Infrared Moisture Balance Device Project Arduino Code.

```
 #include <Wire.h> // Include the Wire library for I2C communication
#include <LiquidCrystal_I2C.h> // Include the LiquidCrystal_I2C library
#include "HX711.h"
#define WIFI_JIO
//#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>

const int relayPin = 15;
int lastSensorValue = 0;    // Store the last sensor reading
int consecutiveReadings = 0;  // Counter for consecutive readings
int consecutiveThreshold = 3;

#include <Wire.h>
#include "time.h"
#include <OneWire.h>
#include <DallasTemperature.h>

#define calibration_factor 1096800.00
#define LOADCELL_DOUT_PIN  16 // Connect load cell DOUT pin to GPIO21
#define LOADCELL_SCK_PIN   4  // Connect load cell SCK pin to GPIO22
#define buttonPin          2  // Pin connected to the reset button
#define resetDelay         2000 // Delay in milliseconds (2 seconds)

#define ONE_WIRE_BUS 13
#define TEMPERATURE_THRESHOLD 105.0

OneWire oneWire(ONE_WIRE_BUS); // Setup a oneWire instance to communicate with
any OneWire devices
DallasTemperature sensors(&oneWire);

HX711 scale;
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD address to 0x27 for a 16 chars
and 2 line display
unsigned long lastButtonPressTime = 0; // Variable to store the time of the last
button press
#include "addons/TokenHelper.h"
```

```
// Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"



// Insert your network credentials and make sure the define at the beginning of
the document is correct
#if defined(WIFI_JIO)
  //#define WIFI_SSID "JioFi_10C2184"
  //#define WIFI_PASSWORD "2ycrszvn8p"
 #define WIFI_SSID "FAB_LAB"
  #define WIFI_PASSWORD "FAB_LAB@12345"
  //#define WIFI_SSID "DESKTOP-55SBV03 9404"
  //#define WIFI_PASSWORD "FABLAB0123"
  #define WIFI_NAME "WS"
#endif

// Insert Firebase project API Key
#define API_KEY "AIzaSyCOIWEWCyBPaJALAvlrrXbiNyAqtlMio6w" //"va-black-soldier-
flies" firebase project

// Insert Authorized Email and Corresponding Password
#define USER_EMAIL "aniruddhaifd07@gmail.com" //you can also choose:
va_esp32_test@gmail.com
#define USER_PASSWORD "123456789"

// Insert RTDB URLefine the RTDB URL
#define DATABASE_URL "https://infrared-balance-device-default-rtdb.asia-
southeast1.firebasedatabase.app/"

// Define Firebase objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Variable to save USER UID
String uid;
int hours;
int minutes;

String databasePath;
String temppath = "/temperature0";
//String prespath = "/temperature1";
String humPath = "/humidity";
String timePath = "/timestamp";
```

```
// Parent Node (to be updated in every loop)
String parentPath;

int timestamp;
FirebaseJson json;

const char* ntpServer = "pool.ntp.org";

int weight;

// Timer variables (send new readings every three minutes)
unsigned long sendDataPrevMillis = 0;
unsigned long timerDelay = 60000*5; //1 minute delay, 1 sec = 1000 for timerDelay
(5min)
//variables for actuators
unsigned long delayTime;

void initWiFi() {
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.print("\nConnected to WiFi!\n");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
}

// Function that gets current epoch time
unsigned long getTime()
{
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    //Serial.println("Failed to obtain time");
    return(0);
  }
  hours = timeinfo.tm_hour;
  minutes = timeinfo.tm_min;
  time(&now);
```

```
  return now;
}



void setup() {
  sensors.begin(); // Start up the library
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
  initWiFi();
  //Serial.begin(9600);
  Serial.begin(115200);
  lcd.begin(); // Assuming 16 columns and 2 rows
  lcd.backlight(); // Turn on the backlight
  lcd.setCursor(0, 0);
  lcd.print("HX711 scale demo");
  delay(2000); // Wait for 2 seconds
  lcd.clear(); // Clear the display



  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(calibration_factor);
  scale.tare();
  pinMode(buttonPin, INPUT_PULLUP); // Set the button pin as input with internal
pull-up resistor
  bool status;
  // default settings



  configTime(0, 0, ntpServer);

  // Assign the api key (required)
  config.api_key = API_KEY;

  // Assign the user sign in credentials
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;

  // Assign the RTDB URL (required)
  config.database_url = DATABASE_URL;

  Firebase.reconnectWiFi(true);
  fbdo.setResponseSize(4096);
```

```
  // Assign the callback function for the long running token generation task
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  // Assign the maximum retry of token generation
  config.max_token_generation_retry = 5;

  // Initialize the library with the Firebase authen and config
  Firebase.begin(&config, &auth);

  // Getting the user UID might take a few seconds
  Serial.println("Getting User UID");
  while ((auth.token.uid) == "") {
    Serial.print('.');
    delay(1000);
}
uid = auth.token.uid.c_str();
  Serial.print("User UID: ");
  Serial.print(uid);

  // Update database path
  databasePath = "/UsersData/"+ uid + "/readings";

  Serial.println("\n\nEnd of void setup function");
}

void loop() {
  // Display load cell readings on LCD
 //float weight = scale.get_units() * 1000;
 sensors.requestTemperatures();
  int weight = scale.get_units() * 1000;
  Serial.print (weight);
  int weightBeforeDecimal = (int)weight;
  lcd.setCursor(0,0 );
  lcd.print("W: ");
  lcd.print(weight,0);
  lcd.print("g");
  timestamp = getTime();
  // Get temperature from DS18B20 sensor
  float temperatureC = sensors.getTempCByIndex(0);
  if (temperatureC != DEVICE_DISCONNECTED_C) { // Check if reading is valid
    Serial.print("Temperature: ");
    Serial.print(temperatureC);
    Serial.println(" °C");
  Serial.print ("current time: ");
```

```
  Serial.println (timestamp);
  Serial.println (hours);
  Serial.println (minutes);
  //Check variables and activate controls
  // Send new readings to database

  if (Firebase.ready() && (millis() - sendDataPrevMillis > timerDelay ||
sendDataPrevMillis == 0)){
    sendDataPrevMillis = millis();

    //Get current timestamp
    timestamp = getTime();
    Serial.print ("current time: ");
    Serial.println (timestamp);

    parentPath= databasePath + "/" + String(timestamp);

    json.set(temppath.c_str(), String(weight));
    //json.set(humPath.c_str(), String(rh));
    json.set(timePath, String(timestamp));

    //We can call that instruction inside a Serial.printf() command to print the
results in the Serial Monitor at the same time the command runs.
    Serial.printf("Set json... %s\n", Firebase.RTDB.setJSON(&fbdo,
parentPath.c_str(), &json) ? "ok" : fbdo.errorReason().c_str());

    if(weight == lastSensorValue) {
    consecutiveReadings++; // Increment the consecutive readings counter
  } else {
    consecutiveReadings = 0; // Reset the counter if the reading doesn't match
  }

  // Update the last sensor reading
  lastSensorValue =weight;

  // If consecutive readings reach the threshold, trigger the relay
  if (consecutiveReadings >= consecutiveThreshold) {
    digitalWrite(relayPin, HIGH); // Turn on the relay
    Serial.println("Relay triggered!");
    // You might want to add a delay here to prevent rapid triggering
    // delay(1000); // Wait for a second
    // Reset the consecutive readings counter
    consecutiveReadings = 0;
    delay(600000);
    //digitalWrite(relayPin, LOW);
```

```
  }

  // You can add further logic here if needed

    // Check if temperature exceeds threshold
    if (temperatureC >= TEMPERATURE_THRESHOLD) {
      // Turn off the relay
      digitalWrite(relayPin, HIGH);
      Serial.println("Relay turned off.");
    } else {
      // Turn on the relay
      digitalWrite(relayPin, LOW);
      Serial.println("Relay turned on.");
    }
    } else {
    Serial.println("Error: Unable to read temperature from sensor.");
  }

  delay(1000); // Wait for 1 second before taking the next reading
 }
}
```