Temperature Sensor (data to Google Sheets)

DuPe (www.dupedup.cz)
Prototype Version: 1.2 (10/2025)

Description:

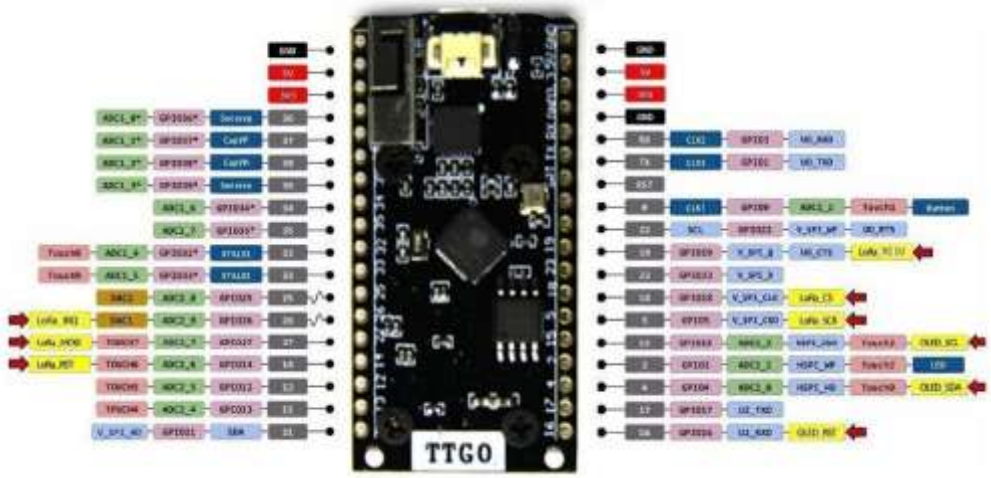
The device measures periodicaly temperature values by a suitable sensor. The device „is sleepig“ between measurements. Data is send via WiFi LAN and Internet to table in the Google Sheets (Internet cloude) and it can be displayed via Google Chart API in browser as the chart. Measurement data is also POST via the WIFI LAN to WEB and store in the CSV file. Data from the CSV may be displayed also in browser via Google Chart API.
Variant: If WIFI is not accessible at sensor device, data from sensor is periodicaly sent via LORA 868 MHz to compatible LORA receiver up to 5km. LORA receiver forwards data from LORA to local WiFi LAN and send to WEB or Goggle Sheets Cloude.

Hardware components:

1) **MCU - TTGO V1 (ESP32 + SX1276 LoRa 868**) (or another ESP32)
   - The main chip using ESPRESSIF ESP32, Tensilica LX6 dual-core processor, clocked at 240MHz, computing power up to 600DMIPS, chip built-in 520 KB SRAM, 802.11 b / g / N HT40 Wi-Fi transceiver, baseband, protocol stack and LWIP, integrated dual-mode Bluetooth (traditional Bluetooth and BLE low power Bluetooth).
   - This product is based on the WI-FI 32 added SX1276 chip, that is, LoRa ™ remote modem, 868/915MHz frequency, about -148dBm high sensitivity, +20 dBm power output, high reliability.
   - Onboard 4M byte(32M Bit) Flash, Wi-Fi antenna, lithium battery charging circuit and interface, CP2102 USB to serial chip
   - Operating voltage: 3.3V to 7V
   - Transmit power: 19.5 dBm @ 11b, 16.5 dBm @ 11g, 15.5 dBm @ 11n

2) **SENSOR DIGITAL PRESSURE AND TEMPERATURE BMP280**
   - Digital interface I^2C
   - Supply voltage : 1.71 V to 3.6 V
   - Current consumption 1.8 µA @ 1 Hz
   - Operating -40 až +85°C, 300-1100hPa
   - Resolution 0.01° C, 0.16Pa
   - Accuracy ± 0.5°C, ± 1.7hPa

3) **Source of Voltage 5V**



Circuit:

MCU	SENZOR	VOLTAGE DIVIDER 0.5 (10kΩ + 10kΩ)	Other
PIN 21	SDA		
PIN 22	SCL		
PIN 38		OUTPUT	
5V		INPUT	
3.3V	VCC (3.3 V)		
USB			Voltage 5V
JST BAT			
GND	GND	GND	GND

Current comsumption during measurement and sending data (WiFi) : 40 – 110 mA
Current comsumption in sleeping mode: < 1mA

Processing:

ANALOG TEMPERATURE SENSOR -> MCU ADC -> MCU CODE (ESP32) -> / LORA RADIO chip (SPI) -> RADIO LORA 868MHz -> LORA TRECEIVER TTGO/ -> WIFI LAN -> DATA POST -> API GOOGLE SHEET -> WWW BROWSER

Software Components:

- C/C++ VSCode / Platform IO + ESP32 Support
- GitHub - mobizt/ESP-Google-Sheet-Client: Arduino Google Sheet REST client library for Arduino
- GitHub - adafruit/Adafruit_BMP280_Library: Arduino Library for BMP280 sensors
- GitHub - sandeepmistry/arduino-LoRa: An Arduino library for sending and receiving data using LoRa radios.

Code:

platformio.ini

```
[env:ttgo-lora32-v1]
platform = espressif32
board = ttgo-lora32-v1
framework = arduino
monitor_speed = 115200
lib_deps =
        adafruit/Adafruit BMP280 Library@^2.6.8
```

mobizt/ESP-Google-Sheet-Client@^1.4.12
sandeepmistry/LoRa@^0.8.0

wifi-settings.h

```
//SETTINGS WIFI 20251010 DUPE
// Google Project ID
#define PROJECT_ID "xxxx"
// Service Account's client email
#define CLIENT_EMAIL "xxxxx"
// Service Account's private key
const char PRIVATE_KEY[] PROGMEM = "-----BEGIN PRIVATE KEY-----\.xxxx\n-----END PRIVATE KEY-----\n";
// The ID of the spreadsheet where you'll publish the data
const char spreadsheetId[] = "xxxxx";
const char* ssid = "xxxx";
const char* password = "….
const char* host = "www.dupedup.cz";
const char* url_data = "http://www.dupedup.cz/....php";
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 3600;
const int   daylightOffset_sec = 0;
```

wifi-settings.h

```
// SETTINGS LORA TTGO-V1
#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
```

MCU code

```
/*
DUPE 20251016 20251021
MEASURE TEMPERATURE & PRESSURE via SENSOR and SEND DATA via WIFI to WEB (POST) and TABLE in
GOOGLESHEETS
VARIANT - SEND DATA via LORA to COMPATIBLE LORA RECEIVER (COMMENT / UNCOMMENT CODE
INSTRUCTIONS)
MCU ESP32: TTGO V1, 240MHz, 320KB RAM, 4MB Flash
I2C: SDA PIN 21, SCL PIN 22
ADC: PIN 38
TEMPERATURE & PRESSURE SENSOR: BMP280 (SPI) Default I2C address: 0x77, Alternate I2C address: 0x76
ESP32 SLEEPING 120 sec between the measurements (BOOT)
CO: https://github.com/mobizt/ESP-Google-Sheet-Client
CO: https://github.com/adafruit/Adafruit_BMP280_Library
CO: https://github.com/sandeepmistry/arduino-LoRa
*/
#include <Arduino.h>
// #include <SPI.h>
// #include <LoRa.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ESP_Google_Sheet_Client.h>
#include <Adafruit_BMP280.h>
#include "esp_adc_cal.h"
#include "time.h"
#include "wifi-settings.h"
//#include "lora-settings.h"
```

```
#define pin_BAT 38
#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) */
#define TYPEBUFFER uint8_t
Adafruit_BMP280 sensor;
bool is_data_sensor = false;
float temp, press, vbat;
unsigned long epochTime;
String DateTime;
String payload;
// int count = 0;
RTC_DATA_ATTR int count, bootCount = 0;

void setup_SENSOR() {
 Serial.print(F("SENSOR START.."));
 //status = sensor.begin(BMP280_ADDRESS_ALT, BMP280_CHIPID);
 //status = sensor.begin();
 if (!sensor.begin()) {
  Serial.println(F("SENSOR FAILED"));
  is_data_sensor = false;
 } else {
  Serial.println(F("SENSOR OK"));
  is_data_sensor = true;
  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());
 }
}

void tokenStatusCallback(TokenInfo info){
 if (info.status == token_status_error){
    GSheet.printf("Token info: type = %s, status = %s\n", GSheet.getTokenType(info).c_str(),
GSheet.getTokenStatus(info).c_str());
    GSheet.printf("Token error: %s\n", GSheet.getTokenError(info).c_str());
 }
 else{
    GSheet.printf("Token info: type = %s, status = %s\n", GSheet.getTokenType(info).c_str(),
GSheet.getTokenStatus(info).c_str());
 }
}

void setup_GOOGLE() {
 // Set the callback for Google API access token generation status (for debug only)
 GSheet.setTokenCallback(tokenStatusCallback);
 // Set the seconds to refresh the auth token before expire (60 to 3540, default is 300 seconds)
 GSheet.setPrerefreshSeconds(10 * 60);
 // Begin the access token generation for Google API authentication
 GSheet.begin(CLIENT_EMAIL, PROJECT_ID, PRIVATE_KEY);
 GSheet.printf("ESP Google Sheet Client v%s\n\n", ESP_GOOGLE_SHEET_CLIENT_VERSION);
}

void connect_WiFi(){
 Serial.print("CONNECTING TO ");
```

```
 Serial.print(ssid);
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(" .. ");
 }
 Serial.print("WIFI CONNECTED OK - ");
 Serial.print("IP: ");
 Serial.println(WiFi.localIP());
}

 String getTime(){
  struct tm timeinfo;
   if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return ("FAIL");
   }
   char timeDat[9];
   char timeTim[6];
   strftime(timeDat,9, "%D", &timeinfo);
   strftime(timeTim,6, "%R" , &timeinfo);
   return String(timeDat) + "|" + String(timeTim);
  }

String prepare_Data() {
    is_data_sensor = true;
    uint16_t raw = analogRead(pin_BAT);
    vbat= 2.2 * (raw * 3.600 / 4095.0) ;
    // if (sensor.takeForcedMeasurement()) {
    temp = sensor.readTemperature();
    press = sensor.readPressure()/100;
    //float a = sensor.readAltitude(1013.25);
    String data = DateTime + ",";
    data = data + String(count) + ",";
    data = data + String(temp, 2) + ",";
    data = data + String(press, 2) + ",";
    data = data + String(vbat, 2);
    Serial.print("PREPARED DATA STRING: ");Serial.println(data);
    return (data);
  }

void send_Web(String postData) {
 Serial.print("POST DATA: ");Serial.println(postData);
 Serial.print("CONNECTING TO: ");Serial.println(host);
 if(WiFi.status()==WL_CONNECTED) {
  WiFiClient client;
  HTTPClient http;
  //http.setAuthorization("REPLACE_WITH_SERVER_USERNAME", "REPLACE_WITH_SERVER_PASSWORD");
  const int httpPort = 80;
   if (!client.connect(host, httpPort)) {
      Serial.println("WEB HOST FAILED");
   return;
   }
  http.begin(client, url_data_sensor);
  Serial.print("SENDING DATA TO: ");Serial.println(url_data_sensor);
  http.addHeader("Content-Type", "text/plain");
```

```
   int httpResponseCode = http.POST(postData);
   Serial.print(String(httpResponseCode));
   Serial.println("  Closing connection.");
   http.end();
  }
  else {
    Serial.println("WIFI-DISCONNECT");
   }
}

void send_GoogleSheet(){
  // For Google Sheet API ref doc, go to
https://developers.google.com/sheets/api/reference/rest/v4/spreadsheets.values/append
  // Append values to the spreadsheet
  Serial.print("GOOGLE DATA: ");

Serial.print(DateTime);Serial.print(",");Serial.print(temp);Serial.print(",");Serial.print(press);Serial.print(",");Serial
.println(vbat);
  FirebaseJson response;
  FirebaseJson valueRange;
  bool google_ready = GSheet.ready();
  if (!google_ready) {
   int ready = 0;
   while (!(google_ready) && ready < 5) {
    ready++;
    Serial.print("GSHEET IS NOT READY - TRYING AGAIN");
    // GSheet.begin(CLIENT_EMAIL, PROJECT_ID, PRIVATE_KEY);
    google_ready = GSheet.ready();
    Serial.println();
    Serial.print(ready); Serial.print(".TRY - STATUS: "); Serial.println(google_ready);
    delay(1500);
   }
  }
  google_ready = GSheet.ready();
  Serial.print("READY STATUS: ");Serial.print(google_ready); Serial.println(is_data_sensor);
  if (google_ready) {
   Serial.println("GSHEET READY - PREPARE DATA SENSOR VALUES ");
   valueRange.add("majorDimension", "COLUMNS");
   valueRange.set("values/[0]/[0]", DateTime);
   valueRange.set("values/[1]/[0]", temp);
   valueRange.set("values/[2]/[0]", press);
   valueRange.set("values/[3]/[0]", vbat);
   Serial.print("APPEND SPREADSHEET VALUES - ");
   bool success = false;
   success = GSheet.values.append(&response, spreadsheetId, "List2!A1", &valueRange);
   if (success){
    Serial.println("APPEND GOOGLE DATA OK");
    response.toString(Serial, true);
    valueRange.clear();
   }
   else{
    Serial.println("APPEND GOOGLE DATA FAILED");
    valueRange.clear();
    Serial.println(GSheet.errorReason());
   }
  }
  ESP.getFreeHeap();
```

```
}

/*
void setupLORA() {
 Serial.println(F("LORA START"));
 LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0);
 if (!LoRa.begin(BAND)) {Serial.println("LORA SETUP FAILED");}
 else {
 Serial.println("LORA SETUP 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
 }
}

void send_LORA(String sendData){
 LoRa.beginPacket();
 LoRa.print(sendData);
 LoRa.endPacket();
}
*/

void setup() {
 Serial.begin(115200);
 ++bootCount; Serial.println("\nBoot number: " + String(bootCount));
 setup_SENSOR();
 //setup_LORA();
 connect_WiFi();
 delay(1000);
 configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
 DateTime = getTime();
 setup_GOOGLE();
 // code from loop if SLEEPING
 count++;
 DateTime = getTime();
 if (is_data_sensor)
 {
  payload = prepare_Data();
  Serial.print("\nMEASUREMENT: "); Serial.print(count);
  Serial.print(" DATA: ");Serial.println(payload);
  send_Web(payload);
  send_GoogleSheet();
  // send_LORA(payload);
  } else {
  payload = DateTime + ",NODATA";
  Serial.println(payload);
  send_Web(payload);
  // send_LORA(payload);
  }
 esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
 Serial.println("\nGoing to sleep now");
 delay(500);
 Serial.flush();
 esp_deep_sleep_start();
```

```
}

void loop() {
//   Serial.println("\n--- WAIT 60s ---");
//   delay(60000);
}
```

HTML code (read data from Gogle Sheet via Google Chart)

```
<!DOCTYPE html>
<html>
 <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
         <meta name="viewport" content="width=device-width, initial-scale=1">
         <title>TEST</title>
        <meta name="description" content="test">
        <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
        <script type="text/javascript">
        // Load the Visualization API
         google.charts.load('current', {'packages':['corechart']});
        // Set a callback to run when the Google Visualization API is loaded.
         google.charts.setOnLoadCallback(drawSheetName);
        // Callback that creates a data table, instantiates the chart, passes in the data and draws it.
        function drawSheetName() {
                var queryString = encodeURIComponent('SELECT A, C');
                var query = new
google.visualization.Query('https://docs.google.com/spreadsheets/d/XXXXXXXXXXXXXX?sheet=List1&headers
=1&tq=' + queryString);
                query.send(handleDataQueryResponse);
           }
        function handleDataQueryResponse(response) {
                if (response.isError()) {
                alert('Error in query: ' + response.getMessage() + ' ' + response.getDetailedMessage());
                return;
                }
        var data = response.getDataTable();
        var chart = new google.visualization.LineChart(document.getElementById('chart_div'));
        var options = {
                        'title':'GoogleSheet - Datalogging ESP32 ',
                'width':1000,
                'height':500,
                        series: {
                 0: {targetAxisIndex: 0}
                 },
                vAxes: {
             0: {title: 'Current [A]', titleTextStyle: {color: 'blue'}}
                        },
                hAxes: {
                        0: {title: 'DateTime ', titleTextStyle: {color: 'grey'}}
                },
                };
        chart.draw(data, options);
        }
  </script>
 </head>
<body>
```

```
        <div id="chart_div"></div>
</body>
</html>
```