Current Sensor (data to Google Sheets)

DuPe (www.dupedup.cz)
Prototype Version: 1.0 - 010/2025

Purpose:

Measure current data up to 30 A by suitable current senzor. Measurement data (mA) is periodicaly sent via WIFI LAN to WEB (PHP code POSTed data writes to CSV file). Data from CSV may be displayed in browser via Google Chart API (JS code using google charts API).
Data is also sends via WiFi LAN and Internet to table in the Google Sheets (Internet cloude) and it can be also displayed via Google Chart API.
Variant – 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)**
   - 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

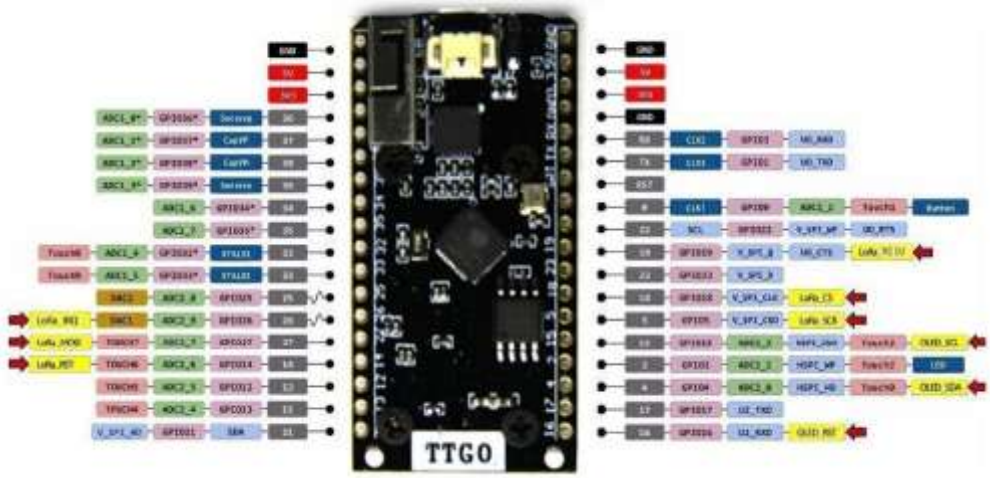
2) **SENSOR - CURRENT ACS712 20A (5A, 30A) - ANALOG**
   - Hall Effect Sensors (is necessary insert to meassured circuit)
   - 5A, 20A, 30A – AC (RMS) or DC bidirectional
   - Analog output: ½ VCC + 66 (resp. 100 resp 185) mV/A
   - VCC 5V

   VARIANT : SENSOR CURRENT WSC1800 - ANALOG
   - Hall Effect Sensors (is not necessary disconnect meassured circuit)
   - 35A DC bidirectional , 25A AC RMS
   - Analog output: ½ VCC + 66 mV/A
   - VCC 3 - 12 V (3mA)

   VARIANT : SENSOR CURRENT INA219 – DIGITAL I2C BUS

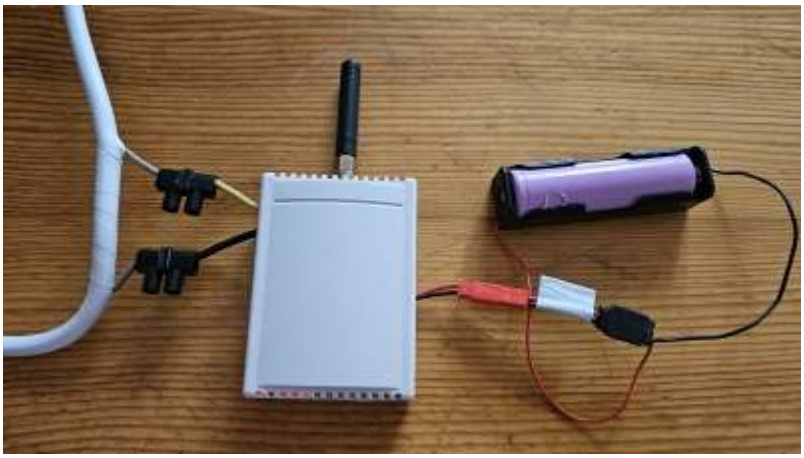
3) **Bat-Boost Converter DIO6605B (Li-Ion -> 5V 0.6A)**
   - Input : 2.7-4.5V
   - Output fix: 5V / 0.6 A (2W)
   - 1.2 MHz
   - 95% (max)
   - 10.2x5.3x2.8mm / 0.6g

4) **Li-Ion1S 2600 mAh 3.7V**



Circuit:

MCU	SENZOR	BAT Boost	Other
PIN 35	OUTPUT		
VCC 5V	VCC 5V	Output 5V	
GND	GND	GND	GND Li-Ion ACU
		Input	3 – 4.2 V Li-Ion ACU



Processing:

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

Current Sensor Analog output: ½ VCC + 66mV/A (5A) , 100mVA (20A), 18 mV/A (30A)

Software Components:

- C/C++ VSCode / Platform IO + ESP32 Support
- GitHub - mobizt/ESP-Google-Sheet-Client: Arduino Google Sheet REST client library for Arduino

Code:

platformio.ini

```
[env:ttgo-lora32-v1]
platform = espressif32
board = ttgo-lora32-v1
framework = arduino
monitor_speed = 115200
lib_deps = mobizt/ESP-Google-Sheet-Client@^1.4.12
```

net_settings.h

```
//NET SETTINGS 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;
```

MCU Source Code

```
/*
ESP32 TTGO V1
CURRENT SENSOR and SEND DATA via WIFI to WEB (POST) and GOOGLESHEETS
DUPE 20251015
CURRENT SENSOR ACS712-20A (or WCS1800) connected to ADC pin 35
The ADC of ESP32 value is a 12-bit number, so the maximum value is 4095 (counting from 0).
To convert the ADC integer value to a real voltage you'll need to divide it by the maximum value of 4095
and multiply by the reference voltage of the ESP32 which is 3.3V.
CO: https://github.com/mobizt/ESP-Google-Sheet-Client
*/
#include <WiFi.h>
#include <HTTPClient.h>
#include <ESP_Google_Sheet_Client.h>
#include "time.h"
#include "net_settings.h"
bool is_data_sensor = false;
const uint8_t ADC_Pin = 35;
const uint8_t sensor_C = 120;
unsigned long epochTime;
String DateTime;

void connectWiFi(){
 // Connect to Wi-Fi network with SSID and password
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.mode(WIFI_STA);
 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());
 }

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);
 }

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 send_Web(String D, float U, float I) {
 String str_U = String(U, 2);
 String str_I = String(I, 2);
 String postData = "";
 postData.concat(D);postData.concat(",");
 postData.concat(U);postData.concat(",");
 postData.concat(I);
 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;
   }
  //Serial.print(postData);
  Serial.print("Sending POST DATA to ");Serial.println(url_data_sensor);
  if (is_data_sensor) http.begin(client, url_data_sensor);
  // if (is_data) http.begin(client, url_data);
  http.addHeader("Content-Type", "text/plain");
  int httpResponseCode = http.POST(postData);
  Serial.println(String(httpResponseCode));
  Serial.println("Closing connection.");
  http.end();
```

```
  }
  else {
    Serial.println("WIFI-DISCONNECT");
   }
}

void send_GoogleSheet(String D, float U, float I){
  // 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("\nGOOGLE DATA: "); Serial.print(D);Serial.print("|");Serial.print(U);Serial.print("|");Serial.println(I);
  FirebaseJson response;
  FirebaseJson valueRange;
  bool data_ready = true;
  bool google_ready = GSheet.ready();
  if (google_ready && is_data_sensor) {
   Serial.println("Prepare DATA SENSOR values...");
   valueRange.add("majorDimension", "COLUMNS");
   valueRange.set("values/[0]/[0]", D);
   valueRange.set("values/[1]/[0]", U);
   valueRange.set("values/[2]/[0]", I);
   }
   //Serial.print(google_ready); Serial.println(data_ready);
   if (google_ready && data_ready) {
    Serial.println("Append spreadsheet values...");
    bool success = false;
    if (is_data_sensor) success = GSheet.values.append(&response, spreadsheetId, "List1!A1", &valueRange);
    //if (is_data) success = GSheet.values.append(&response, spreadsheetId, "List3!A1", &valueRange);
    if (success){
     Serial.println("GOOGLE DATA OK");
     response.toString(Serial, true);
     valueRange.clear();
     }
     else{
     Serial.println("GOOGLE DATA FAILED");
     valueRange.clear();
     Serial.println(GSheet.errorReason());
     }
    }
   Serial.println();
   Serial.println(ESP.getFreeHeap());
  }

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);
}

  float get_AC(){
    is_data_sensor = true;
    float result;
    int readValue;
```

```
    // value read from the sensor
    int maxValue = 0;
    // store max value here
    int minValue = 4095;
    // store min value here
    uint32_t
    start_time = millis();
    while((millis()-start_time) < 1000) //sample for 1 Sec
    { readValue = analogRead(ADC_Pin);
     // see if you have a new maxValue
     if (readValue > maxValue)
     { /*record the maximum sensor value*/ maxValue = readValue; }
     if (readValue < minValue)
     { /*record the minimum sensor value*/ minValue = readValue; } }
     // Subtract min from max
     result = ((maxValue - minValue) * 3300)/4095.0;
     float VRMS = (result/2.0) * 0.707;
     return VRMS;
   }

  float get_DC() {
  int sensor_offset = 2500;
  int samples = 100;
  float value = 0;
  float adc_C = 3300.0/4095.0;
  for (int i = 0; i < samples; i++) {
   //value = value + sq(adc_C * (analogRead(ADC_Pin)-sensor_offset));
   value = value + analogRead(ADC_Pin);
   delay(5);
  }
  //value = sqrt(value/samples);
  //value = value * 0.707;
  value = (adc_C * value)/samples;
  value = value - sensor_offset;
  return (value);
}

void setup() {
  Serial.begin(115200);
  connectWiFi();
  delay(100);
  setup_GOOGLE();
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void loop() {
  Serial.println();
  DateTime = getTime();
  float voltage_AC = get_AC();
  float current_AC = voltage_AC/sensor_C;
  float voltage_DC = get_DC();
  float current_DC = voltage_DC/sensor_C;
  Serial.print("Voltage_AC[mV]: ");Serial.print(voltage_AC); Serial.print("  Current_AC[A]: ");
Serial.println(current_AC);
  Serial.print("Voltage_DC[mV]: "); Serial.print(current_DC);Serial.print("  Current_DC[A]: ");
Serial.println(current_DC);
  send_GoogleSheet(DateTime, voltage_AC, current_AC);
```

```
 send_Web(DateTime, voltage_AC, current_AC);
 delay(10000);
}
```

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>