```
#define BLYNK_TEMPLATE_ID
"TMPL6QNjrv5_n"
#define BLYNK_TEMPLATE_NAME "Restaurant"
#define BLYNK_AUTH_TOKEN
"4oanph7SRHhWeuN634bwlzUVQCvTdYoZ"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <FastLED.h>
#include <HTTPClient.h>
#include <DHT.h>

// Joystick pin definitions
#define joyX 34
```

```
// FastLED definitions
#define LED_PIN     27    // Use a GPIO pin that supports output
#define NUM_LEDS    12    // Number of LEDs in your strip
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB

// Thresholds for detecting left/right movement
#define LEFT_THRESHOLD 1500
#define RIGHT_THRESHOLD 2500

// Proximity sensor pins
#define TRIG_PIN 12
#define ECHO_PIN 13

// Speaker pin
#define SPEAKER_PIN 19

// Order Ready LED
#define ORDER_LED 17

// DHT22 sensor
#define DHTPIN 21
#define DHTTYPE DHT22
```

```
DHT dht(DHTPIN, DHTTYPE);

// Define the array of LEDs
CRGB leds[NUM_LEDS];

// WiFi credentials
char ssid[] = "RUNI-Wireless";
char pass[] = "";

BlynkTimer timer;
bool canSendNewRequest = true;
bool orderBlinkState = false;

void playOrderReadySound() {
  tone(SPEAKER_PIN, 1000, 300); // First note
  delay(350);
  tone(SPEAKER_PIN, 1200, 300); // Second
note
  delay(350);
  tone(SPEAKER_PIN, 1400, 300); // Third note
  delay(350);
  noTone(SPEAKER_PIN);
}

void toggleOrderLED() {
```

```
  orderBlinkState = !orderBlinkState;
  digitalWrite(ORDER_LED, orderBlinkState ?
HIGH : LOW);
}

BLYNK_WRITE(V0) {
  int orderReadyState = param.asInt();

  if (orderReadyState == 1) {
    Serial.println("Order ready - Blinking LED");
    playOrderReadySound();
    timer.setInterval(500L, toggleOrderLED);
  } else {
    Serial.println("Order not ready - LED Off");
    digitalWrite(ORDER_LED, LOW);
  }
}

BLYNK_WRITE(V1) {
  int rating = param.asInt();
  for (int i = 0; i < NUM_LEDS; i++) {
    if (i < rating * 2) {
      if (rating < 2) {
        leds[i] = CRGB::Red;
      } else if (rating < 4) {
```

```
        leds[i] = CRGB::Yellow;
      } else {
        leds[i] = CRGB::Green;
      }
    } else {
      leds[i] = CRGB::Black;
    }
  }
  FastLED.show();
}

void checkJoystick() {
  int xVal = analogRead(joyX);
  Serial.print("xVal: ");
  Serial.println(xVal);

  bool requestMade = false;

  if (xVal < LEFT_THRESHOLD) {
    if (canSendNewRequest) {
      Serial.println("Cleaning");
      Blynk.logEvent("cleaning");
      Blynk.virtualWrite(V2, "Cleaning Required");
      requestMade = true;
      canSendNewRequest = false;
```

```
      delay(1000);
      canSendNewRequest = true;
    }
  }
  else if (xVal > RIGHT_THRESHOLD) {
    if (canSendNewRequest) {
      Serial.println("Waiter Needed");
      Blynk.logEvent("waiter_required");
      Blynk.virtualWrite(V2, "Manager is needed");
      requestMade = true;
      canSendNewRequest = false;
      delay(1000);
      canSendNewRequest = true;
    }
  }

  if (requestMade) {
    Blynk.virtualWrite(V5, 1);
  }
}

void checkProximity() {
  float distance;
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
```

```
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  distance = pulseIn(ECHO_PIN, HIGH) * 0.034 /
2;

  Serial.print("Distance: ");
  Blynk.virtualWrite(V7, distance);
  Serial.print(distance);
  Serial.println(" cm");

  if (distance < 50) {
    Serial.println("Customer Entered");
    Blynk.logEvent("customer_entered");
    HTTPClient http;
    http.begin("https://hook.eu2.make.com/
ky83pxm1hchpccmdjlb7cbfcvltf9vu4");
    http.GET();
    http.end();
  }
}

void checkDHT() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
```

```
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT
sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C  Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  Blynk.virtualWrite(V3, temperature);
  Blynk.virtualWrite(V4, humidity);

  if (temperature > 28 || humidity > 90) {
    Blynk.logEvent("music_tempo");
    Serial.println("Critical Temperature/Humidity -
Triggering Webhook");
    HTTPClient http;
    http.begin("https://maker.ifttt.com/trigger/
play_music/with/key/bXkmZXDxUdlQq-
ZBI76KNA");
    http.GET();
```

```
    http.end();
  }
}

void resetV5() {
  Blynk.virtualWrite(V5, 0);
  Serial.println("Reset V5 to 0");
}

BLYNK_WRITE(V6) {
  int menuSelection = param.asInt();
  if (menuSelection == 1) {
    tone(SPEAKER_PIN, 800, 300); // Play first
note
  } else if (menuSelection == 2) {
    tone(SPEAKER_PIN, 1000, 300); // Play
second note
  } else if (menuSelection == 3) {
    tone(SPEAKER_PIN, 1200, 300); // Play third
note
  }
  delay(350);
  noTone(SPEAKER_PIN);
  Blynk.virtualWrite(V6, 0); // Reset V6 to default
}
```

```
void setup() {
  Serial.begin(115200);
  FastLED.addLeds<LED_TYPE, LED_PIN,
COLOR_ORDER>(leds,
NUM_LEDS).setCorrection(TypicalLEDStrip);
  FastLED.setBrightness(50);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(SPEAKER_PIN, OUTPUT);
  pinMode(ORDER_LED, OUTPUT);
  dht.begin();

  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("WiFi connected");
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid,
pass);
  Serial.println("Connected to Blynk");
```

```
  // Reset V5 every 10 seconds
  timer.setInterval(10000L, resetV5);

  timer.setInterval(5000L, checkDHT);
  timer.setInterval(1000L, checkProximity);
  timer.setInterval(500L, checkJoystick);
}

void loop() {
  Blynk.run();
  timer.run();
}
```