VIRTUAL INSTRUMENT IN LABVIEW FOR DISPLAYING AND TRANSMITTING DATA FROM SENSORS

Viorel-Mihai Diaconu

Table of figures:

Table of contents:

1. Intruduction

Air quality has become a major global concern, with the World Health Organization estimating that over 6.7 million premature deaths were caused by air pollution in 2019 [1]. In this context, monitoring environmental parameters becomes essential both in urban environments and indoors. The present project proposes the creation of a virtual instrument in LabVIEW, connected to an ESP32 microcontroller, that collects data from temperature, humidity and air quality sensors (DHT11 and MQ-135), displays the values in real time and automatically saves them in a Google Spreadsheet for further analysis.

This system combines the affordable hardware capabilities of the ESP32 with the graphical flexibility offered by LabVIEW, providing an efficient solution for rapid prototyping or local environmental monitoring. Integration with Google Sheets also allows the extension of functionality to cloud-based analysis platforms, without the need for complex or expensive infrastructures.

2. Components used

a. ESP32

The ESP32 is a low-cost microcontroller, available starting at around €5-6, making it accessible to the general public. It is a low-power device, making it ideal for IoT applications that require long battery life, thanks to power-saving modes such as deep sleep. It integrates Wi-Fi capabilities, allowing you to connect to an existing network or create your own, thus facilitating communication between devices. Most ESP32 boards are equipped with a dual-core processor (32-bit Xtensa LX6), offering high performance. It has a rich input/output interface, including support for capacitive touch, ADC and DAC converters, as well as communication protocols such as UART, SPI, I^{2}C and PWM. The ESP32 is compatible with both the Arduino programming language and MicroPython, thus providing flexibility for developers, whether they are beginners or advanced. [2]

Figure 1: ESP32

b. DHT11

The DHT11 temperature and humidity sensor has a temperature and humidity sensor complex with a calibrated digital signal output. By using the exclusive digital signal acquisition technique and temperature and humidity detection technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive type humidity measurement component and an NTC temperature measurement component and connects to a high-performance 8-bit microcontroller, providing excellent quality, fast response, anti-interference ability and cost-effectiveness. [3]

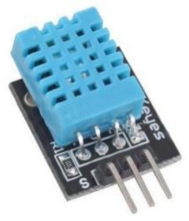

Figure 2: DHT11

c. MQ-135

The MQ-135 sensor is a wide-range gas sensor, sensitive to ammonia (NH_3), nitrous oxide (NO_x), alcohol, benzene, smoke, carbon dioxide (CO_2), and other compounds that affect air quality. Its operating principle is based on a sensitive material made of tin dioxide (SnO_2), whose resistance level decreases in the presence of the target gases, thus generating an electrical signal that can be measured. [4]

Figure 3: MQ-135

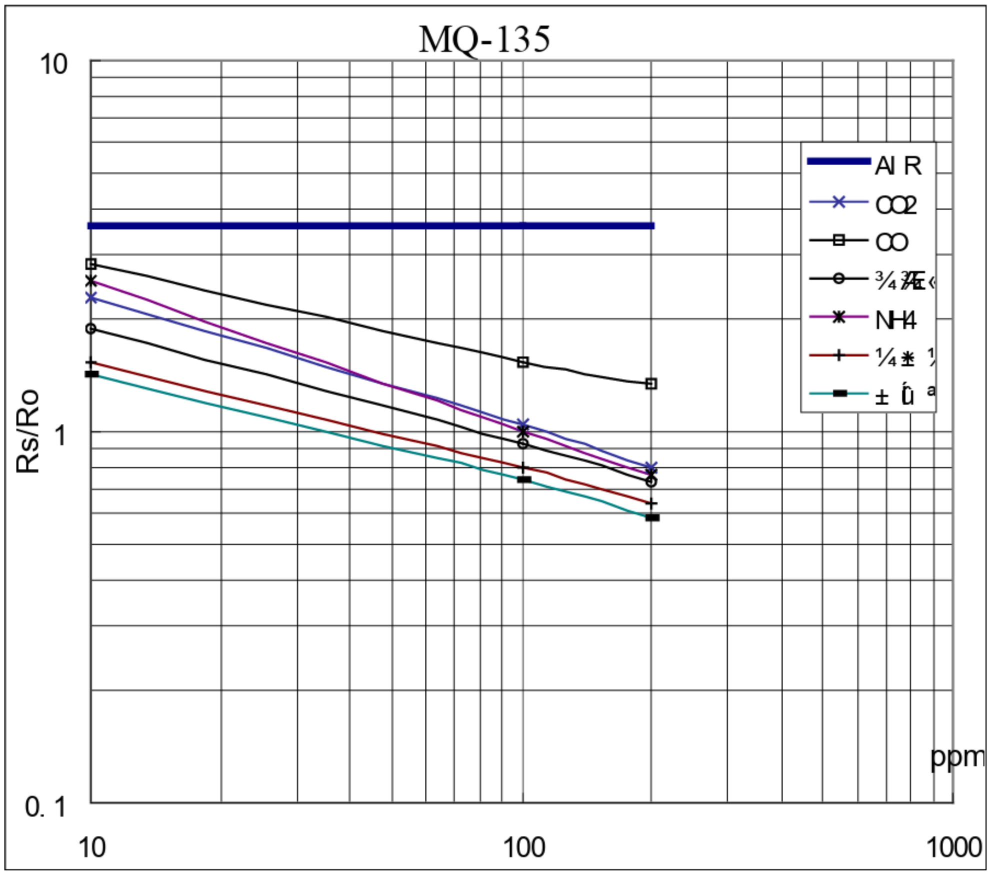

Figure 4: MQ-135 Sensibility

Sensor calibration:

For calibration of the sensor we need the values Rs and R0. To calculate the value Rs we need to know the value of the load resistance Rl whose measured value is 9.89 Kohms and the value of the voltage Vout. These values are inserted into the relationship below.

$$Rs = \frac{(Vin - Vout) * Rl}{Vout}$$

where:

- Vin is the sensor power supply voltage (3.3 V),
- Vout is the measured output voltage,
- Rl is the load resistor (9.89 kΩ).

After determining the Rs value, we can calculate the reference value of R0, needed for estimating the gas concentration. Calibration is done in a clean environment using the formula below:

$$R0 = \frac{Rs}{3.6}$$

3. The measuring system

a. Sensors connection diagram

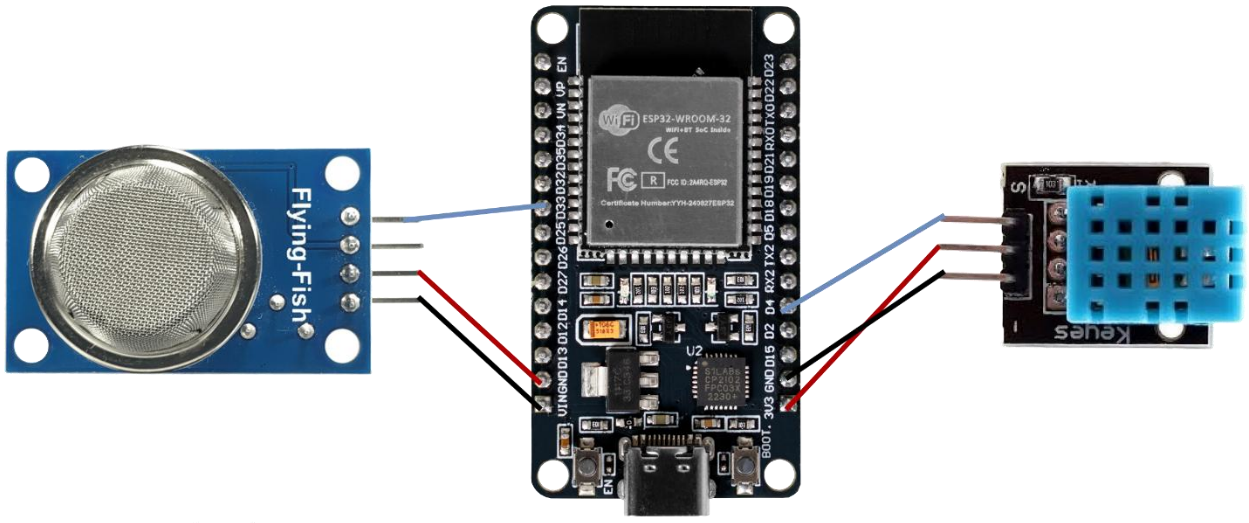

Figure 5: Diagram for sensor connection

The data pin used for DHT11 is D4 and the pin for MQ-135 is D33.

b. Code and libraries used:

```
#include <DHT.h>
#define PPM_PIN 33       // Pin connected to MQ-135
#define DHTPIN 4         // Pin connected to DHT11
#define DHTTYPE DHT11    // DHT 11 sensor
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  analogReadResolution(12);
  dht.begin();
}
void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  float RL = 9.89; // 9.89 kohm
  float R0= 4.98;
  int analogValue = analogRead(PPM_PIN);
  float Vout = analogValue * (3.3 / 4095.0);
  float Rs = ((3.3 - Vout) * RL) / Vout;
  // float R0 = Rs / 3.6
  float ratio = Rs / R0;
 //https://davidegironi.blogspot.com/2014/01/cheap-co2-meter-using-mq135-sensor-with.html
  float ppm = 116.6020682 * pow(ratio, -2.769034857);
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Error reading DHT11!");
  } else {
    Serial.print(ppm);
    Serial.print(",");
    Serial.print(temperature);
    Serial.print(",");
    Serial.println(humidity);
  }
  delay(500);
}
```

The MQ-135 sensor provides an analog signal proportional to the concentration of pollutant gases. It is connected to one of the analog-to-digital converter (ADC) pins of the ESP32, and the value is then read with the analogRead() function. Thus, the output voltage on the sensor pin (in the range of 0–3.3V) is converted into a 12-bit digital value (0–4095). This value is proportional to the detected gas concentration. The following calculation relationship is used to determine PPM. [5]

$$PPM = 116.6020682 * \frac{Rs}{R0}^{-2.769034857}$$

The DHT11 sensor provides digital temperature and humidity data via a specific protocol over a single wire. To communicate with the sensor, the ESP32 program uses the DHT.h library, which automatically handles sending the initialization signal, waiting for the response, and decoding the 40-bit packet sent by the sensor.

After reading, the temperature (in °C), relative humidity (in %) and PPM values are transmitted serially delimited by commas and retrieved into Labview.

4. Spreadsheet and appscript

After the data is retrieved from the sensors via the serial interface and processed in LabVIEW, it is automatically sent to a Google Spreadsheet, where it is stored for further analysis. This allows for a history of the measured values (temperature, humidity, ppm), as well as their visualization and analysis in an easily accessible way from the cloud. To enable automatic writing of data to a spreadsheet, it is necessary to configure an Apps Script, i.e. a small JavaScript program attached to the Google Sheets document, which defines how and where the data is inserted.

The process includes the next steps:

- Creating a new Google Sheets file;
- Opening the App Script editor (Extensions > Apps Script);
- Adding code that defines a doPost(e) function to transmit data via the HTTP POST method and a doGet(e) function to retrieve the data in JSON form and display it on a graph;
- Publish the script as a Web app (Deploy > Manage deployments), with permission „Anyone with the link”.

```
function doGet(e) {
  return handleRequest(e);
}
function doPost(e) {
  return handleRequest(e);
}
function handleRequest(e) {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();

  if (e.parameter.action == "read") {
    var data = sheet.getDataRange().getValues();
    return ContentService
      .createTextOutput(JSON.stringify(data))
      .setMimeType(ContentService.MimeType.JSON);
  } else {
    var ppm = parseFloat(e.parameter.ppm);
    var temp = parseFloat(e.parameter.temp);
    var hum = parseFloat(e.parameter.hum);
    sheet.appendRow([new Date(), ppm, temp , hum]);
    return ContentService.createTextOutput("Success");
  }
}
```

5. Labview

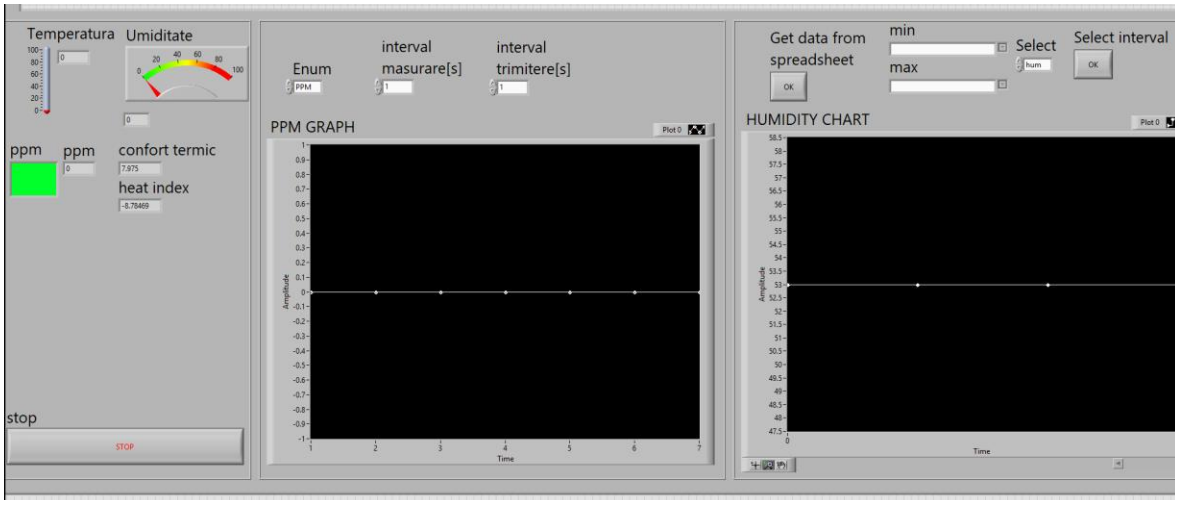

Figure 6: Front panel

The structure of the LabVIEW application is divided into 4 while loops that run in parallel, each having a well-defined role in the data acquisition, processing and storage process. The loops communicate with each other through queues, ensuring a continuous and orderly flow of information.

a. Reading from the serial port

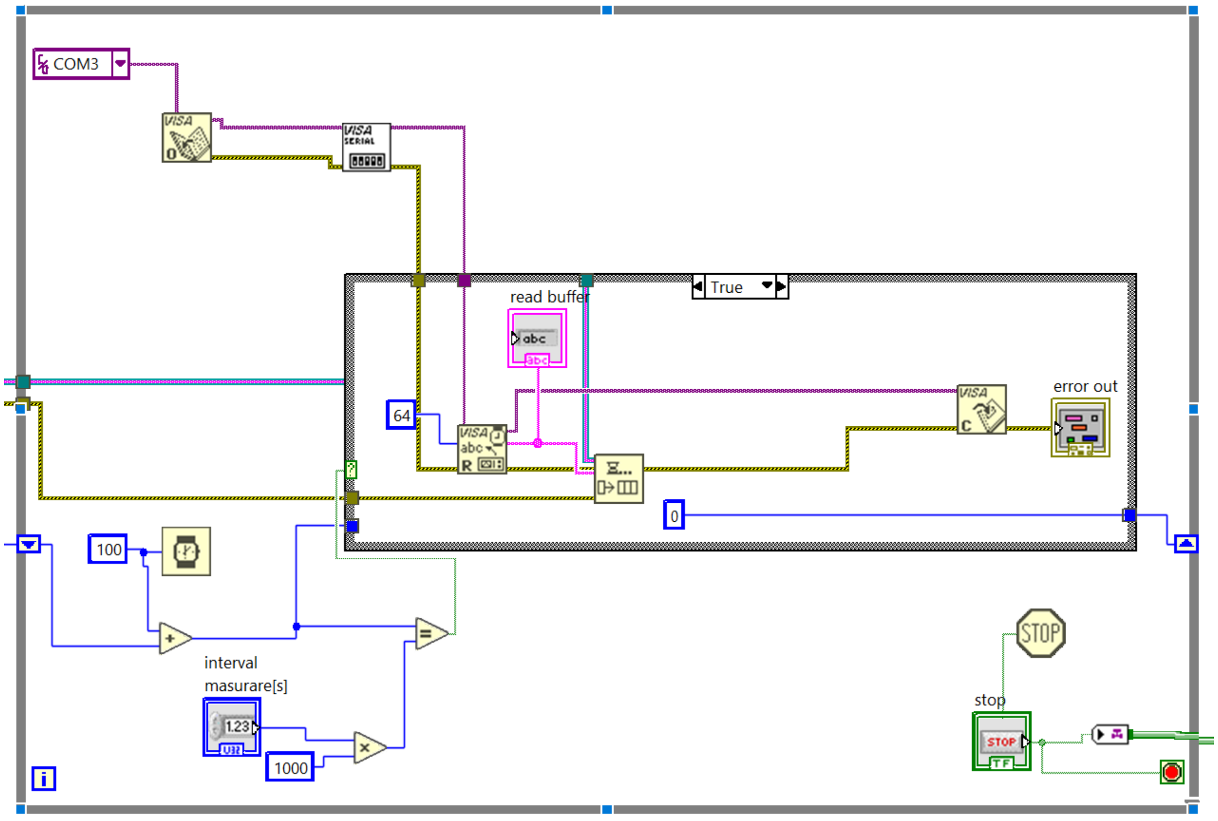

Figure 7: Loop for serial reading

In the first loop, the data transmitted by the ESP32 via the serial port is acquired. The DHT11 and MQ-135 sensors send the measured values in a text format, which is read, stored and sent further for processing.

b. Processing and displaying data

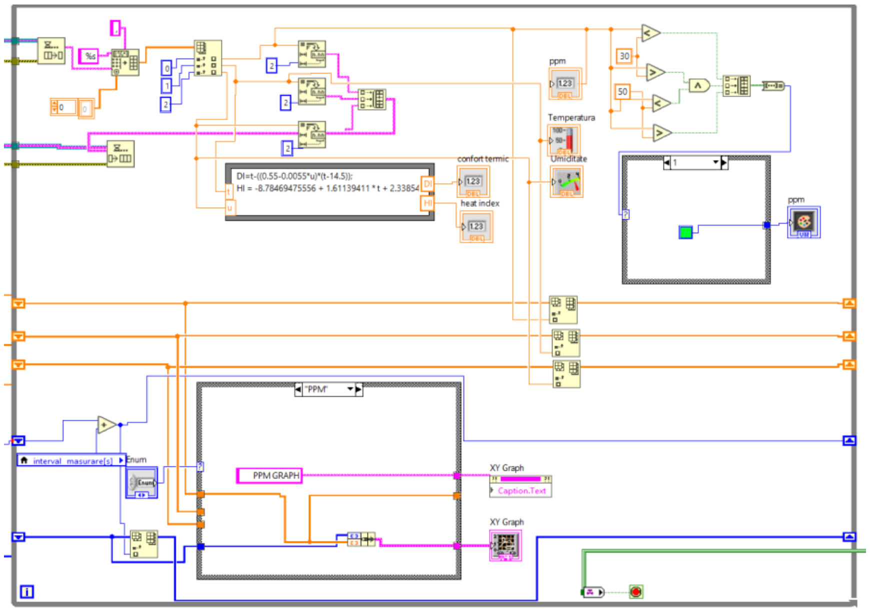

Figure 8: Loop for processing and displaying data

The second loop receives the data from the first loop through a queue, separates them (temperature, humidity, air quality), sends them to the second queue, then displays them both numerically and graphically (on waveform graphs) on the front panel. Here the thermal comfort index and the heat index in degrees Celsius are also calculated. This real-time presentation allows for easy visual analysis of the measured values.

c. Transmitting data to Google Spreadsheet

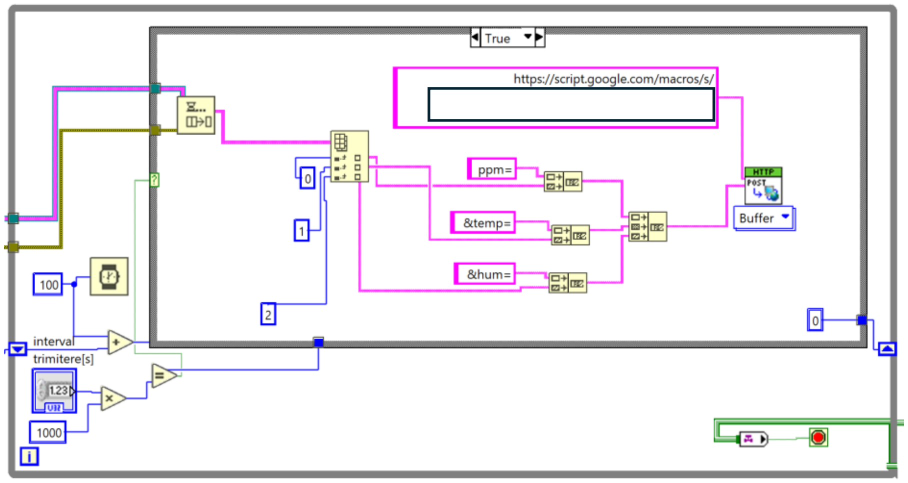

Figure 9: Loop for transmitting data to Spreadsheet

The third loop is to retrieve the data from the second queue and periodically send it to a Google Spreadsheet, using the HTTP POST method. This allows the data to be stored in an accessible and manageable format, as well as possible later analysis in Google Sheets.

d. Reading from Spreadsheet and displaying on selected ranges

The last loop is responsible for reading the values stored in the Spreadsheet. The user can select a time period (start and end date), and the application will display the corresponding values only for that interval, thus allowing a historical and comparative view of the data. This loop is divided into two case-structures.

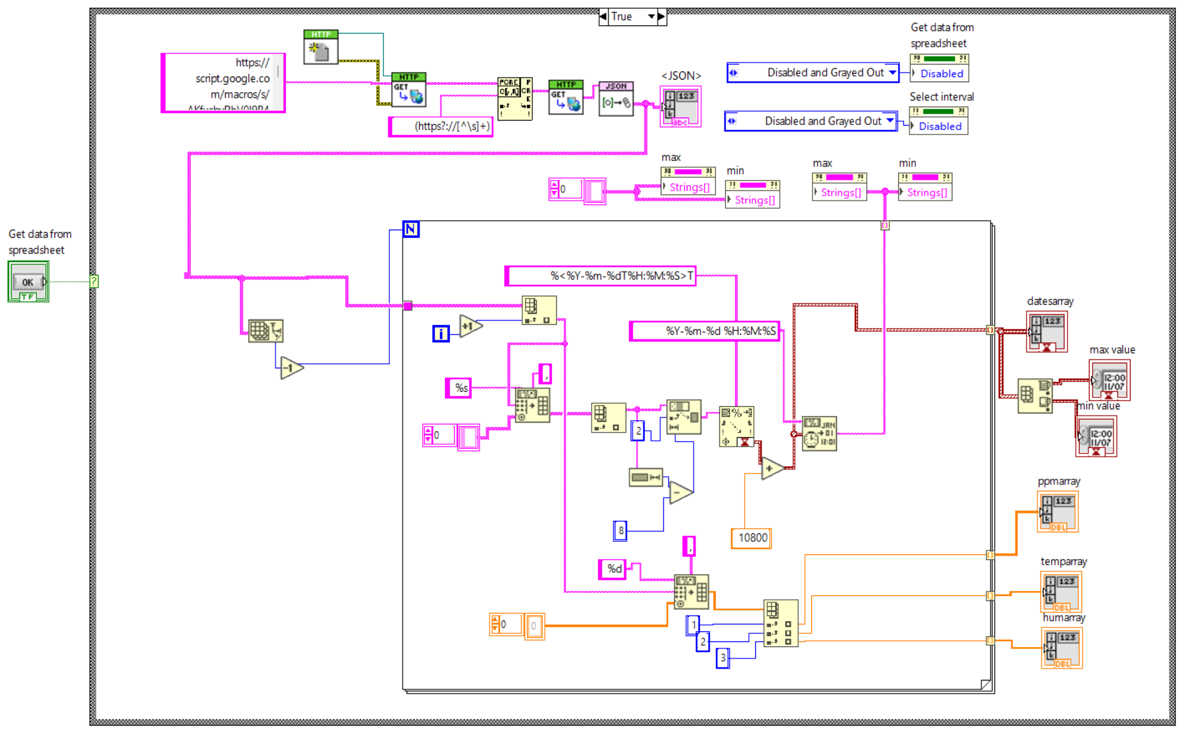

Figure 10: Data reading structure from Spreadsheet

The first case structure takes care of retrieving data from the spreadsheet and storing it in arrays to make it available to the second case structure.

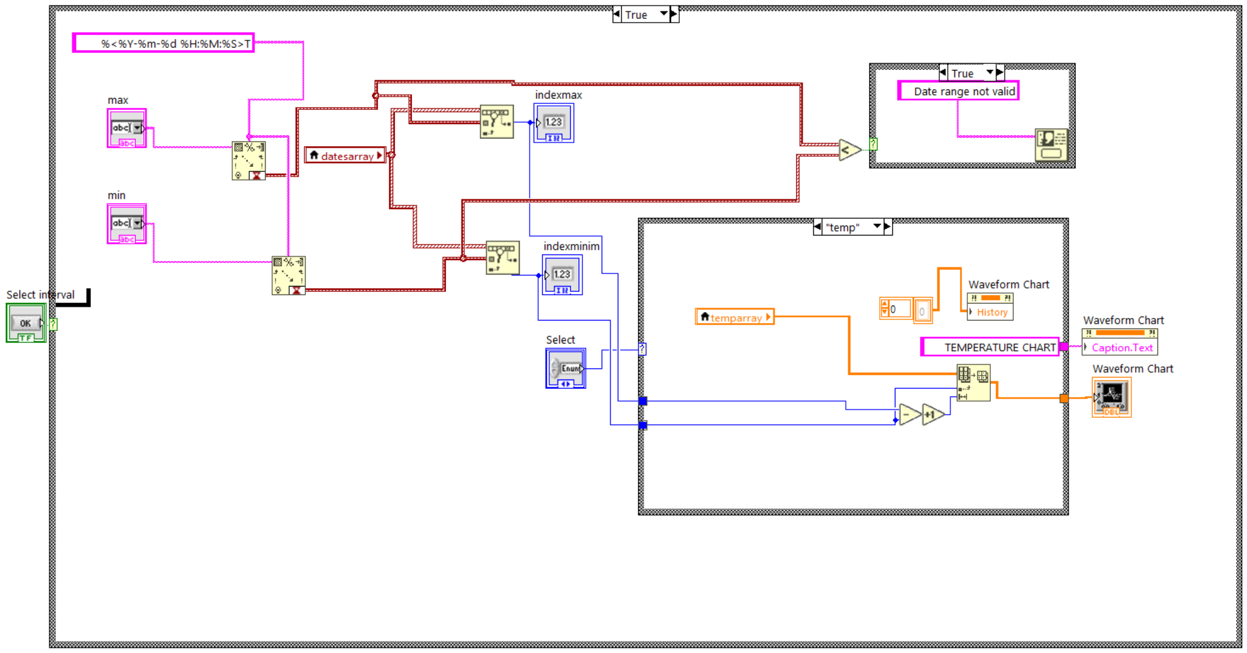

Figure 11: Data displaying structure from Spreadsheet

The second case-structure deals with displaying the data on the graph. When the interval selection button is pressed from the arrays where the data was stored, a subarray is retrieved that contains only the corresponding values from the selected time interval.

6. Bibliography

[1] World Health Organization, *Air pollution*, 2023. https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health/

[2] https://randomnerdtutorials.com/getting-started-with-esp32/

[3] https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[4] https://www.winsen-sensor.com/d/files/PDF/Semiconductor%20Gas%20Sensor/MQ135%20(Ver1.4)%20-%20Manual.pdf

[5] https://davidegironi.blogspot.com/2014/01/cheap-co2-meter-using-mq135-sensor-with.html