LoRa Sender / LoRa Receiver

DuPe (www.dupedup.cz)
Prototype Version: 1/2025-2

Purpose:

Data from sensors are sent as small data packets from LoRa Sender via LoRa radio (433 MHz) to LoRa receiver. There is display on OLED and forwarded (POSTed) via WiFi to WWW server into the CSV file.

Hardware components:

1) **MCU Sender - Arduino Mini Pro 3.3V**
   - CPU ATmega328P / 8-bit Atmel AVR RISC / 8 MHz
   - Flash Program Memory: 32kB / EEPROM: 1kB / SRAM: 2kB
   - Absolute maxiumum VCC: 6V Maximum current for chip: 200mA / Maximum current per pin: 40mA / Recommended current per pin: 20mA
   - ADC: 10-bit PWM: 8bit
   - Power Raw: 5V-16V (6V-12V recommended) / DC VCC: input/ouput 5V , output 3.3V/
   - Maximum current: 150mA @5V

2) **MCU Receiver - Wemos D1 MINI ESP8266**
   - CPU Tensilica L106 32-bit processor 80 MHz
   - 4 MB Flash / 64+96k RAM
   - Operating Current Average value: 80 mA
   - DC VCC 3.3 V
   - 34,2 x 25,6 mm
   - USB CH340

3) **2x LoRa Radio Modul RA-02 Ai-Thinker (Sender + Receiver)**
   - SPI BUS
   - Support FSK, GFSK, MSK, GMSK, LoRa™ and OOK modulation methods
   - Support frequency band 410MHz~525MHz
   - The working voltage is 3.3V, the maximum output is +20dBm, and the maximum working current is105mA
   - It has low power consumption characteristics in the receiving state, the receiving current is 12.15mA,the standby current is 1.6mA
   - High sensitivity: as low as -140dBm
   - Small size double row stamp hole patch package
   - The module adopts SPI interface, half-duplex communication, with CRC, up to 256-byte packet engine

4) **Sensors**
   - Temperature - S18B20 (-55 +125 °C, ±0.2°C) - OneWire , VCC 3.3 - 5V
   - Temperature - ASM2301 /DHT21/ (-40 +80°C, ±0.5°C) - OneWire, VCC3.3 - 5V
   - Temperature / Humdity - SHT40 (-40-125°C ±0.2°C, 0-100% RH) - I2C, VCC 3.3V, 16 bit
   - Pressure / Temperature / Humi - BMP280 (-40 až +85°C ± 0.5°C , 300-1100hPa ± 1.7hPa) - I2C, VCC 3.3V
   - Pressure / Temperature- BME280 (-40 až +85°C + 1°C, 300-1100hPa ± 1hPa, 0 -100% RH) - I2C, VCC 3.3V

5) **2x Display OLED 0.96" or 1.3 " (Sender & Receiver)**
   - OLED 128x64 pixels
   - I2C - controller SSD1306 or SH1106 (the SH1106 controller has an internal RAM of 132x64 pixel, SSD1306 has 128x64 pixel)
   - VCC 3.3-5V / 100mA (max), 34 x 36 mm.

6) **DC - DC Step Down 3.3 V (Receiver)**

7) **LiPo 1S 1000 mAh (Sender)**
8) **LiPo 2 -3S 1000 mAh (Receiver)**

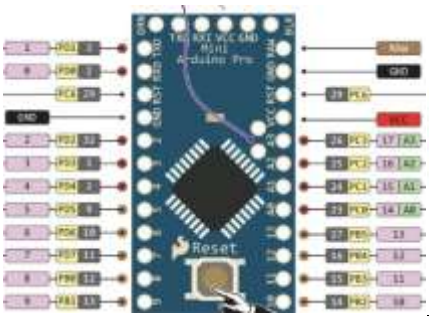
Circuit:

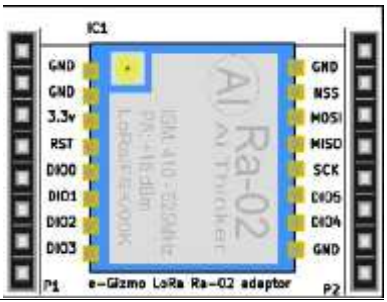
SENDER

Arduino Pro Mini	**RA02** (SPI)	**OLED** (I2C)	**SENSOR** (ONE WIRE)	**Other**
D10 – SS	En/Ns			
D11 – MOSI	MOSI			
D12 – MISO	MISO			
D13 – SCK	SCK			
D2	DIO0			
D9	RST			
A5 – SCL I2C		SCL		
A4 – SDA I2C		SDA		
A0			DATA	
RAW				LiPo 1S
VCC 3.3 V	VCC	VCC	VCC	
GND	GND	GND	GND	

RECEIVER

Wemos D1	**RA02** (SPI)	**OLED** (I2C)	**STEP DOWN** INPUT 6-23V (LiPo 2-3S)
D8/GPIO15	En/Ns		
D7/GPIO13 MOSI	MOSI		
D6/GPIO12 MISO	MISO		
D5/GPIO14 SCK	SCK		
D4/GPI02	DIO0		
D0/GPIO16	RST		
D1/GPIO5		SCL	
D2/GPIO4 SDA		SDA	
WEMOS 3.3 V	VCC	VCC	OUTPUT 3.3V
WEMOS 5V			
GND	GND	GND	GND





Current Consumption:

Receiver: 40 mA
Sender: 10-20 mA

Software Components:

- VSCode + Platform IO + Arduino/ESP8266 Support
- GitHub - sandeepmistry/arduino-LoRa: An Arduino library for sending and receiving data using LoRa radios.
- GitHub - adafruit/Adafruit_SSD1306: Arduino library for SSD1306 monochrome 128x64 and 128x32 OLEDs
- https://github.com/milesburton/Arduino-Temperature-Control-Library
- https://github.com/PaulStoffregen/OneWire
- (GitHub - adafruit/Adafruit_SH110x: Arduino library for SH110x based monochrome OLEDs)
- (GitHub - olikraus/u8g2: U8glib library for monochrome displays, version 2)
- (https://github.com/adafruit/DHT-sensor-library)
- (https://github.com/adafruit/Adafruit_SHT4x)
- (https://github.com/adafruit/Adafruit_BMP280_Library)
- (https://github.com/adafruit/Adafruit_BME280_Library)

Processing:

SENSOR DATA -> BUS -> MCU SENDER -> BUS -> LORA MODEM -> LORA RADIO (UP TO 5km) -> LORA MODEM -> BUS- > MCU RECEIVER -> BUS-> (DISPLAY) -> WIFI - > SERVER HTTP POST -> TXT FILE

SENDER - Code

platformio.ini

```
[env:pro8MHzatmega328]
platform = atmelavr
board = pro8MHzatmega328
framework = arduino
lib_deps =
        paulstoffregen/OneWire@^2.3.8
        adafruit/Adafruit GFX Library@^1.11.11
        sandeepmistry/LoRa@^0.8.0
        milesburton/DallasTemperature@^4.0.4
```

```
/*
LORA SENDER 433MHz + OLED
Arduino Mini Pro 386 3.3V
OLED - SH1106 I2C 0x3C
LORA - SX1278 - Ai-Thinker RA02 - SPI
```

```
SENZOR - DS18B20 ONEWIRE
SPI = En/Nss 10, MOSI 11, MISO 12, SCK 13, G0/DIO0 2, RST 9
I2C = SDA A4, SCL A5
ONE WIRE A0
CO: github.com/sandeepmistry/arduino-LoRa
PD20211020 20250102
*/
#include <Arduino.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SPI.h>
#include <LoRa.h>
#include <OneWire.h>
#include "DallasTemperature.h"
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C
// -- pins used by OLED
#define OLED_SDA A4
#define OLED_SCL A5
// -- pins used by LoRa
#define LORA_SCK 13
#define LORA_MISO 11
#define LORA_MOSI 12
#define LORA_CS 10
#define LORA_RST 9
#define LORA_DIO0 2
#define BAND 433E6
#define BAUD 9600
#define txPower 17
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
#define ONE_WIRE_BUS A0
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensor(&oneWire);
int count = 0;
int packet = 0;
String message = "OK";
String payload ="";

void setup_LORA() {
  LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0);
  if (!LoRa.begin(BAND)) { message = "LoRa Failed";}
  //LoRa.setSpreadingFactor(10); // 6-12 dft 7
  LoRa.setTxPower(txPower); //17
  // LoRa.setSignalBandwidth(signalBandwidth);  //62.5E3 125E3 250E3
  // LoRa.setFrequency(frequency); // 433E6
  // LoRa.setPreambleLength(preambleLength); // 6-65535 dft 8
  // LoRa.setCodingRate4(codingRateDenominator); // 5, 8 dft 5
}
void setup_OLED() {
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS) ;
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setRotation(2);
  display.setCursor(0,0);
  display.display();
}
void setup() {
  Serial.begin(9600);
  sensor.begin();
  setup_OLED();
  setup_LORA();
```

```
}
void display_oled() {
      count++;
      Serial.println(message);
      Serial.println(payload);
      display.setTextSize(1);
      display.print(payload);
      display.print(" ");
      display.println(message);
      display.display();
      if (!(count % 8)) {
        display.clearDisplay();
        display.setCursor(0,0);
        count=0;
      }
}
void send_dataPacket() {
  packet++;
  //const uint8_t txBuffer[6]={'>','D','A','T','A'};
  payload = "DATA:";
  payload = payload + String(packet);
  LoRa.beginPacket();
  //LoRa.write(txBuffer, sizeof(txBuffer));
  LoRa.print(payload);
  LoRa.endPacket();
}
void send_sensorPacket() {
  packet++;
  sensor.requestTemperatures();
  // Serial.println("DONE");
  float tempC = sensor.getTempCByIndex(0);
  // Check if reading was successful
  if (tempC != DEVICE_DISCONNECTED_C)
  {
    Serial.print("Temperature: ");
    Serial.println(tempC);
  }
  else
  {
    message = "Error: Could not read temperature data";
  }
  int temp = int (tempC * 100);
  payload = String(temp) + "," + String(packet);
  LoRa.beginPacket();
  //LoRa.write(txBuffer, sizeof(txBuffer));
  LoRa.print(payload);
  LoRa.endPacket();
}

void loop() {
    //send_dataPacket();
    send_sensorPacket();
    display_oled();
    delay (5000);
  }
```

RECEIVER - Code

platformio.ini

```
[env:d1_mini_pro]
platform = espressif8266
board = d1_mini_pro
```

```
framework = arduino
lib_deps =
        adafruit/Adafruit GFX Library@^1.11.11
        sandeepmistry/LoRa@^0.8.0
        adafruit/Adafruit SSD1306@^2.5.13

/*
LORA RECEIVER 433 MHz
LORA RA02 + WEMOSD1MINI + OLED + WIFI + WEB POST
OLED SH1106 I2C 0x3C / / LORA SX 1278 (Ai-Thinker RA02)
LORA  MOSI D7/GPIO13, MISO D6/12, SCK D5/14, CS/En/Nss D8/GPIO15, DIO0 D4/GPI02, RST D0/GPIO16
SDA D2/GPIO4, SCL D1/GPIO5,
CO: github.com/sandeepmistry/arduino-LoRa)
PD20211020 20241212
*/

#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <LoRa.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_ADDRESS 0x3C
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET    -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define BAND 433E6
#define LORA_CS 15
#define LORA_RST 16
#define LORA_DIO0 2
#define txPower 17
#define TIMEOUT 5000
// REPLACE xxx
const char* ssid = "xxx“
const char* password = "xxx“ const char* host = "xxx“
const char* url = "xxx“
String message= "LORAOK ";
String message_wifi = "WiFi..";
String rssi = "0";
String snr = "0";
String size = "0";
int packet = 0, nopacket = 0;
int lastpacket = 0, lastnopacket = 0, lasttime =0;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// U8G2_SH1106_128X64_NONAME_F_HW_I2C display(U8G2_R0);

void setup_LORA() {
 LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0);
 if (!LoRa.begin(BAND)) { message = "LORAFAIL";}
 // LoRa.setSpreadingFactor(10); // 6-12 dft 7
 // LoRa.setTxPower(txPower); //17
 // LoRa.setSignalBandwidth(signalBandwidth);  //62.5E3 125E3 250E3
 // LoRa.setFrequency(frequency); // 433E6
 // LoRa.setPreambleLength(preambleLength); // 6-65535 dft 8
 // LoRa.setCodingRate4(codingRateDenominator); // 5, 8 dft 5
}
```

```
void setup_OLED() {
 display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
 display.clearDisplay();
 display.setTextColor(SSD1306_WHITE);
 display.setTextSize(1);
 display.setRotation(2);
 display.setCursor(0,0);
 display.display();
}
void connectToNet() {
 Serial.print("WiFiConnecting:");
 display.print(message_wifi);
 display.display();
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(":");
  }
 Serial.println("WiFi connected");
 message_wifi = WiFi.localIP().toString().c_str();
 Serial.println();
 Serial.println(message_wifi);
 display.println(message_wifi);
 display.display();
}
void setup() {
 Serial.begin(9600);
 // while (! Serial);
 setup_OLED();
 setup_LORA();
 connectToNet();
 lasttime = millis();
}
void display_oled(String display_p) {
    display.clearDisplay();
    display.setCursor(0,0);
    display.print(F("RSS/SNR:")); display.print(rssi); display.print(F("/")); display.println(snr);
    display.print(F("SIZE:")); display.println(size);
    display.print(F("-> ")); display.println(display_p);
    display.display();
}
void sendToWeb(String postData) {
 Serial.print("Connecting: ");
 Serial.println(host);
 if(WiFi.status()==WL_CONNECTED) {
  message_wifi = WiFi.localIP().toString().c_str();
  WiFiClient client;
  HTTPClient http;
  //http.setAuthorization("REPLACE_WITH_SERVER_USERNAME", "REPLACE_WITH_SERVER_PASSWORD");
  const int httpPort = 80;
   if (client.connect(host, httpPort)) {
     Serial.print(postData);
     Serial.print(" --> ");
     Serial.println(host);
     http.begin(client, url);
     http.addHeader("Content-Type", "text/plain");
     int httpResponseCode = http.POST(postData);
     message_wifi = message_wifi + "/" + String(httpResponseCode);
     Serial.println(message_wifi);
```

```
      Serial.println("Closing connection.");
      http.end();
     } else {
      message_wifi = "HOSTFAIL";
      Serial.print(message_wifi);
     }
  } else {
   message_wifi = "WIFIDISCON";
   Serial.println(message_wifi);
  }
}

void loop() {
 char payload[15] = "";
 int packetSize = LoRa.parsePacket();
 if (packetSize) {
  for (int i = 0; i < packetSize; i++) {
   payload[i] = (char)LoRa.read();
   // Serial.print((char)LoRa.read());
  }
  packet++;
  rssi = String(LoRa.packetRssi());
  snr = String(LoRa.packetSnr());
  size = String(packetSize);
  String payloads = String(payload);
  String payloadw = payloads + "," + rssi + "," + snr;
  // long freqErr = LoRa.packetFrequencyError();
  Serial.println();
  Serial.println(payloadw);
  display_oled(payloads);
  sendToWeb(payloadw);
  display.setCursor(0,40);
  display.print("OK/KO: ");
  display.setCursor(40,40);
  display.print(packet);display.print("/");display.println(nopacket);
  display.println(message);
  display.println(message_wifi);
  display.display();
  lasttime = millis();
 }
 else {
  if  ((millis() - lasttime) > TIMEOUT){
  nopacket++;
  Serial.print("NODATA-");
  display.setCursor(0,40);
  display.print("OK/KO: ");
  display.setCursor(100,40);
  display.print("!!!");
  display.setCursor(40,40);
  display.setTextColor(SSD1306_BLACK);
  display.print(lastpacket);display.print("/");display.println(lastnopacket);
  display.setCursor(40,40);
  display.display();
  display.setTextColor(SSD1306_WHITE);
  display.print(packet);display.print("/");display.println(nopacket);
  display.println(message);
  lastnopacket = nopacket;
  lastpacket = packet;
  display.println(message_wifi);
  display.display();
```

```
    lasttime = millis();
   }
  }
}
```