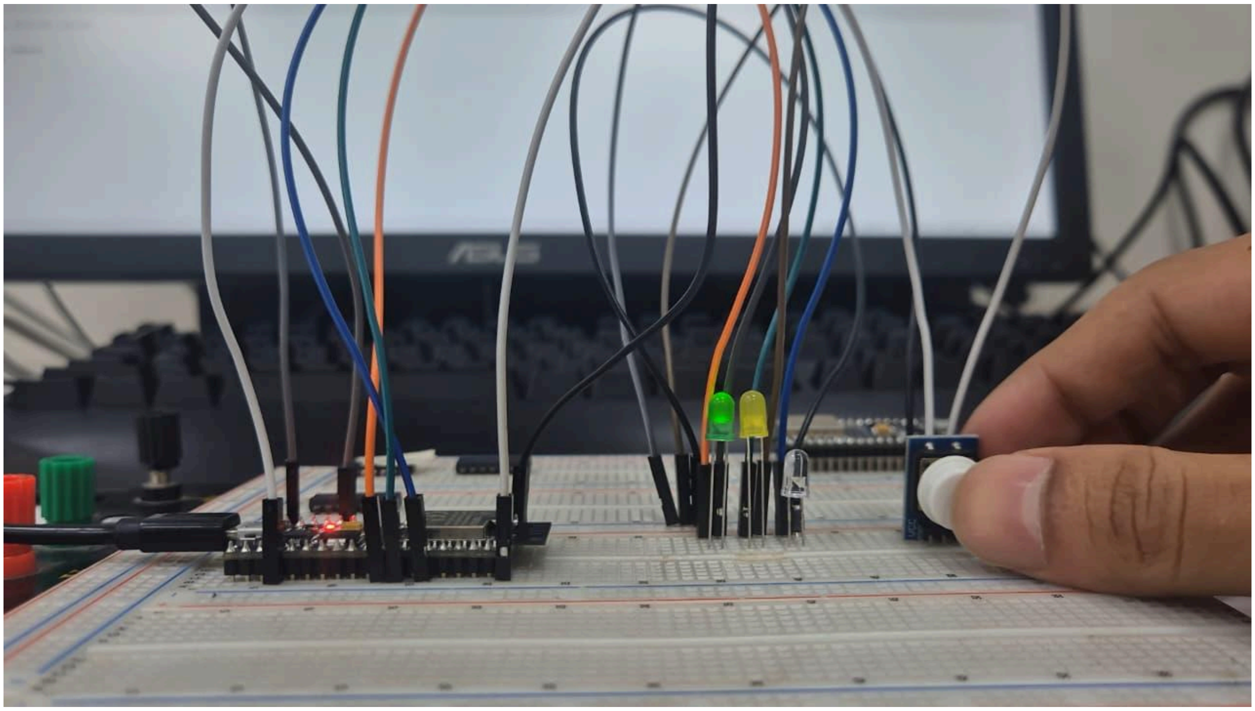
Source Code :

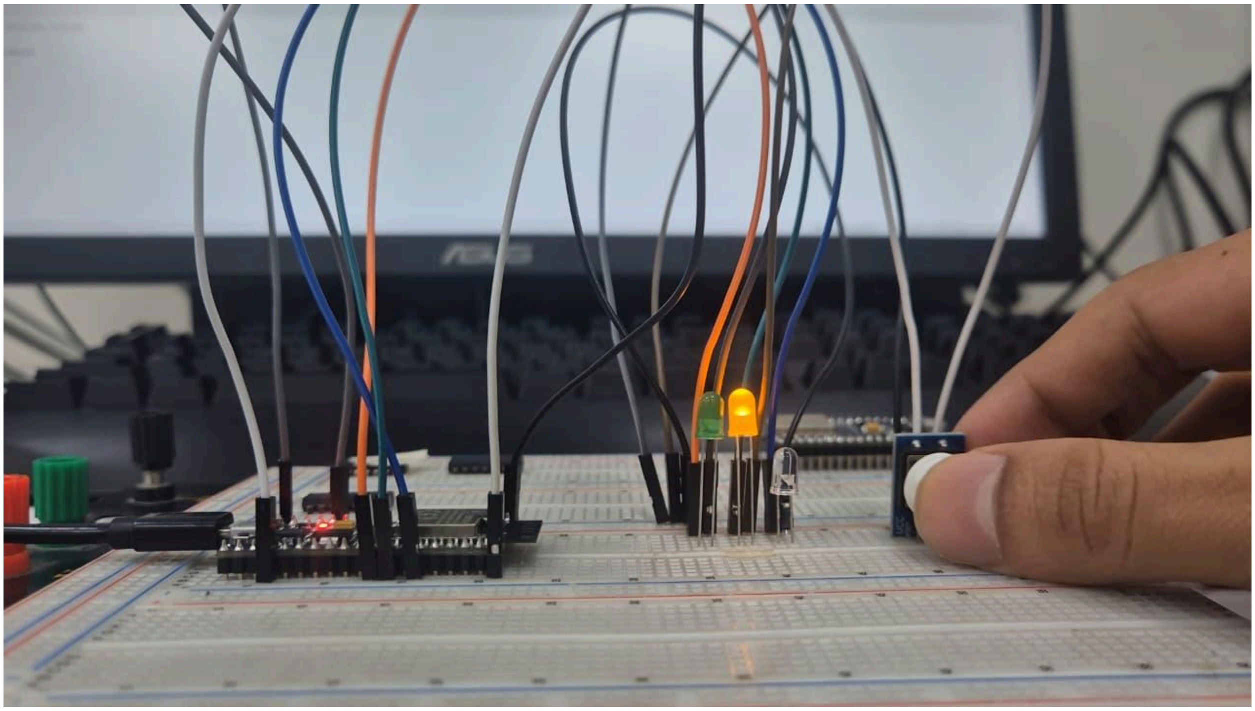
```
from machine import Pin, PWM
import time

# --- Pin Setup ---
button = Pin(15, Pin.IN, Pin.PULL_UP)   # Push button
led_r = PWM(Pin(18), freq=1000)         # Red LED with PWM
led_y = PWM(Pin(19), freq=1000)         # Yellow LED with PWM
led_g = Pin(21, Pin.OUT)                # Green LED (digital)

# --- Globals ---
mode = 0
last_press = 0  # debounce timestamp
stop_flag = False

# --- Interrupt handler ---
def button_handler(pin):
    global mode, last_press, stop_flag
    now = time.ticks_ms()
    if time.ticks_diff(now, last_press) > 300:  # debounce
        mode += 1
        print("Mode ->", mode)
        if mode == 4:   # last function → stop program
            stop_flag = True
        elif mode > 4:
            mode = 1
        last_press = now

# Attach external interrupt
button.irq(trigger=Pin.IRQ_FALLING, handler=button_handler)

# --- Helper Functions ---
def fade(led, delay=5):
    for duty in range(0, 1024, 64):   # Fade in
        if stop_flag: return
        led.duty(duty)
        time.sleep_ms(delay)
    for duty in range(1023, -1, -64): # Fade out
        if stop_flag: return
        led.duty(duty)
```

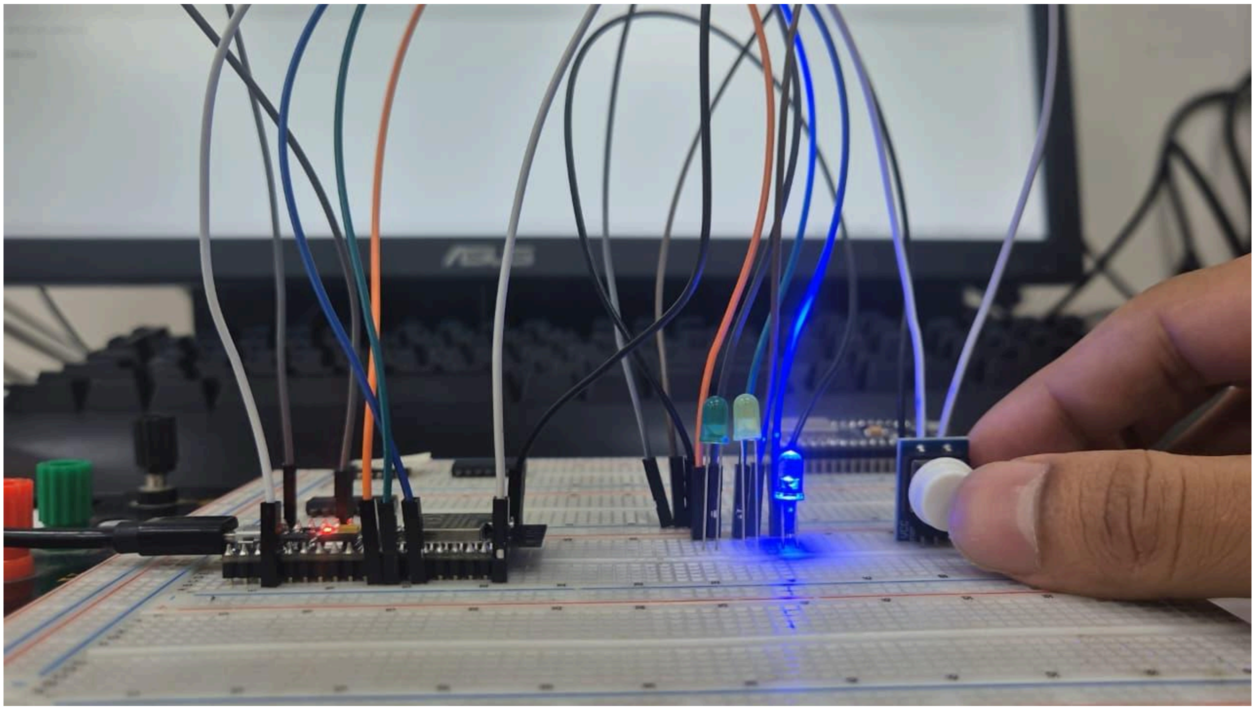


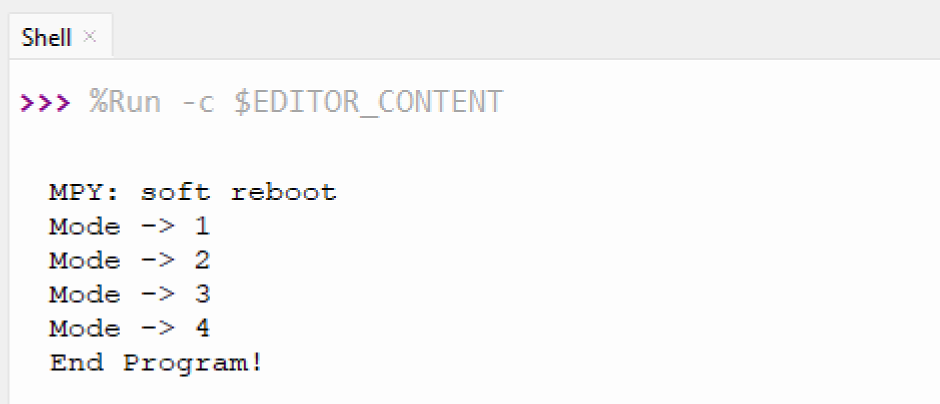
```
        time.sleep_ms(delay)

def green_cycle():
    for i in range(3):
        if stop_flag: return
        led_g.value(1)
        time.sleep(3)
        led_g.value(0)
        time.sleep(1.5)

# --- Main Loop ---
while True:
    if stop_flag:
        # Turn everything OFF before exit
        led_r.duty(0)
        led_y.duty(0)
        led_g.value(0)
        print("End Program!")
        break

    if mode == 1:  # Function 1: Red LED fade
        led_y.duty(0)
        led_g.value(0)
        fade(led_r, delay=5)

    elif mode == 2:  # Function 2: Yellow LED fade
        led_r.duty(0)
        led_g.value(0)
        fade(led_y, delay=5)

    elif mode == 3:  # Function 3: Green LED ON/OFF cycle
        led_r.duty(0)
        led_y.duty(0)
        green_cycle()
        mode = 0  # reset so it waits for next button press
```

Test Records :









```
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
  Mode -> 1
  Mode -> 2
  Mode -> 3
  Mode -> 4
  End Program!
```

Lab2 – On ESP32 and Wokwi , complete the following tasks:

A. Sensor and RGB LED

- ESP32 (physical): Sound sensor (digital output) + RGB LED.
- Wokwi (simulation): PIR motion sensor (digital output) + RGB LED.

B. GPIO Connections (replace XX with actual pins before deployment)

LED_R → GPIO XX,LED_Y → GPIO XX and LED_G → GPIO XX.

C. ThingSpeak Fields

- Create Field1: Sensor (to log sensor status or event)
- (Optional) Create Field2: Show Status (to display status string such as On/Off)

D. External Interrupt and Behavior

When the program starts, set up an external interrupt, with the source being:

- Wokwi: PIR sensor
- ESP32 (physical): Sound sensor

E. Each time an interrupt occurs, perform the following:

- Connect to ThingSpeak Upload data:

1. Sensor: write 1 (or event count value)
2. Show Status: write On (optional)

- Interrupt count vs. RGB behavior:

Funtion 1: First interrupt: R LED quickly fades in/out continuously until the next interrupt, and log the detection record to ThingSpeak

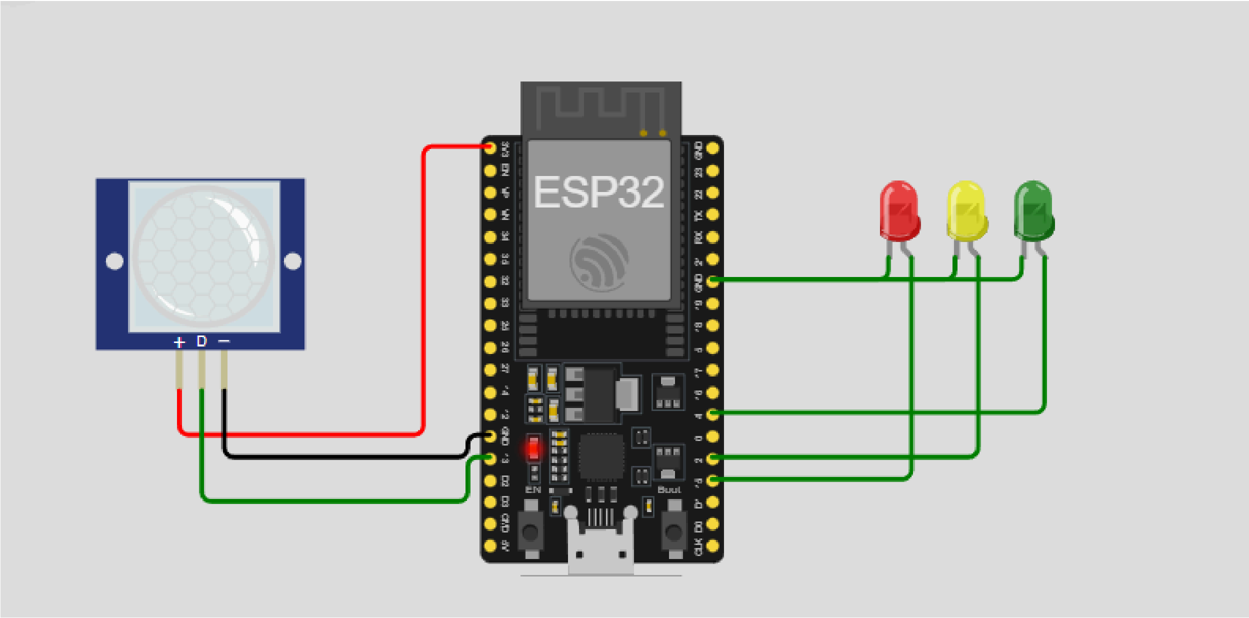
Funtion 2: Second interrupt: Y LED quickly fades in/out continuously until the next interrupt, and log the detection record to ThingSpeak

Funtion 3: Third interrupt: G LED stays ON for 3 seconds and OFF for 1.5 seconds, repeated 3 times, and log the detection record to ThingSpeak

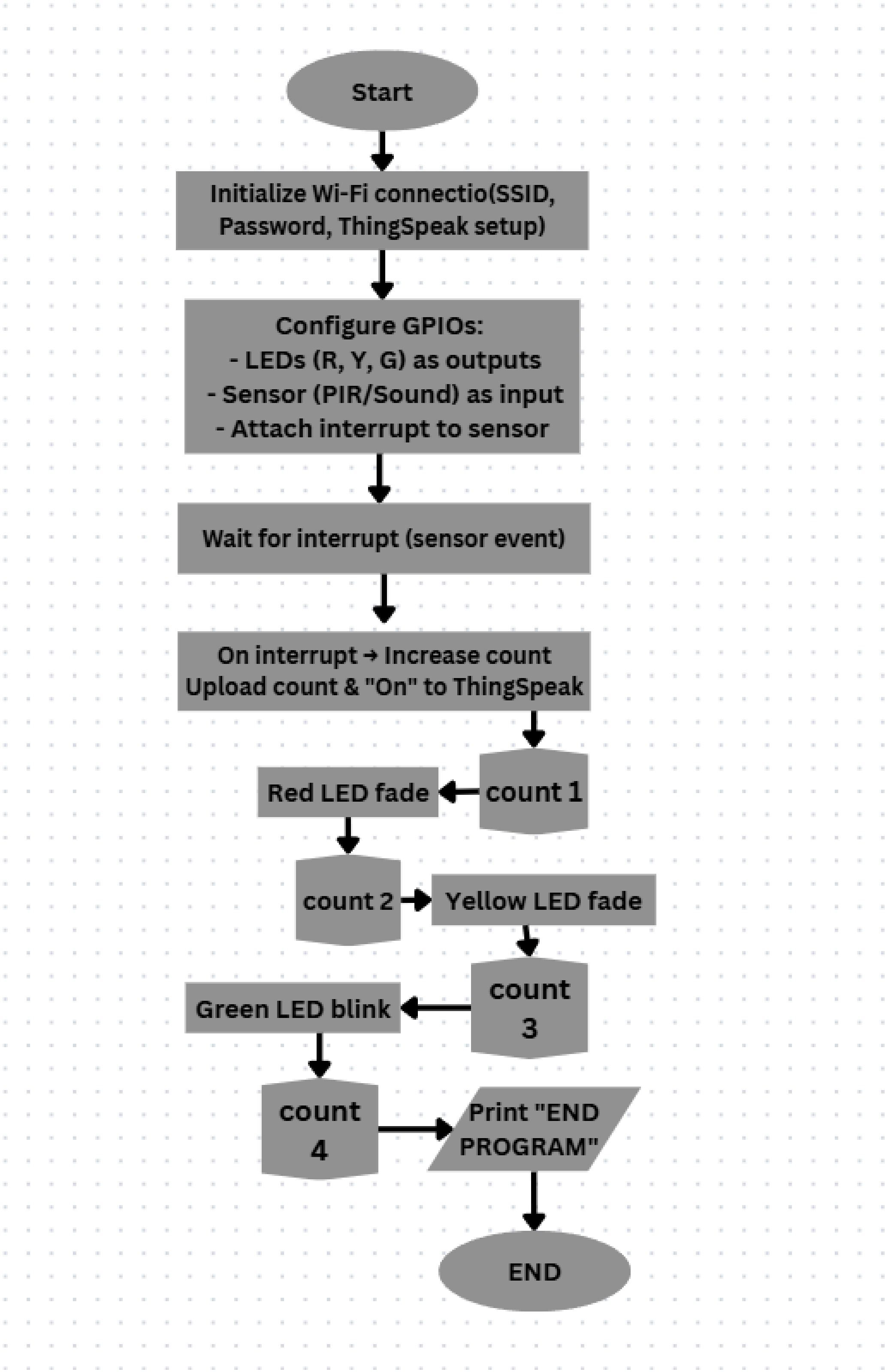
Funtion 4:Fourth interrupt: Display End Program! in the Shell, set Show Status=On on ThingSpeak, and stop the program

Connections :

Red LED (+)	GPIO15
Yellow LED (+)	GPIO2
Green LED (+)	GPIO4
Sensor OUT	GPIO13

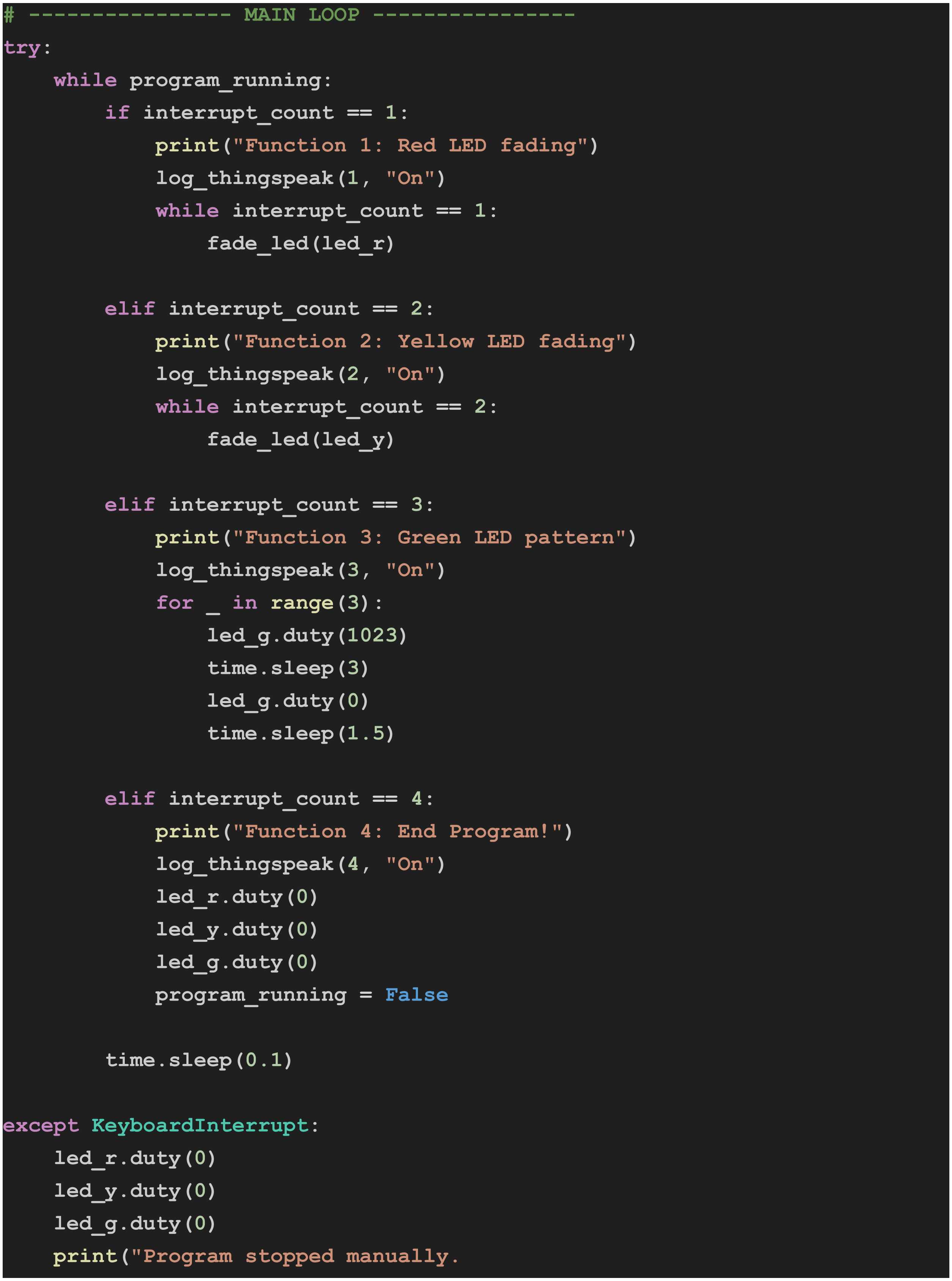


Flowchart :


Start
Initialize Wi-Fi connectio(SSID, Password, ThingSpeak setup)
Configure GPIOs:
- LEDs (R, Y, G) as outputs
- Sensor (PIR/Sound) as input
- Attach interrupt to sensor
Wait for interrupt (sensor event)
On interrupt → Increase count
Upload count & "On" to ThingSpeak
count 1
Red LED fade
count 2
Yellow LED fade
count 3
Green LED blink
count 4
Print "END PROGRAM"
END

Source code : [ESP32]

```
from machine import Pin, PWM
import time
import network
import urequests

# ---------------- USER SETTINGS ----------------
LED_R_PIN = 15
LED_Y_PIN = 2
LED_G_PIN = 4
SENSOR_PIN = 13  # PIR in Wokwi or Sound sensor in physical ESP32

WIFI_SSID = "ISILAB CR"
WIFI_PASSWORD = "isilab.ncut.CR"

THINGSPEAK_API_KEY = "CFGELDYE0PXQRXF0"
THINGSPEAK_URL = "https://thingspeak.mathworks.com/channels/3074763"

# ---------------- INITIALIZATION ----------------
led_r = PWM(Pin(LED_R_PIN), freq=500)
led_y = PWM(Pin(LED_Y_PIN), freq=500)
led_g = PWM(Pin(LED_G_PIN), freq=500)

sensor = Pin(SENSOR_PIN, Pin.IN)

interrupt_count = 0
program_running = True
last_trigger_time = 0  # for lockout (ms)

# ---------------- FUNCTIONS ----------------
def connect_wifi(ssid, password):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    print(f"Connecting to Wi-Fi: {ssid} ...")
    timeout = 10
    while not wlan.isconnected() and timeout > 0:
        time.sleep(1)
        timeout -= 1
    if wlan.isconnected():
        print("Wi-Fi connected:", wlan.ifconfig())
    else:
        print("Failed to connect to Wi-Fi")

def log_thingspeak(field1_val, field2_val=""):
    try:
        url =
f"{THINGSPEAK_URL}?api_key={THINGSPEAK_API_KEY}&field1={field1_val}"
        if field2_val:
            url += f"&field2={field2_val}"
        response = urequests.get(url)
        response.close()
        print(f"ThingSpeak uploaded -> field1: {field1_val}, field2:
{field2_val}")
    except Exception as e:
        print("ThingSpeak error:", e)

def fade_led(led, duration=0.01):
    for i in range(0, 1024, 20):
        led.duty(i)
        time.sleep(duration)
    for i in range(1023, -1, -20):
        led.duty(i)
        time.sleep(duration)

def interrupt_handler(pin):
    global interrupt_count, last_trigger_time, program_running
    now = time.ticks_ms()
    # lockout: accept only one interrupt every 3000 ms
    if time.ticks_diff(now, last_trigger_time) > 3000:
        last_trigger_time = now
        interrupt_count += 1
        print("Interrupt detected! Count:", interrupt_count)
        if interrupt_count >= 4:
            sensor.irq(handler=None)   # stop after 4th

# ---------------- SETUP ----------------
sensor.irq(trigger=Pin.IRQ_RISING, handler=interrupt_handler)
connect_wifi(WIFI_SSID, WIFI_PASSWORD)
```



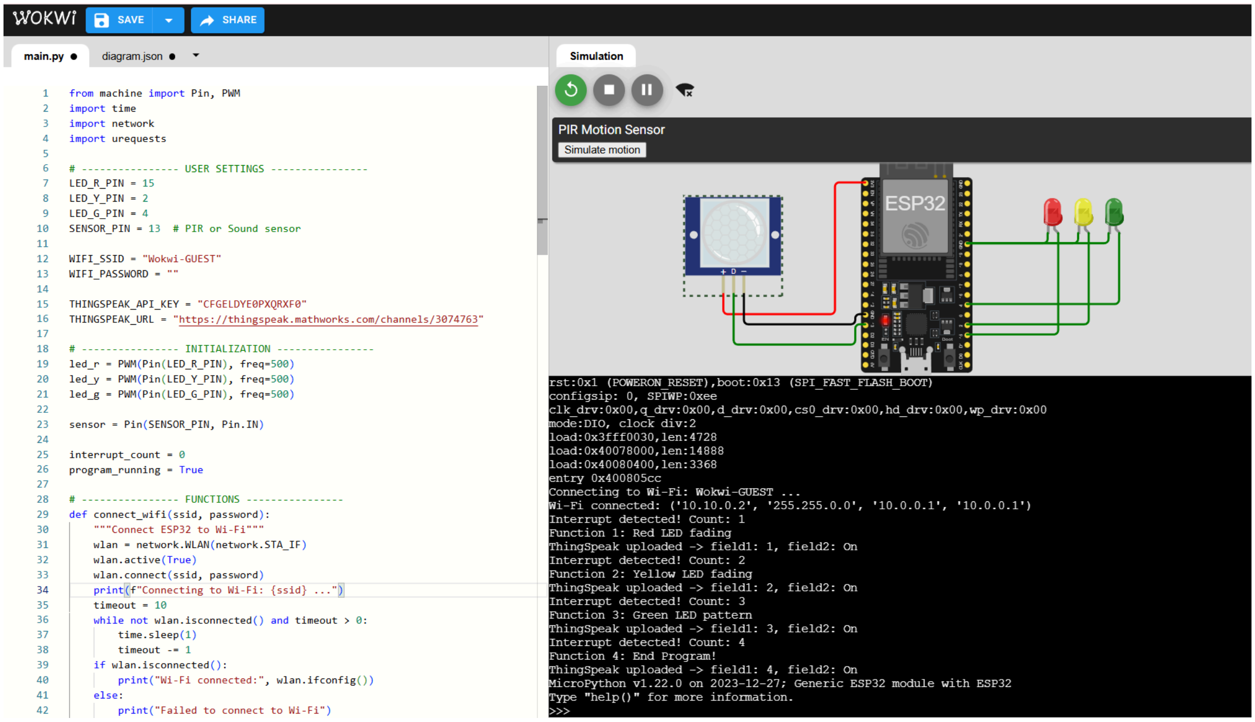
```python
# ---------------- MAIN LOOP ----------------
try:
    while program_running:
        if interrupt_count == 1:
            print("Function 1: Red LED fading")
            log_thingspeak(1, "On")
            while interrupt_count == 1:
                fade_led(led_r)

        elif interrupt_count == 2:
            print("Function 2: Yellow LED fading")
            log_thingspeak(2, "On")
            while interrupt_count == 2:
                fade_led(led_y)

        elif interrupt_count == 3:
            print("Function 3: Green LED pattern")
            log_thingspeak(3, "On")
            for _ in range(3):
                led_g.duty(1023)
                time.sleep(3)
                led_g.duty(0)
                time.sleep(1.5)

        elif interrupt_count == 4:
            print("Function 4: End Program!")
            log_thingspeak(4, "On")
            led_r.duty(0)
            led_y.duty(0)
            led_g.duty(0)
            program_running = False

        time.sleep(0.1)

except KeyboardInterrupt:
    led_r.duty(0)
    led_y.duty(0)
    led_g.duty(0)
    print("Program stopped manually.
```

Code II [WOKWI]

```
from machine import Pin, PWM
import time
import network
import urequests

# ---------------- USER SETTINGS ----------------
LED_R_PIN = 15
LED_Y_PIN = 2
LED_G_PIN = 4
SENSOR_PIN = 13  # PIR or Sound sensor

WIFI_SSID = "Wokwi-GUEST"
WIFI_PASSWORD = ""

THINGSPEAK_API_KEY = "CFGELDYE0PXQRXF0"
THINGSPEAK_URL =
"https://thingspeak.mathworks.com/channels/3074763"

# ---------------- INITIALIZATION ----------------
led_r = PWM(Pin(LED_R_PIN), freq=500)
led_y = PWM(Pin(LED_Y_PIN), freq=500)
led_g = PWM(Pin(LED_G_PIN), freq=500)

sensor = Pin(SENSOR_PIN, Pin.IN)

interrupt_count = 0
program_running = True

# ---------------- FUNCTIONS ----------------
def connect_wifi(ssid, password):
    """Connect ESP32 to Wi-Fi"""
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    print(f"Connecting to Wi-Fi: {ssid} ...")
    timeout = 10
    while not wlan.isconnected() and timeout > 0:
        time.sleep(1)
        timeout -= 1
    if wlan.isconnected():
```

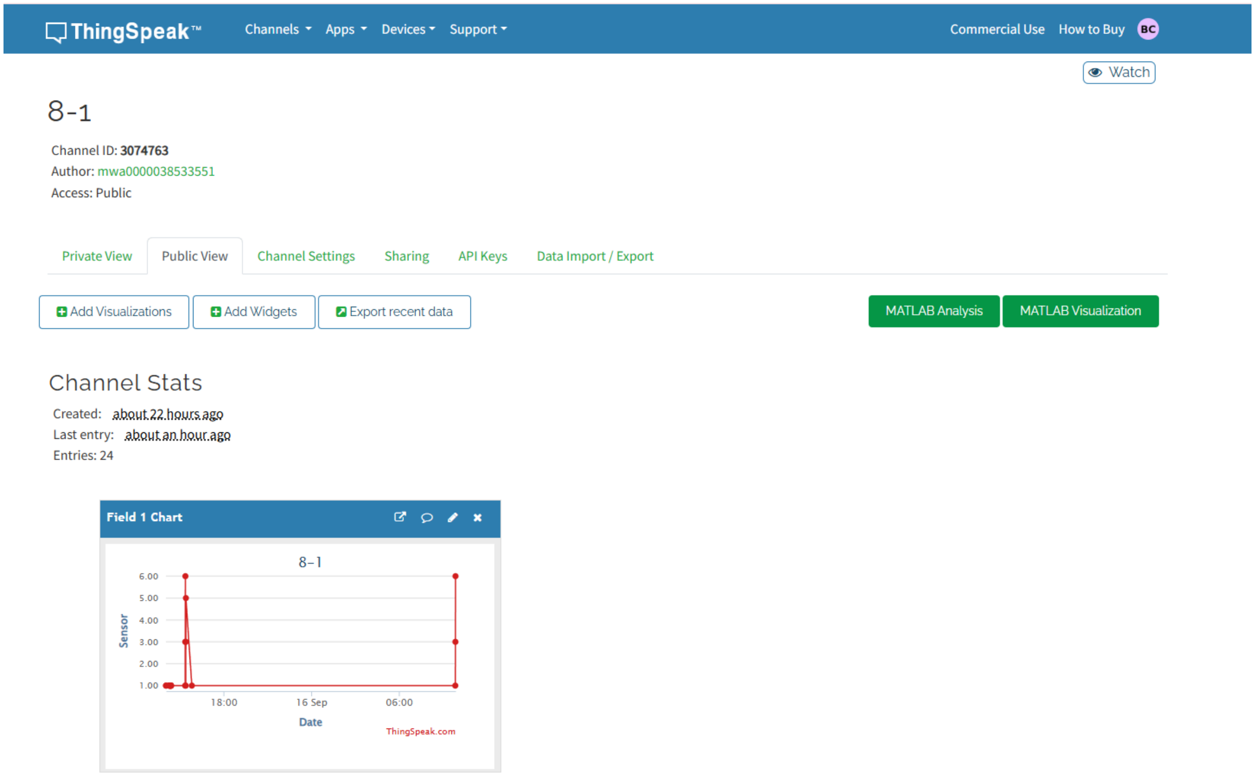
```python
        print("Wi-Fi connected:", wlan.ifconfig())
    else:
        print("Failed to connect to Wi-Fi")

def log_thingspeak(field1_val, field2_val=""):
    """Send data to ThingSpeak"""
    try:
        url =
f"{THINGSPEAK_URL}?api_key={THINGSPEAK_API_KEY}&field1={field1_val}
"
        if field2_val:
            url += f"&field2={field2_val}"
        response = urequests.get(url)
        response.close()
        print(f"ThingSpeak uploaded -> field1: {field1_val},
field2: {field2_val}")
    except Exception as e:
        print("ThingSpeak error:", e)

def fade_led(led, duration=0.01):
    """Fade LED in and out once"""
    for i in range(0, 1024, 20):
        led.duty(i)
        time.sleep(duration)
    for i in range(1023, -1, -20):
        led.duty(i)
        time.sleep(duration)

def interrupt_handler(pin):
    global interrupt_count
    interrupt_count += 1
    print("Interrupt detected! Count:", interrupt_count)

# ---------------- SETUP ----------------
sensor.irq(trigger=Pin.IRQ_RISING, handler=interrupt_handler)
connect_wifi(WIFI_SSID, WIFI_PASSWORD)

# ---------------- MAIN LOOP ----------------
try:
    while program_running:
        if interrupt_count == 1:
            print("Function 1: Red LED fading")
            log_thingspeak(1, "On")
            while interrupt_count == 1:
                fade_led(led_r)

        elif interrupt_count == 2:
            print("Function 2: Yellow LED fading")
            log_thingspeak(2, "On")
            while interrupt_count == 2:
                fade_led(led_y)

        elif interrupt_count == 3:
            print("Function 3: Green LED pattern")
            log_thingspeak(3, "On")
            for _ in range(3):
                led_g.duty(1023)
                time.sleep(3)
                led_g.duty(0)
                time.sleep(1.5)
            # no reset here → let it go to 4

        elif interrupt_count == 4:
            print("Function 4: End Program!")
            log_thingspeak(4, "On")
            led_r.duty(0)
            led_y.duty(0)
            led_g.duty(0)
            sensor.irq(handler=None)   # disable interrupt
            program_running = False    # stop loop

        time.sleep(0.1)

except KeyboardInterrupt:
    led_r.duty(0)
    led_y.duty(0)
    led_g.duty(0)
    print("Program stopped manually.")
```

Test Records :



```python
from machine import Pin, PWM
import time
import network
import urequests

# --------------- USER SETTINGS ---------------
LED_R_PIN = 15
LED_Y_PIN = 2
LED_G_PIN = 4
SENSOR_PIN = 13  # PIR or Sound sensor

WIFI_SSID = "Wokwi-GUEST"
WIFI_PASSWORD = ""

THINGSPEAK_API_KEY = "CFGELDYE0PXQRXF0"
THINGSPEAK_URL = "https://thingspeak.mathworks.com/channels/3074763"

# --------------- INITIALIZATION ---------------
led_r = PWM(Pin(LED_R_PIN), freq=500)
led_y = PWM(Pin(LED_Y_PIN), freq=500)
led_g = PWM(Pin(LED_G_PIN), freq=500)

sensor = Pin(SENSOR_PIN, Pin.IN)

interrupt_count = 0
program_running = True

# --------------- FUNCTIONS ---------------
def connect_wifi(ssid, password):
    """Connect ESP32 to Wi-Fi"""
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    print(f"Connecting to Wi-Fi: {ssid} ...")
    timeout = 10
    while not wlan.isconnected() and timeout > 0:
        time.sleep(1)
        timeout -= 1
    if wlan.isconnected():
        print("Wi-Fi connected:", wlan.ifconfig())
    else:
        print("Failed to connect to Wi-Fi")
```

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
Connecting to Wi-Fi: Wokwi-GUEST ...
Wi-Fi connected: ('10.10.0.2', '255.255.0.0', '10.0.0.1', '10.0.0.1')
Interrupt detected! Count: 1
Function 1: Red LED fading
ThingSpeak uploaded -> field1: 1, field2: On
Interrupt detected! Count: 2
Function 2: Yellow LED fading
ThingSpeak uploaded -> field1: 2, field2: On
Interrupt detected! Count: 3
Function 3: Green LED pattern
ThingSpeak uploaded -> field1: 3, field2: On
Interrupt detected! Count: 4
Function 4: End Program!
ThingSpeak uploaded -> field1: 4, field2: On
MicroPython v1.22.0 on 2023-12-27; Generic ESP32 module with ESP32
Type "help()" for more information.
>>>
```

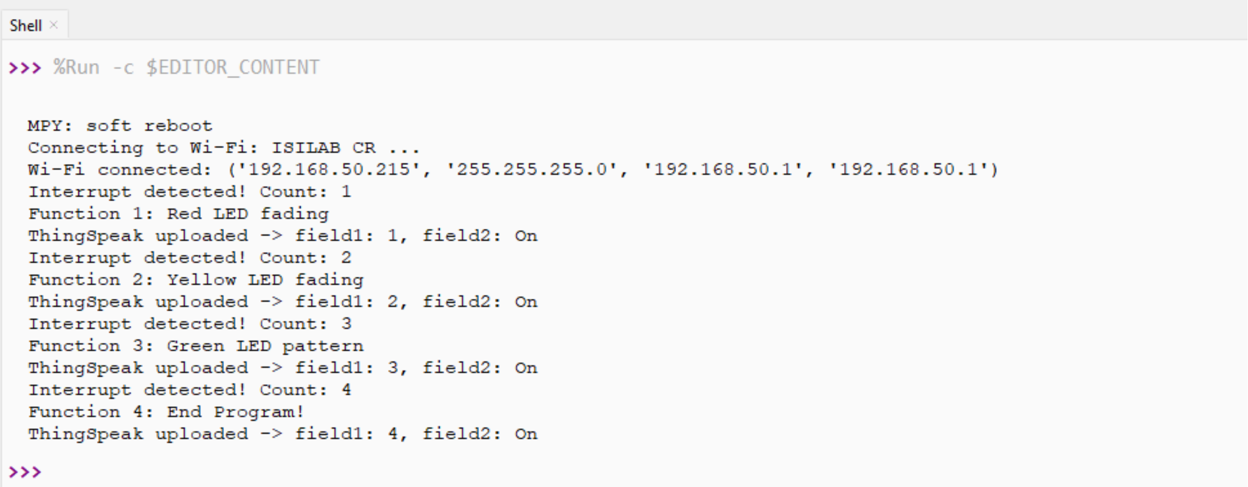


8-1

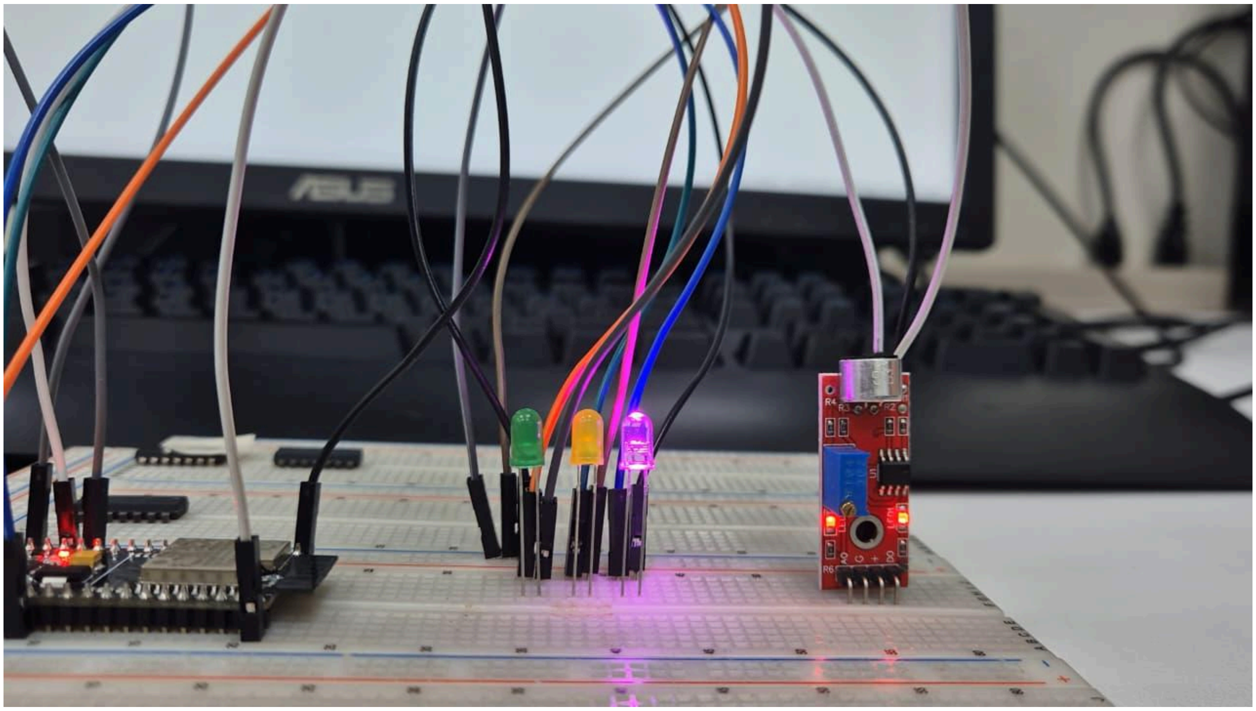
Channel ID: **3074763**
Author: mwa0000038533551
Access: Public

Channel Stats

Created: about 22 hours ago
Last entry: about an hour ago
Entries: 24



```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
 Connecting to Wi-Fi: ISILAB CR ...
 Wi-Fi connected: ('192.168.50.215', '255.255.255.0', '192.168.50.1', '192.168.50.1')
 Interrupt detected! Count: 1
 Function 1: Red LED fading
 ThingSpeak uploaded -> field1: 1, field2: On
 Interrupt detected! Count: 2
 Function 2: Yellow LED fading
 ThingSpeak uploaded -> field1: 2, field2: On
 Interrupt detected! Count: 3
 Function 3: Green LED pattern
 ThingSpeak uploaded -> field1: 3, field2: On
 Interrupt detected! Count: 4
 Function 4: End Program!
 ThingSpeak uploaded -> field1: 4, field2: On

>>>
```



Lab3 – OnESP32, complete the following tasks:

A. Build an ultrasonic module with the Trigger pin connected to GPIO XX and the Echo pin connected to GPIO XX. Connect an RGB LED as follows:LED_R → GPIO XX,LED_Y → GPIO XX and LED_G → GPIO XX.

B. When the program starts, implement the ultrasonic measurement as a custom class, and complete the following functions:

Connect to the ThingSpeak platform and send the current distance value to the IoT platform.

- Funtion 1: If the distance is greater than 30 cm: the G LED slowly fades in and slowly fades out.
- Funtion 2: If the distance is greater than 10 cm but less than 30 cm: the B LED slowly fades in and slowly fades out.
- Funtion 3: If the distance is less than 10 cm: the R LED slowly fades in and slowly fades out.

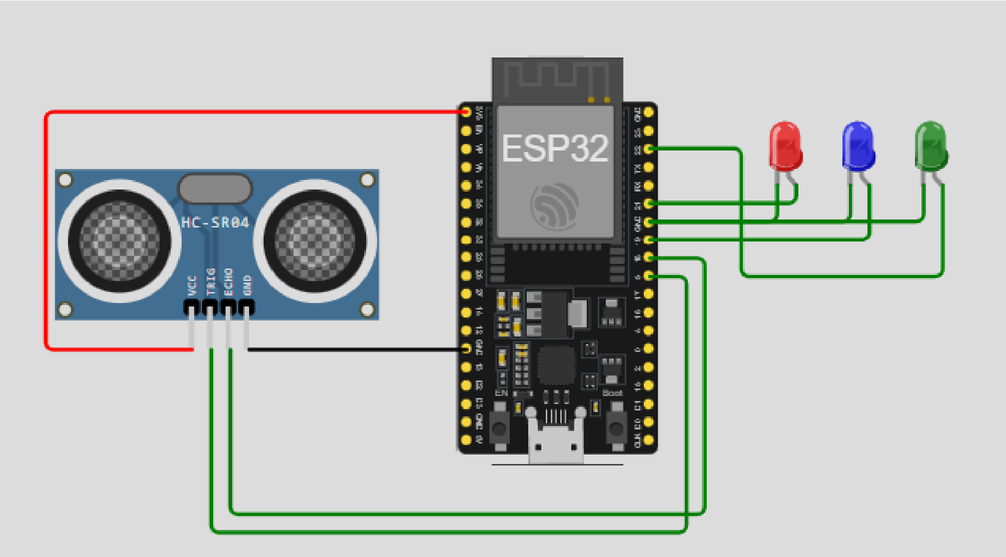
Connections :

Ultrasonic Sensor (HC-SR04)

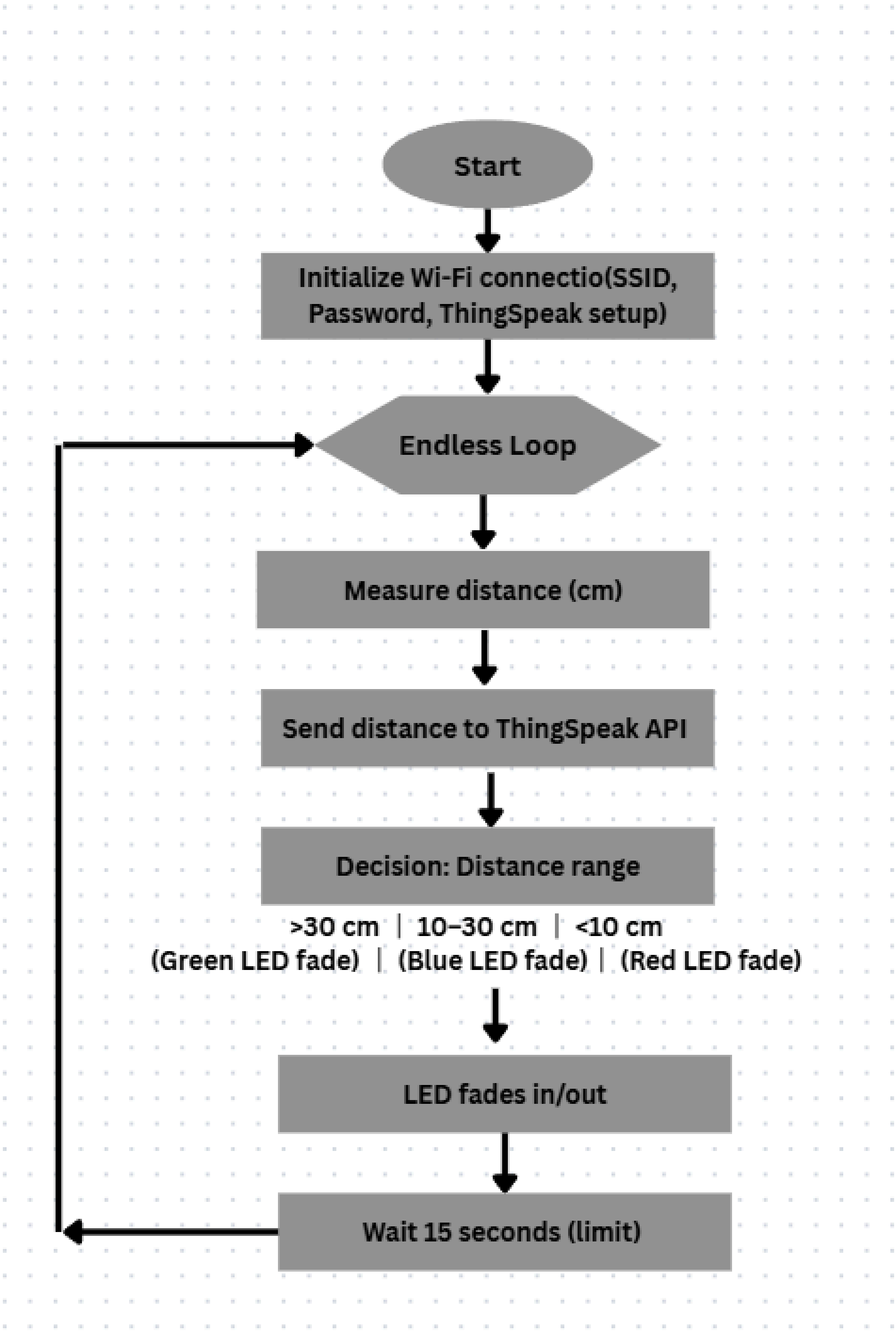
- VCC → 5V
- GND → GND
- Trigger → GPIO 5
- Echo → GPIO 18

RGB LED (Common Cathode type, with resistors ~220Ω each)

- Red → GPIO 21
- Blue → GPIO 19
- Green → GPIO 22



Flowchart :

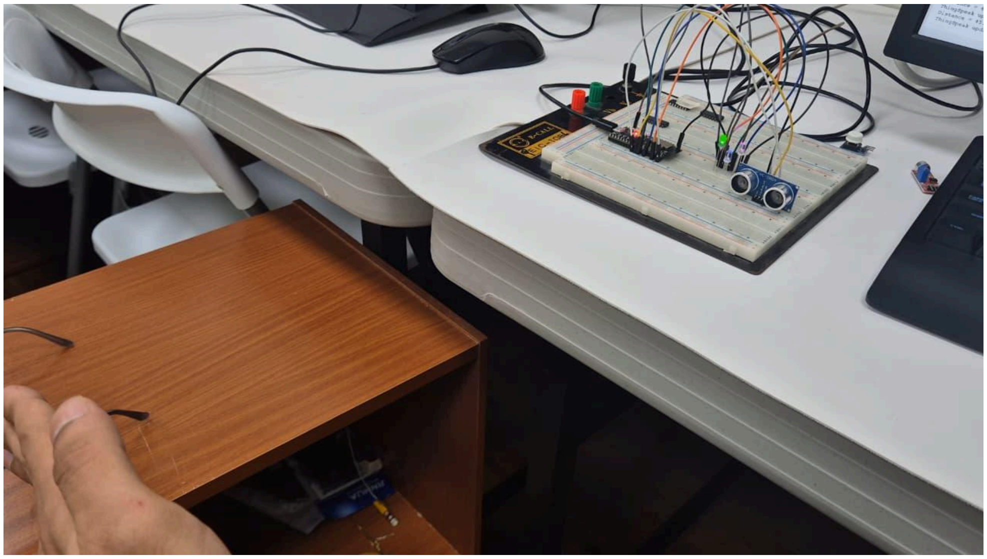

Start
Initialize Wi-Fi connectio(SSID, Password, ThingSpeak setup)
Endless Loop
Measure distance (cm)
Send distance to ThingSpeak API
Decision: Distance range
>30 cm | 10–30 cm | <10 cm
(Green LED fade) | (Blue LED fade) | (Red LED fade)
LED fades in/out
Wait 15 seconds (limit)

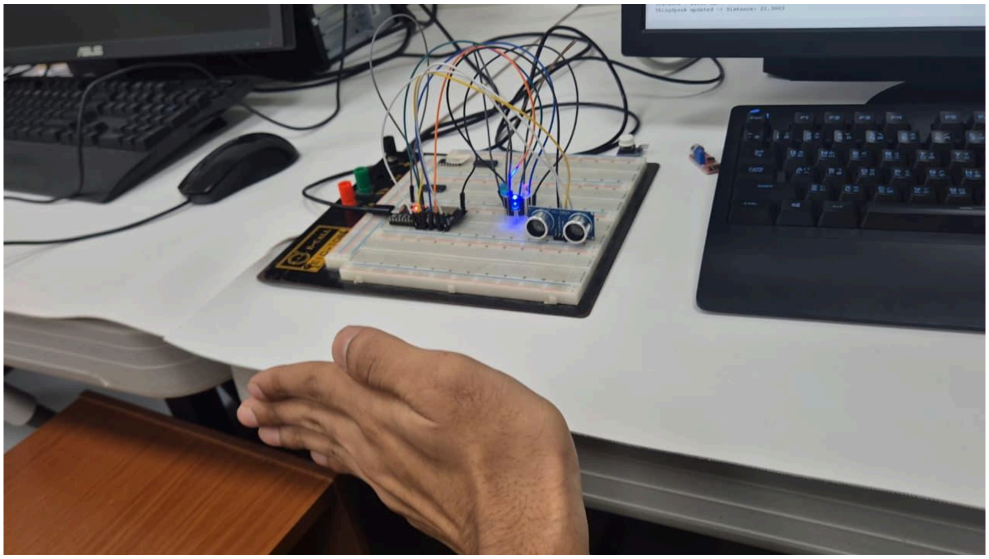
Source code :

```
from machine import Pin, PWM
import time, network, urequests

# ------------ USER SETTINGS ------------
TRIG_PIN = 5
ECHO_PIN = 18
LED_R_PIN = 21
LED_B_PIN = 19
LED_G_PIN = 22
WIFI_SSID = "ISILAB CR"
WIFI_PASS = "isilab.ncut.CR"
THINGSPEAK_API_KEY = "3258JHZI9GMKSN6L"
THINGSPEAK_URL = "http://api.thingspeak.com/update"  # must use /update
# ------------ ULTRASONIC CLASS ------------
class Ultrasonic:
    def __init__(self, trig, echo):
        self.trig = Pin(trig, Pin.OUT)
        self.echo = Pin(echo, Pin.IN)
    def distance_cm(self):
        self.trig.off()
        time.sleep_us(2)
        self.trig.on()
        time.sleep_us(10)
        self.trig.off()
        while self.echo.value() == 0:
            pass
        start = time.ticks_us()
        while self.echo.value() == 1:
            pass
        end = time.ticks_us()
        duration = time.ticks_diff(end, start)
        dist = (duration * 0.0343) / 2
        return dist
# ------------ CONNECT TO WIFI ------------
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(WIFI_SSID, WIFI_PASS)
    print("Connecting to Wi-Fi", end="")
    while not wlan.isconnected():
        print(".", end="")
        time.sleep(1)
    print("\nConnected:", wlan.ifconfig())


# ------------ SEND TO THINGSPEAK ------------
def send_thingspeak(distance):
    try:
        url = "{}?api_key={}&field1={:.2f}".format(THINGSPEAK_URL, THINGSPEAK_API_KEY, distance)
        res = urequests.get(url)
        res.close()
        print("ThingSpeak updated -> Distance:", distance)
    except Exception as e:
        print("Error sending to ThingSpeak:", e)
```

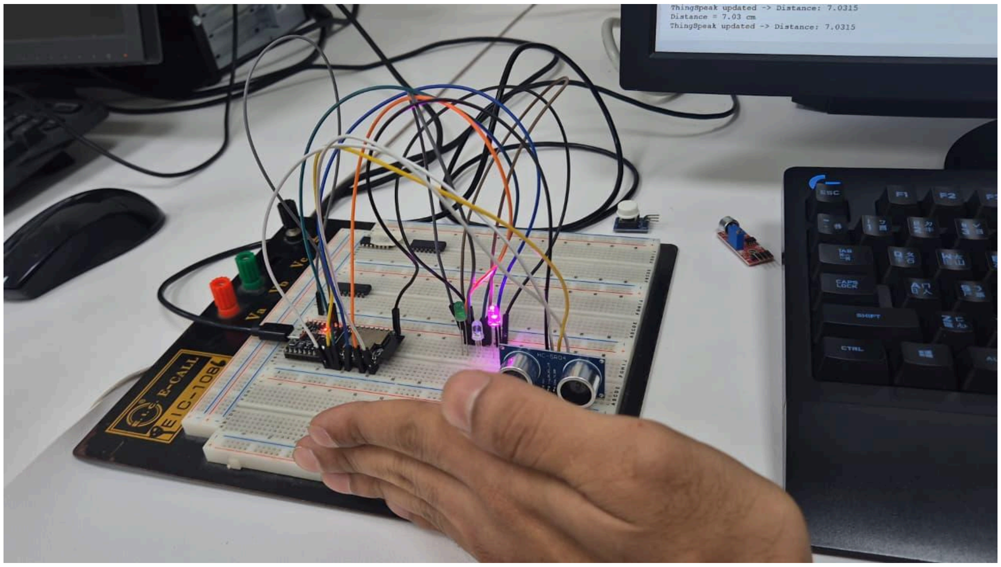
```
# ------------ LED FADE FUNCTION ------------
def fade_led(pwm):
    for duty in range(0, 1024, 20):
        pwm.duty(duty)
        time.sleep(0.01)
    for duty in range(1023, -1, -20):
        pwm.duty(duty)
        time.sleep(0.01)
# ------------ MAIN PROGRAM ------------
connect_wifi()
ultra = Ultrasonic(TRIG_PIN, ECHO_PIN)

led_r = PWM(Pin(LED_R_PIN), freq=1000, duty=0)
led_b = PWM(Pin(LED_B_PIN), freq=1000, duty=0)
led_g = PWM(Pin(LED_G_PIN), freq=1000, duty=0)
while True:
    dist = ultra.distance_cm()
    print("Distance = {:.2f} cm".format(dist))
    send_thingspeak(dist)
    if dist > 30:
        fade_led(led_g)   # Function 1
    elif 10 < dist <= 30:
        fade_led(led_b)   # Function 2
    else:
        fade_led(led_r)   # Function 3
```

Test Records :

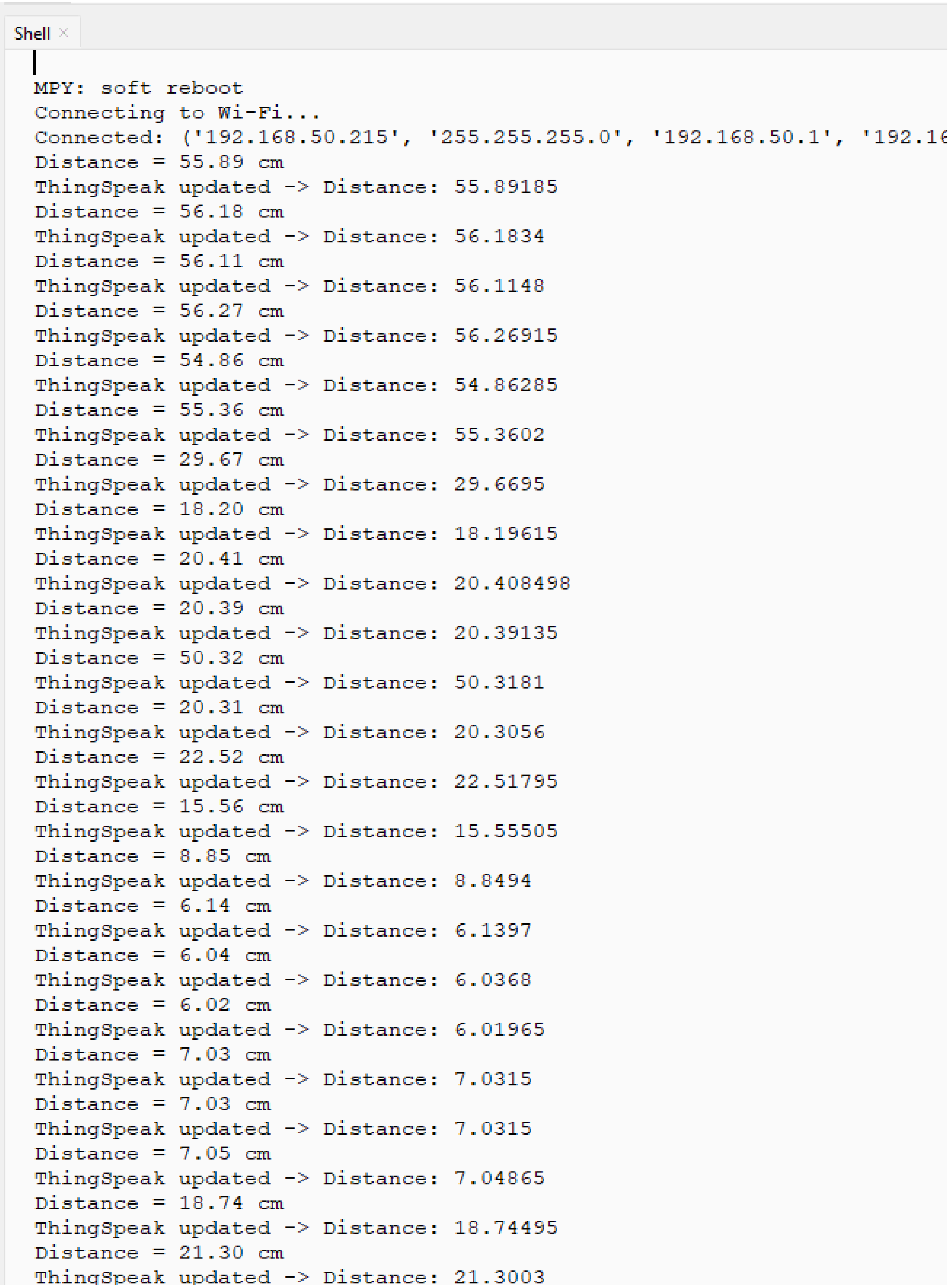




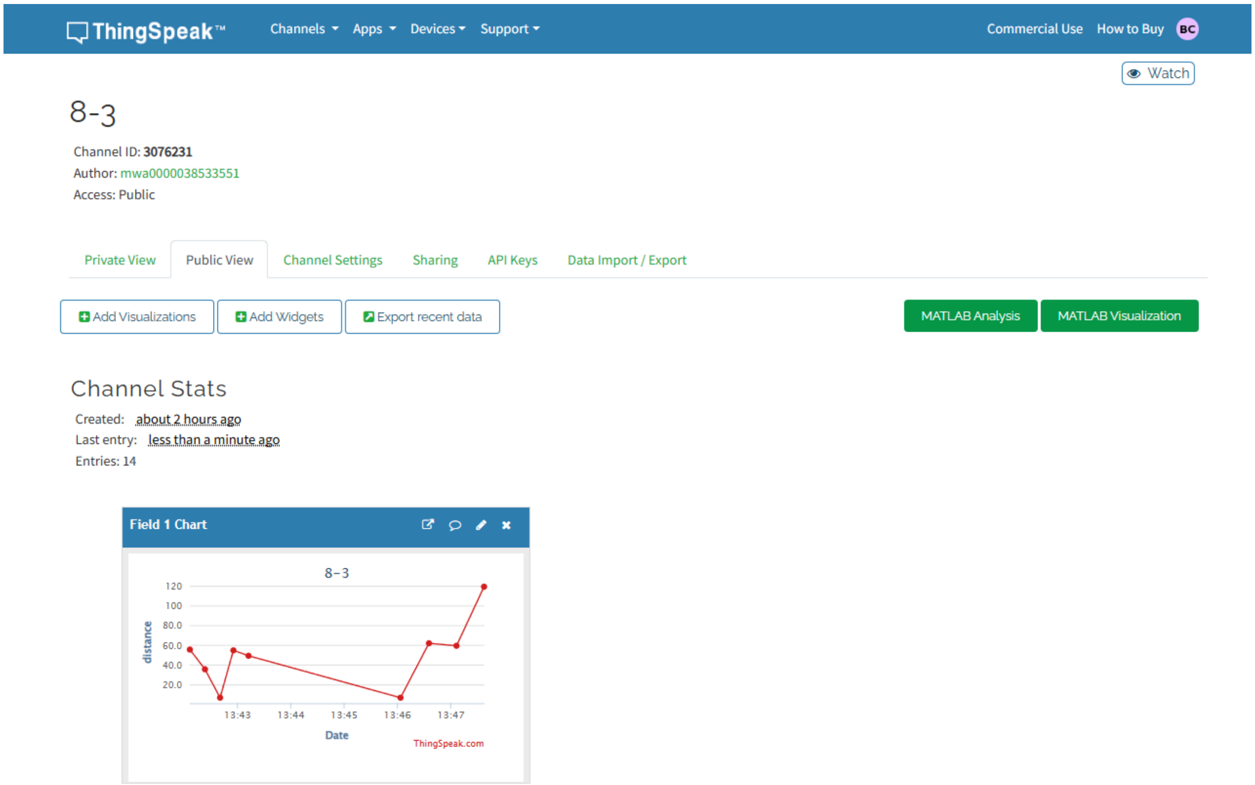
```
ThingSpeak updated -> Distance: 21.3003
```



```
ThingSpeak updated -> Distance: 7.0315
Distance = 7.03 cm
ThingSpeak updated -> Distance: 7.0315
```



```
MPY: soft reboot
Connecting to Wi-Fi...
Connected: ('192.168.50.215', '255.255.255.0', '192.168.50.1', '192.16
Distance = 55.89 cm
ThingSpeak updated -> Distance: 55.89185
Distance = 56.18 cm
ThingSpeak updated -> Distance: 56.1834
Distance = 56.11 cm
ThingSpeak updated -> Distance: 56.1148
Distance = 56.27 cm
ThingSpeak updated -> Distance: 56.26915
Distance = 54.86 cm
ThingSpeak updated -> Distance: 54.86285
Distance = 55.36 cm
ThingSpeak updated -> Distance: 55.3602
Distance = 29.67 cm
ThingSpeak updated -> Distance: 29.6695
Distance = 18.20 cm
ThingSpeak updated -> Distance: 18.19615
Distance = 20.41 cm
ThingSpeak updated -> Distance: 20.408498
Distance = 20.39 cm
ThingSpeak updated -> Distance: 20.39135
Distance = 50.32 cm
ThingSpeak updated -> Distance: 50.3181
Distance = 20.31 cm
ThingSpeak updated -> Distance: 20.3056
Distance = 22.52 cm
ThingSpeak updated -> Distance: 22.51795
Distance = 15.56 cm
ThingSpeak updated -> Distance: 15.55505
Distance = 8.85 cm
ThingSpeak updated -> Distance: 8.8494
Distance = 6.14 cm
ThingSpeak updated -> Distance: 6.1397
Distance = 6.04 cm
ThingSpeak updated -> Distance: 6.0368
Distance = 6.02 cm
ThingSpeak updated -> Distance: 6.01965
Distance = 7.03 cm
ThingSpeak updated -> Distance: 7.0315
Distance = 7.03 cm
ThingSpeak updated -> Distance: 7.0315
Distance = 7.05 cm
ThingSpeak updated -> Distance: 7.04865
Distance = 18.74 cm
ThingSpeak updated -> Distance: 18.74495
Distance = 21.30 cm
ThingSpeak updated -> Distance: 21.3003
```



8-3

Channel ID: **3076231**
Author: mwa0000038533551
Access: Public

Channel Stats

Created: about 2 hours ago
Last entry: less than a minute ago
Entries: 14

Lab4– On Wokwi, complete the following tasks:

A. Create two ultrasonic modules:

- The first ultrasonic sensor: Trigger Pin → GPIO XX, Echo Pin → GPIO XX.
- The second ultrasonic sensor: Trigger Pin → GPIO XX, Echo Pin → GPIO XX.

B. Connect an RGB LED as follows :

- LED_R → GPIO XX, LED_Y → GPIO XX and LED_G → GPIO XX

C. When the program starts, implement the ultrasonic measurement as a custom class, and complete the following functions:

D. On the ThingSpeak platform, create a field named Sensor to receive the status of the sound sensor (ESP32 physical) / PIR sensor (Wokwi simulation).

A. Complete the following functions:

- · Connect to the ThingSpeak platform and display the current distance values from both ultrasonic modules on ThingSpeak.
- · Function 1: If both L and R distances are greater than 30 cm: the G LED slowly fades in and slowly fades out.
- · Function 2: If either L or R distance is greater than 0 cm but less than 30 cm: the B LED slowly fades in and slowly fades out.
- · Function 3: If either L or R distance is less than 0 cm: the R LED slowly fades in and slowly fades out.

Connections:

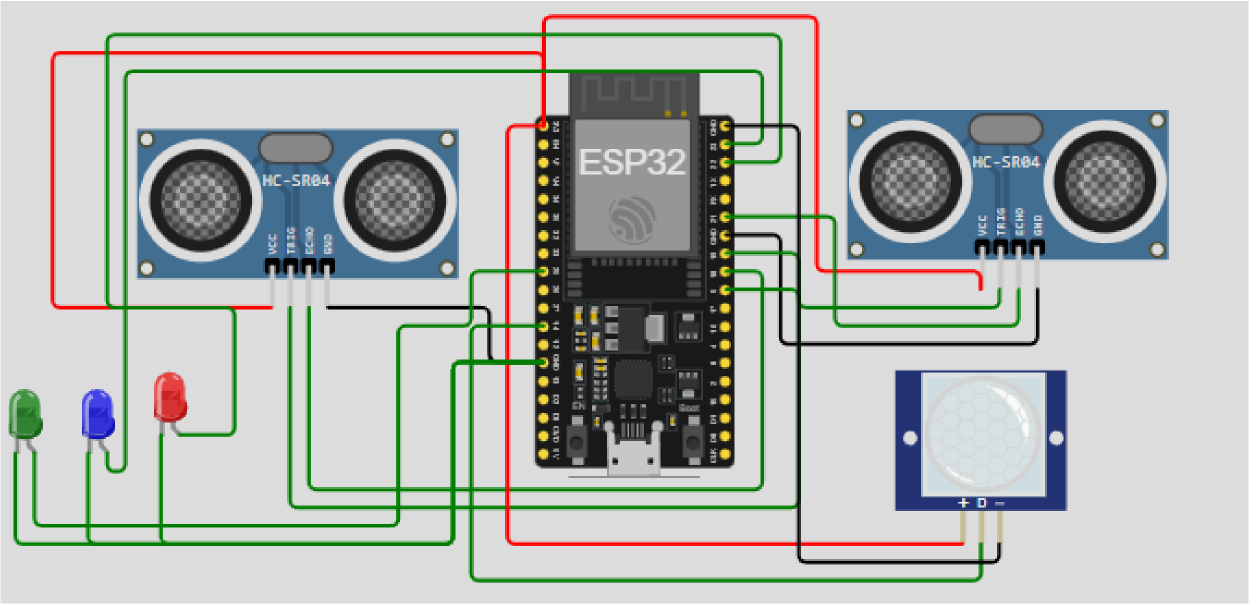
Ultrasonic Sensors

- Ultrasonic Left
  - ○ TRIG → GPIO 5
  - ○ ECHO → GPIO 18
- Ultrasonic Right
  - ○ TRIG → GPIO 19
  - ○ ECHO → GPIO 21

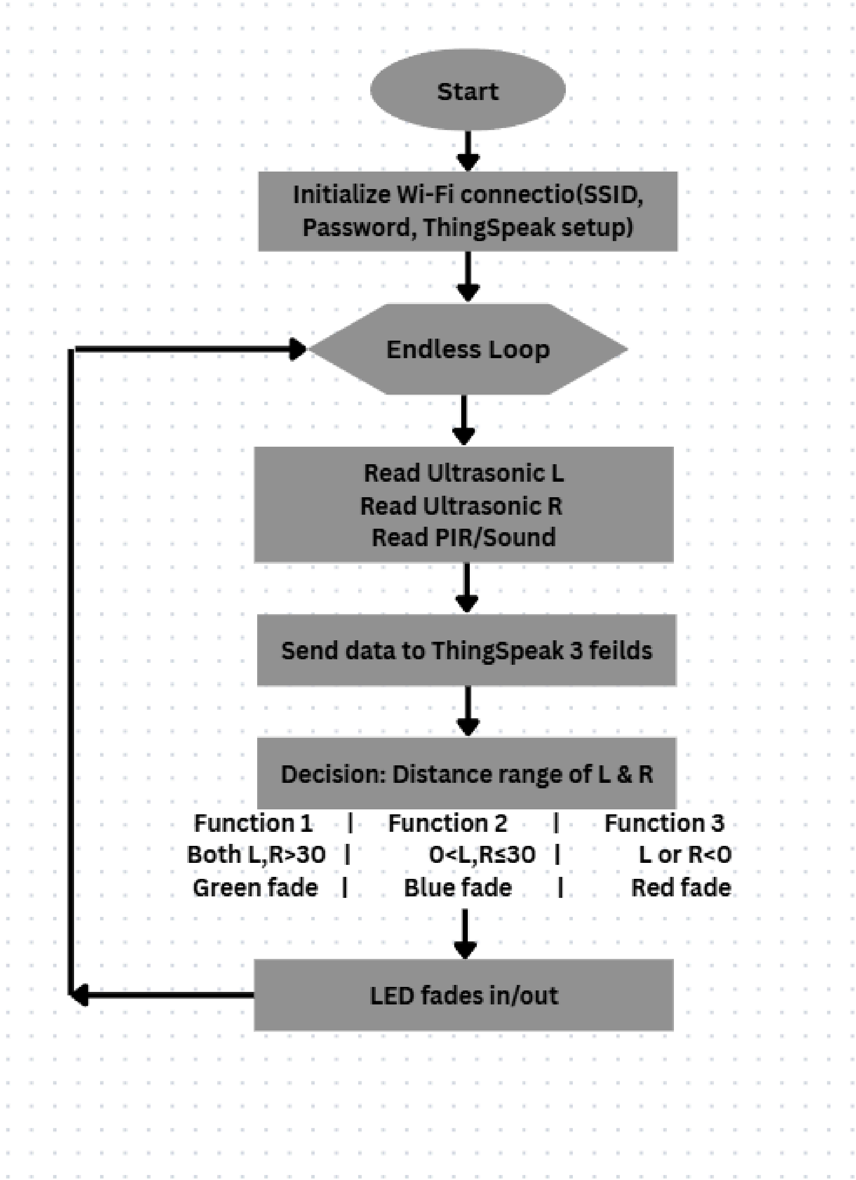
RGB LED (Common Cathode recommended)

- LED_R → GPIO 22
- LED_B → GPIO 23
- LED_G → GPIO 25

PIR / Sound Sensor- GPIO 14



Flowchart :

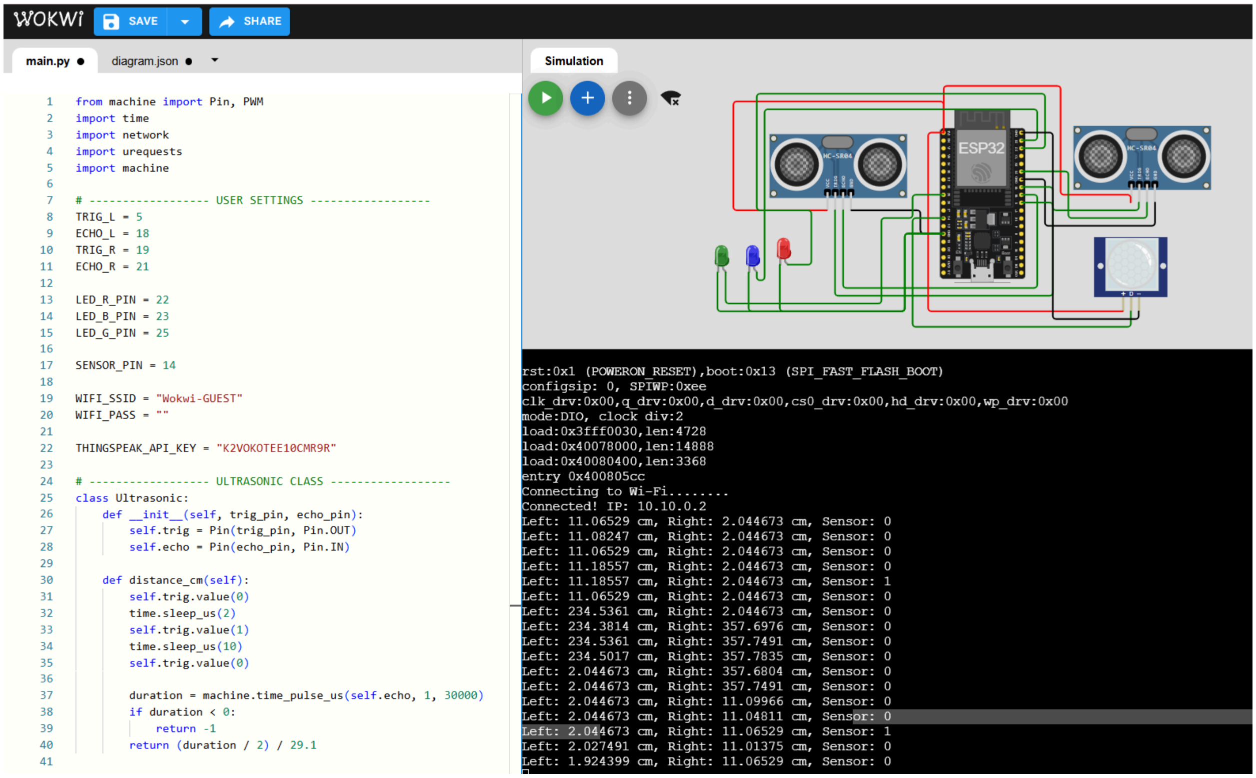

Start
Initialize Wi-Fi connectio(SSID, Password, ThingSpeak setup)
Endless Loop
Read Ultrasonic L
Read Ultrasonic R
Read PIR/Sound
Send data to ThingSpeak 3 feilds
Decision: Distance range of L & R
Function 1 | Function 2 | Function 3
Both L,R>30 | 0<L,R≤30 | L or R<0
Green fade | Blue fade | Red fade
LED fades in/out

Source code:

```
from machine import Pin, PWM
import time
import network
import urequests
import machine

# ------------------ USER SETTINGS ------------------
TRIG_L = 5
ECHO_L = 18
TRIG_R = 19
ECHO_R = 21

LED_R_PIN = 22
LED_B_PIN = 23
LED_G_PIN = 25

SENSOR_PIN = 14

WIFI_SSID = "Wokwi-GUEST"
WIFI_PASS = ""

THINGSPEAK_API_KEY = "K2VOKOTEE10CMR9R"

# ------------------ ULTRASONIC CLASS ------------------
class Ultrasonic:
    def __init__(self, trig_pin, echo_pin):
        self.trig = Pin(trig_pin, Pin.OUT)
        self.echo = Pin(echo_pin, Pin.IN)

    def distance_cm(self):
        self.trig.value(0)
        time.sleep_us(2)
        self.trig.value(1)
        time.sleep_us(10)
        self.trig.value(0)
        duration = machine.time_pulse_us(self.echo, 1, 30000)
        if duration < 0:
            return -1
        return (duration / 2) / 29.1
```

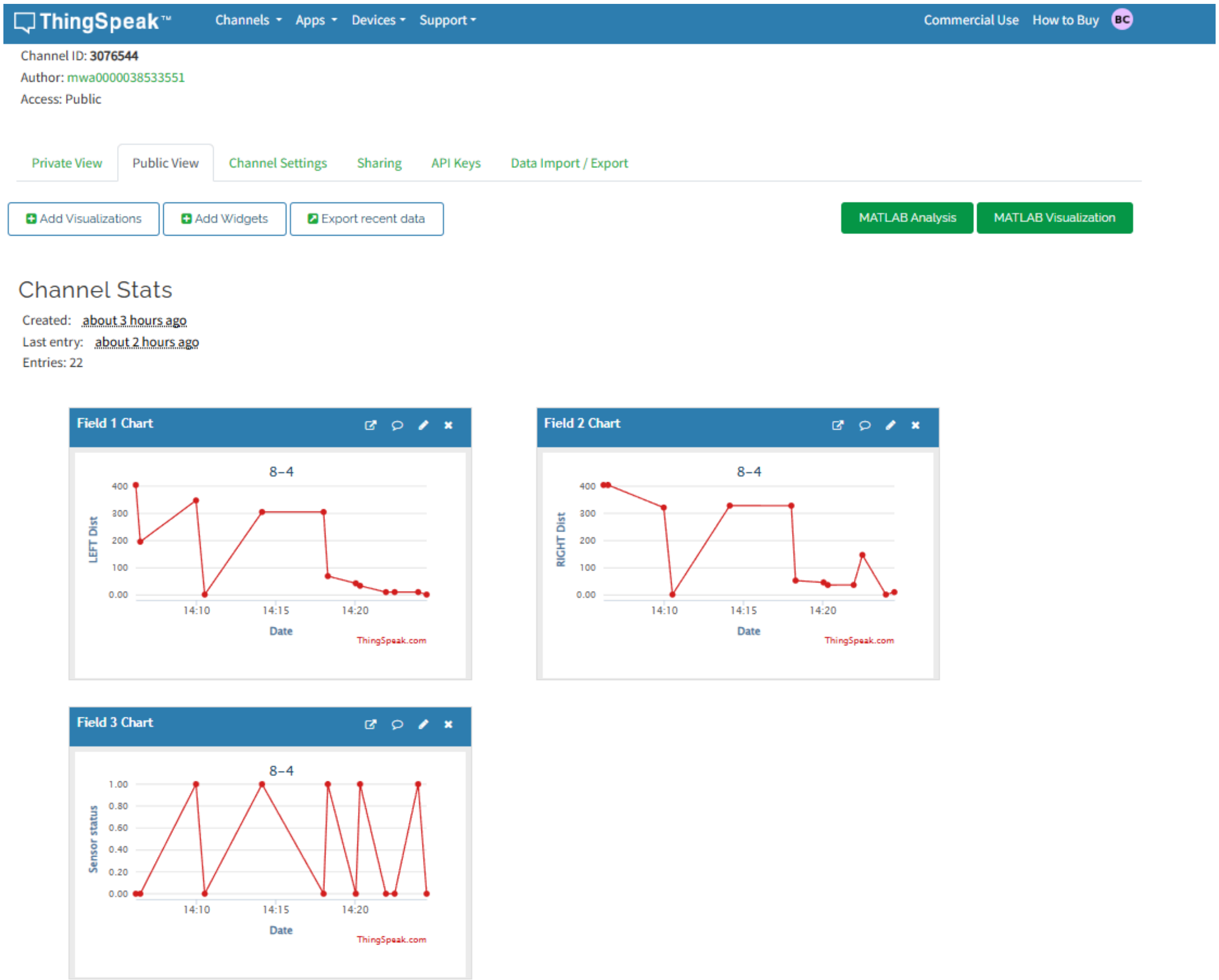
```
# ------------------ WIFI & ThingSpeak ------------------
def connect_wifi(ssid, password):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    print("Connecting to Wi-Fi", end="")
    while not wlan.isconnected():
        print(".", end="")
        time.sleep(0.5)
    print("\nConnected! IP:", wlan.ifconfig()[0])

def update_thingspeak(dist_L, dist_R, sensor_status):
    url =
f"https://api.thingspeak.com/update?api_key={THINGSPEAK_API_KEY}&fi
eld1={dist_L}&field2={dist_R}&field3={sensor_status}"
    try:
        response = urequests.get(url)
        response.close()
    except:
        print("ThingSpeak update failed")

# ------------------ INITIALIZATION ------------------
sensor_L = Ultrasonic(TRIG_L, ECHO_L)
sensor_R = Ultrasonic(TRIG_R, ECHO_R)
sensor = Pin(SENSOR_PIN, Pin.IN)

led_r = PWM(Pin(LED_R_PIN), freq=1000)
led_b = PWM(Pin(LED_B_PIN), freq=1000)
led_g = PWM(Pin(LED_G_PIN), freq=1000)

# Current PWM duty for non-blocking fade
led_duty = {"R": 0, "B": 0, "G": 0}
led_dir = {"R": 1, "B": 1, "G": 1}  # 1=up, -1=down
connect_wifi(WIFI_SSID, WIFI_PASS)

# ------------------ MAIN LOOP ------------------
while True:
    # Read sensors
    dist_L = sensor_L.distance_cm()
    dist_R = sensor_R.distance_cm()
    sensor_status = sensor.value()

    print(f"Left: {dist_L} cm, Right: {dist_R} cm, Sensor: {sensor_status}")
    # Update ThingSpeak
    update_thingspeak(dist_L, dist_R, sensor_status)

    # Determine active LED
    active_led = None
    if dist_L > 30 and dist_R > 30:
        active_led = "G"
    elif (0 < dist_L <= 30) or (0 < dist_R <= 30):
        active_led = "B"
    elif dist_L < 0 or dist_R < 0:
        active_led = "R"

    # Non-blocking LED fade
    for color in ["R", "B", "G"]:
        if color == active_led:
            # Update duty
            led_duty[color] += led_dir[color] * 20
            if led_duty[color] >= 1023:
                led_duty[color] = 1023
                led_dir[color] = -1
            elif led_duty[color] <= 0:
                led_duty[color] = 0
                led_dir[color] = 1
        else:
            # Turn off inactive LEDs
            led_duty[color] = 0
        # Apply PWM duty
        if color == "R":
            led_r.duty(led_duty[color])
        elif color == "B":
            led_b.duty(led_duty[color])
        elif color == "G":
            led_g.duty(led_duty[color])

    time.sleep(0.05)  # Small delay for smooth fading
```

Test Records :



```python
from machine import Pin, PWM
import time
import network
import urequests
import machine

# ------------------ USER SETTINGS ------------------
TRIG_L = 5
ECHO_L = 18
TRIG_R = 19
ECHO_R = 21

LED_R_PIN = 22
LED_B_PIN = 23
LED_G_PIN = 25

SENSOR_PIN = 14

WIFI_SSID = "Wokwi-GUEST"
WIFI_PASS = ""

THINGSPEAK_API_KEY = "K2VOKOTEE10CMR9R"

# ------------------ ULTRASONIC CLASS ------------------
class Ultrasonic:
    def __init__(self, trig_pin, echo_pin):
        self.trig = Pin(trig_pin, Pin.OUT)
        self.echo = Pin(echo_pin, Pin.IN)

    def distance_cm(self):
        self.trig.value(0)
        time.sleep_us(2)
        self.trig.value(1)
        time.sleep_us(10)
        self.trig.value(0)

        duration = machine.time_pulse_us(self.echo, 1, 30000)
        if duration < 0:
            return -1
        return (duration / 2) / 29.1

```

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
Connecting to Wi-Fi........
Connected! IP: 10.10.0.2
Left: 11.06529 cm, Right: 2.044673 cm, Sensor: 0
Left: 11.08247 cm, Right: 2.044673 cm, Sensor: 0
Left: 11.06529 cm, Right: 2.044673 cm, Sensor: 0
Left: 11.18557 cm, Right: 2.044673 cm, Sensor: 0
Left: 11.18557 cm, Right: 2.044673 cm, Sensor: 1
Left: 11.06529 cm, Right: 2.044673 cm, Sensor: 0
Left: 234.5361 cm, Right: 2.044673 cm, Sensor: 0
Left: 234.3814 cm, Right: 357.6976 cm, Sensor: 0
Left: 234.5361 cm, Right: 357.7491 cm, Sensor: 0
Left: 234.5017 cm, Right: 357.7835 cm, Sensor: 0
Left: 2.044673 cm, Right: 357.6804 cm, Sensor: 0
Left: 2.044673 cm, Right: 357.7491 cm, Sensor: 0
Left: 2.044673 cm, Right: 11.09966 cm, Sensor: 0
Left: 2.044673 cm, Right: 11.04811 cm, Sensor: 0
Left: 2.044673 cm, Right: 11.06529 cm, Sensor: 1
Left: 2.027491 cm, Right: 11.01375 cm, Sensor: 0
Left: 1.924399 cm, Right: 11.06529 cm, Sensor: 0
```



Channel ID: 3076544
Author: mwa0000038533551
Access: Public

Channel Stats

Created: about 3 hours ago
Last entry: about 2 hours ago
Entries: 22

Note : Lab 5 and Lab 6 will be completed soon and submitted later in a separate file, building on the skills developed so far.

Lab7 learning reflection or comments(100 words)

Lab 1, I learned how to configure an external interrupt using a push button to control program modes. I also applied PWM to fade RGB LEDs, which helped me understand timing, duty cycles, and event-driven programming.

In **Lab 2**, I worked with digital sensors such as sound (physical) and PIR (simulation). By integrating the ESP32 with **ThingSpeak**, I practiced uploading real-time data and visualizing it on a cloud dashboard. This gave me a clearer picture of how embedded systems interact with IoT platforms.

In **Lab 3**, I used an ultrasonic distance sensor and developed a custom class, which improved my object-oriented programming skills on microcontrollers. Mapping distance values to LED behaviors highlighted the importance of processing raw sensor data into useful outputs.

Lab 4 required handling two ultrasonic sensors at the same time. I learned to manage conditional logic for multiple inputs while sending continuous data to ThingSpeak. This reinforced the challenges of real-time IoT data logging and multi-sensor integration.

Lab8 Video Link

- **Link: https://youtu.be/2dN5aHkSYT8**

Thank you