//10A

#include <WiFi.h>

#include <esp\_now.h>

//Read Water and Light Sensor

#include <stdint.h>

#define Light\_Y 1

#define Temp\_Y 2

#define Light\_F 3

#define Temp\_F 4

#define WATER\_SENSOR 5

//GPIO pin

// Replace with the MAC address of the receiver ESP32S3 board

uint8\_t broadcastAddress[] = {0x34, 0x85, 0x18, 0x91, 0xB4, 0xF4}; //34:85:18:8E:29:18 60B: 34:85:18:AC:BD:6C; 59A: 0x34, 0x85, 0x18, 0x91, 0x30, 0xA4

// Structure example to send data

// Must match the receiver structure

typedef struct struct\_message {

 // char a[32];

 int id;

 float b;

 float c;

 float d;

 float e;

 float f;

 float g;

 float h;

 float i;

 // uint8\_t crc; //CRC for error checking

 // bool d;

} struct\_message;

/\*------------------ CRC-8 Calculation-----------------------\*/

// CRC-8 lookup table for polynomial 0x8C (reverse of 0x31)

const uint8\_t crc8\_table[256] = {

0x00, 0x8C, 0x94, 0x18, 0xA4, 0x28, 0x30, 0xBC, 0xC4, 0x48, 0x50, 0xDC, 0x60, 0xEC, 0xF4, 0x78,

 0x04, 0x88, 0x90, 0x1C, 0xA0, 0x2C, 0x34, 0xB8, 0xC0, 0x4C, 0x54, 0xD8, 0x64, 0xE8, 0xF0, 0x7C,

 0x08, 0x84, 0x9C, 0x10, 0xAC, 0x20, 0x38, 0xB4, 0xCC, 0x40, 0x58, 0xD4, 0x68, 0xE4, 0xFC, 0x70,

 0x0C, 0x80, 0x98, 0x14, 0xA8, 0x24, 0x3C, 0xB0, 0xC8, 0x44, 0x5C, 0xD0, 0x6C, 0xE0, 0xF8, 0x74,

 0x10, 0x9C, 0x84, 0x08, 0xB4, 0x38, 0x20, 0xAC, 0xD4, 0x58, 0x40, 0xCC, 0x70, 0xFC, 0xE4, 0x68,

 0x14, 0x98, 0x80, 0x0C, 0xB0, 0x3C, 0x24, 0xA8, 0xD0, 0x5C, 0x44, 0xC8, 0x74, 0xF8, 0xE0, 0x6C,

 0x18, 0x94, 0x8C, 0x00, 0xBC, 0x30, 0x28, 0xA4, 0xDC, 0x50, 0x48, 0xC4, 0x78, 0xF4, 0xEC, 0x60,

 0x1C, 0x90, 0x88, 0x04, 0xB8, 0x34, 0x2C, 0xA0, 0xD8, 0x54, 0x4C, 0xC0, 0x7C, 0xF0, 0xE8, 0x64,

 0x20, 0xAC, 0xB4, 0x38, 0x84, 0x08, 0x10, 0x9C, 0xE4, 0x68, 0x70, 0xFC, 0x40, 0xCC, 0xD4, 0x58,

 0x24, 0xA8, 0xB0, 0x3C, 0x80, 0x0C, 0x14, 0x98, 0xE0, 0x6C, 0x74, 0xF8, 0x44, 0xC8, 0xD0, 0x5C,

 0x28, 0xA4, 0xBC, 0x30, 0x8C, 0x00, 0x18, 0x94, 0xEC, 0x60, 0x78, 0xF4, 0x48, 0xC4, 0xDC, 0x50,

 0x2C, 0xA0, 0xB8, 0x34, 0x88, 0x04, 0x1C, 0x90, 0xE8, 0x64, 0x7C, 0xF0, 0x4C, 0xC0, 0xD8, 0x54,

 0x30, 0xBC, 0xA4, 0x28, 0x94, 0x18, 0x00, 0x8C, 0xF4, 0x78, 0x60, 0xEC, 0x50, 0xDC, 0xC4, 0x48,

 0x34, 0xB8, 0xA0, 0x2C, 0x90, 0x1C, 0x04, 0x88, 0xF0, 0x7C, 0x64, 0xE8, 0x54, 0xD8, 0xC0, 0x4C,

 0x38, 0xB4, 0xAC, 0x20, 0x9C, 0x10, 0x08, 0x84, 0xFC, 0x70, 0x68, 0xE4, 0x58, 0xD4, 0xCC, 0x40,

 0x3C, 0xB0, 0xA8, 0x24, 0x98, 0x14, 0x0C, 0x80, 0xF8, 0x74, 0x6C, 0xE0, 0x5C, 0xD0, 0xC8, 0x44

};

// Function to calculate CRC-8

uint8\_t calculateCRC8(const void\* data, size\_t length) {

   uint8\_t crc = 0;

   uint8\_t\* buffer = (uint8\_t\*)data;

   for (size\_t i = 0; i < length; i++) {

       crc = crc8\_table[crc ^ buffer[i]];

   }

   return crc;

}

// Create a struct\_message called myData

struct\_message myData;

esp\_now\_peer\_info\_t peerInfo;

// callback when data is sent

void OnDataSent(const uint8\_t \*mac\_addr, esp\_now\_send\_status\_t status) {

 Serial.print("\r\nLast Packet Send Status:\t");

 Serial.println(status == ESP\_NOW\_SEND\_SUCCESS ? "Delivery Success" : "Delivery Fail");

}

void setup() {

 // Init Serial Monitor

 Serial.begin(115200);

 //pinMode(Light,INPUT);

 pinMode(Light\_Y,INPUT);

 pinMode(Temp\_Y,INPUT);

 pinMode(Light\_F,INPUT);

 pinMode(Temp\_F,INPUT);

 pinMode(WATER\_SENSOR, INPUT);

 // Set device as a Wi-Fi Station

 WiFi.mode(WIFI\_STA);

 // Init ESP-NOW

 if (esp\_now\_init() != ESP\_OK) {

   Serial.println("Error initializing ESP-NOW");

   return;

 }

 // Once ESPNow is successfully Init, we will register for Send CB to

 // get the status of Trasnmitted packet

 esp\_now\_register\_send\_cb(OnDataSent);

 // Register peer

 memcpy(peerInfo.peer\_addr, broadcastAddress, 6);

 peerInfo.channel = 0;

 peerInfo.encrypt = false;

 // Add peer

 if (esp\_now\_add\_peer(&peerInfo) != ESP\_OK){

   Serial.println("Failed to add peer");

   return;

 }

}

void loop() {

 int sensorValue\_opY = analogRead(Light\_Y);

 float voltage\_opY = sensorValue\_opY \* (3.3 / 4096.0);

 float Lux\_Y = (1570.1\*exp(-0.866\*voltage\_opY));

 int sensorValue\_TY = analogRead(Temp\_Y);

 float voltage\_TY = sensorValue\_TY \* (3.3 / 4096.0);

 float Tem\_Y = -12.161\*voltage\_TY+56.241;

 int sensorValue\_opF = analogRead(Light\_F);

 float voltage\_opF = sensorValue\_opF \* (3.3 / 4096.0);

 float Lux\_F = (540\*pow(voltage\_opF,-1.067));

 int sensorValue\_TF = analogRead(Temp\_F);

 float voltage\_TF = sensorValue\_TF \* (3.3 / 4096.0);

 float Tem\_F = -20.134\*voltage\_TF+67.779;

 int SHUI = analogRead(WATER\_SENSOR);

 int WATER = ((SHUI-4095)\*(-1));

 myData.id=1;

 myData.b = Lux\_Y;

 myData.c = Tem\_Y;

 myData.d = Lux\_F;

 myData.e = Tem\_F;

 myData.f = WATER;

 // Calculate CRC

 uint8\_t CRC =calculateCRC8(&myData, sizeof(myData));

 // Send data including CRC using ESP-NOW

 uint8\_t dataToSend[sizeof(myData)+1];

 memcpy(dataToSend,&myData,sizeof(myData));

 dataToSend[sizeof(myData)] = CRC;

 // Send message via ESP-NOW

 esp\_err\_t result = esp\_now\_send(broadcastAddress, dataToSend, sizeof(dataToSend));

 if (result == ESP\_OK)

 {

   Serial.print("Lux\_Y:");

   Serial.println(Lux\_Y);

   Serial.print("Tem\_Y:");

   Serial.println(Tem\_Y);

   Serial.print("Lux\_F:");

   Serial.println(Lux\_F);

   Serial.print("Tem\_F:");

   Serial.println(Tem\_F);

   if(WATER > 800)

        {

           Serial.print("degree of water:");

           Serial.println(WATER);

           Serial.println("Detected Water");

        }

   else{

           Serial.print("degree of water:");

           Serial.println(WATER);

           Serial.println(" Dry");

       }

   Serial.println(" Sender 2: Sent with success");

   Serial.print("  Calculated CRC: ");

   Serial.println(dataToSend[sizeof(myData)]);

 }

 else {

   Serial.println("Error sending the data");

 }

 delay(1500);

}