### Slave2:

#include <WiFi.h> #include <esp\_now.h> #include "DHT.h" #include "DHT20.h" #include <stdint.h> #define MoisturePin 1 DHT20 DHT;

//#define DHTPIN 5 //GPIO 5 is connected to the DHT sensor

//#define DHTTYPE DHT20 //DHT 22(AM2302, AM2301)

//DHT dht(DHTPIN, DHTTYPE);

// Replace with the MAC address of the receiver ESP32S3 board uint8\_t count = 0;

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;

} 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);

 //Serial.print(F("DHT20 Test!"));

 Serial.println( FILE );

 Serial.print("DHT20 LIBRARY VERSION: ");

 Serial.println(DHT20\_LIB\_VERSION);

 Serial.println();

 Wire.begin();

 DHT.begin();

 delay(1000);

 // 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() {

 // Set values to send

 // Reading temperature or humidity takes about 250 milliseconds!

 // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

   int sensorValue = analogRead(MoisturePin);

   int moisture = sensorValue;

 // Convert the analog reading ADC:12bit (which goes from 0 - 4095) to a voltage (0 - 5V):

 //float moisture = ( 100 - ( (sensorValue/4095.00) \* 100 ) );

 if (millis() - DHT.lastRead() >= 1000)

 {

   //  READ DATA

   uint32\_t start = micros();

   int status = DHT.read();

   uint32\_t stop = micros();

   if ((count % 10) == 0)

   {

     count = 0;

     Serial.println();

     Serial.println("Type\tHumidity (%)\tTemp (°C)\tTime (µs)\tStatus");

   }

   count++;

   Serial.print("DHT20 \t");

   //  DISPLAY DATA, sensor has only one decimal.

   //Serial.print(DHT.getHumidity(), 1);

   //Serial.print("\t\t");

   //Serial.print(DHT.getTemperature(), 1);

   //Serial.print("\t\t");

   //Serial.print(stop - start);

   //Serial.print("\t\t");

 myData.id=2;  // Board ID

 // strcpy(myData.a, "This is Sender1-59C ");

 myData.g = moisture;

 myData.h = (DHT.getHumidity());

 myData.i = (DHT.getTemperature());

 }

 //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(F("Humidity: "));

   Serial.print(DHT.getHumidity());

   Serial.println("%");

   Serial.print(F("Temperature: "));

   Serial.print(DHT.getTemperature());

   Serial.println(F("C "));

   Serial.print(F("Moisture = "));

   Serial.print(moisture);

 if (moisture<300)

{

      Serial.println("Dry ");

         }

       else

       {

       if (moisture>300 &&moisture<700)

      {

        Serial.println("About to dry ");

      }

         else

         {

       if (moisture>700)

           {

          Serial.println("Wet ");

           }

         }

    }

   Serial.println("  Sent with success");

 }

 else {

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

 }

 delay(2000); //Based on TDMA consideration

}