Data LoRa Sender (GPS/SENSOR)

DuPe (www.dupedup.cz)
Prototype Version: 04/2025-4

Purpose:

Send periodically data from data sensor (e.g temperature) or GPS via LoRa Radio 868 MHz to compatible LoRa Receiver. Display sensor/GPS data on the local OLED.

Hardware components

1) **MCU (OLED) LilyGO LoRa32 T3_V1.6 868Mhz** (LoRa32 V2.1_1.6 – LILYGO®)

- Espressif ESP32 microcontroller (two Tensilica Xtensa LX7 cores)
- 240MHz, 512kB on-board static RAM , 4MB flash memory
- SX1276 [868/915Mhz]
- single-band 2.4GHz Wi-Fi / Bluetooth 4.2 Low Energy (BLE)
- integrated 128×64 0.96" OLED display (I2C Controller SH1106) / integrated TF Card Slot
- Reset \ Power switch
- Support USB Micro / Li-Po Battery Dual Power Supply (JST GH 2pin 1.25mm) [USB can power the battery]
- USB- TTL Serial Chip CH9102
- 27 x 65 mm

2) **MCU (variant no OLED) TTGO LoRa32 V1 868Mhz**

- ESP32
- SX1276 [868/915Mhz]
- USB- TTL Serial Chip CH9102
- Support USB Micro / Li-Po Battery Dual Power Supply (JST GH 2pin 1.25mm) [USB can power the battery]

3) **GPS BN-280 - GNSS Module + Antenna (UART)**

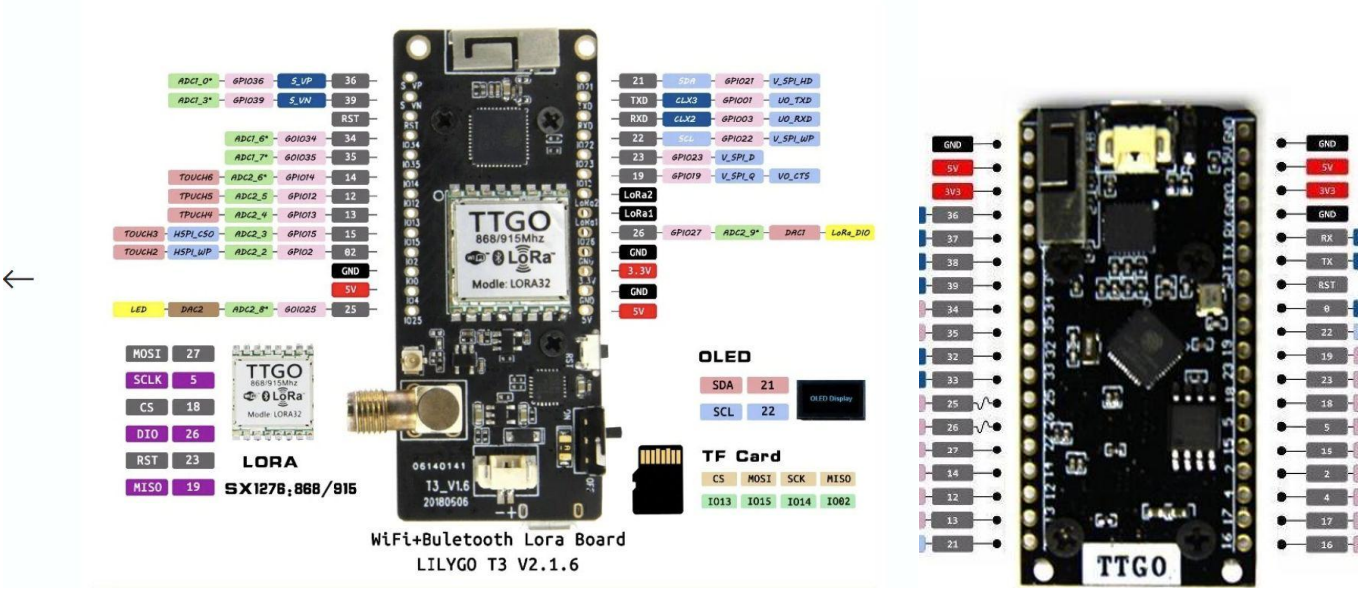
- Chipset M8030-KT
- Frequency GPS L1, GLONASS L1, BDS B1, GALILEO E1, SBAS L1, QZSS L1
- Receiving Format GPS, GLONASS, BDS, GALILEO, SBAS, QZSS.
- Default GPS, GLONASS, SBAS, QZSS.
- Channels 72 Searching Channel
- Sensitivity Tracking -167dBm Reacquisition -160dBm Cold Start -148dBm Hot Start -156dBm
- Accuracy Position Horizontal 2.0 m CEP 2D RMS SBAS Enable (Typical Open Sky) Velocity 0.1m/sec 95% (SA off)
- Timing 1us synchronized to GPS time
- Acquisition Time Cold Start 26s Warm Start 25s Hot Start 1s
- Data Output Support Rate 4800bps to 921600bps, Default 9600bps Data Level TTL Level
- Data Protocol NMEA-0183 NMEA Message RMC, VTG, GGA, GSA, GSV, GLL
- Update Rate 1Hz-10Hz, Default 1Hz
- FLASH 4M FLASH, Store the configuration permanently
- Power Consumption VCC DC Voltage 3.6V-5.5V, Typical: 5.0V
- Current Capture 50mA/5.0V
- Dimension 28mm*28mm*8mm , Weight 11.8g
- Connector 1.25mm 6pins connector
- TX LED: blue. The data output, TX LED flashing
- PPS LED: red. PPS LED not bright when GPS not fixed

4) **Sensor DIGITAL TEMPERATURE/HUMIDITY/ PRESSURE - BMP280**

- Digital interface I^2C (up to 3.4 MHz)
- Supply voltage: 1.71 V to 3.6 V (3.3V DC)
- Operating range Temp: -40 +85°C, + 1°C, 0.01° C
- Operating range Press: 300-1100hPa, ± 1hPa, 0.18Pa

Circuit:

MCU	Sensor Modul	GPS Modul	Other
GPIO 34 - UART RX2		TX	
GPIO 12 - UART TX2		RX	
GPIO 21 – I2C SDA	SDA		
GPIO 22 – I2C SCL	SCL		
GPIO 38 (only TTGO)			Voltage divider 10k/10k
3.3 V	VCC 3.3 V		
5 V		VCC 3.5 - 5V	
Battery Connector JST			1S Lipo / LiION (3.7 - 4,2V)
GND	GND	GND	GND



Current consumption of the device is max 100 mA.

Processing:

Data is periodically readed from temperature sensor or GPS receiver. Temperature data from single sensor measure (connected via I2C bus) is arranged to packet as chars, where first char is ‚&'. MCU between measures (5 minutes) goes to sleep.
If GPS data is received from GPS receiver (UART bus), there is used TinyGPSPLus library to parse GPS data. If GPS is changed (each minut is checked) GPS data are arranged to packet as bytes. First byte is ‚$'. Voltage from GPS is readed on ADC GPIO and send as data packet where first byte is ‚@'.
If is no GPS or no sensor data, dummy packet is sent.
Data (and no data) packet are sent to LoRa modem buffer to send it via LoRa radio to compatible receiver. There is parsed and processed (displayed, forwarded to data server etc).

Software Components:

- C/C++ VSCode / Platform IO
- GitHub - sandeepmistry/arduino-LoRa: An Arduino library for sending and receiving data using LoRa radios.
- GitHub - mikalhart/TinyGPSPlus: A new, customizable Arduino NMEA parsing library
- GitHub - adafruit/Adafruit_BMP280_Library: Arduino Library for BMP280 sensors

platformio.ini (LilyGO T3 + GPS)

```
[env:lilygo-t-display]
platform = espressif32
```

```
board = lilygo-t-display
framework = arduino
lib_deps =
        sandeepmistry/LoRa@^0.8.0
        mikalhart/TinyGPSPlus@^1.1.0
        adafruit/Adafruit SSD1306@^2.5.13
```

platformio.ini (TTGO V1 + SENSOR)

```
[env:ttgo-lora32-v1]
platform = espressif32
board = ttgo-lora32-v1
framework = arduino
lib_deps =
        adafruit/Adafruit BMP280 Library@^2.6.8
        sandeepmistry/LoRa@^0.8.0
```

settings.h (LilyGO T3 + GPS)

```
// pins used by the HW UART2 for GPS
#define RX2 34
#define TX2 12
#define OLED_WIDTH 128
#define OLED_HEIGHT 64
#define OLED_ADDRESS 0x3C
#define SENSOR_ADDRESS 0x77
#define SDAPIN 21
#define SCLPIN 22
#define OLED_RESET 4
#define LORA_SCK 5
#define LORA_MISO 19
#define LORA_MOSI 27
#define LORA_CS 18
#define LORA_RST 23
#define LORA_DIO0 26
#define BAND 866E6
#define txPower 17
#define spreadFactor 7
#define signalBandwidth 125E3
```

settings.h (TTGO V1 + SENSOR)

```
// SETTINGS TTGO-V1
#define SENSOR_ADDRESS 0x77
#define SDAPIN 21
#define SCLPIN 22
#define OLED_RESET 4
#define LORA_SCK 5
#define LORA_MISO 19
#define LORA_MOSI 27
#define LORA_CS 18
#define LORA_RST 14
#define LORA_DIO0 26
#define BAND 866E6
#define txPower 17
#define spreadFactor 7
#define signalBandwidth 125E3
#define pin_BAT 38
```

Source Code (TTGO V1 + SENSOR)

```
// ESP32 TTGO-V1
// SEND LORA - SENSOR DATA
// DUPE 20250403
// CO: https://github.com/sandeepmistry/arduino-LoRa
#include <Arduino.h>
#include <SPI.h>
```

```
#include <LoRa.h>
#include <Wire.h>
#include <Adafruit_BMP280.h>
#include "esp_adc_cal.h"
#include <settings.h>
#define SEA_LEVEL_PRESSURE_HPA      1026.25
#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 120 /* Time ESP32 will go to sleep (in seconds) */
Adafruit_BMP280 sensor;
String message_lora = "L-NO";
String message_sensor = "S-NO";
RTC_DATA_ATTR int sensorPacket, nodataPacket, bootCount = 0;
float t, p, a, vbat;

void setupLORA() {
  Serial.println(F("LORA START"));
  LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0);
  if (!LoRa.begin(BAND)) { message_lora = "L-KO";}
  else {
  message_lora = "L-OK";
  LoRa.setSpreadingFactor(spreadFactor); // 6-12 dft 7
  LoRa.setTxPower(txPower);
  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
  }
  Serial.println(message_lora);
}

void setupSENSOR() {
  Serial.println(F("SENSOR START"));
  //status = sensor.begin(BMP280_ADDRESS_ALT, BMP280_CHIPID);
  //status = sensor.begin();
  if (!sensor.begin()) {
    Serial.println(F("Could not find a valid sensor."));
    Serial.println(F("SENSOR FAILED"));
    message_sensor = "S-KO";
  } else {
    message_sensor = "S-OK";
    sensor.setSampling(Adafruit_BMP280::MODE_NORMAL,
    Adafruit_BMP280::SAMPLING_X2,
    Adafruit_BMP280::SAMPLING_X16,
    Adafruit_BMP280::FILTER_X16,
    Adafruit_BMP280::STANDBY_MS_500);
    //Serial.println(sensor.getStatus());
    //Serial.println(sensor.sensorID());
  }
  Serial.println(message_sensor);
}

void sendLora(String sendData){
  char payload[20];
  uint8_t dataBuffer[20] = {0};
  sendData.toCharArray(payload, 20);
  strcpy ((char*)dataBuffer, payload);
  LoRa.beginPacket();
  LoRa.write(dataBuffer, sizeof(dataBuffer));
  LoRa.endPacket();
}

void displayData(String data) {
  Serial.print(F("DATA-"));
  Serial.println(data);
```

```
  Serial.print(F("Temperature = "));
  Serial.print(t);
  Serial.println(" *C");
  Serial.print(F("Pressure = "));
  Serial.print(p);
  Serial.println(" Pa");
  Serial.print(F("Approx altitude = "));
  Serial.print(a); Serial.println(" m");
  Serial.print(F("Vbat = "));
  Serial.print(vbat); Serial.println(" V");
  Serial.println(message_sensor + " " + message_lora + " VBAT:" + String(vbat));
}

void send_SensorPacket() {
     sensorPacket++;
     //if (sensor.takeForcedMeasurement()) {
     t = sensor.readTemperature();
     p = sensor.readPressure();
     a = sensor.readAltitude(1013.25);
     String payloads = "&";
     payloads = payloads + (sensorPacket+1000); //4B
     int temp = (int)((t+50)*100); //4B
     int press = (int)(p+10000); //6B
     int bat = (int)(vbat*100); //3B
     payloads.concat(temp);
     payloads.concat(press);
     payloads.concat(bat);
     displayData(payloads);
     sendLora(payloads);
   }

void send_NodataPacket() {
  nodataPacket++;
  String payloads = "NODATA-" + String(nodataPacket)+" VBAT:" + String(vbat);
  Serial.println(payloads);
  Serial.println(message_sensor + " " + message_lora + " VBAT:" + String(vbat));
  sendLora(payloads);
}

uint32_t readADC_Cal(int ADC_Raw) {
esp_adc_cal_characteristics_t adc_chars;
esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_12, ADC_WIDTH_BIT_12, 1100, &adc_chars);
return(esp_adc_cal_raw_to_voltage(ADC_Raw, &adc_chars));
}

void setup() {
  Serial.begin(9600);
  delay (1000);
  ++bootCount;
  Serial.println("\nBoot number: " + String(bootCount));
  // esp_sleep_wakeup_cause_t wakeup_reason;
  // Serial.println(esp_sleep_get_wakeup_cause());
  // Wire.begin(SDAPIN, SCLPIN);
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);
  setupLORA();
  setupSENSOR();
  delay(500);
  uint16_t raw = analogRead(pin_BAT);
  vbat= 2.2 * (raw * 3.600 / 4095.0) ;
  //uint16_t raw = (int16_t) (~adc1_get_raw (ADC1_CHANNEL_7) & 0xFFF);
  //vbat = 2 * (raw * 3.600 / 4095.0) ;
  //vbat = (readADC_Cal(ADCvalue))/500;
  (message_sensor == "S-OK") ? send_SensorPacket() : send_NodataPacket();
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
```

```
  Serial.println("Going to sleep now");
  delay(1000);
  Serial.flush();
  esp_deep_sleep_start();
}

void loop() {
  // SLEEPING
}
```

Source Code (LiLyGO T3 + GPS)

```
// LILIGO V3 OLED
// SEND LORA - GPS DATA
// GPS UART RX2-34 TX2-12
// DUPE 20250409-4
// CO: https://github.com/sandeepmistry/arduino-LoRa
#include <Arduino.h>
#include <HardwareSerial.h>
#include <Wire.h>
#include <SPI.h>
#include <LoRa.h>
#include <Adafruit_SSD1306.h>
#include <TinyGPSplus.h>
#include <WiFi.h>
#include <BluetoothSerial.h>
#include <esp_bt.h>
#include <esp_wifi.h>
#include <esp_adc_cal.h>
#include "settings.h"
#define BAUD 9600
#define TIMEOUT 1000
HardwareSerial gpsSerial(2);
Adafruit_SSD1306 display(OLED_WIDTH, OLED_HEIGHT, &Wire, OLED_RESET);
TinyGPSPlus gps;
String message_lora = "L-NO";
String message_gps = "GPS-NO";
int gpsPacket = 0, no_gpsPacket = 0,  gpsUpdate = 0, lasttime = 0, count;
uint32_t latitudeBinary, longitudeBinary;
uint8_t gps_hdop, gps_sat;
uint16_t gps_alt;
double gps_lat, gps_lng;
float vbat;

void setupLORA() {
  Serial.println(F("LORA START"));
  LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0);
  if (!LoRa.begin(BAND)) { message_lora = "L-KO";}
  else {
    message_lora = "L-OK";
    LoRa.setSpreadingFactor(spreadFactor); // 6-12 dft 7
    LoRa.setTxPower(txPower);
    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
     }
  Serial.println(message_lora);
}
void setupOLED() {
  if(!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDRESS)) {
    Serial.println(F("SSD1306 Failed"));
    }
  display.clearDisplay();
```

```
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
}
void disableWiFi_BT(){
  //adc_power_off();
  WiFi.disconnect(true);
  WiFi.mode(WIFI_OFF);
  delay(500);
  btStop();
  esp_bt_controller_disable();
  delay(1000);
}

void gps_print_SERIAL() {
  Serial.print("GPS-PACKETS: "); Serial.print(gpsPacket);
  Serial.print(" / ");Serial.print(gpsUpdate);
       /*
      Serial.print("LAT: ");
      Serial.println(gps_lat, 6);
      Serial.print("LONG: ");
      Serial.println(gps_lng, 6);
      Serial.print("SPEED (km/h) = ");
      Serial.println(gps.speed.kmph());
      Serial.print("ALT (min)= ");
      Serial.println(gps_alt);
      Serial.print("HDOP = ");
      Serial.println(gps_hdop / 100.0);
      Serial.print("Time in UTC: ");
      Serial.println(String(gps.date.year()) + "/" + String(gps.date.month()) + "/" + String(gps.date.day()) + "," +
String(gps.time.hour()) + ":" + String(gps.time.minute()) + ":" + String(gps.time.second()));
      */
  Serial.print("Satellites = ");
  Serial.println(gps_sat);
  }

void gps_print_OLED(){
 display.clearDisplay();
 display.setCursor(0,0);
 display.print("GPS DATA: " + String(gpsPacket));
 display.setCursor(0,10);
 display.print("LAT: " + String(gps_lat, 6));
 display.setCursor(0,20);
 display.print("LONG: " + String(gps_lng, 6));
 display.setCursor(0,30);
 display.print("ALT:  " + String(gps_alt));
 display.setCursor(0,40);
 display.print("Satellites = " + String(gps_sat));
 display.setCursor(0,50);
 display.print(message_lora + message_gps + " BAT:" + String(vbat));
 display.display();
}

void send_nogpsPacket() {
  count++;
  if (!(count % 20)) {
    String payloads = "NOGPS:" + String(count/20) + ":" + String(vbat);
    display.clearDisplay();
    display.setCursor(0,10);
    display.print(payloads);
    display.setCursor(0,50);
    display.print(message_lora + message_gps + " BAT:" + String(vbat));
    display.display();
    Serial.print(payloads);
    LoRa.beginPacket();
```

```
    LoRa.print(payloads);
    LoRa.endPacket();
  }
}

void send_dataPacket() {
  String payloads = "@" + String(vbat);
  Serial.println(payloads);
  LoRa.beginPacket();
  LoRa.print(payloads);
  LoRa.endPacket();
}

void send_gpsPacket() {
  gpsPacket++;
  uint8_t gpsBuffer[10] = {0};
  latitudeBinary = ((gps_lat + 90) / 180.0) * 16777215;
  longitudeBinary = ((gps_lng + 180) / 360.0) * 16777215;
  gpsBuffer[0] = '$';
  gpsBuffer[1] = (latitudeBinary >> 16) & 0xFF;
  gpsBuffer[2] = (latitudeBinary >> 8) & 0xFF;
  gpsBuffer[3] = latitudeBinary & 0xFF;
  gpsBuffer[4] = (longitudeBinary >> 16) & 0xFF;
  gpsBuffer[5] = (longitudeBinary >> 8) & 0xFF;
  gpsBuffer[6] = longitudeBinary & 0xFF;
  gpsBuffer[7] = (gps_alt >> 8) & 0xFF;
  gpsBuffer[8] = gps_alt & 0xFF;
  gpsBuffer[9] = gps_hdop/10 & 0xFF;
  LoRa.beginPacket();
  LoRa.write(gpsBuffer, sizeof(gpsBuffer));
  LoRa.endPacket();
}

void setup() {
  Serial.begin(9600);
  disableWiFi_BT();
  gpsSerial.begin(BAUD, SERIAL_8N1, RX2, TX2);
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);
  setupLORA();
  setupOLED();
}

void loop() {
  uint16_t raw = (int16_t) (~adc1_get_raw (ADC1_CHANNEL_7) & 0xFFF);
  vbat = raw * 3.600 / 4095.0 * 2;
  if (gpsSerial.available() > 0) {
    char gpsData = gpsSerial.read();
    //Serial.write(gpsData);
    gps.encode(gpsData);
    if (gps.location.isUpdated()) {
      gpsUpdate++;
      message_gps = " GPS-OK";
      Serial.print(message_gps);
      Serial.print(gpsUpdate);
      lasttime = millis();
      gps_lat = gps.location.lat();
      gps_lng = gps.location.lng();
      gps_alt = gps.altitude.meters();
      gps_hdop = gps.hdop.value();
      gps_sat = gps.satellites.value();
      gps_print_SERIAL();
      gps_print_OLED();
      if (!(gpsUpdate % 10)) {
        send_gpsPacket();
```

```
        }
       if (!(gpsUpdate % 25)) {
         send_dataPacket();
        }
       }
    }
    else {
     if ((millis() - lasttime) > TIMEOUT) {
       no_gpsPacket++;
       message_gps = " GPS-KO";
       if (!(no_gpsPacket % 1000)) {
         send_nogpsPacket();
        }

     }
   }
}
```