Smart Temperature and Humidity Monitoring

Willy Isranda Sihombing

Polytechnic State of Batam, Batam 29461, Indonesia

*Email: willyisranda1234@gmail.com

***Abstract*— The temperature and humidity monitor system is an important parameter of room air due to temperature. Humidity and temperature has a major influence on machines, materials, and component. Humidity and temperature monitors have wide industrial applications in many areas such as automobile industry, food processing and any other industries. However, most of these monitors had limited functionality. Hence, this paper was offer the new features that can detect, notify, record the humidity and temperature instantaneously in order to have stable, controllable atmospheric conditions. This research was used of temperature humidity sensor DHT-11 to detect level of humidity and temperature changes inside the room. With Comparative analysis of the hardware data with readings from the HTC-2 device revealed minimal total error rates, with an average of 1.594% affirming the reliability and consistent performance of this device. The data and information from DHT-11 sensor was transferred by ESP32 using Arduino IDE Program and analyzed graphically on Google Spreadsheet platform using Apps Script.**

Keyword: ***Apps Script, Temperature and Humidity Monitoring, ESP32.***

I. INTRODUCTION

The convergence of Internet of Things (IoT) technologies with environmental monitoring has propelled the evolution of smart systems capable of real-time data acquisition and analysis. Within this domain, the monitoring of temperature and humidity stands as a cornerstone for ensuring optimal conditions in various contexts, from indoor environments to industrial processes. This journal embarks on a survey of recent developments in IoT-based temperature and humidity monitoring systems, exploring their design, implementation, and potential impact [1].

The significance of temperature and humidity monitoring spans across diverse sectors, including agriculture, healthcare, logistics, and infrastructure management. With IoT-enabled sensors and data analytics, stakeholders can gain valuable insights into environmental conditions, preemptively address anomalies, and optimize resource utilization. Such capabilities are instrumental in enhancing product quality, operational efficiency, and regulatory compliance [2].

This journal endeavors to provide a comprehensive overview of the methodologies and technologies driving the advancement of IoT-based temperature and humidity monitoring systems. Through a synthesis of empirical studies, technological innovations, and industry applications, it elucidates key factors influencing system performance, such as sensor accuracy, communication protocols, data processing algorithms, and energy efficiency

Furthermore, this survey explores the broader implications of IoT-driven environmental monitoring, including its potential to revolutionize sustainability practices, urban planning, and disaster resilience. By analyzing recent trends, challenges, and future prospects, this journal aims to inform researchers, practitioners, and policymakers about the state-of-the-art in IoT-based temperature and humidity monitoring and inspire collaborative efforts towards further advancements in this field.

II. METHOD

A. *Implementation Flow*

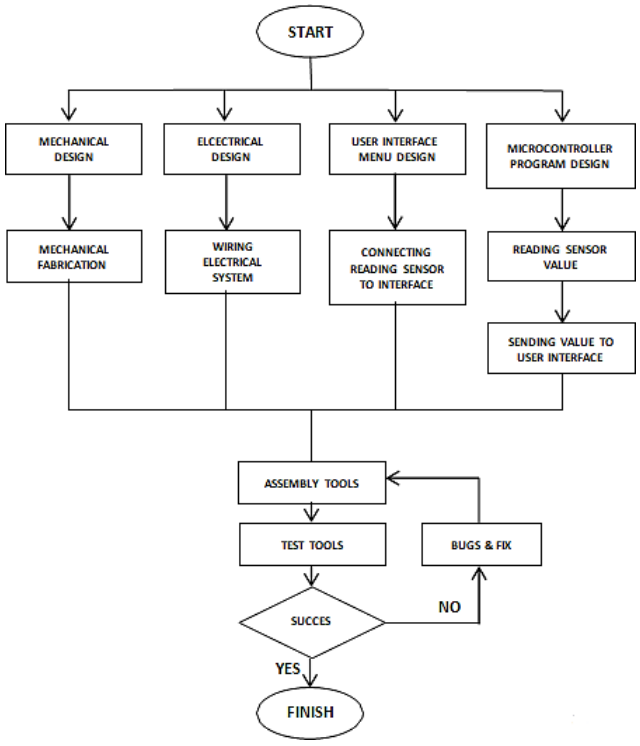

Fig 1. Implementation Flowchart

Figure 1 shows the research development flow of the proposed temperature, humidity monitoring and controlling system. The idea is to implement a complete temperature, humidity monitoring and controlling system. The required hardware components were purchased after the identification of the hardware based on the functional requirements of the system. Then, the temperature, humidity monitoring and controlling system was implemented using the purchased hardware, and the required software will be developed at the same time. Both

hardware and software components were tested to ensure the requirements of the system were met. If the system did not meet the requirements, troubleshooting process will be carried out to find out the problems and possible solutions to solve it. Otherwise, the temperature, humidity monitoring and controlling system cannot be developed successfully.

B. Electrical Design

Electrical circuit design is the stage of making tools which includes the process of making electrical designs, wiring cable processes, soldering cables, installing microcontrollers, and laying components to be used as shown in figure 2.

.

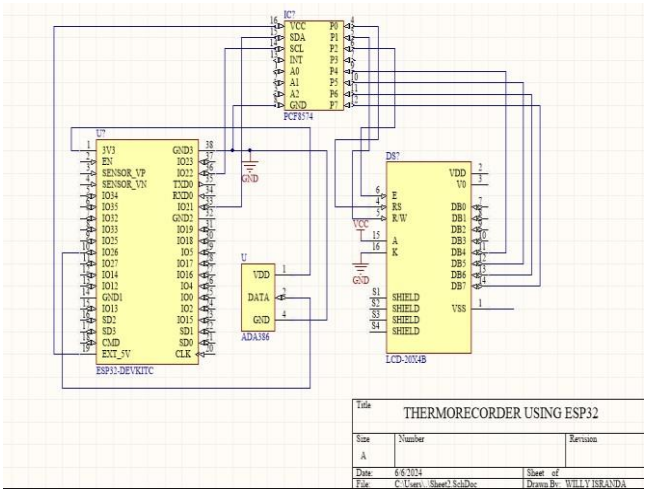

Fig 2.Electrical Diagram

Figure 2 shown a simple circuit diagram for prototype, The DHT 11 temperature and humidity sensor, as shown in Fig. 2, was used in this experiment. This sensor can measure temperature in the range of 0°C to 50°C, with an accuracy of ±2°C. Meanwhile, the relative humidity is measured between 20% and 90% RH with an accuracy of ±5% RH and the operating temperature range of 0°C to 50°C. With an 8-bit resolution, the measurement values are returned [3].

The signals from DHT 11 sensor were read and processed by the microcontroller. In this experiment, ESP32-C3-WROOM2 was used as a microcontroller, as shown in Fig. 2. It is a general-purpose Wi-Fi module. The ESP32 may function as a stand-alone system or as a slave device to a large number of MCUs, lowering communication stack overhead on most application processors. Through its SPI / SDIO or I2C / UART interfaces, the ESP32 can connect to other systems and provide Wi-Fi and Bluetooth capability. The rich set of peripherals and high performance make this module an ideal choice for smart homes, industrial automation, health care, consumer electronics, etc. ESP32-C3-WROOM2 is a 32-bit RISC-V single-core processor up to 160 MHz with 384KB ROM, 400KB SRAM and 8KB SRAM in RTC memory. ESP32-C3 integrates an upscale set of peripherals, starting from UART, I2C, I2S, remote peripheral, LED PWM controller, general DMA controller, TWAI® controller, USB Serial/JTAG controller, temperature sensor, and ADC [4].

LCD I2C modules are compatible with a wide range of LCD displays, including standard character LCDs (e.g., 16x2, 20x4) and graphical LCDs. They are compatible with popular microcontroller platforms such as Arduino, Raspberry Pi, and ESP8266/ESP32. Integration with microcontroller platforms is straightforward, typically involving the installation of appropriate libraries and writing minimal code to initialize the display and send text or commands.

As technology evolves, the LCD I2C module may see enhancements and innovations to address current limitations and cater to emerging trends. This may include improvements in communication protocols, increased compatibility with newer microcontroller platforms, and integration with advanced display technologies [5].

C. Interface Design

Google Apps Script (GAS) is a powerful scripting language developed by Google to automate tasks, extend Google products, and build custom applications within the Google ecosystem. Google Apps Script serves as a cloud-based scripting platform that enables developers to write code to automate processes and integrate with various Google services such as Gmail, Google Sheets, Google Docs, and Google Drive. It is based on JavaScript, making it accessible to a wide range of developers with varying levels of programming experience. The platform provides a browser-based integrated development environment (IDE), allowing developers to write, test, and deploy scripts directly within the browser [6].

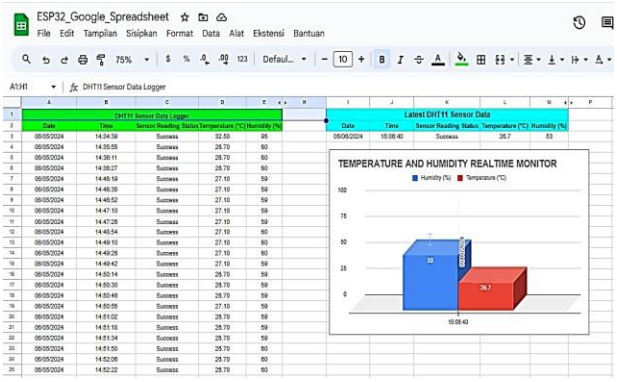

Fig 3.Interface and Data Store

For monitoring process itself need to ease the process by combining several program, On this project not only used Arduino IDE to program, also used Apps Script to display the value of the temperature and humidity from the ESP32, Apps Script itself able to print the value that ESP32 transfer to the Google Spreadsheet that often used to monitoring and collecting data. It works after the program deployment, copy the URL and add several programs behind according the value that need to transfer.

D. Microcontroller Program Design

In designing this temperature and humidity monitoring used ESP32 as microcontrollers to run the program that already design, the DHT11 are receiving the value from the actual

temperature and humidity and the ESP32 as the microcontrollers transmitting the data to the interface and LCD Module to display the value.

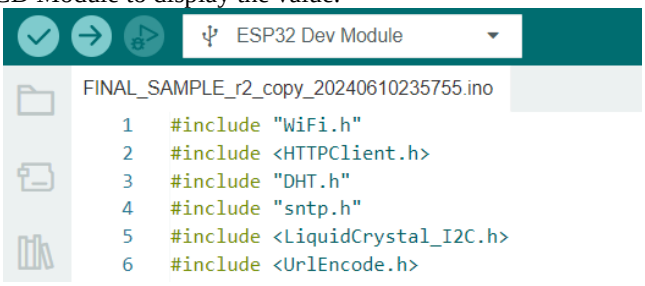

Fig 4.Microcontroller Library

Fig 5.Microcontroller Program

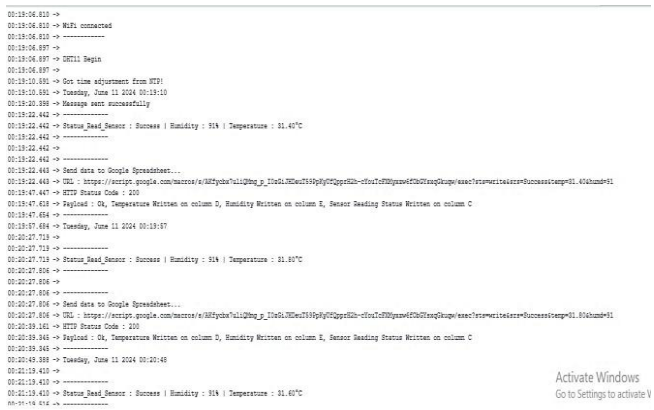

Fig 6.Microcontroller Output

In designing this temperature and humidity monitoring used ESP32 as microcontrollers to run the program the already design, the DHT11 are receiving the value from the actual temperature and humidity and the ESP32 as the microcontrollers transmitting the data to the interface and LCD Module to display the value. To get started with the Apps Script it necessary to make the Script at Google Sheet Extension, Apps Script, The Apps Script need to make a bit of program that write the temperature and humidity value at Google Sheet, and also need to synchronize the Name of the Sheet, Coolum, Row. After that, the program can be deploy and Paste the deploy script at Arduino Program.

For the notification need to use Whatsapp bot to notify the user if specification is out of the limit. From that Whatsapp bot, APIkey will be send from that bot and apply that APIkey at the Arduino Program.

E. Mechanical Design

Mechanical design is the stage of making the mechanical framework of the tool which is equipped with the dimensions of the drawing, the parts of the tool, and the size of the tool to be made as shown in Figure 7. In this mechanical design, the control box material used for the Temperature and Humidity Control is made of PVC Polyvinyl Chloride.

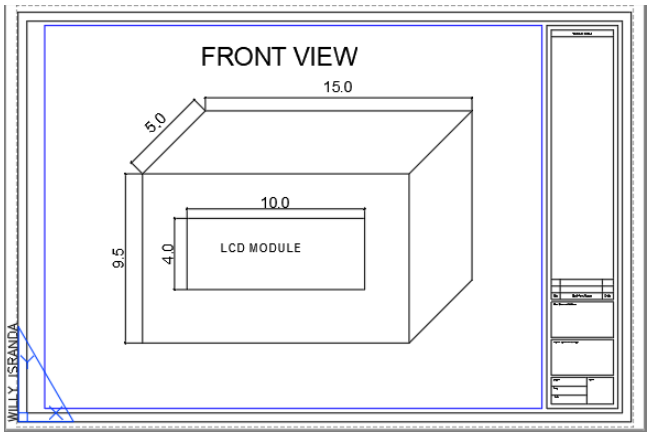

Fig 7.Front View Design

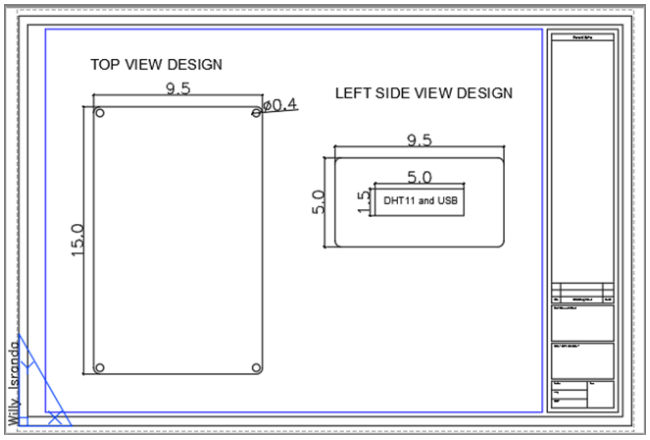

Fig 8.Top And Left side View Design

III. RESULT AND DISCUSSION

A. Mechanical Assembly Result

Fig 9.Mechanical Assembly

In the mechanical assembly there are several important components, namely: Electronic box that is used as a place of the PCB itself, and There is DHT11 sensor that visible at left side of the box to sense the temperature and humidity. Also there are an LCD Module to display the Value of temperature and humidity, also able to display time and date.

B. *Electrical Assembly Result*

The working principle of this electrical system is quite simple by use a 3.7 V Battery that supply to ESP32, Then the ESP32 itself used as the power source of the DHT11 Sensor and LCD Module.

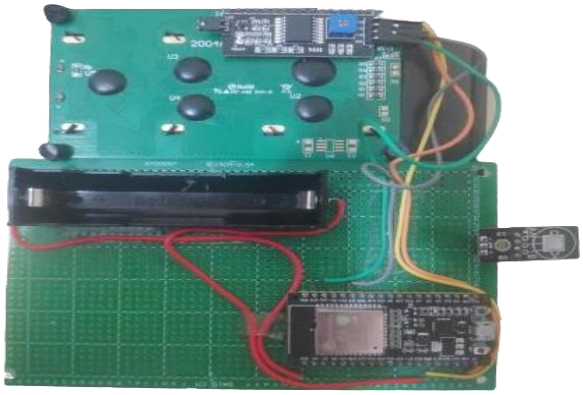

Fig 10.Electrical Assembly

Below are some of the important components used and their functions in the electrical assembly, namely:

Table 1. Component names and Function

NO	COMPONENT NAME	FUNCTION
1	ESP32-WROOM DEVKIT	ESP32 used as a microcontroller that controls various electronic components that have programmed and used to sending data using Wi-Fi to store data logger.
2	DHT11	DHT11 Sensor used to get the value of temperature and humidity for area nearby the sensor.
3	I2C MODULE	I2C Module used to reduce pin usage of using LCD module.
4	LCD MODULE	LCD Module used to display value of the temperature and humidity, and also display real-time.
5	SOCKET BATTERY 3.7 VOLT	Socket battery 3.7V used to supply ESP32 to power the component, 3.7 V are use because the ESP32 Operate on 3.3V and 3.7V still safe to use.

C. *Programming Result*

In making this Temperature and Humidity Monitoring Tool, the application used to program the ESP32 so the component able to run according its function, In this case Arduino IDE is the application program, The following is the output of serial monitor of the program after running.

Fig 11.Serial Output Program

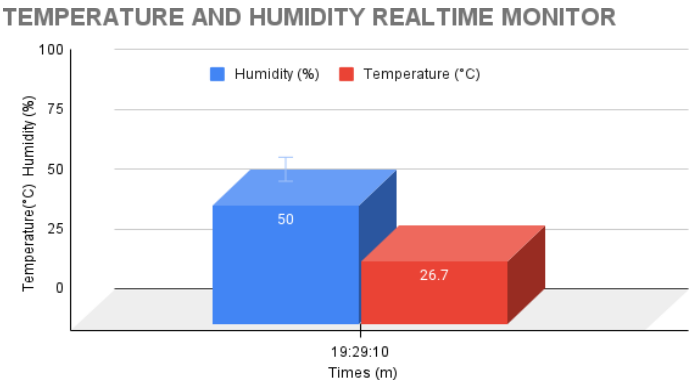

Fig 12.Real Time Monitoring Chart

Based on the Figure 12 the latest value shown on the chart is the value that latest store in the Table 2. Figure 12 shown the latest reading from sensor and will be overwrite by the newest value that DHT11 sensor read. Otherwise, the value that already DHT11 sensor read is store on the table on the left side of the Real Time Chart.

Fig 13.ESP32 Notification to User

```
if (Temp > 32) {
sendAlert("Warning! Temperature out of Range");
}
if (Temp < 18) {
sendAlert("Warning! Temperature out of Range");
}
if (Humd > 90) {
sendAlert("Warning! Humidity out of Range");
}
if (Humd < 40) {
sendAlert("Warning! Humidity out of Range");
}
```

Fig 14.Program Specification Limit

This tool also provide with notification whenever the tools activate and also it can notify the user if the temperature and the humidity out of the specification limit.

Fig 15.Temperature Out of Limit Display

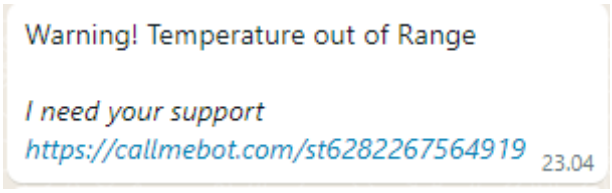

Fig 16.Temperature Out of Limit Notification

Fig 17.Humidity Out of Limit Display

Fig 18.Humidity Out of Limit Notification

Fig 19.Temperature and Humidity Out of Limit Display

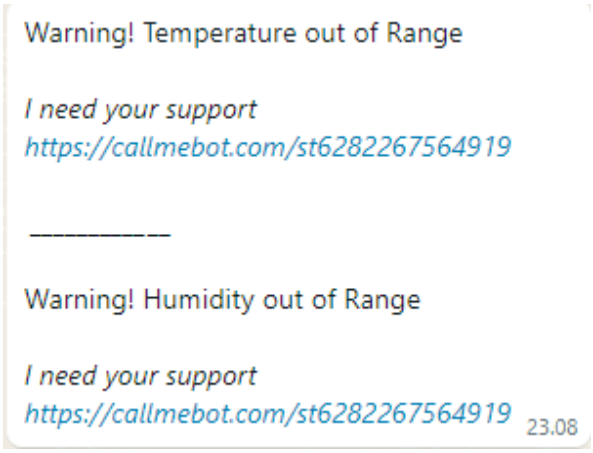

Fig 20.Temperature and Humidity Out of Limit Notification

This tool able to do this certain action by using Whatsapp bot as the notification system, by collecting the APIkey from the bot and using URLEncode Library the program able to use this notification. And also able to change the specification on Figure 14 at the Arduino IDE according the user requirement.

D. Sensor Linearity Test

To ensure the consistency of the sensor, Linearity test needed to prove it. In this test sensor will be tested on the different room for 30 minute for each room to find the consistency of DHT11 sensor, below are the result that shown the value of the from the DHT11 from different room at Figure 21.

Time	AC Room Temperature (°C)	AC Room Humidity (%)	No AC Room Temperature (°C)	No AC Room Humidity (%)
1	28.5	60	30.2	91
2	28.5	54	30.2	91
3	28.5	53	30.2	91
4	28.5	53	30.2	91
5	28.5	54	30.2	91
6	28.1	54	30.5	91
7	28.5	55	30.2	91
8	28.5	60	30.2	91
9	28.5	57	30.2	91
10	28.0	54	30.2	91
11	28.0	56	30.2	91
12	28.0	56	30.2	91
13	27.6	55	30.2	91
14	27.6	55	30.2	91
15	27.6	54	30.2	91
16	27.6	56	30.2	91
17	27.6	57	30.2	91
18	27.6	55	30.4	91
19	27.6	56	30.2	91
20	27.6	57	30.2	91
21	27.6	58	30.2	91
22	27.6	57	30.2	91
23	27.6	58	30.2	91
24	27.6	58	30.0	91
25	27.6	57	30.2	91
26	27.6	57	30.2	91
27	27.6	61	30.0	91
28	27.6	59	30.2	91
29	27.9	61	30.1	91
30	27.6	57	30.2	91
AVG	27.9	56.5	30.2	91.0
MAX	28.5	61	30.5	91
MIN	27.6	53.0	30.0	91.0
STDEV	0.392984455	2.239663357	0.087098834	0

Fig 21.Temperature and Humidity Testing Result

Figure 21 shown that the sensor reading between to different room with different situation, because there are one room with AC (Air Conditioner) and another room without it. The deviation on the different room quite significant this is because the each room has different condition and factor that effect the value of the DHT11 sensor read.

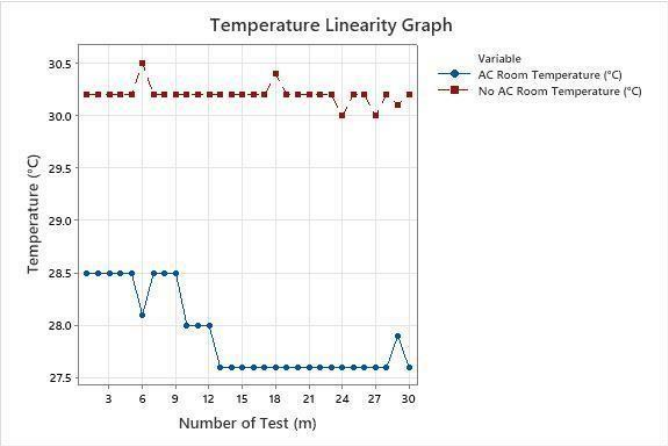

Fig 22.Temperature Linearity Graph DHT11 Sensor

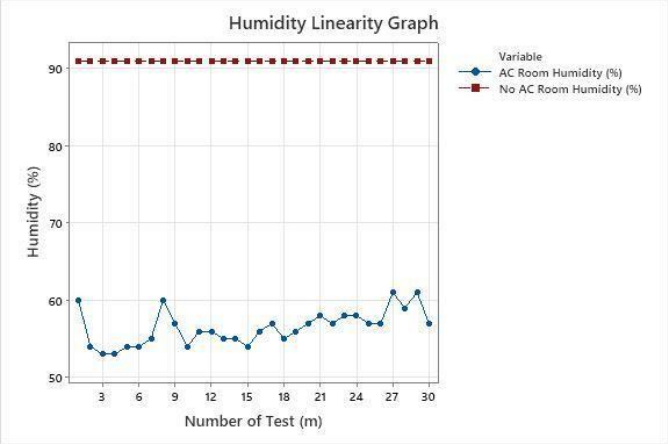

Fig 23.Humidity Linearity Graph DHT11 Sensor

Figure 22 and Figure 23 Shown that the linearity of the sensor Graphically, the figure shown that the temperature and humidity on the AC Room has a lots of change on the value, this is because the room environment condition was change by following the AC condition while the other room has a flat environment condition because no factor that impacting the temperature and humidity.

E. Measurement Calibration

This process is run to test the error of the equipment measurement, this process are run by comparing the prototype with a Thermo recorder from HTC-2 (Used as a Standard for Calibration) and tested in TFME Cleanroom to get the value and comparing it.

To calculate the total error, it need to calculate the error temperature and humidity error first for each measurement. Then, the total error is calculated as the average from temperature error and humidity error. Here is the formula to calculate temperature error and humidity error:

Error Temperature (%) = |(Temperature HTC-2 - Temperature)| / Temperature HTC-2 * 100%

Error Humidity (%) = |(Humidity HTC-2 - Humidity)| / Humidity HTC-2 * 100%

Error Total (%) = (Error Temperature + Error Humidity) / 2

Below are the example of the calculation from the first measurement :

Temperature HTC-2 = 26.9°C
Temperature = 26.7°C
Error Temperature (%) = |(26.9 – 26.7)| / 26.9*100% = 0.74%
Humidity HTC-2 = 53%
Humidity = 52%
Error Humidity (%) = |(53 - 52)| / 53 * 100% = 1.8%
Error Total (%) = (0.74 + 1.8) / 2 = 1.27%

Table 2. DHT11 Sensor Verification

NO	Temperature (°C)	Humidity (%)	Temperature HTC-2 (°C)	Humidity HTC-2 (%)	Error Total (%)
1	26.70	52	26.9	53	1.27
2	26.70	52	26.9	53	1.27
3	26.70	54	27.0	52	2.45
4	26.70	52	27.1	52	0.7
5	26.70	52	27.1	52	0.7
6	26.70	53	27.2	51	2.85
7	27.10	51	27.0	52	1.2
8	27.60	51	27.1	52	1.85
9	27.60	51	27.2	52	1.65
10	28.00	49	27.4	52	2.0
AVG	27.05	51.7	27.09	52.1	1.594

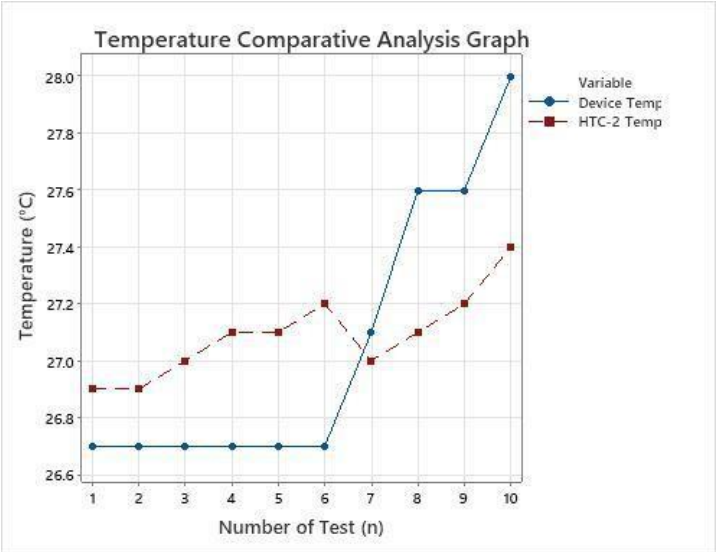

Fig 24.Temperature Comparison Chart

In Fig 24, show the comparison of temperature data collected from the hardware with temperature data from the HTC-2 device. This graph provides a visual overview of the extent to which the hardware consistently monitors temperature in the data center space and how accurate the data comparisons are with the HTC-2 device. If the graph shows a pattern line that approaches the 1:1 reference line, this indicates that hardware and the HTC-2 delivers results quite accurate in monitoring temperature.

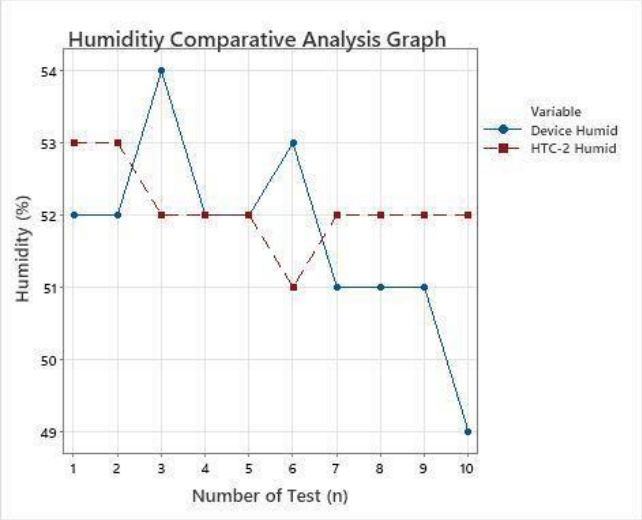

Fig 25.Humidity Comparison Chart

Fig 25 provides a humidity comparison collected from hardware with humidity data from the HTC-2 device. Through this graph, we can assess the extent where the hardware provides humidity readings which is consistent and close to the results of the HTC-2. If line on this chart it is approaching the 1:1 reference line, that means hardware and the HTC-2 shows similarities quite good

In this testing process DHT11 Reading is still within the DHT11 accuracy for the temperature and the humidity from the beginning until the end of the test. According the accuracy of the DHT11 which is ±2°C for the temperature and ±5%RH for the humidity, With total error average 1.594% the measurement testing on the device still within the DHT11 Accuracy which mean the reading is Pass.

IV. Conclusion

Humidity monitoring and controlling system will serve as an optimal solution for all the concerned parties such as electrical component storage facilities, automobile manufacturers and food storage departments to improve their efficiency in monitoring, controlling as well as managing the environmental parameters such as temperature and humidity where these factors play an important role in their productivity and finance. The system is capable of obtaining the temperature and humidity of a space, this providing a real time temperature and humidity monitoring either using an application in a smartphone via WIFI over IoT, a LCD display embedded in the system or a computer interface.

The application mentioned above keeps track of the temperature and humidity readings as well, which helps the user to monitor these parameters in a table that able to turn into a graph. Besides that, the system is also able to send an alert to the user Whatsapp if either one or both of these parameters fall in the critical region; as well as making the necessary step to bring back the particular parameter to safe region such as turning on the humidifier if the humidity falls below the assigned value.

In testing process DHT11 Reading is still within the DHT11 accuracy for the temperature and the humidity. With total error average of 1.594 %. The measurement testing still within the DHT11 Accuracy which means the reading is pass the specification of the DHT11 sensor.

This research was built upon basic components such as DHT11 as the sensor. Further improvements can be made to this project to increase its capability and efficiency such as temperature control. By using serial communication between two or more microcontrollers to monitor and control. With this system in place, it can eliminate the necessity of workers to manually obtain the reading of these parameters and maintaining them frequently. Many industries especially those require the controlled storages in which the temperature and humidity plays an important role to ensure the lifespan and efficiency of their products. The system will be a great help in boosting the efficiency and reducing the cost.

References

[1] Patel, S., & Gupta, R. (2023). "IoT-Enabled Smart Agriculture: A Case Study on Real-Time Monitoring of Soil Moisture and Temperature." Journal of Agricultural Engineering, vol. 30, no. 4, pp. 432-445

[2] Chen, X., et al. (2023). "Recent Advances in IoT-Based Environmental Monitoring Systems: A Comprehensive Review." IEEE Internet of Things Journal, vol. 15, no. 6, pp. 567-580.

[3] W. W. Gay, “DHT11 Sensor,” in Experimenting with Raspberry Pi, Springer., Apress, Berkeley, CA, 2014, pp. 1–13.

[4] Espressif Systems, “ESP32 Series Datasheet,” Espressif Systems, 2021. www.espressif.com (accessed Jun. 12, 2021).

[5] Banzi, M., & Shiloh, M. (2014). Arduino: A Quick-Start Guide (2nd ed.). Maker Media, Inc.

[6] Bazilian, R. (2019). Google Apps Script: Web Application Development Essentials. Packt Publishing Ltd.

Attachment

A. Arduino IDE Program Appendix

```
#include "WiFi.h"
#include <HTTPClient.h>
#include "DHT.h"
#include "sntp.h"
#include <LiquidCrystal_I2C.h>
#include <UrlEncode.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);
byte derajat[8] = {
B00111,
B00101,
B00111,
B00000,
B00000,
B00000,
B00000,
B00000
};
byte suhu[8] = {
B00100,
B01010,
B01010,
B01110,
B11111,
B11111,
B01110,
B00000
};
byte kelembapan[8] = {
B00100,
B01010,
B01010,
B10001,
B10001,
B10001,
B01110,
B00000
};
#define DHTPIN  26 // PIN Mengikuti rangkaian
#define DHTTYPE DHT11
const char* ntpServer1 = "pool.ntp.org";
const char* ntpServer2 = "time.nist.gov";
const long  gmtOffset_sec = 25200;
const int   daylightOffset_sec = 0;
const char* ssid = "POCO";  //--> Username Wifi Bisa Diubah
const char* password = "11112222"; //--> wifi password
String phoneNumber = "+6282267564919"; //country_code + phone number
String apiKey = "6089169";//--> Apikeyy dari bot yang dipakai
// Google script Web_App_URL.
String Web_App_URL = "https://script.google.com/macros/s/AKfycbx7uliQMmg_p_I0zGiJHDeuT59PpKyUfQpprH2h-cYouTcFXMyxzw6fObGYsxqGkuqw/exec"; // URL APPSCRIPT
String Status_Read_Sensor = "";
float Temp;
int Humd;
DHT dht11(DHTPIN, DHTTYPE);
```

```
void Getting_DHT11_Sensor_Data() {
Humd = dht11.readHumidity();
Temp = dht11.readTemperature();
if (isnan(Humd) || isnan(Temp)) {
Serial.println();
Serial.println(F("Failed to read from DHT sensor!"));
Serial.println();
Status_Read_Sensor = "Failed";
Temp = 0.00;
Humd = 0;
} else {
Status_Read_Sensor = "Success";
}
Serial.println();
Serial.println(" -------------");
Serial.print(F("Status_Read_Sensor : "));
Serial.print(Status_Read_Sensor);
Serial.print(F(" | Humidity : "));
Serial.print(Humd-7);
Serial.print(F("% | Temperature : "));
Serial.print(Temp);
Serial.println(F("°C"));
Serial.println(" -------------");
}
void printLocalTime(){
struct tm timeinfo;
if(!getLocalTime(&timeinfo)){
Serial.println("No time available (yet)");
return;
}
Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
lcd.setCursor(0,3);
lcd.print(&timeinfo,"%a, %d%b%y, %H:%M");
}
void timeavailable(struct timeval *t){
Serial.println("Got time adjustment from NTP!");
printLocalTime();
}
void setup() {
Serial.begin(115200);
Serial.println();
delay(1000);
lcd.init();
lcd.backlight();
lcd.createChar(0, suhu);
lcd.createChar(1, kelembapan);
lcd.createChar(2, derajat);
lcd.setCursor(1, 0);
lcd.print("ESP32 MONITORING");
lcd.setCursor(1, 1);
lcd.print("TEMPERATURE:");
lcd.setCursor(1, 2);
lcd.print("HUMIDITY   :");
sntp_set_time_sync_notification_cb( timeavailable );
sntp_servermode_dhcp(1);     // (optional)
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer1, ntpServer2);
Serial.println();
Serial.println(" -------------");
Serial.println("WIFI mode : STA");
WiFi.mode(WIFI_STA);
```

```
Serial.println(" ----------");
Serial.println();
Serial.println(" ----------");
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
int connecting_process_timed_out = 20; //--> 20 = 20 seconds.
connecting_process_timed_out = connecting_process_timed_out * 2;
while (WiFi.status() != WL_CONNECTED) {
Serial.print(".");
delay(5000);
if (connecting_process_timed_out > 0) connecting_process_timed_out--;
if (connecting_process_timed_out == 0) {
delay(1000);
ESP.restart();
}
}
Serial.println();
Serial.println("WiFi connected");
Serial.println(" ----------");
Serial.println();
Serial.println("DHT11 Begin");
Serial.println();
dht11.begin();
sendAlert("Your system is ready!");
delay(2000);
}
void loop() {
Getting_DHT11_Sensor_Data();
if (WiFi.status() == WL_CONNECTED) {
String Send_Data_URL = Web_App_URL + "?sts=write";
Send_Data_URL += "&srs=" + Status_Read_Sensor;
Send_Data_URL += "&temp=" + String(Temp);
Send_Data_URL += "&humd=" + String(Humd-7); //-7 Adjustment After Calibration With Thermorecorder HTC-2
Serial.println();
Serial.println(" ----------");
Serial.println("Send data to Google Spreadsheet.  ");
Serial.print("URL : ");
Serial.println(Send_Data_URL);
HTTPClient http;
http.begin(Send_Data_URL.c_str());
http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
int httpCode = http.GET();
Serial.print("HTTP Status Code : ");
Serial.println(httpCode);
String payload;
if (httpCode > 0) {
payload = http.getString();
Serial.println("Payload : " + payload);
}
http.end();
Serial.println(" ----------");
}
if (Temp > 32) {
sendAlert("Warning! Temperature out of Range");
}
if (Temp < 18) {
sendAlert("Warning! Temperature out of Range");
}
if (Humd > 100) {
```

```
sendAlert("Warning! Humidity out of Range");
}
if (Humd < 40) {
sendAlert("Warning! Humidity out of Range");
}
delay(10000); //---------- Interval Pembacaan dan Pengiriman Data
lcd.setCursor(0, 1);
lcd.write((byte)0);
lcd.setCursor(0, 2);
lcd.write((byte)1);
lcd.setCursor(13, 1);
lcd.print(Temp);
lcd.setCursor(13, 2);
lcd.print(Humd-7);
lcd.setCursor(18, 1);
lcd.write((byte)2);
lcd.setCursor(19, 1);
lcd.print("C");
lcd.setCursor(19, 2);
lcd.print("%");
printLocalTime();
delay(30000);
}
void sendAlert(String message) {
String url = "https://api.callmebot.com/whatsapp.php?phone=" + phoneNumber + "&apikey=" + apiKey + "&text=" +
urlEncode(message);
HTTPClient http;
http.begin(url);
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
int httpResponseCode = http.POST(url);
if (httpResponseCode == 200) {
Serial.print("Message sent successfully");
}
else {
Serial.println("Error sending the message");
Serial.print("HTTP response code: ");
Serial.println(httpResponseCode);
}
http.end();
}
```

B. Apps Script Program Appendix

```
function doGet(e) {
  Logger.log(JSON.stringify(e));
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1tGtfjrrRth8RJJ330LsSiJQutM3soo8_D9v9Y3fbK1E';     // Spreadsheet ID.
    var sheet_name = "ESP32_Google_Sheets_Sheet";  // Sheet Name in Google Sheets.
    var sheet_open = SpreadsheetApp.openById(sheet_id);
    var sheet_target = sheet_open.getSheetByName(sheet_name);
    var newRow = sheet_target.getLastRow() + 1;
    var rowDataLog = [];
    var Data_for_I3;
    var Data_for_J3;
    var Data_for_K3;
    var Data_for_L3;
    var Data_for_M3;

    var Curr_Date = Utilities.formatDate(new Date(), "Asia/Jakarta", 'dd/MM/yyyy');
    rowDataLog[0] = Curr_Date;  // Date will be written in column A (in the "DHT11 Sensor Data Logger" section).
    Data_for_I3 = Curr_Date;  // Date will be written in column I3 (in the "Latest DHT11 Sensor Data" section).

    var Curr_Time = Utilities.formatDate(new Date(), "Asia/Jakarta", 'HH:mm:ss');
    rowDataLog[1] = Curr_Time;  // Time will be written in column B (in the "DHT11 Sensor Data Logger" section).
    Data_for_J3 = Curr_Time;  // Time will be written in column J3 (in the "Latest DHT11 Sensor Data" section).

    var sts_val = '';

    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      switch (param) {
        case 'sts':
          sts_val = value;
          break;

        case 'srs':
          rowDataLog[2] = value;  // Sensor Reading Status will be written in column C (in the "DHT11 Sensor Data Logger" section).
          Data_for_K3 = value;  // Sensor Reading Status will be written in column K3 (in the "Latest DHT11 Sensor Data" section).
          result += ', Sensor Reading Status Written on column C';
          break;

        case 'temp':
          rowDataLog[3] = value;  // The temperature value will be written in column D (in the "DHT11 Sensor Data Logger" section).
          Data_for_L3 = value;  // The temperature value will be written in column L3 (in the "Latest DHT11 Sensor Data" section).
          result += ', Temperature Written on column D';
          break;

        case 'humd':
          rowDataLog[4] = value; // The humidity value will be written in column E (in the "DHT11 Sensor Data Logger" section).
          Data_for_M3 = value;  // The humidity value will be written in column M3 (in the "Latest DHT11 Sensor Data" section).
          result += ', Humidity Written on column E';
          break;
```

```
        default:
          result += ", unsupported parameter";
      }
    }

    // Conditions for writing data received from ESP32 to Google Sheets.
    if (sts_val == 'write') {
      // Writes data to the "DHT11 Sensor Data Logger" section.
      Logger.log(JSON.stringify(rowDataLog));
      var newRangeDataLog = sheet_target.getRange(newRow, 1, 1, rowDataLog.length);
      newRangeDataLog.setValues([rowDataLog]);

      // Write the data to the "Latest DHT11 Sensor Data" section.
      var RangeDataLatest = sheet_target.getRange('I3:O3');
      RangeDataLatest.setValues([[Data_for_I3, Data_for_J3, Data_for_K3, Data_for_L3, Data_for_M3]]);

      return ContentService.createTextOutput(result);
    }

    // Conditions for sending data to ESP32 when ESP32 reads data from Google Sheets.
    if (sts_val == 'read') {
      // Use the line of code below if you want ESP32 to read data from columns I3 to O3 (Date,Time,Sensor Reading Status,Temperature,Humidity,Switch 1, Switch 2).
      // var all_Data = sheet_target.getRange('I3:O3').getDisplayValues();

      // Use the line of code below if you want ESP32 to read data from columns K3 to O3 (Sensor Reading Status,Temperature,Humidity,Switch 1, Switch 2).
      var all_Data = sheet_target.getRange('K3:O3').getValues();
      return ContentService.createTextOutput(all_Data);
    }
  }
}
function stripQuotes( value ) {
  return value.replace(/^["']|["']$/g, "");
}
```

FORMULIR LOOGBOOK BIMBINGAN DAN PENGAJUAN
SEMINAR PROPOSAL/SIDANG TUGAS AKHIR*

Nama : Willy Isranda Sihombing
NIM : 3222111004
Pembimbing I : Nadhrah Wivanius, S.Si., M.Si.
Pembimbing II* : -
Judul : Smart Temperature and Humidity Monitor

No	Hari/Tgl	Rincian Kegiatan	TTD Pembimbing I & II	
1	27-Feb-2024	Melaporkan perencanaan pengerjaan proyek akhir		
2	14-Mei-2024	Membahas pengumpulan data pada prototype untuk laporan tugas akhir		
3	16-Mei-2024	Perencanaan penyusunan laporan tugas akhir bab-4		
4	23-Mei-2024	Membahas hasil pengerjaan bab 4 dan 5		
5	29-Mei-2024	Perencanaan Penggunaan template Jurnal Laporan tugas akhir		
6	11-Juni-2024	Membahas mengenai penulisan program dan pelampiran program pada laporan tugas akhir.		
7	14-Juni-2024	Menentukan letak penulisan program pada laporan tugas akhir.		
8	21-Juni-2024	Melakukan bimbingan dengan membahas Penulisan Laporan Tugas akhir		
9	26-Juni-2024	Melakukan bimbingan dengan membahas Penulisan Laporan Tugas akhir		
10	28-Juni-2024	Melaporkan Hasil revisi dari bimbingan sebelumnya mengenai penulisan laporan		

Berdasarkan hasil bimbingan yang telah dilaksanakan selama 4 bulan dan telah disetujui oleh dosen pembimbing, maka dengan ini saya mengajukan diri sebagai peserta Seminar Proposal /Sidang Tugas Akhir*.

Batam,28-Juni-2024
Peserta

Willy Isranda Sihombing
NIM: 3222111004

**Hapus yang tidak perlu.*
Jumlah bimbingan minimal 10 kali. Dalam satu minggu maksimal bimbingan yang dihitung adalah 2 kali.