//MASTER

#include <esp\_now.h>

#include <WiFi.h>

#include <HardwareSerial.h>

HardwareSerial SerialPort(2); //use UART2

#define UART\_TX\_PIN 43

#define UART\_RX\_PIN 44

#define BAUD\_RATE 115200

// Structure example to receive data

// Must match the sender structure

typedef struct struct\_message {

    int id;

    // char a[32];

    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;

typedef struct UART\_message {

    int id;

    float b;

    float c;

    float d;

    float e;

    float f;

    float g;

    float h;

    float i;

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

    // bool d;

} UART\_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;

struct\_message sender1;

struct\_message sender2;

struct\_message sender3;

struct\_message sender4;

struct\_message sender5;

struct\_message sender6;

struct\_message sender7;

struct\_message sender8;

UART\_message dataUARTSend;

// Create an array with all the structures

struct\_message boardsStruct[8] = {sender1, sender2, sender3, sender4, sender5, sender6, sender7, sender8};

// callback function that will be executed when data is received

void OnDataRecv(const uint8\_t \* mac\_addr, const uint8\_t \*incomingData, int len) {

  char macStr[18];

 // Serial.print("Packet received from: ");

  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",

           mac\_addr[0], mac\_addr[1], mac\_addr[2], mac\_addr[3], mac\_addr[4], mac\_addr[5]);

  Serial.println(macStr);

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

  uint8\_t ReceivedCRC = incomingData [len - 1];

  uint8\_t calculatedCRC;

  int CRCCheckSum;

  byte\* byteArray;

/\*\*\*\*\*\*\*\*\*\*----Process different types of incoming data----\*\*\*\*\*\*\*\*\*\*/

 switch (myData.id){

    case 1:

            // Calculate CRC-8 for the received data

            calculatedCRC = calculateCRC8(&myData, sizeof(myData));  // Mind how many data types are sent, //Board ID 1

            boardsStruct[myData.id-1].b = myData.b;

            boardsStruct[myData.id-1].c = myData.c;

            boardsStruct[myData.id-1].d = myData.d;

            boardsStruct[myData.id-1].e = myData.e;

            boardsStruct[myData.id-1].f = myData.f;

            Serial.printf("    Board ID %u: %u bytes\n", myData.id, len);

            Serial.printf("    Lux\_Y: %.2f lux\n", boardsStruct[myData.id-1].b);

            Serial.printf("    Tem\_Y: %.2f C\n", boardsStruct[myData.id-1].c);

            Serial.printf("    Lux\_F: %.2f lux\n", boardsStruct[myData.id-1].d);

            Serial.printf("    Tem\_F: %.2f C\n", boardsStruct[myData.id-1].e);

            Serial.printf("    WATER: %.2f \n", boardsStruct[myData.id-1].f);

           Serial.printf("    Calcultaed CRC: %d \n", calculatedCRC);

            Serial.printf("    Received CRC: %d \n", ReceivedCRC);

            if (calculatedCRC == ReceivedCRC){

              Serial.printf("    Data Tramission Right!");

              CRCCheckSum = 1;

            }

            else{

              Serial.printf("    Data Tramission Wrong!");

              CRCCheckSum = 0;

            }

            Serial.println();

            // UART Data Transmission

            dataUARTSend.id=myData.id;

            dataUARTSend.b=boardsStruct[myData.id-1].b;

            dataUARTSend.c=boardsStruct[myData.id-1].c;

            dataUARTSend.d=boardsStruct[myData.id-1].d;

            dataUARTSend.e=boardsStruct[myData.id-1].e;

            dataUARTSend.f=boardsStruct[myData.id-1].f;

            dataUARTSend.CRC\_checksum=CRCCheckSum;

            byteArray = reinterpret\_cast<byte\*>(&dataUARTSend);

            SerialPort.write(byteArray,sizeof(dataUARTSend));

            delay(1500);

            break;

    case 2:

            calculatedCRC = calculateCRC8(&myData, sizeof(myData));  // Mind how many data types are sent,//Board ID 2

            boardsStruct[myData.id-1].g = myData.g;

            boardsStruct[myData.id-1].h = myData.h;

            boardsStruct[myData.id-1].i = myData.i;

            /\*boardsStruct[myData.id-1].e = myData.e;

            boardsStruct[myData.id-1].f = myData.f;\*/

            Serial.printf("    Board ID %u: %u bytes\n", myData.id, len);

            Serial.printf("    moisture: %.2f \n", boardsStruct[myData.id-1].g);

            Serial.printf("    Humidity(): %.2f %%\n", boardsStruct[myData.id-1].h);

            Serial.printf("    Temperature(): %.2f C\n", boardsStruct[myData.id-1].i);

            Serial.printf("    Calcultaed CRC: %d \n", calculatedCRC);

            Serial.printf("    Received CRC: %d \n", ReceivedCRC);

            if (calculatedCRC == ReceivedCRC){

              Serial.printf("    Data Tramission Right!");

              CRCCheckSum = 1;

            }

            else{

              Serial.printf("    Data Tramission Wrong!");

              CRCCheckSum = 0;

            }

            Serial.println();

            // UART Data Transmission

            dataUARTSend.id=myData.id;

            dataUARTSend.g=boardsStruct[myData.id-1].g;

            dataUARTSend.h=boardsStruct[myData.id-1].h;

            dataUARTSend.i=boardsStruct[myData.id-1].i;

            dataUARTSend.CRC\_checksum=CRCCheckSum;

            byteArray = reinterpret\_cast<byte\*>(&dataUARTSend);

            SerialPort.write(byteArray,sizeof(dataUARTSend));

            delay(2000);

            break;

    default:

            calculatedCRC = calculateCRC8(&myData, sizeof(myData)-sizeof(myData.c));  // Mind how many data types are sent //Board ID 3;

            boardsStruct[myData.id-1].b = myData.b;

            Serial.printf("    Board ID %u: %u bytes\n", myData.id, len);

            //Serial.printf("    TSL2591 Light Density: %.2f lux\n", boardsStruct[myData.id-1].b);

            Serial.printf("    Calcultaed CRC: %d \n", calculatedCRC);

            Serial.printf("    Received CRC: %d \n", ReceivedCRC);

            if (calculatedCRC == ReceivedCRC){

              Serial.printf("    Data Tramission Right!");

              CRCCheckSum = 1;

            }

            else{

              Serial.printf("    Data Tramission Wrong!");

              CRCCheckSum = 0;

            }

            Serial.println();

            // UART Data Transmission

            dataUARTSend.id=myData.id;

            dataUARTSend.b=boardsStruct[myData.id-1].b;

            dataUARTSend.CRC\_checksum=CRCCheckSum;

            byteArray = reinterpret\_cast<byte\*>(&dataUARTSend);

            SerialPort.write(byteArray,sizeof(dataUARTSend));

            delay(1000);

            break;

  }

}

void setup() {

  // Initialize Serial Monitor

  //Serial.begin(115200);

  Serial.begin(BAUD\_RATE);

  SerialPort.begin(BAUD\_RATE, SERIAL\_8N1, UART\_RX\_PIN, UART\_TX\_PIN);

  // 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 recv CB to

  // get recv packer info

  esp\_now\_register\_recv\_cb(OnDataRecv);

}

void loop() {

}