# INTELLIGENT TEMPERATURE AND HUMIDITY MANAGEMENT SYSTEM

L'industry 4.0

2024-2025

GASMI RIFKA & DRIDI DHIA & JLASSI IHEB
Master RAIA2

## Summary

# List of Figures

# Intelligent temperature and humidity management system with LabVIEW and a DHT11 sensor

## I. Introduction

The objective of this project is to monitor and control the environmental parameters in a home using a **DHT11 sensor**, a microcontroller and **LabVIEW**. The thresholds set for temperature and humidity activate a fan and an air conditioner respectively. In this prototype, these devices are simulated by LEDs in LabVIEW. In addition, all collected data is saved in an **Excel CSV file and then in a cloud** for further analysis.

## II. Objectives

1. **Monitoring** : Read and display temperature and humidity in real time.
2. **Control** :
   - Activate a fan (LED) when the temperature exceeds **25°C**.
   - Activate an air conditioner (LED) when humidity exceeds **50%.**
3. **Recording** : Save environmental data (temperature, humidity, LED status) in a CSV file.

## III. Theoretical part

### 1. Hardware and Software Components

**Hardware:**

- **DHT11 Sensor** : Measures temperature and humidity.
- **Microcontroller** (Arduino or other): Data collection and communication with LabVIEW.
- **LEDs** : Simulate the status of the fan and air conditioner.
- **Resistors** : Protect the LEDs.
- **Wiring** : Connections between components.
- **PC** : Running LabVIEW.

**Software:**

- **LabVIEW** : For GUI design, data acquisition and processing, and saving results to a CSV file.
- **Arduino Library** : Communication between LabVIEW and Arduino.

## 2 . Methodology

**Data acquisition:**

- Connect the **DHT11** sensor to the microcontroller to read temperature and humidity measurements.
- Use a serial protocol to transmit data from the microcontroller to LabVIEW.

**Processing and display in LabVIEW:**

- Display the data received in real time on a graphical interface.
- Add graphs to visualize the evolution of temperature and humidity.

**Control logic:**

- Implement the following conditions in LabVIEW:
    - **Temperature> 25°C** : Activate the LED simulating the fan.
    - **Humidity > 50%** : Activate the LED simulating the air conditioner.
- Transmit on/off signals to the microcontroller.

**Data logging:**

- Create a CSV file in LabVIEW.
- Add the following entries to the file every second:
    - Timestamp.
    - Temperature.
    - Humidity

# IV. Technical part

## 1 . Arduino and LabVIEW communication:

For our project, we used a LabVIEW interface, an Arduino Uno board, a DHT11 sensor, and two LEDs. To establish communication between LabVIEW and Arduino, we installed the LabVIEW Interface for Arduino Toolkit via the VI Package Manager (a tool for adding libraries and functionality to LabVIEW). This configuration is shown in the following figure.

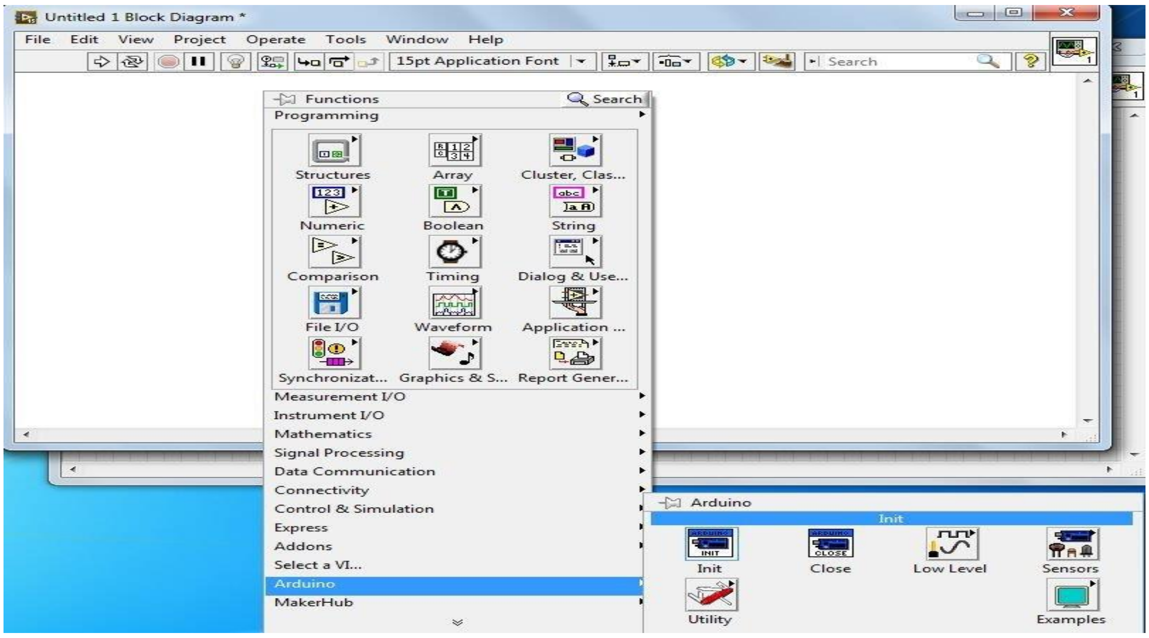


Figure 1:Interface for Arduino Toolkit via le VI Package

After installation, we uploaded the LIFA codebase, but we encountered a problem in the code, a defect related to the NI developers. The problem lies in declaring the checksum variable, which should be set as *"unsigned char checksum = 0;"* instead of just *"unsigned char checksum"*.

```
unsigned char checksum_Compute(unsigned char command[])
{
  unsigned char checksum=0;
  for (int i=0; i<(COMMANDLENGTH-1); i++)
  {
```

Table 1: functioning of the deferents blocks

| The function | Description |
|---|---|
| Init.vi<br>ARDUINO<br>INIT | This block initializes the communication between LabVIEW and the Arduino board. It establishes the connection needed to send and receive data between the two systems. |
| Close.vi<br>ARDUINO<br>CLOSE | This block closes the communication with the Arduino board. It is used at the end of the run to ensure that the connection is properly closed. |
| Set Digital Pin Mode.vi<br>ARDUINO<br>I/O | This block allows you to configure a specific pin of the Arduino in input or output mode. It determines whether the pin will be used to read data (input) or to send signals (output). |

| | |
|---|---|
| Digital Write Pin.vi | This block allows you to send a digital signal (up or down) to a pin of the Arduino. It is used to control devices such as LEDs or relays by sending commands of 0 or 1 (off/on) |
| Read Data.vi | This block allows data to be read from a sensor, such as the DHT11. It extracts temperature and humidity information from the sensor and transmits it to LabVIEW. |

## 2. Structure used:

A State Machine on LabVIEW with a While loop is a programming architecture where the program is divided into different states (actions or steps). Each state is represented by a case in a Case Structure, and the While loop is used to repeat execution until a stop condition is met.

**Operation:**

1. While Structure: Continuously repeats logic until an end condition.
2. Case Structure: Contains the different states.
3. Enum or String: A variable (often an Enum or a text string) determines which state should run.
4. Transition: At the end of a state, the program decides which state to run next by updating the state variable.

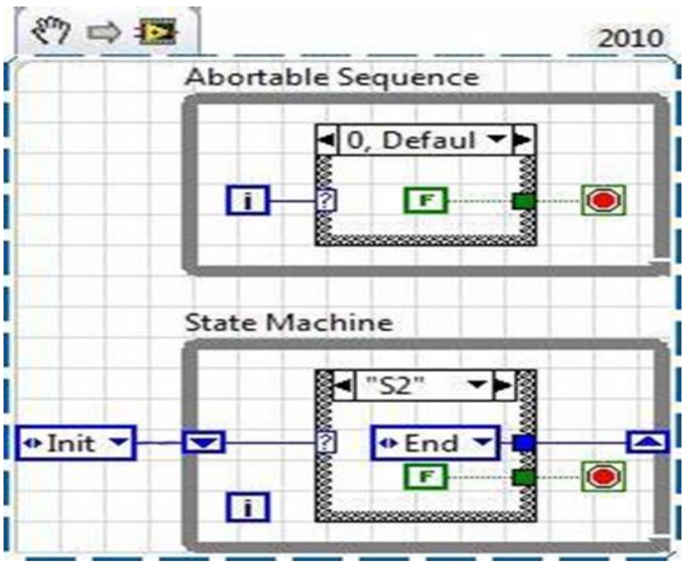


*Figure 2:Machine State example*

For our project, as shown in the following figure, we have defined three **Enums**:

- The first is dedicated to the initialization of all the parameters,
- The second corresponds to the automatic mode,
- And the third in manual mode.

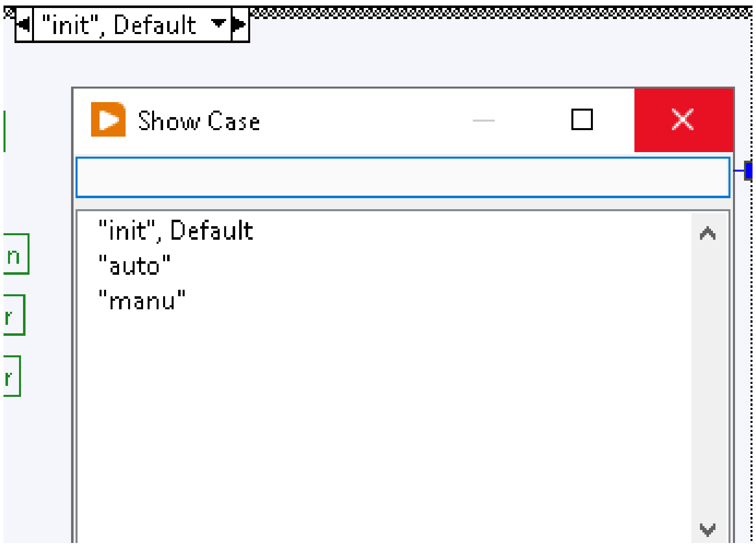


*Figure 3:d different modes*

## 3. Interfaces in LabVIEW

### a. L'interface graphique :

Our interface contains these different components:

1. **Operating status**
   - Status Indicator: Likely displays the current state of the system (started, stopped, or pending). The STOP button appears to be a switch to disable the system.

2. **Sensor and actuator control**
   - DHT11 Sensor: A temperature and humidity sensor that measures these two environmental parameters.
   - Initialization: A button to start or reset the system.
   - Fan: Enables or disables the operation of the fan.
   - Air conditioner: Allows you to activate air conditioners to regulate temperature and humidity.

3. **Humidity measurement**
   - Circular Gauge: Displays the measured value of humidity in percent (%).
   - Scale from 0 to 255: This could indicate that the values are captured as raw (unconverted) sensor data.

4. **Temperature measurement**
   - Linear Slider: Allows you to display or set a target value for temperature.
   - Scale 0 to 255: Indicates raw data similar to humidity.

5. **Operating Modes**
   - AUTO_MANU: A button to switch between automatic (sensor control) and manual (user control) mode.
   - IOT_DATA: Likely related to sending data to an IoT (Internet of Things) platform.

6. **Data visualization section**
   - XY graph: Represents in real time the evolution of parameters (temperature or humidity) over time.
   - Amplitude (Y-axis): Indicates the measured values.
   - Time (X-axis): Shows the duration or intervals of data collection.

7. **Information Submission Section**
   - Email Field: Allows the user to enter an email address to receive information.
   - "Send_info" button: To send the collected data via e-mail.

8. **"VISA resource" selector**
   - Allows you to configure communication with connected devices via a standard protocol (GPIB, USB, or other).

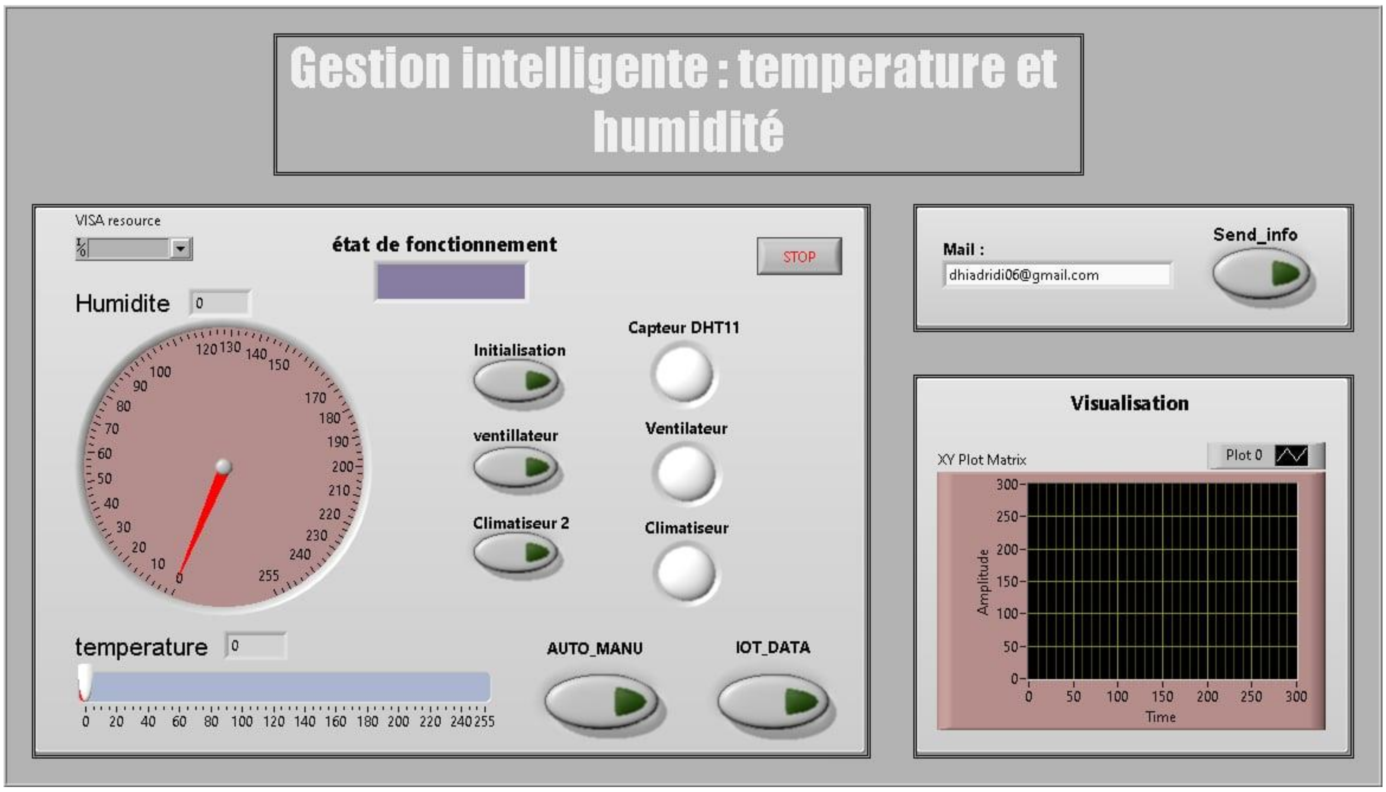


*Figure 4:L'interface graphique*

## b. Operating modes

**Mode init**

In the initialization mode, we make sure to configure and validate all the parameters essential to our project. to ensure data consistency, avoid confusion and prevent loss of information, which contributes to optimal execution of subsequent phases.

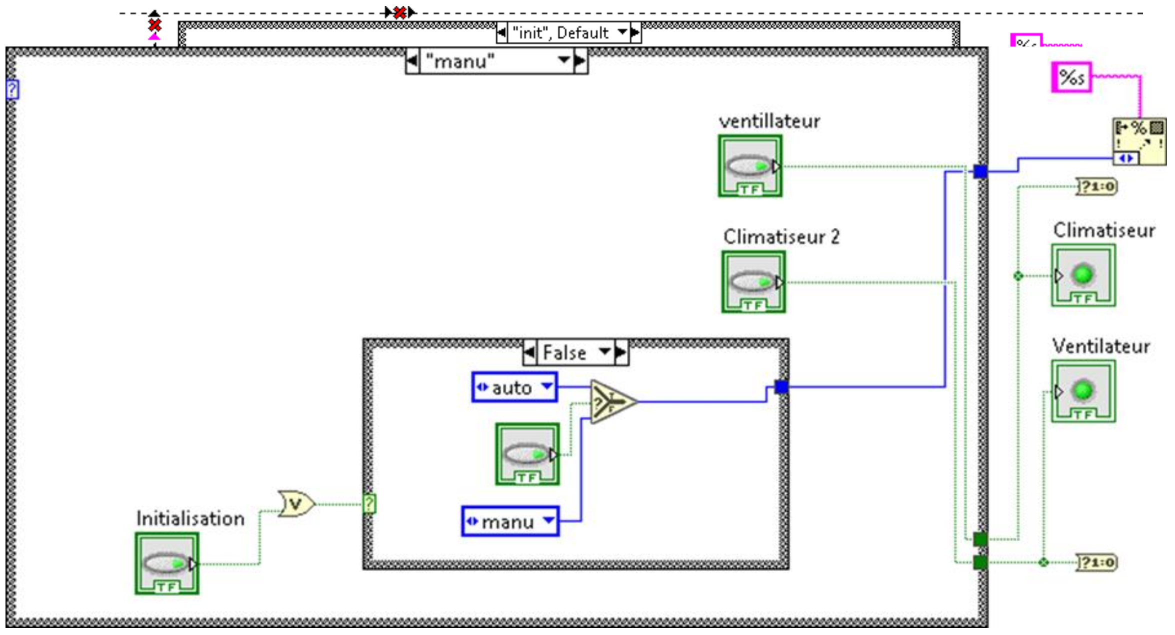


*Figure 6:manual mode*

**Manuel mode:**

In the manual mode, as shown in the following figure, we implemented direct control of the fan and air conditioner using two separate buttons. This mode allows manual management, without the intervention of temperature and humidity sensors.

**Automatic mode :**

In the automatic mode, the system works intelligently, where the DHT11 sensor plays a vital role in providing the temperature and humidity data. This information is sent from Arduino to LabVIEW, and based on the conditions being measured, the system makes automatic decisions to turn the fan and air conditioner on or off, optimizing the environment without manual intervention.

To ensure the performance and accuracy of the system, we have integrated several advanced data management and decision-making algorithms. These algorithms allow a rapid and adequate response to variations in environmental parameters.

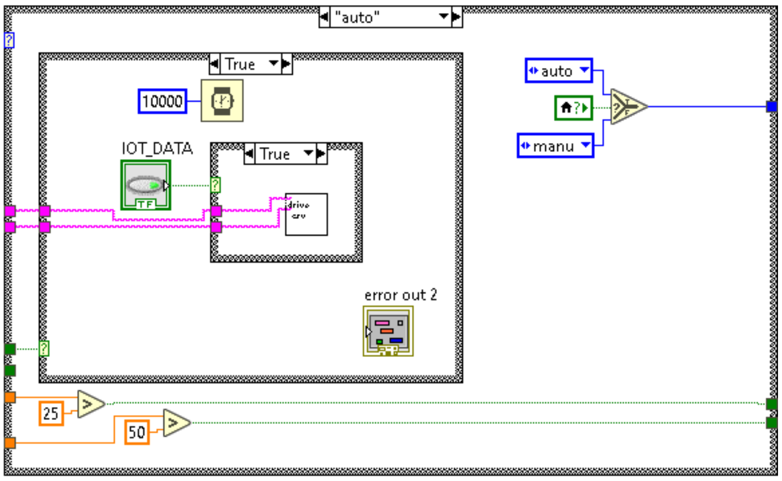
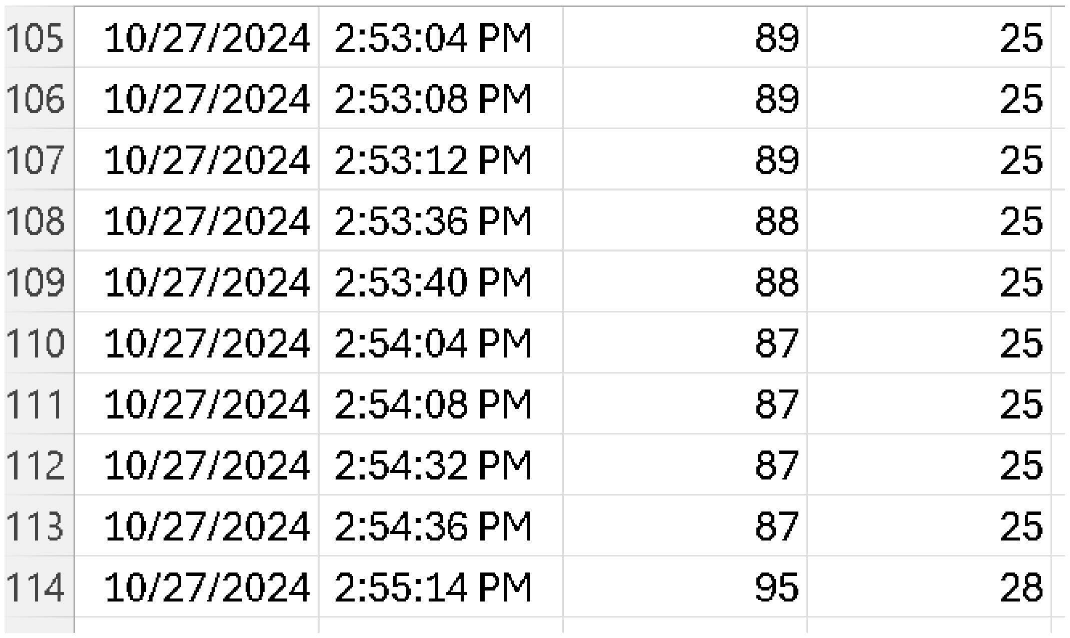


Figure 7:Automatic mode

In this mode, all data captured by the sensors is saved in a CSV file, as shown in image 8. This allows for accurate tracking of data over a period of time.



| | | | | |
|---|---|---|---|---|
| 105 | 10/27/2024 | 2:53:04 PM | 89 | 25 |
| 106 | 10/27/2024 | 2:53:08 PM | 89 | 25 |
| 107 | 10/27/2024 | 2:53:12 PM | 89 | 25 |
| 108 | 10/27/2024 | 2:53:36 PM | 88 | 25 |
| 109 | 10/27/2024 | 2:53:40 PM | 88 | 25 |
| 110 | 10/27/2024 | 2:54:04 PM | 87 | 25 |
| 111 | 10/27/2024 | 2:54:08 PM | 87 | 25 |
| 112 | 10/27/2024 | 2:54:32 PM | 87 | 25 |
| 113 | 10/27/2024 | 2:54:36 PM | 87 | 25 |
| 114 | 10/27/2024 | 2:55:14 PM | 95 | 28 |

Figure 8:data structure save

Then, we use this information to generate a curve representing the variation in temperature as a function of humidity, which is visualized in Figure 9.

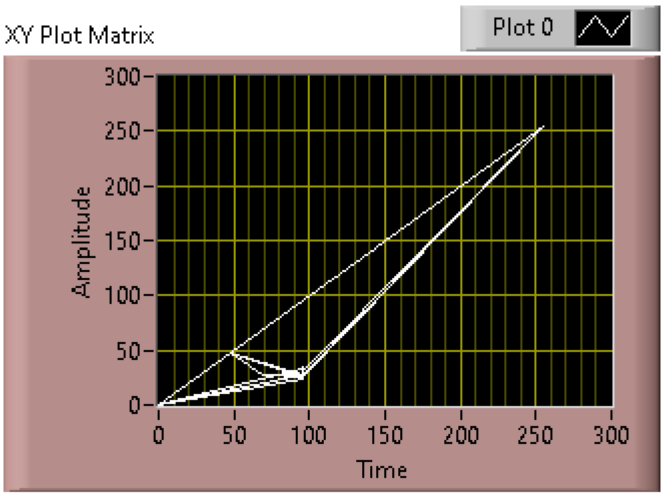
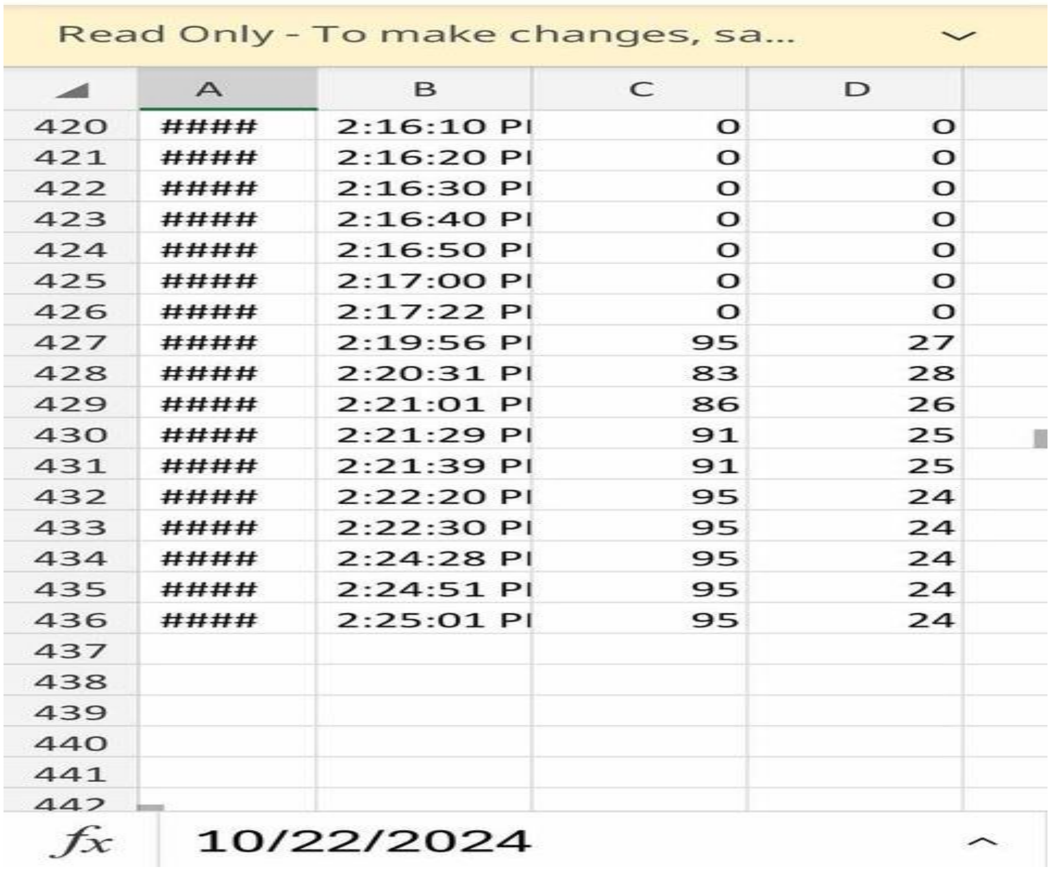


Figure 9:Data Visualization Section

In addition, a key feature of the project is the integration of IoT principles. The data is recorded in real-time and can be viewed remotely, providing continuous monitoring of the system's status. This feature is shown in Figure 10, allowing for remote management and optimized control of the system.



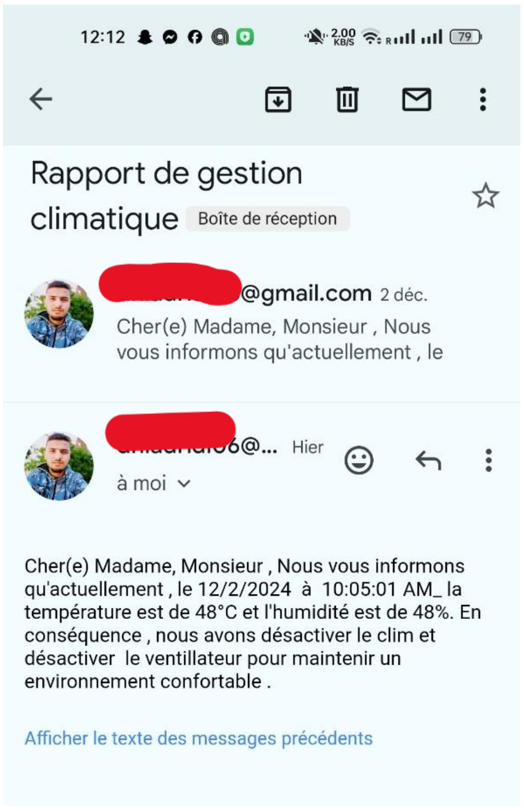
Read Only - To make changes, sa...

| | A | B | C | D |
|---|---|---|---|---|
| 420 | #### | 2:16:10 PI | 0 | 0 |
| 421 | #### | 2:16:20 PI | 0 | 0 |
| 422 | #### | 2:16:30 PI | 0 | 0 |
| 423 | #### | 2:16:40 PI | 0 | 0 |
| 424 | #### | 2:16:50 PI | 0 | 0 |
| 425 | #### | 2:17:00 PI | 0 | 0 |
| 426 | #### | 2:17:22 PI | 0 | 0 |
| 427 | #### | 2:19:56 PI | 95 | 27 |
| 428 | #### | 2:20:31 PI | 83 | 28 |
| 429 | #### | 2:21:01 PI | 86 | 26 |
| 430 | #### | 2:21:29 PI | 91 | 25 |
| 431 | #### | 2:21:39 PI | 91 | 25 |
| 432 | #### | 2:22:20 PI | 95 | 24 |
| 433 | #### | 2:22:30 PI | 95 | 24 |
| 434 | #### | 2:24:28 PI | 95 | 24 |
| 435 | #### | 2:24:51 PI | 95 | 24 |
| 436 | #### | 2:25:01 PI | 95 | 24 |
| 437 | | | | |
| 438 | | | | |
| 439 | | | | |
| 440 | | | | |
| 441 | | | | |
| 442 | | | | |

fx 10/22/2024

Figure 10:remote viewing

**Sends email automatically:**
To ensure the robustness and professionalism of our project, we have developed a communication feature between Python and LabVIEW. This feature allows you to send an automatic email if you need information about the status of the project. When a specific situation arises, an email is automatically generated and sent, as explained in the following example:



Rapport de gestion climatique Boîte de réception

[redacted]@gmail.com 2 déc.
Cher(e) Madame, Monsieur , Nous vous informons qu'actuellement , le

[redacted]6@... Hier
à moi

Cher(e) Madame, Monsieur , Nous vous informons qu'actuellement , le 12/2/2024 à 10:05:01 AM_ la température est de 48°C et l'humidité est de 48%. En conséquence , nous avons désactiver le clim et désactiver le ventillateur pour maintenir un environnement confortable .

Afficher le texte des messages précédents

*Figure 11: example of an email*

This feature, as shown in Figure 12, provides real-time tracking and smooth communication when additional project information is needed.

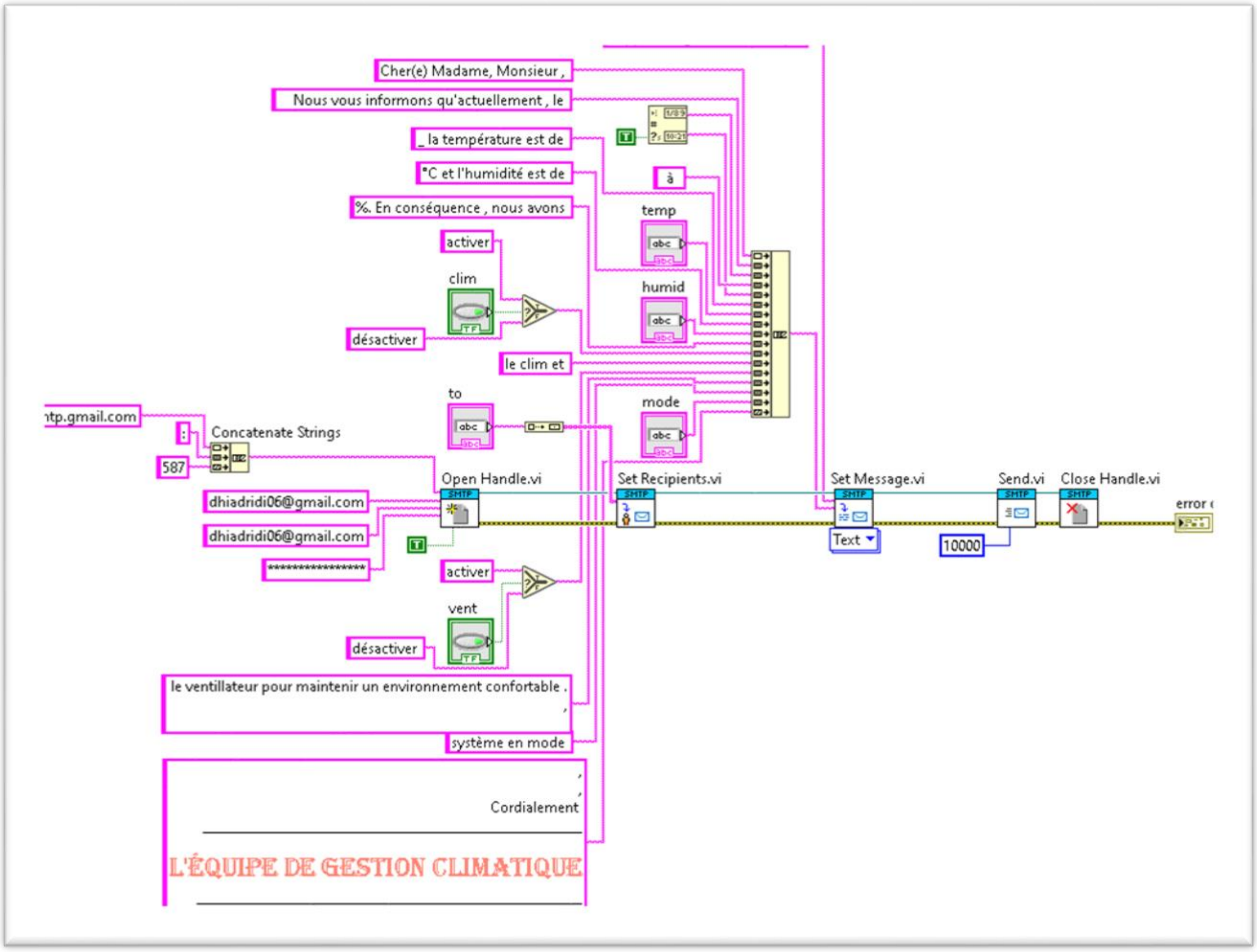


*Figure 12: block for the mail*

## Results achieved

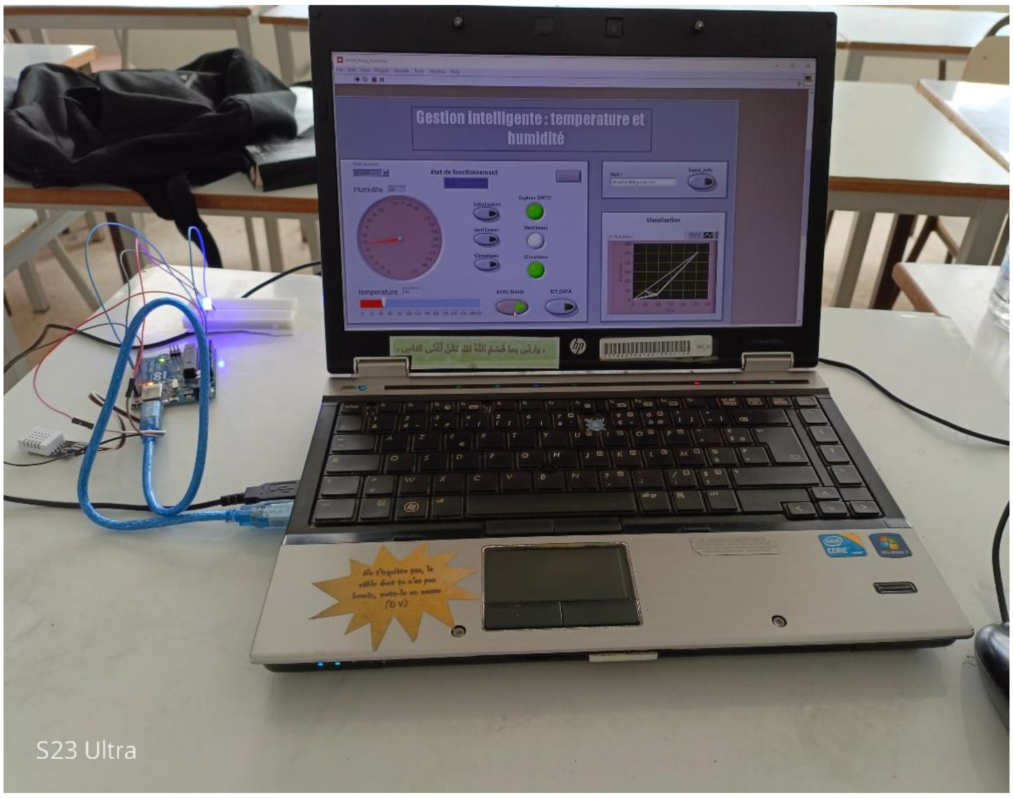


*Figure 13:Result*

## VI. Conclusion

This project offers a practical and cost-effective solution for monitoring and controlling a home's environmental parameters. Using LabVIEW simplifies the development of the user interface, and saving the data to a CSV file allows for later analysis. The prototype can be easily expanded to include other features or sensors.