C++

```
#include <WiFi.h>

#include <PubSubClient.h>

#include "HX711.h"


// === Wi-Fi Credentials ===

const char* ssid = "YOUR_WIFI_SSID";        // Replace with your Wi-Fi SSID

const char* password = "YOUR_WIFI_PASSWORD"; // Replace with your Wi-Fi password


// === MQTT Broker Settings ===

const char* mqtt_server = "broker.hivemq.com";  // Public MQTT broker for testing

const int mqtt_port = 1883;

const char* mqtt_topic = "smartwaste/bin1/data"; // Topic to publish sensor data


WiFiClient espClient;

PubSubClient client(espClient);


// === Ultrasonic Sensor Pins ===

const int trigPin = 5;  // GPIO5 connected to HC-SR04 Trig

const int echoPin = 18; // GPIO18 connected to HC-SR04 Echo


// === Load Cell Pins ===

const int LOADCELL_DOUT_PIN = 4;  // GPIO4 connected to HX711 DOUT

const int LOADCELL_SCK_PIN = 15;  // GPIO15 connected to HX711 SCK


HX711 scale;  // HX711 object for load cell


// === Function to connect to Wi-Fi ===

void setup_wifi() {

  delay(10);

  Serial.println();
  Serial.print("Connecting to Wi-Fi: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  // Wait until connected
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("Wi-Fi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

// === Function to reconnect MQTT if disconnected ===
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect with client ID "ESP32Client"
    if (client.connect("ESP32Client")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
```

```
}

// === Function to read distance from ultrasonic sensor ===
long readUltrasonicDistance() {
  // Send 10us pulse to trigger pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read echo pin pulse duration
  long duration = pulseIn(echoPin, HIGH, 30000); // Timeout 30ms to avoid blocking

  // Calculate distance in cm (speed of sound = 343 m/s)
  long distance = duration * 0.034 / 2;

  // If no echo received, return -1
  if (duration == 0) {
    return -1;
  }
  return distance;
}

void setup() {
  Serial.begin(115200);

  // Initialize ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // Initialize load cell
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(2280.f);  // Calibration factor (adjust for your load cell)
  scale.tare();             // Reset scale to zero

  // Connect to Wi-Fi
  setup_wifi();

  // Setup MQTT server
  client.setServer(mqtt_server, mqtt_port);
}

void loop() {
  // Ensure MQTT connection
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Read sensors
  long distance = readUltrasonicDistance();
  float weight = scale.get_units(10);  // Average of 10 readings

  // Prepare JSON payload string
  String payload = "{";
  payload += "\"distance_cm\":";
  payload += distance;
  payload += ", \"weight_kg\":";
  payload += weight;
  payload += "}";
  // Print to Serial Monitor
  Serial.print("Publishing data: ");
  Serial.println(payload);

  // Publish to MQTT topic
  client.publish(mqtt_topic, payload.c_str());

  // Wait 10 seconds before next reading
  delay(10000);
}
```