****

**IOT102 PROJECT**

**WASTE MANAGEMENT SYSTEM**

**Group member: Nguyen Huu Minh Tai SE170053**

**Truong Nhat Hoang SE170464**

**Dang Cong Hung SE171500**

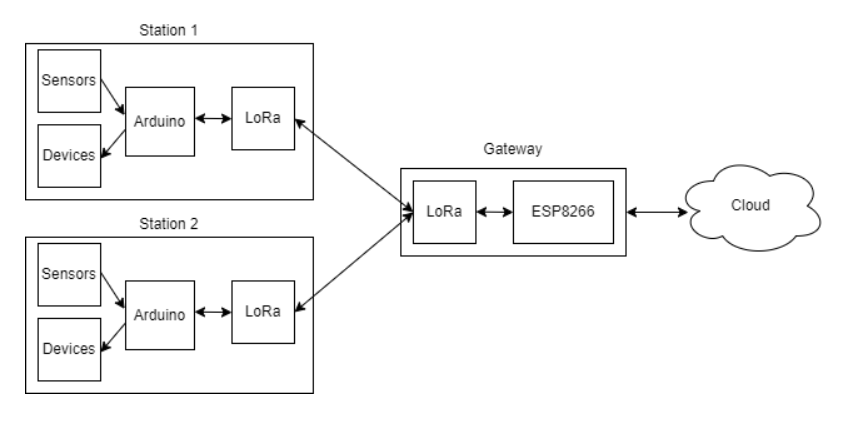
**Nguyen Tan Hung QE170205**

**Nguyen Van Hieu SE173336**

**HO CHI MINH CITY, 2023**

**I. PROJECT DESCRIPTIONS**

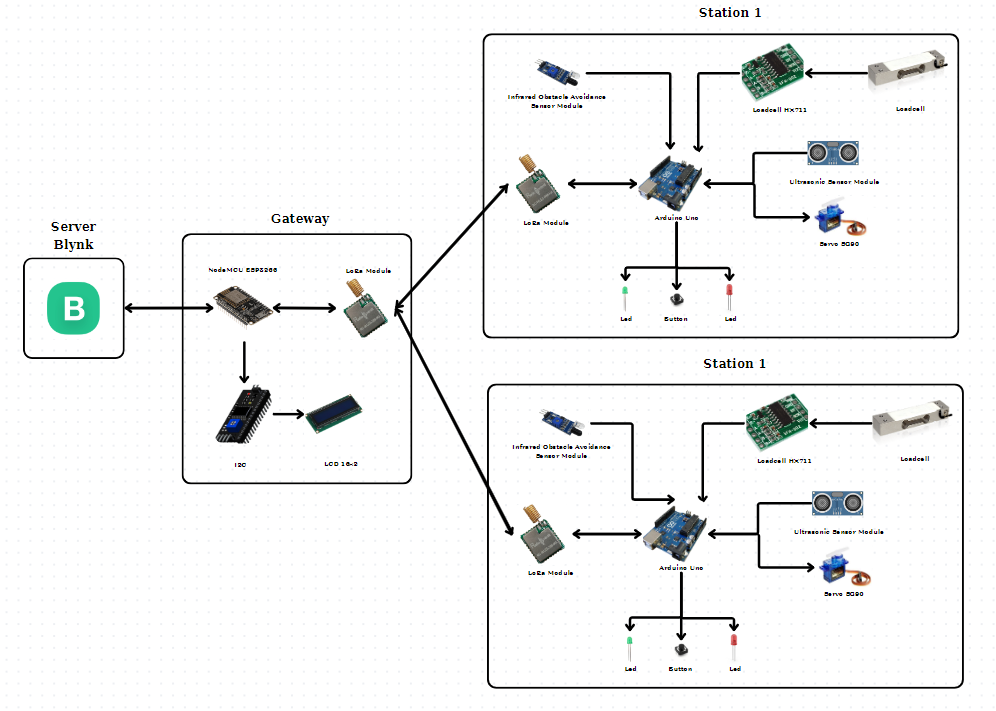
**Xác định vấn đề/chủ đề dự án**: Quản lý rác thải hiệu quả là một thách thức lớn trong các thành phố hiện đại. Dự án này tập trung vào việc phát triển một hệ thống quản lý rác thông minh dựa trên công nghệ LoRa và IoT.



Chúng ta có thể quản lý từng thùng rác;

* Kiểm soát được lượng rác thải của toàn thành phố thông qua việc báo cáo khối lượng rác của mỗi thùng. Biết được thùng rác nào đầy.
* Hạn chế việc thùng rác quá đầy, không thể đạy được nắp gây ảnh hưởng đến mỹ quan dô thị
* Tích hợp chế độ tự dộng mở thùng rác khi đến gần nhằm năng cao chất lượng sống, và tạo ra sự tiện loại cho người dùng. Giúp khuyến kích người dùng vứt khác đúng nơi quy định.

**Cách dự án hoạt động:** Hệ thống của chúng tôi bao gồm một bộ Arduino điều khiển tích hợp các cảm biến, bao gồm 1 động cơ servo SG90 180 độ dùng để đóng mở thùng rác, cảm biến Ultrasonic Sensor Module để nhận diện khoảng cách và có thể tự động đóng mở khi có người tới gần.

*Và chế độ này có thẻ bật tắt thông qua sever Blynk, nếu bật chế độ auto open thì đèn* ***màu xanh sẻ on****, còn tắt thì* ***đèn off****. Và lúc này nếu thùng rác chưa đầy thì chúng ta có thể ấn nút* ***button*** *để vứt rác*.

Và cảm biến Infrared Obstacle Avoidance Sensor Module để xác định tình trạng đầy của thùng rác. Bên cạnh đó, chúng tôi sử dụng một động cơ Servo để điều khiển việc mở nắp thùng rác.

* *Nếu thùng rác chứa đầy thì mở nắp thùng rác bình thường.* ***Đèn led màu đỏ sẽ off****.*
* *Nếu thùng rác đầy rồi thì chúng ta không thể mở được thùng rác.* ***Đèn led màu đỏ sẽ on.***
* *Lúc này chúng ta phải ấn* ***button*** *để thu gom rác.*

Cảm biến Loadcell để thu thập thông tin về khối lượng hiện tại của thùng rác. Dữ liệu thu thập từ các cảm biến được truyền bằng công nghệ LoRa, đồng thời LoRa cũng chuyển dữ liệu đến một Gateway để tiếp nhận thông tin. Các thông tin này sau đó được gửi lên server Blynk, nơi mà người dùng có thể dễ dàng theo dõi thông tin về tình trạng thùng rác hiện tại và thậm chí có khả năng điều khiển thùng rác thông qua giao diện tương tác, cho phép họ truyền tín hiệu để thùng rác thực hiện các chức năng cần thiết. Điều này tạo ra một hệ thống quản lý rác thông minh, cho phép quản lý nhiều thùng rác một cách hiệu quả và tiết kiệm nhân lực.

**Mục tiêu của dự án**: Mục tiêu chung của dự án là tối ưu hóa quá trình thu gom rác thải thông qua việc giám sát thùng rác trong thời gian thực. Mục tiêu cụ thể là giảm thiểu thời gian và nguồn lực tiêu hao trong quá trình thu gom rác. Nâng cao được chất lượng sống của xã hội.

**Phạm vi dự án**: Dự án này bao gồm thiết kế và triển khai hệ thống quản lý rác thông minh, bao gồm cả phần cứng (thùng rác thông minh) và phần mềm (hệ thống giám sát và thông báo).

**Đóng góp của dự án:** Dự án này đóng góp vào việc cải thiện hiệu suất quản lý rác thải, giảm thiểu tác động môi trường và cải thiện chất lượng cuộc sống trong thành phố.

**Bố cục dự án:** Dự án được chia thành nhiều giai đoạn, bao gồm:

+ Lên ý tưởng

+ Thiết kế hệ thống, vẽ sơ đồ hệ thống

+ Lắp đặt cảm biến, vẽ mạch điện

+ Viết code

+ Chạy thử

+ Hoàn thành mạch

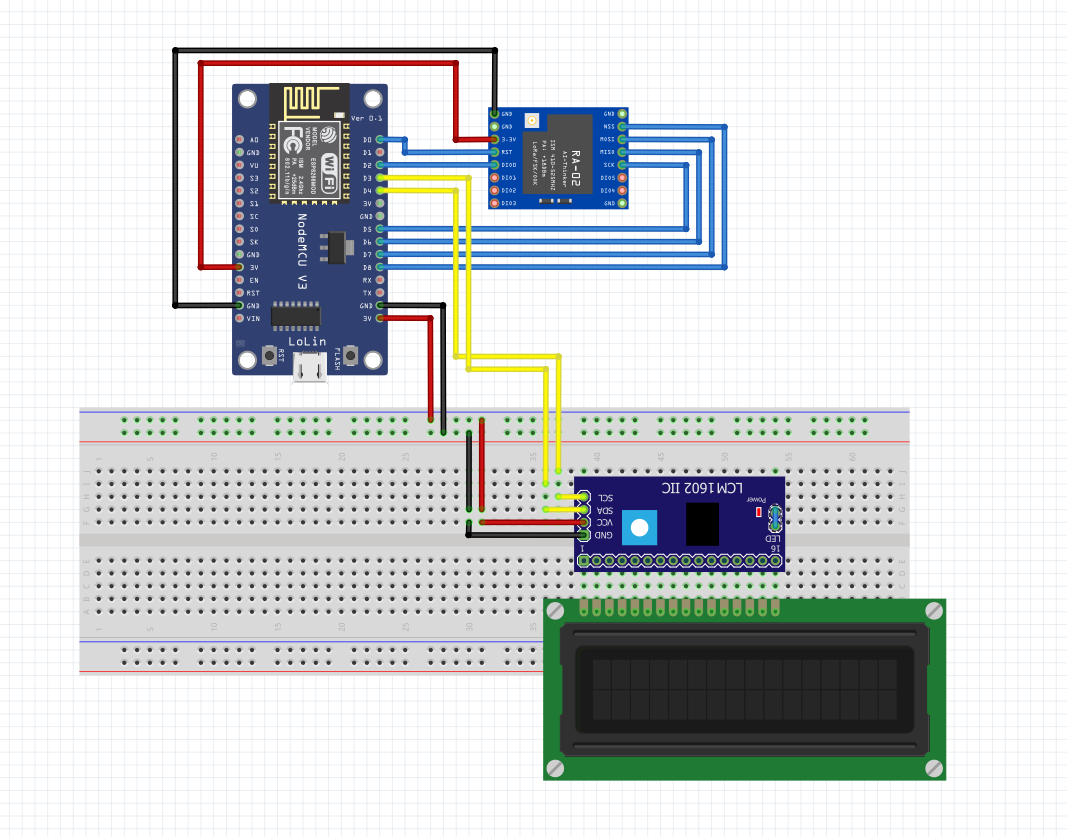
+ Viết báo cáo

**II. REQUIRED HARWARE**

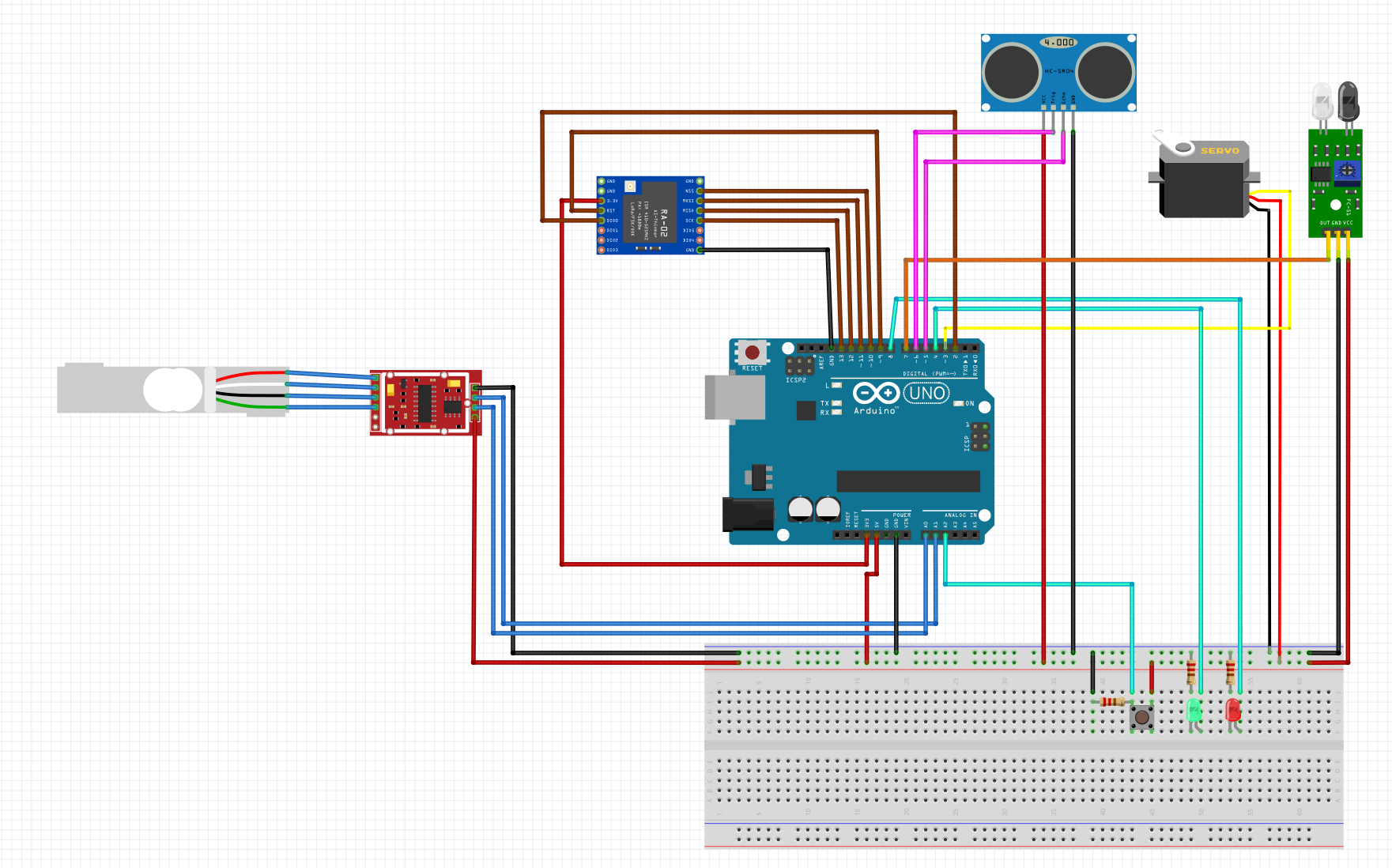
* Arduino Uno
* Module thu phát Wifi ESP8266 NodeMCU
* Mạch Thu Phát RF Lora SX1278 433Mhz Ra-01
* HC-SR04 Ultrasonic Sensor Module
* Động cơ servo SG90 180 độ
* Cảm Biến Loadcell 1kg + Mạch Chuyển Đổi ADC 24bit Loadcell HX711
* Cảm Biến Vật Cản Hồng Ngoại
* Đèn Led
* Momentary button or Switch
* Breadboard
* 330 ohm resistor
* Hook up wires
* Màn hình LCD1602 + Mạch Chuyển Đổi Giao Tiếp I2C Cho LCD
* Device used to control Blynk server

**III. CIRCUIT**

**GATEWAY**

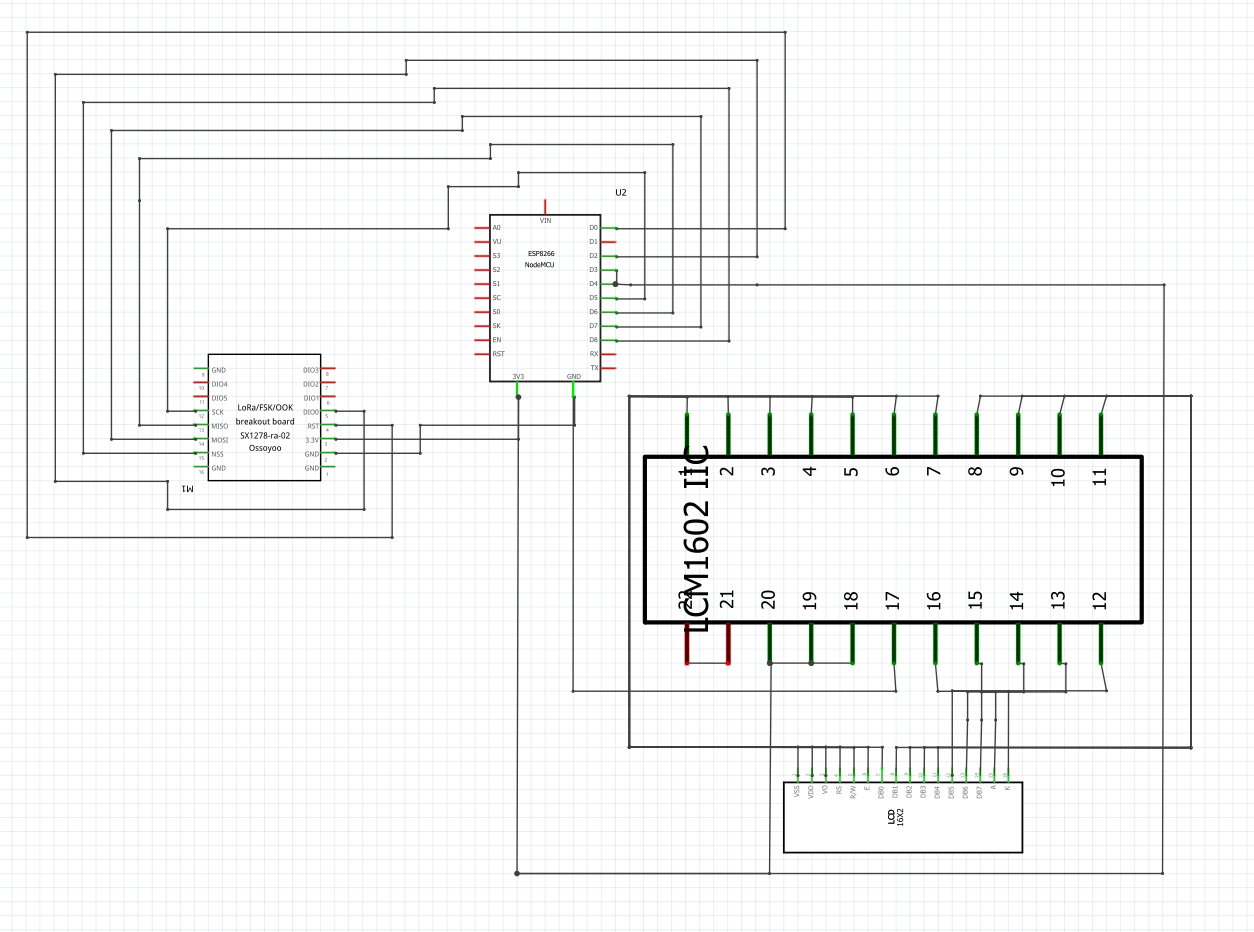
****

**STATION**

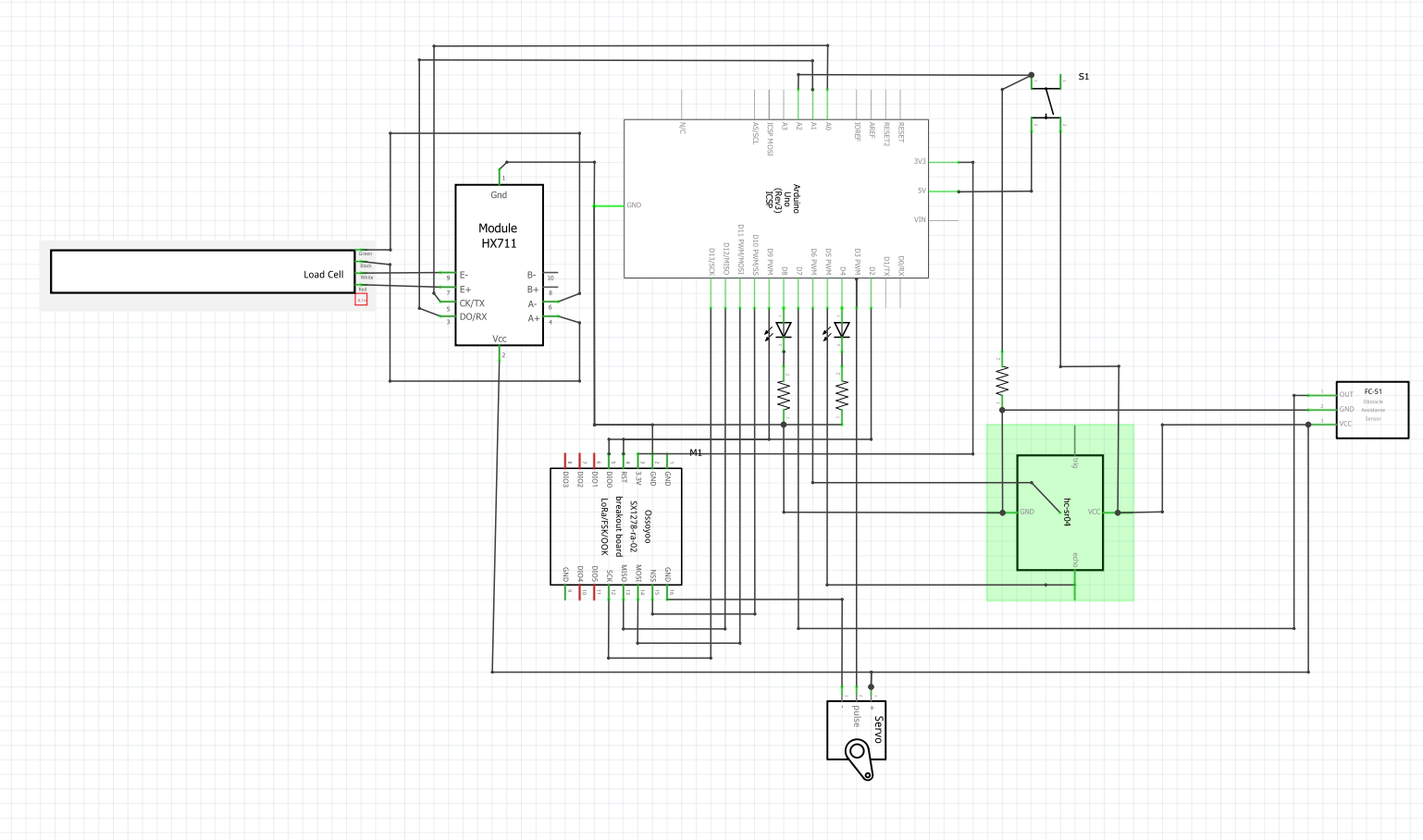


**III. SCHEMATIC**

**GATEWAY**

****

**STATION**



**IV. CODE**

**CODE OF GATEWAY**

#include <SPI.h>

#include <LoRa.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

//---------------------------------------- LoRa Pin / GPIO configuration.

#define ss 15

#define rst 16

#define dio0 2

//----------------------------------------

//---------------------------------------- LCD Pin.

LiquidCrystal\_I2C lcd(0x27,16,2);

//----------------------------------------

//---------------------------------------- Blynk.

#define BLYNK\_PRINT Serial

#define BLYNK\_TEMPLATE\_ID "TMPL6uz3V8Iyk"

#define BLYNK\_TEMPLATE\_NAME "Waste Management System"

#define BLYNK\_AUTH\_TOKEN "tgK7Rz9KY\_b36KhVDYpFHF-1IB0pt6Zj"

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

char ssid[] = "Minh Tai";

char pass[] = "71120079";

//----------------------------------------

//---------------------------------------- Variable declaration to hold incoming and outgoing data.

String Incoming = "";

String Message = "";

//----------------------------------------

//---------------------------------------- LoRa data transmission configuration.

byte LocalAddress = 0x01;                 //--> address of this device (Master Address).

byte Destination\_Arduino\_Slave\_1 = 0x02;    //--> destination to send to Slave 1 (Arduino).

//----------------------------------------

//---------------------------------------- Variable declaration for Millis/Timer.

unsigned long previousMillis\_SendMSG = 0;

const long interval\_SendMSG = 100;

//----------------------------------------

// Variable declaration to count slaves.

byte Slv = 0;

int autoOpen;

// Variable rece.

float w;

int ledAuto, ledOpen;

void onReceive(int packetSize) {

  if (packetSize == 0) return;  //--> if there's no packet, return

  //---------------------------------------- read packet header bytes:

  int recipient = LoRa.read();        //--> recipient address

  byte sender = LoRa.read();          //--> sender address

  byte incomingLength = LoRa.read();  //--> incoming msg length

  //----------------------------------------

  // Clears Incoming variable data.

  Incoming = "";

  //---------------------------------------- Get all incoming data.

  while (LoRa.available()) {

    Incoming += (char)LoRa.read();

  }

  //----------------------------------------

  //---------------------------------------- Check length for error.

  if (incomingLength != Incoming.length()) {

    Serial.println();

    Serial.println("error: message length does not match length");

    return; //--> skip rest of function

  }

  //----------------------------------------

  //---------------------------------------- Checks whether the incoming data or message for this device.

  if (recipient != LocalAddress) {

    Serial.println();

    Serial.println("This message is not for me.");

    return; //--> skip rest of function

  }

  //----------------------------------------

  //---------------------------------------- if message is for this device, or broadcast, print details:

  Serial.println();

  Serial.println("Received from: 0x" + String(sender, HEX));

  //Serial.println("Message length: " + String(incomingLength));

  Serial.println("Message: " + Incoming);

  //Serial.println("RSSI: " + String(LoRa.packetRssi()));

  //Serial.println("Snr: " + String(LoRa.packetSnr()));

  //----------------------------------------

  w = 0.0;

  ledAuto = 0;

  ledOpen = 0;

  int viTriDauPhay1 = Incoming.indexOf(',');

  int viTriDauPhay2 = Incoming.indexOf(',', viTriDauPhay1 + 1);

  String soThuNhatStr = Incoming.substring(viTriDauPhay1 + 1, viTriDauPhay2);

  String soThuHaiStr = Incoming.substring(viTriDauPhay2 + 1, Incoming.indexOf(',', viTriDauPhay2 + 1));

  String soThuBaStr = Incoming.substring(Incoming.lastIndexOf(',') + 1);

  w = soThuNhatStr.toFloat();

  double doubleValue = static\_cast<double>(w);

  ledAuto = soThuHaiStr.toInt();

  ledOpen = soThuBaStr.toInt();

  String tmp;

  if (ledOpen == 1) {

    tmp = "Y";

    WidgetLED LED(V2);

    LED.on();

  } else {

    tmp = "N";

    WidgetLED LED(V2);

    LED.off();

  }

  lcd.setCursor(10,1);

  lcd.print("Over:" + tmp);

  lcd.setCursor(0,0);

  lcd.print("Amount:");

  lcd.setCursor(7,0);

  lcd.print(w);

  Blynk.virtualWrite(V1, doubleValue);

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void sendMessage(String Outgoing, byte Destination) {

  LoRa.beginPacket();             //--> start packet

  LoRa.write(Destination);        //--> add destination address

  LoRa.write(LocalAddress);       //--> add sender address

  LoRa.write(Outgoing.length());  //--> add payload length

  LoRa.print(Outgoing);           //--> add payload

  LoRa.endPacket();               //--> finish packet and send it

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ VOID SETUP

void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600);

  Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

  Wire.begin(D3,D4);

  lcd.init();

  lcd.clear();

  lcd.backlight();

  //---------------------------------------- Settings and start Lora Ra-01.

  LoRa.setPins(ss, rst, dio0);

  Serial.println("Start LoRa init...");

  if (!LoRa.begin(433E6)) {             // initialize ratio at 915 or 433 MHz

    Serial.println("LoRa init failed. Check your connections.");

    while (true);                       // if failed, do nothing

  }

  Serial.println("LoRa init succeeded.");

  //----------------------------------------

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ VOID LOOP

void loop() {

  // put your main code here, to run repeatedly:

  Blynk.run();

  //---------------------------------------- Millis or Timer to send message / command data to slaves every 1 second (see interval\_SendCMD variable).

  // Messages are sent every one second is alternately.

  // > Master sends message to Slave 1, delay 1 second.

  // > Master sends message to Slave 2, delay 1 second.

  unsigned long currentMillis\_SendMSG = millis();

  if (currentMillis\_SendMSG - previousMillis\_SendMSG >= interval\_SendMSG) {

    previousMillis\_SendMSG = currentMillis\_SendMSG;

    Slv++;

    if (Slv > 2) Slv = 1;

    if (autoOpen == 1) {

      lcd.setCursor(0,1);

      lcd.print("Auto:ON ");

    } else {

      lcd.setCursor(0,1);

      lcd.print("Auto:OFF");

    }

    Message = "SDS" + String(Slv) + String(autoOpen);

    //::::::::::::::::: Condition for sending message / command data to Slave 1 (UNO Slave 1).

    if (Slv == 1) {

      Serial.println();

      Serial.print("Send message to UNO Slave " + String(Slv));

      Serial.println(" : " + Message);

      sendMessage(Message, Destination\_Arduino\_Slave\_1);

    }

    //:::::::::::::::::

    //::::::::::::::::: Condition for sending message / command data to Slave 2 (UNO Slave 2).

    // if (Slv == 2) {

    //   Serial.println();

    //   Serial.print("Send message to ESP32 Slave " + String(Slv));

    //   Serial.println(" : " + Message);

    //   sendMessage(Message, Destination\_ESP32\_Slave\_2);

    // }

    //:::::::::::::::::

  }

  //----------------------------------------

  //---------------------------------------- parse for a packet, and call onReceive with the result:

  onReceive(LoRa.parsePacket());

  //----------------------------------------

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

BLYNK\_WRITE(V0) {

  autoOpen = param.asInt();

}

**CODE OF STATION**

#include <SPI.h>

#include <LoRa.h>

#include <Servo.h>

#include <HX711\_ADC.h>

#define infrared  7

const int ledAuto = 4;

const int ledOver = 8;

const int button = A2;

const unsigned int TRIG\_PIN = 6;

const unsigned int ECHO\_PIN = 5;

HX711\_ADC LoadCell(A1,A0);

Servo myservo;

float w = 0.0;

int pos = 0;

byte ledStateAuto;

byte ledStateOver;

int state = 0;

String Incoming = "";

String Message = "";

//---------------------------------------- LoRa data transmission configuration.

////////////////////////////////////////////////////////////////////////////

// PLEASE UNCOMMENT AND SELECT ONE OF THE "LocalAddress" VARIABLES BELOW. //

////////////////////////////////////////////////////////////////////////////

byte LocalAddress = 0x02;       //--> address of this device (Slave 1).

//byte LocalAddress = 0x03;       //--> address of this device (Slave 2).

byte Destination\_Master = 0x01; //--> destination to send to Master (ESP32).

//----------------------------------------

//---------------------------------------- Millis / Timer to update weight values from sensor.

unsigned long previousMillis\_UpdateDHT11 = 0;

const long interval\_UpdateDHT11 = 1000;

//----------------------------------------

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Subroutines for sending data (LoRa Ra-02).

void sendMessage(String Outgoing, byte Destination) {

  LoRa.beginPacket();             //--> start packet

  LoRa.write(Destination);        //--> add destination address

  LoRa.write(LocalAddress);       //--> add sender address

  LoRa.write(Outgoing.length());  //--> add payload length

  LoRa.print(Outgoing);           //--> add payload

  LoRa.endPacket();               //--> finish packet and send it

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Subroutines for receiving data (LoRa Ra-02).

void onReceive(int packetSize) {

  if (packetSize == 0) {

    Serial.println("There's no packet");

    return;

  }   //--> if there's no packet, return

  //---------------------------------------- read packet header bytes:

  int recipient = LoRa.read();        //--> recipient address

  byte sender = LoRa.read();          //--> sender address

  byte incomingLength = LoRa.read();  //--> incoming msg length

  //----------------------------------------

  // Clears Incoming variable data.

  Incoming = "";

  //---------------------------------------- Get all incoming data.

  while (LoRa.available()) {

    Incoming += (char)LoRa.read();

  }

  //----------------------------------------

  //---------------------------------------- Check length for error.

  if (incomingLength != Incoming.length()) {

    Serial.println();

    Serial.println("error: message length does not match length");

    return; //--> skip rest of function

  }

  //----------------------------------------

  //---------------------------------------- Checks whether the incoming data or message for this device.

  if (recipient != LocalAddress) {

    Serial.println();

    Serial.println("This message is not for me.");

    return; //--> skip rest of function

  }

  //----------------------------------------

  //---------------------------------------- if message is for this device, or broadcast, print details:

  Serial.println();

  Serial.println("Received from: 0x" + String(sender, HEX));

  //Serial.println("Message length: " + String(incomingLength));

  Serial.println("Message: " + Incoming);

  //Serial.println("RSSI: " + String(LoRa.packetRssi()));

  //Serial.println("Snr: " + String(LoRa.packetSnr()));

  //----------------------------------------

  // Calls the Processing\_incoming\_data() subroutine.

  Processing\_incoming\_data();

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Subroutines to process data from incoming messages, then send messages to the Master.

void Processing\_incoming\_data() {

  //---------------------------------------- Conditions for sending messages to Master.

/////////////////////////////////////////////////////////////////////////////////////////

// PLEASE UNCOMMENT THE LINE OF CODE BELOW IF THIS CODE OR THIS DEVICE IS FOR SLAVE 1. //

/////////////////////////////////////////////////////////////////////////////////////////

  if (Incoming == "SDS11" && (digitalRead (infrared) == HIGH)) {

    digitalWrite(ledAuto, HIGH);

    digitalWrite(ledOver, LOW);

    digitalWrite(TRIG\_PIN, LOW);

    delayMicroseconds(2);

    digitalWrite(TRIG\_PIN, HIGH);

    delayMicroseconds(10);

    digitalWrite(TRIG\_PIN, LOW);

    const unsigned long duration = pulseIn(ECHO\_PIN, HIGH);

    int distance = duration/29/2;

    if(duration==0){

      Serial.println("Warning: no pulse from sensor");

    }

    else{

      Serial.print("Distance to nearest object: ");

      Serial.print(distance);

      Serial.println(" cm");

    }

    if (distance > 10 ) {

      myservo.write(180);

    }

    else if (distance <= 10 && distance >0) {

      myservo.write(60);

      delay(4000);

    }

    ledStateAuto = digitalRead(ledAuto);

    ledStateOver = digitalRead(ledOver);

    Message = "";

    Message = "SL1," + String(w) + "," + String(ledStateAuto) + "," + String(ledStateOver);

    Serial.println();

    Serial.println("Send message to Master");

    Serial.print("Message: ");

    Serial.println(Message);

    sendMessage(Message, Destination\_Master);

  }

  if (Incoming == "SDS11" && (digitalRead (infrared) == LOW)) {

    digitalWrite(ledAuto, HIGH);

    digitalWrite(ledOver, HIGH);

    ledStateAuto = digitalRead(ledAuto);

    ledStateOver = digitalRead(ledOver);

    Message = "";

    Message = "SL1," + String(w) + "," + String(ledStateAuto) + "," + String(ledStateOver);

    Serial.println();

    Serial.println("Send message to Master");

    Serial.print("Message: ");

    Serial.println(Message);

    sendMessage(Message, Destination\_Master);

  }

  if (Incoming == "SDS10" && (digitalRead (infrared) == LOW)) {

    digitalWrite(ledAuto, LOW);

    digitalWrite(ledOver, HIGH);

    ledStateAuto = digitalRead(ledAuto);

    ledStateOver = digitalRead(ledOver);

    Message = "";

    Message = "SL1," + String(w) + "," + String(ledStateAuto) + "," + String(ledStateOver);

    Serial.println();

    Serial.println("Send message to Master");

    Serial.print("Message: ");

    Serial.println(Message);

    sendMessage(Message, Destination\_Master);

  }

  if (Incoming == "SDS10" && (digitalRead (infrared) == HIGH)) {

    digitalWrite(ledAuto, LOW);

    digitalWrite(ledOver, LOW);

    ledStateAuto = digitalRead(ledAuto);

    ledStateOver = digitalRead(ledOver);

    Message = "";

    Message = "SL1," + String(w) + "," + String(ledStateAuto) + "," + String(ledStateOver);

    Serial.println();

    Serial.println("Send message to Master");

    Serial.print("Message: ");

    Serial.println(Message);

    sendMessage(Message, Destination\_Master);

  }

  //----------------------------------------

  //---------------------------------------- Conditions for sending messages to Master.

/////////////////////////////////////////////////////////////////////////////////////////

// PLEASE UNCOMMENT THE LINE OF CODE BELOW IF THIS CODE OR THIS DEVICE IS FOR SLAVE 2. //

/////////////////////////////////////////////////////////////////////////////////////////

//  if (Incoming == "SDS2") {

//    digitalWrite(LED\_1\_Pin, !digitalRead(LED\_1\_Pin));

//    digitalWrite(LED\_2\_Pin, !digitalRead(LED\_2\_Pin));

//

//    LED\_1\_State = digitalRead(LED\_1\_Pin);

//    LED\_2\_State = digitalRead(LED\_2\_Pin);

//

//    Message = "";

//    Message = "SL2," + String(h) + "," + String(t) + "," + String(LED\_1\_State) + "," + String(LED\_2\_State);

//

//    Serial.println();

//    Serial.println("Send message to Master");

//    Serial.print("Message: ");

//    Serial.println(Message);

//

//    sendMessage(Message, Destination\_Master);

//  }

  //----------------------------------------

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void setup() {

  Serial.begin(9600);

  pinMode (ledAuto, OUTPUT);

  pinMode (ledOver, OUTPUT);

  myservo.attach(3);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  pinMode(infrared, INPUT);

  pinMode(button, INPUT);

  for (pos = 180; pos >= 60; pos -= 1) {

      myservo.write(pos);

  }

  for (pos = 60; pos <= 180; pos += 1) {

      myservo.write(pos);

  }

  LoadCell.begin();

  LoadCell.start(2000);

  LoadCell.setCalFactor(1000.0);

  //---------------------------------------- Settings and start Lora Ra-01.

  Serial.println();

  Serial.println("Start LoRa init...");

  if (!LoRa.begin(433E6)) {             // initialize ratio at 915 or 433 MHz

    Serial.println("LoRa init failed. Check your connections.");

    while (true);                       // if failed, do nothing

  }

  Serial.println("LoRa init succeeded.");

  //----------------------------------------

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ VOID LOOP

void loop() {

  // put your main code here, to run repeatedly:

  //---------------------------------------- Millis / Timer to update the temperature and humidity values ​​from the DHT11 sensor every 2 seconds (see the variable interval\_UpdateDHT11).

  unsigned long currentMillis\_UpdateDHT11 = millis();

  if (currentMillis\_UpdateDHT11 - previousMillis\_UpdateDHT11 >= interval\_UpdateDHT11) {

    previousMillis\_UpdateDHT11 = currentMillis\_UpdateDHT11;

    // Reading weight takes about 250 milliseconds!

    LoadCell.update();

    w = LoadCell.getData();

    // Check if any reads failed and exit early (to try again).

    if (isnan(w)) {

      Serial.println(F("Failed to read from LoadCell sensor!"));

      w = 0.0;

    }

  }

  //----------------------------------------

  state = digitalRead(button);

  if (state == 1) {

    Serial.println("Open");

    myservo.write(60);

    delay(3000);

    return;

  }

  //---------------------------------------- parse for a packet, and call onReceive with the result:

  onReceive(LoRa.parsePacket());

  //----------------------------------------

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**V. DEMONSTRATING RESULTS**

Khi mạch chạy thành công thì, satation sẽ đợi tín hiệu từ gateway, khi gateway hoạt động thành công thì gateway sẽ gửi tín hiệu bật tắt chế độ Auto Open. Và Station sẽ gửi khối lượng thùng rác hiện tại về cho gateway, và gateway sẽ hiển thị lên LCD và update data lên Sever Blynk.

Chế độ Auto được bật thì đèn LED MÀU XANH sẽ sáng, thùng rác sẽ tự động đống mở khi có người đến gần nếu thùng rác chưa đầy.

Khi thùng rác đầy thì chế độ auto sẽ bị vô hiệu hóa và đền LED MÀU ĐỎ sẽ sáng và sẽ gửi tín hiệu lên sever Blynk để cập nhật trạng thái. Lúc này ta phải ấn button để thu thập rác thải.

Khi ta tắt chế độ auto thì buộc chúng ta phải ấn button để có thể vứt rác.