## Data Logger and Telemetry System Script

### Transmitter Code

#include <SPI.h>

#include <LoRa.h>

//Libraries for LoRa

#include "DHT.h"

#include "DHT.h"

//Libraries for OLED Display

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define DHTPIN 4 //pin where the dht11 is connected

DHT dht(DHTPIN, DHT11);

#define SENSOR 27 //fuelflow sensor

//define the pins used by the LoRa transceiver module

#define ss 5

#define rst 14

#define dio0 2

#define BAND 866E6 for Europe

//OLED pins

#define OLED\_SDA 21

#define OLED\_SCL 22

#define OLED\_RST -1

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

//packet counter

int readingID = 0;

int counter = 0;

String LoRaMessage = "";

float temperature = 0;

float wheel\_speed = 0;

float fuel\_flow = 0;

float coolant\_temperature = 0;

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RST);

//Initialize OLED display

void startOLED(){

//reset OLED display via software

pinMode(OLED\_RST, OUTPUT);

digitalWrite(OLED\_RST, LOW);

delay(20);

digitalWrite(OLED\_RST, HIGH);

//initialize OLED

Wire.begin(OLED\_SDA, OLED\_SCL);

if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for 128x32

Serial.println(F("SSD1306 allocation failed"));

for(;;); // Don't proceed, loop forever

}

display.clearDisplay();

display.setTextColor(WHITE);

display.setTextSize(1);

display.setCursor(0,0);

display.print("LORA SENDER");

}

//Initialize LoRa module

void startLoRA()

{

LoRa.setPins(ss, rst, dio0); //setup LoRa transceiver module

while (!LoRa.begin(BAND) && counter < 10) {

Serial.print(".");

counter++;

delay(500);

}

if (counter == 10)

{

// Increment readingID on every new reading

readingID++;

Serial.println("Starting LoRa failed!");

}

Serial.println("LoRa Initialization OK!");

delay(2000);

}

void startDHT()

{

if (isnan(wheel\_speed) || isnan(temperature) || isnan(fuel\_flow) || isnan(coolane\_temperature))

{

Serial.println("Failed to read from the sensors!");

return;

}

}

void getReadings(){

temperature = dht.readTemperature();

Serial.print(F("% Temperature: "));

Serial.print(temperature);

Serial.println(F("°C "));

// read hall effect sensor value

Wheel\_speed= hallRead(); // print the results to the serial monitor Serial.println(Wheel\_speed);

delay(1000);

}

void sendReadings() {

LoRaMessage = String(readingID) + "/" + String(temperature) + "&" + String(Wheel\_speed) + "&" + String(fuel\_flow) + "&" + String(coolant\_temperature);

//Send LoRa packet to receiver

LoRa.beginPacket();

LoRa.print(LoRaMessage);

LoRa.endPacket();

Serial.print("Sending packet: ");

Serial.println(readingID);

readingID++;

Serial.println(LoRaMessage);

}

void setup() {

//initialize Serial Monitor

Serial.begin(115200);

dht.begin();

startDHT();

startOLED();

startLoRA();

}

void loop() {

getReadings();

sendReadings();

delay(5000);

}

### Receiver Code

#include <WiFi.h>

#include "ESPAsyncWebServer.h"

#include <SPIFFS.h>

//Libraries for LoRa

#include <SPI.h>

#include <LoRa.h>

// Libraries to get time from NTP Server

#include <NTPClient.h>

#include <WiFiUdp.h>

//define the pins used by the LoRa transceiver module

#define ss 5

#define rst 14

#define dio0 2

#define BAND 433E6 //433E6 for Asia, 866E6 for Europe, 915E6 for North America

// Replace with your network credentials

const char\* ssid = "myhome";

const char\* password = "9496234603";

// Define NTP Client to get time

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP);

// Variables to save date and time

String formattedDate;

String day;

String hour;

String timestamp;

// Initialize variables to get and save LoRa data

int rssi;

String loRaMessage;

String temperature;

String wheel\_speed;

String fuel\_flow;

String coolant\_temperature;

String readingID;

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

// Replaces placeholder with DHT values

String processor(const String& var){

//Serial.println(var);

if(var == "TEMPERATURE"){

return temperature;

}

else if(var == "WHEEL\_SPEED"){

return wheel\_speed;

}

else if(var == “FUEL\_FLOW"){

return fuel\_flow;

}

else if(var == "COOLANT\_TEMPERATURE"){

return coolant\_temperature;

}

else if(var == "TIMESTAMP"){

return timestamp;

}

else if (var == "RRSI"){

return String(rssi);

}

return String();

}

//Initialize LoRa module

void startLoRA(){

int counter;

//setup LoRa transceiver module

LoRa.setPins(ss, rst, dio0); //setup LoRa transceiver module

while (!LoRa.begin(BAND) && counter < 10) {

Serial.print(".");

counter++;

delay(500);

}

if (counter == 10) {

// Increment readingID on every new reading

Serial.println("Starting LoRa failed!");

}

Serial.println("LoRa Initialization OK!");

display.setCursor(0,10);

display.clearDisplay();

display.print("LoRa Initializing OK!");

display.display();

delay(2000);

}

void connectWiFi(){

// Connect to Wi-Fi network with SSID and password

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

// Print local IP address and start web server

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

display.setCursor(0,20);

display.print("Access web server at: ");

display.setCursor(0,30);

display.print(WiFi.localIP());

display.display();

}

// Read LoRa packet and get the sensor readings

void getLoRaData() {

Serial.print("Lora packet received: ");

// Read packet

while (LoRa.available()) {

String LoRaData = LoRa.readString();

// LoRaData format: readingID/temperature&soilMoisture#batterylevel

// String example: 1/27.43&654#95.34

Serial.print(LoRaData);

// Get readingID, temperature and moisture

int pos1 = LoRaData.indexOf('/');

int pos2 = LoRaData.indexOf('&');

readingID = LoRaData.substring(0, pos1);

temperature = LoRaData.substring(pos1 +1, pos2);

humidity = LoRaData.substring(pos2+1, LoRaData.length());

}

// Get RSSI

rssi = LoRa.packetRssi();

Serial.print(" with RSSI ");

Serial.println(rssi);

}

// Function to get date and time from NTPClient

void getTimeStamp() {

while(!timeClient.update()) {

timeClient.forceUpdate();

}

// The formattedDate comes with the following format:

// 2018-05-28T16:00:13Z

// We need to extract date and time

formattedDate = timeClient.getFormattedDate();

Serial.println(formattedDate);

// Extract date

int splitT = formattedDate.indexOf("T");

day = formattedDate.substring(0, splitT);

Serial.println(day);

// Extract time

hour = formattedDate.substring(splitT+1, formattedDate.length()-1);

Serial.println(hour);

timestamp = day + " " + hour;

}

void setup() {

// Initialize Serial Monitor

Serial.begin(115200);

startLoRA();

connectWiFi();

if(!SPIFFS.begin()){

Serial.println("An Error has occurred while mounting SPIFFS");

return;

}

// Route for root / web page

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/index.html", String(), false, processor);

});

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", temperature.c\_str());

});

server.on("/wheel\_speed", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", wheel\_speed.c\_str());

});

server.on("/fuel\_flow", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", fuel\_flow.c\_str());

});

server.on("/coolant\_temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", coolant\_temperature.c\_str());

});

server.on("/timestamp", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", timestamp.c\_str());

});

server.on("/rssi", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", String(rssi).c\_str());

});

// Start server

server.begin();

// Initialize a NTPClient to get time

timeClient.begin();

// Set offset time in seconds to adjust for your timezone, for example:

// GMT +1 = 3600

timeClient.setTimeOffset(0);

}

void loop() {

// Check if there are LoRa packets available

int packetSize = LoRa.parsePacket();

if (packetSize) {

getLoRaData();

getTimeStamp();

}

}

### Webpage Code

<!DOCTYPE HTML><html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" href="data:,">

<title>ESP32 (LoRa + Server)</title>

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">

<style>

body {

margin: 0;

font-family: Arial, Helvetica, sans-serif;

text-align: center;

}

header {

margin: 0;

padding-top: 5vh;

padding-bottom: 5vh;

overflow: hidden;

background-size: cover;

color: white;

}

h2 {

font-size: 2.0rem;

}

p { font-size: 1.2rem; }

.units { font-size: 1.2rem; }

.readings { font-size: 2.0rem; }

</style>

</head>

<body>

<header>

<h2>ESP32 (LoRa + Server)</h2>

<p><strong>Last received packet:<br/><span id="timestamp">%TIMESTAMP%</span></strong></p>

<p>LoRa RSSI: <span id="rssi">%RSSI%</span></p>

</header>

<main>

<p>

<i class="fas fa-thermometer-half" style="color:#059e8a;"></i> Temperature: <span id="temperature" class="readings">%TEMPERATURE%</span>

<sup>&deg;C</sup>

</p>

<p>

<i class="fas fa-tint" style="color:#00add6;"></i> Wheel Speed: <span id="wheel\_speed" class="readings">%WHEEL\_SPEED%</span>

<sup>&#37;</sup>

</p>

<p>

<i class="fas fa-tint" style="color:#00add6;"></i> Fuel Flow: <span id="fuel\_flow" class="readings">%FUEL\_FLOW%</span>

<sup>&#37;</sup>

</p>

<p>

<i class="fas fa-tint" style="color:#00add6;"></i> Coolant Temperature: <span id=" coolant\_temperature " class="readings">%COOLANT\_TEMPERATURE%</span>

<sup>&#37;</sup>

</p>

</main>

<script>

setInterval(updateValues, 10000, "temperature");

setInterval(updateValues, 10000, "wheel\_speed");

setInterval(updateValues, 10000, "fuel\_flow");

setInterval(updateValues, 10000, "coolant\_temperature");

setInterval(updateValues, 10000, "rssi");

setInterval(updateValues, 10000, "timestamp");

function updateValues(value) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById(value).innerHTML = this.responseText;

}

};

xhttp.open("GET", "/" + value, true);

xhttp.send();

}

</script>

</body>

</html>