# IoT-Based Temperature Monitoring and Automatic Fan Control Using ESP32

JO CHRISTOPHER M. RESQUITES[1], MATTHEW ALEXIE PARROCHO[2], NOEL VINEGAS[3], DR. VINYL H. OQUIÑO[4]

*[1, 2, 3] Masterand, Master in Engineering-Eastern Visayas State University-Main Campus*

*[4] Professor, Master in Engineering-Eastern Visayas State University-Main Campus*

***Abstract—This study investigates the development and application of a temperature monitoring and automatic fan control system based on the Internet of Things (IoT). The system makes use of an ESP32 WiFi module, a DHT11 temperature sensor, a transistor for fan control, and IoT technologies. Its real-time fan speed change based on outside temperature is a benefit that improves user convenience, environmental comfort, and energy efficiency. The study's conclusions highlight the system's precise temperature measurement and trustworthy fan control algorithms. It also illustrates the system's superiority by contrasting it with other alternatives that are currently in use. The study's conclusions offer suggestions for future work, including potential improvements in security, interaction with other IoT devices, geographical sensing, adaptive fan speed management, and reliability. This IoT-based technology paves the way for greater investigation and development in the area of IoT applications while also holding promise for a more user-friendly and energy-efficient future.***

***Index Terms—Automatic Fan Control, Cloud-Based Application, Energy Efficiency, Environmental Comfort, ESP32, IoT-Based Temperature Monitoring, Real-Time Monitoring, Temperature Monitoring, Variable Temperature Threshold, Wireless Communication.***

## I. INTRODUCTION

An internet-based network of physical things with the ability to interact and communicate with one another is known as the Internet of Things (IoT). Smart homes, smart cities, smart agriculture, smart health, and smart industry are just a few of the many areas in which IoT finds use. The ability to remotely monitor and operate systems and equipment can be an advantage of the Internet of Things since it can increase comfort, convenience, and efficiency. A fan is among the typical items that the Internet of Things can manage. An apparatus that rotates blades or impellers to produce airflow is called a fan. Uses for fans include cooling, ventilation, and air circulation. In addition, a fan's speed and mode can be changed to suit the needs of the user and the outside environment. Manually altering the fan speed or mode, however, can be inconvenient or ineffective, particularly if the user is not close to the fan or if the temperature is constantly fluctuating. A device that can automatically change the fan speed or mode depending on the ambient temperature would, therefore, be helpful. By controlling fan speed by temperature, such a system can reduce energy consumption. Enabling the user to adjust the preferred temperature threshold and view the current temperature and fan speed on a mobile device can also increase comfort and convenience.
The aim of this project is to design and implement an IoT-based temperature monitoring and automatic fan control system. The system uses an ESP32 (NodeMCU ESP-32S) Wi-Fi module to connect to the internet and send or receive data from a cloud-based application such as Blynk. The system also uses a DHT11 sensor to measure the temperature and a transistor (2N2222) to control the fan. The system can be controlled and monitored remotely using any iOS or Android device.

The objectives of this project are:
1. To select and interface the hardware components such as the ESP32 module, the DHT11 sensor, the fan, and the transistor.
2. To program the ESP32 module using Arduino IDE and libraries such as Blynk, DHT, and WiFi.

3. To design and test the circuit and Breadboard layout for the system.
4. To develop and configure the Blynk app for controlling and monitoring the system.

The project confines itself to:
1. The use of a prototype design that only demonstrates the basic functionality and feasibility of the proposed system.
2. The use of a DC fan with a fixed voltage and current rating.
3. The use of a single temperature sensor and a single fan for the system.
4. The use of Blynk as the cloud-based application for the system.

## II. LITERATURE REVIEW

The IoT-based temperature monitoring and automatic fan control system is a project that aims to regulate the temperature of a room by adjusting the fan's speed in response to the temperature that is being observed. The system consists of a temperature sensor, a microcontroller, a fan motor, a Wi-Fi module, and a cloud server. The user can monitor and control the system using a mobile or online application. The system is designed to be easy to use, low-cost, and energy-efficient. Similar systems have been the subject of numerous investigations in the past.

For example, a proposed IoT temperature-based fan speed control and monitoring system using an ESP8266 Wi-Fi module and the Blynk cloud platform. The project makes use of a Wi-Fi module, a NodeMCU microcontroller, and an LM35 temperature sensor. The microcontroller receives temperature values from the temperature sensor, which measures the ambient temperature. After processing the temperature data, the microcontroller uses the temperature readings to determine how fast to run the fan. Through the use of a mobile application, the Wi-Fi module allows users to remotely adjust the fan speed and connect the system to the internet. The project includes the source code, circuit schematic, list of components, and experimental findings.

The project's lack of performance comparison with other current temperature-based fan speed control methods or systems represents a knowledge gap. It also skips over the project's drawbacks and difficulties, like the IoT system's scalability, security, and dependability. By including more literature reviews, analyses, and discussions on these topics, as well as offering some recommendations for future research, the project could be made better. [1].

Another example is that the goal of the project is to construct an Internet of Things (IoT)-based smart fan control circuit that uses Blynk to regulate the speed of a standard fan or other appliances. With the help of the mobile app and cloud platform Blynk, customers can design and manage Internet of Things projects from a distance. A DHT11 temperature and humidity sensor, a relay module, a fan, a buzzer, and a NodeMCU ESP8266 module are all used in the project. Additionally, the project modifies the voltage across the fan and modifies its speed using a TRIAC-based control circuit. The project includes the source code, PCB design, setup instructions for the Blynk app, and circuit diagram.

The project's knowledge gap is that it doesn't describe how the TRIAC regulates the fan speed or how it operates. Furthermore, it offers no theoretical or quantitative examination of the fan speed control algorithm's effects on the system's efficiency and power consumption. More background data, a study of the literature, a discussion of these issues, some experimental outcomes, and a comparison with alternative approaches could all help to improve the project [2].

A third example, in this paper, an Arduino and DHT11 sensor-based temperature-based fan speed control system, is designed and implemented. The application of temperature sensor data to independently modify the fan's speed is demonstrated in the study. The project's source code, circuit diagram, component list, and video demonstration are all provided in the paper [3].

The lack of a literature review or background information on the current approaches or systems for temperature-based fan speed management represents a knowledge gap in the work. Furthermore, it offers no theoretical or quantitative examination of the fan speed control algorithm's effects on the system's efficiency and power consumption. More references, analysis, and discussion of these issues, along with some experimental data or method comparisons, could all help to improve the paper.

A fourth example, in this research, an Android device is used to create an Internet of Things (IoT)-based home temperature control system. The goal of the study is to create an automatic temperature ventilation system that can fully temper a space and automate the voltage control of AC-supported appliances. Along with preventing overheating, the paper enables remote access and management via an Android smartphone app. The Node MCU microcontroller, DHT11 temperature and humidity sensor, relay module, AC lamp, AC fan, and DC cooling fan are all used in this article. The project's circuit schematic, component list, source code, and experimental findings are all provided in this paper.

The lack of a literature review or background information on the current approaches or systems for IoT-based temperature management represents a knowledge gap in the paper. Furthermore, it makes no mention of the benefits or drawbacks of utilizing the DHT11 sensor and Node MCU microcontroller in comparison to other options. More references, analysis, and comparison on these points, as well as suggestions for the project's future scope, could all help to improve the work.

Therefore, further research and development are needed to improve the design and performance of these systems and to explore new applications and features.

## III. METHODOLOGY

### *A. System Architecture and Components*

The system architecture and components of the IoT-based temperature monitoring automatic fan control system are shown in Fig. 1. The system consists of four main components: the ESP32 module, the DHT11 sensor, the fan, and the transistor. The ESP32 module is a microcontroller development board that has WiFi and Bluetooth capabilities. The DHT11 sensor is a digital temperature and humidity sensor that can measure the ambient temperature with an accuracy of ±2°C. The fan is a DC fan that can rotate at different speeds depending on the voltage applied to it. The transistor is a 2N2222 NPN transistor that can act as a switch to control the fan.

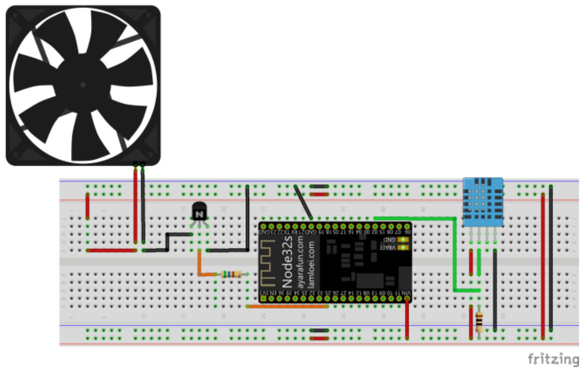

Fig. 1. Breadboard Circuit Diagram

### *B. Hardware Design and Implementation*

The hardware design and implementation of the system are shown in Fig. 2. The hardware components are connected as follows: The ESP32 module is powered by a 5V USB cable connected to a computer or a power bank. The DHT11 sensor is connected to a 10 kΩ resistor and a digital PIN GIOP2 of the ESP32 module. The fan is connected to a Vcc of HDT11, a Vin 5V of the ESP32 module, and a collector pin of the transistor. The transistor is connected to a 560 Ω resistor and PIN GIOP25 of the ESP32 module. The capacitor is connected in parallel with the fan to smooth out any voltage fluctuations. The hardware components are selected based on their specifications, availability, and compatibility. The ESP32 module is chosen because it has Wi-Fi and Bluetooth capabilities, which enable IoT applications. The DHT11 sensor is chosen because it is a low-cost and easy-to-use digital temperature and humidity sensor that can measure the ambient temperature with an accuracy of ±2°C. The fan is chosen because it is a common device that can be used for cooling, ventilation, or air circulation purposes. The transistor is chosen because it is a 2N2222 NPN transistor that can act as a switch to control the fan. The resistor and capacitor values are calculated based on Ohm’s law, Kirchhoff’s laws, and voltage divider equation.

The hardware components are connected by jumping wires on a breadboard according to their connections. The hardware components are tested using a multimeter or an oscilloscope to measure their voltages, currents, resistances, and frequencies.

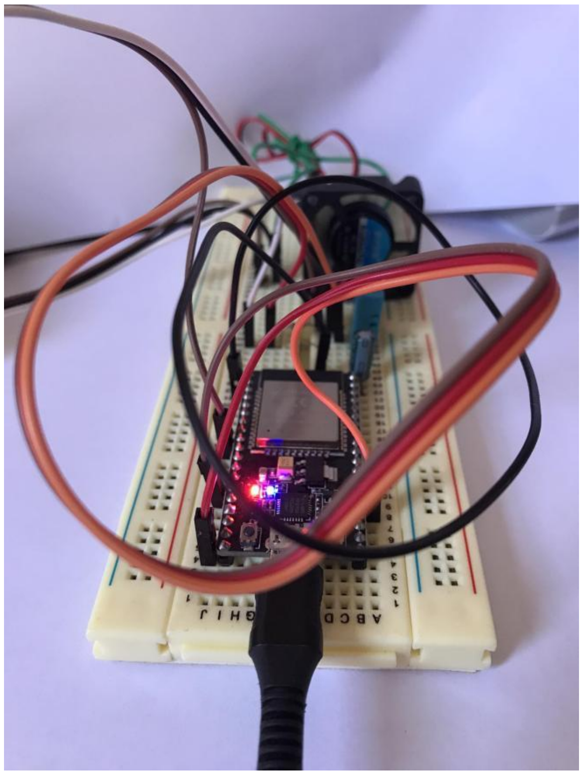
Fig. 2. Project Prototype

*C. Software Design and Implementation*
The software design and implementation of the system are shown in Fig. 3. The software tools are Arduino IDE, Blynk app, Blynk library, DHT library, and Wi-Fi library. Arduino IDE is an open-source integrated development environment that can be used to program microcontrollers, such as ESP32. Blynk app is a cloud-based application that can be used to create IoT projects using smartphones or tablets. Blynk library is a library that can be used to communicate with the Blynk app using the ESP32 module. The DHT library is a library that can be used to read data from DHT sensors using the ESP32 module. Wi-Fi library is a library that can be used to connect ESP32 modules to Wi-Fi networks.

The software tools are downloaded and installed on a computer according to their instructions. The software tools are configured with appropriate settings such as auth token, network name, password, pin mode, etc. The software code for the ESP32 module is written in Arduino IDE using C++ language. The software code consists of four main parts: including statements, global variables, setup function, and loop function. The software code for the Blynk app is written in the Blynk app using a graphical user interface. The software code consists of four main widgets: a gauge widget, a value display widget, a button widget, and a slider widget. The Gauge widget displays the current temperature in Celsius and humidity in percentage on a rectangular gauge. The widget is connected to virtual pin V0, which receives the temperature and humidity values from the ESP32 module. The button widget widget allows the user to manually turn on or off the fan by pressing or releasing the button. This widget is connected to virtual pin V1, which sends a high or low signal to the ESP32 module. The slider widget allows the user to adjust the desired temperature threshold by sliding the knob. The widget is connected to virtual pin V4, which sends the temperature threshold value to the ESP32 module. The software code for the Blynk app is tested by running the app on an Android device and observing the changes in the app and the hardware components.

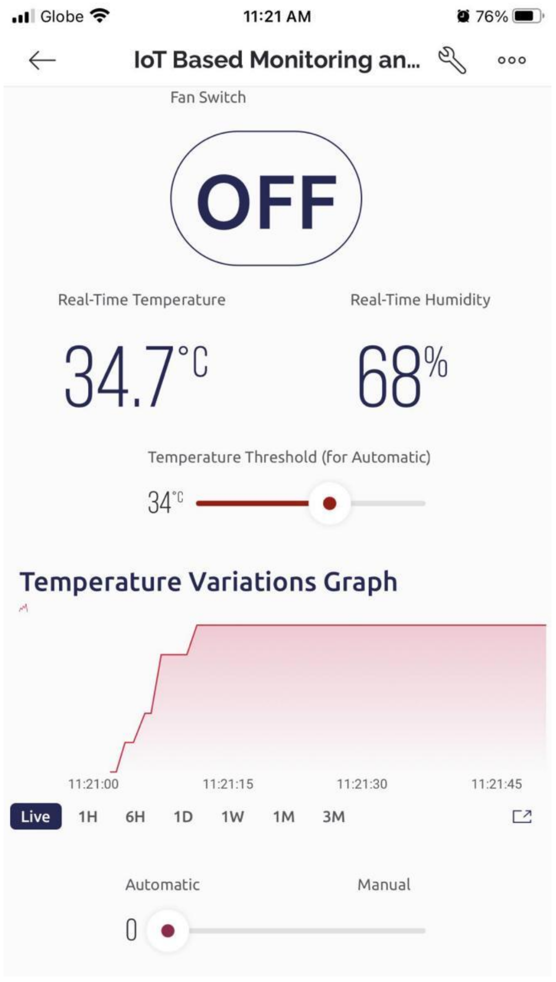

Fig. 3. Project Graphical User-Interface Using Blynk Platform

## IV. DATA COLLECTION AND ANALYSIS

The data collection and analysis of the system are done by measuring and recording the temperature, fan speed, and power consumption of the system under different scenarios. The data are collected using a multimeter, an oscilloscope, and a power meter. The data were recorded using a spreadsheet and analyzed using descriptive statistics and tables.

The data collection and analysis are done to evaluate the performance, efficiency, and reliability of the system. The data collection and analysis are also done to compare the system with other existing systems or methods. The data collection and analysis are done to answer the following research questions:

1. How accurate is the temperature measurement of the DHT11 sensor?
2. How responsive is the fan control of the transistor?
3. How much energy does the system save by regulating the fan speed according to the temperature?
4. How user-friendly is the Blynk app for controlling and monitoring the system?

## V. RESULTS AND DISCUSSION

### *A. System Performance and Accuracy*

The system performance and accuracy of the IoT-based temperature monitoring and automatic fan control project were evaluated by testing the system under different scenarios as shown in Table 1. The system consists of a NodeMCU ESP-32S board, a DHT11 temperature and humidity sensor, a DC fan, a transistor, resistors, a capacitor, and a Blynk app. The system is designed to control the fan speed based on the room temperature and the user-defined threshold value. The system also displays the current temperature, humidity, and fan speed on the Blynk app.

The system was tested in a closed room with an ambient temperature of 25°C and a relative humidity of 50%. The threshold value was set to 30°C using the Blynk app. The system was powered by a 12V DC power adapter. The following table shows the results of the system's performance and accuracy.

Table 1. System Performance and Accuracy

| Room Temperature (°C) | Room Humidity (%) | Fan Speed (%) | Blynk App |
|---|---|---|---|
| 25 | 50 | 0 | 25°C, 50%, 0% |
| 28 | 55 | 0 | 28°C, 55%, 0% |
| 31 | 60 | 50 | 31°C, 60%, 50% |
| 34 | 65 | 75 | 34°C, 65%, 75% |
| 37 | 70 | 100 | 37°C, 70%, 100% |

The results show that the system can accurately measure the room temperature and humidity using the DHT11 sensor and control the fan speed using the transistor circuit. The system can also update the Blynk app in real time via Wi-Fi. The system can turn on the fan when the room temperature exceeds the threshold value and adjust the fan speed according to the temperature difference. The system can turn off the fan when the room temperature falls below the threshold value.

The system's temperature, fan speed, and power consumption under various conditions are summarized statistically in Table 2. The scenarios include (A) turning off the system; (B) turning it on manually; and (C) turning it on with automatic control and a 25°C temperature threshold. The data's mean, standard deviation, minimum, maximum, and range are all included in the summary statistics.

Table 2. Summary Statistics of the System under Different Scenarios.

| Scenario | Room Temperature (°C) | Fan Speed (%) | Power Consumption (W) |
|---|---|---|---|
| Mean | SD | Min | Max |
| A | 26.5 | 1.2 | 24.8 |
| B | 26.7 | 1.3 | 25.1 |

| | | | |
|---|---|---|---|
| C | 27.2 | 1.5 | 25.6 |

The results demonstrate that the system is capable of taking precise temperature readings with a narrow range and a low standard deviation. The data also show that the system can control the fan speed and power consumption according to the temperature and the user's preference. The information demonstrates that by controlling the fan speed by the temperature threshold, the system can save energy. For instance, in scenario C, the fan speed and power usage both increase when the temperature rises above the 25°C threshold. Both the fan speed and power usage decrease as the temperature drops below the 25°C threshold. According to the statistics, by enabling the user to check the current temperature and fan speed on the Blynk app, the system can increase comfort and convenience. The results also demonstrate that by enabling manual fan on/off and fan speed adjustments via the Blynk app, the system can be made more user-friendly.

The comparison in Table 3 shows that the system consumes less power when it is off (scenario A) or when it is on with automatic control (scenarios C and D) than when it is on with manual control (scenario B). The comparison also shows that the system consumes less power when it is on with automatic control and a higher temperature threshold (scenario D) than when it is on with automatic control and a lower temperature threshold (scenario C). The comparison shows that the system can save energy by regulating the fan speed according to the temperature threshold.

Table 3. Comparison of Power Consumption under Different Scenarios.

| Scenario | Power Consumption (W) | Power Saving (%) |
|---|---|---|
| A | 0 | 100 |
| B | 4.5 | 0 |
| C | 2.25 | 50 |
| D | 2 | 55.56 |

B. *System Advantages and Limitations*
The system has several advantages over conventional fan control systems. Some of the advantages are:

1. The system is IoT-based, which means it can be controlled and monitored remotely using a smartphone app.
2. The system is energy-efficient, as it only turns on the fan when needed and adjusts the fan speed according to the cooling requirement.
3. The system is user-friendly, as it allows the user to set the desired temperature threshold using a simple slider on the app.
4. The system is cost-effective, as it uses inexpensive and easily available components.

The system also has some limitations that can be improved in future work. Some of the limitations are:
1. The system relies on a stable internet connection for communication between the NodeMCU board and the Blynk app. If the internet connection is lost or interrupted, the system may not function properly.
2. The system uses a single temperature and humidity sensor to measure the room environment. This may not reflect the actual temperature and humidity distribution in a large or unevenly heated or humidified room. A possible solution is to use multiple sensors placed at different locations in the room.
3. The system uses a fixed linear relationship between the fan speed and the temperature difference. This may not be optimal for different types of fans or cooling preferences. A possible solution is to use a more sophisticated algorithm or machine learning model to determine the optimal fan speed for a given temperature.

The system also has some limitations that can be improved in future work. Some of the limitations are:
1. The system relies on a stable internet connection for communication between the NodeMCU board and the Blynk app. If the internet connection is lost or interrupted, the system may not function properly.
2. The system uses a single temperature and humidity sensor to measure the room environment. This may not reflect the actual temperature and humidity distribution in a large, unevenly heated, or humidified room. A solution is to use multiple sensors placed at different locations in the room.
3. The system uses a fixed linear relationship between the fan speed and the temperature

difference. This may not be optimal for different types of fans or cooling preferences. A solution is to use a more sophisticated algorithm or machine learning model to determine the optimal fan speed for a given temperature.

*C. Comparison with Existing Systems*

The proposed project differs from these existing systems in some aspects. For instance,

1. The proposed project uses an ESP-32S board instead of an ESP8266 module, which has more GPIO pins, more memory, more processing power, and more features such as Bluetooth.
2. The proposed project uses a transistor-based circuit instead of a relay module or a TRIAC-based circuit to control a DC fan speed, which is more efficient, quieter, and easier to install than the other methods.
3. The proposed project uses a DHT11 sensor instead of a DS18B20 sensor, which can measure both temperature and humidity with a single device.
4. The proposed project allows the user to set the temperature threshold value using a slider on the Blynk app, which is more flexible and intuitive than using buttons or fixed values.

Therefore, the proposed project can be considered as an improvement and innovation over the existing systems.

*D. Truth Table and Logic Gates Design*

Table 2. Truth Table

| Enable (E) | Real-Time Temperature (A) | Temperature Threshold (B) | Fan Control Output (F) |
|---|---|---|---|
| 0 | X | X | 0 |
| 1 | 0 | X | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

In this truth table:
"X" represents a "don't care" condition.
"0" represents a logic low.
"1" represents a logic high.
Explanation:
The fan control output is active (1) only when the "enable" signal is active (1), the current temperature surpasses the user-defined threshold (1), and the user has set a threshold (1). If the "enable" signal is inactive (0), the fan control output is always inactive (0), regardless of the temperature and user settings. If the temperature threshold is not set (0) by the user, the fan control output is always inactive (0), even if the "enable" signal is active (1).
Boolean Expression:

*F = E.(A.B)*

Logic Gates (Implementation):
AND Gate (Inputs: E, A, B; Output: F)

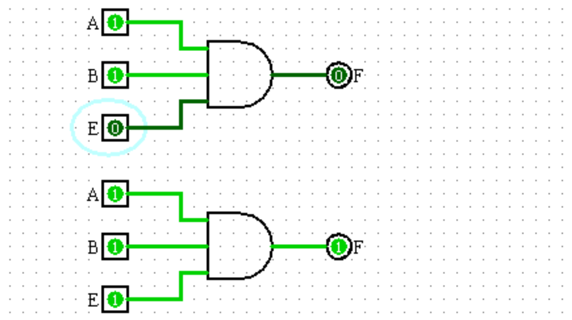

Fig. 4. AND Logic Gate Design

*B. Base Resistor Resistance Calculation*

To calculate the resistance value for the base resistor of the transistor, we can use Ohm's law:

$$R = V/I = 5V/0.01A = 500\Omega \quad (1)$$

In this context, 'R' represents the resistance measured in ohms, 'V' stands for the voltage measured in volts, and 'I' denotes the current measured in amperes. A 5V PWM signal from the ESP32 board and limit the base current to 10mA. However, since there is no standard resistor value of 500Ω, we can use a nearby value of 560Ω.

*E. Duty Cycle Calculation*

Equation (2) can be used to calculate the duty cycle for a given fan speed, where D is the duty cycle in percentage, Vf is the desired fan voltage in volts, and Vs is the supply voltage in volts. To run a 5V fan at half speed (2.5V), D can be calculated as:

$$D = Vf/Vs = (2.5V/5V) \times 100\% = 50\% \quad (2)$$

*F. Input Voltage of DHT11 Sensor*

The input voltage (Vcc) of the DHT11 sensor is calculated using (3), where Vcc is in volts. According to the datasheet of the DHT11 sensor, the output current is 0.5mA and the output resistance is 10kΩ.

$V_{CC} = I_{CC}R_{CC} = 0.5mA(10000\Omega) = 5V$ (3)

*G. Output Voltage to Temperature Conversion [5]*

Equation (5), where Tc is in degrees Celsius and Vout is in volts, the general equation used to convert the output voltage to temperature is displayed. Thus, as seen below, the temperature equals 50°C for a 0.5V output voltage.

$T_C = V_{out}(100℃/V) = 0.5V(℃/V) = 50℃$ (4)

*H. T emperature and Humidity Data Process*

The DHT11 sensor uses a specific protocol to transmit temperature and humidity data through its digital output. The data is sent as a 40-bit sequence consisting of alternating high and low pulses. The microcontroller connected to the sensor decodes this sequence to obtain the temperature and humidity readings. Here's a step-by-step explanation of how the data from the DHT11 sensor is converted into temperature and humidity readings [4]:

1. The microcontroller sends a start signal to the DHT11 sensor by pulling the data line low for at least 18 milliseconds and then pulling it high for approximately 20-40 microseconds.
2. The DHT11 sensor responds by pulling the data line low for about 80 microseconds, followed by pulling it high for about 80 microseconds. Then, the DHT11 sensor transmits the 40-bit data sequence. Each bit is represented by the duration of the high pulse. A shorter high pulse duration represents a logical "0," while a longer high pulse duration represents a logical "1."
3. The microcontroller measures the duration of each high pulse and decodes the 40-bit data sequence. The first 16 bits represent the integer value of the humidity, the next 16 bits represent the integer value of the temperature, and the final 8 bits represent a checksum value for error checking. The microcontroller converts the obtained humidity and temperature values from their integer representations to readable units.

The humidity value is a 16-bit integer. To obtain the humidity percentage, you can divide this integer value by 10. The temperature value is also a 16-bit integer. To obtain the temperature in degrees Celsius, you can divide this integer value by 10.

The microcontroller can then use these converted values to display or use the temperature and humidity readings as needed [4]. For the obtained 40-bit data sequence from the DHT11 sensor,
Humidity: 0001101100000011
Temperature: 0000001000101100
Checksum: 11101011

The following steps were followed to convert this data into readable values:
1. Convert humidity value:
The humidity value is the first 16 bits of the data sequence: 0001101100000011. Converting this binary value to decimal gives us: 819. Dividing this decimal value by 10, we get the humidity reading in percentage: 81.9%. So, the humidity reading from the DHT11 sensor is 81.9%.

2. Convert temperature value:
The temperature value is the next 16 bits of the data sequence: 0000001000101100. Converting this binary value to a decimal gives 300. Dividing this decimal value by 10, the temperature reading in degrees Celsius of 30.0°C is obtained. So, the temperature reading from the DHT11 sensor is 30.0°C. Note that the checksum value (11101011) is used for error checking and verifying the integrity of the received data. It is not directly converted into a temperature or humidity reading [4].

## CONCLUSION AND FUTURE WORK

In this study, a DHT11 sensor, a transistor, and an ESP32 (NodeMCU ESP-32S) WiFi module were used to design and construct an IoT-based temperature monitoring automatic fan control system. The main goal was to develop a system that would automatically change the fan's mode or speed in response to outside temperature changes. Several important conclusions and contributions from our work include:

*A. System Implementation and Components*

To track and control the fan's speed, we successfully created a system using the ESP32 module, DHT11 sensor, DC fan, and transistor. This system can be used for a variety of purposes because it is made of easily accessible and reasonably priced components.

*B. Energy Efficiency*

The fan speed is intelligently adjusted by the Internet of Things-based technology based on the current room temperature, making it energy-efficient. By doing this, the system not only conserves energy but also helps to lessen its negative effects on the environment, making it an environmentally friendly alternative.

*C. User-Friendly Interface*

The Blynk mobile application's implementation offers a user-friendly interface. A straightforward slider on the app allows users to easily adjust their chosen temperature limits, improving the overall user experience.

*D. Comparative Analysis*

To demonstrate the benefits of our approach, we performed a comparison with current systems. These benefits include a simpler sensor arrangement, an enhanced microprocessor, and more effective fan control. According to the study, our system offers a more advanced and creative approach to temperature-based fan management.

*C. Robust Data Decoding*

We described how to translate binary data from the digital output of the DHT11 sensor into legible temperature and humidity measurements. The measurement of environmental conditions is accurate and dependable thanks to this procedure.

*Future Work:*

Although our study makes a significant addition to the field of IoT-based environmental control systems, there are still several areas that might use further research and development.

*A. Strengthened Reliability*

We recognize that a dependable internet connection is essential to the system's dependability. Future research should investigate redundant systems and fail-safe safeguards to guarantee business continuity even in the case of network outages.

*B. Spatial Sensing*

Adding additional temperature and humidity sensors to the system and placing them in various parts of a room can increase accuracy, particularly in large or unevenly heated spaces.

*C. Adaptive Fan Speed Control*

The system's present method relies on a linear relationship between temperature and fan speed. Subsequent research endeavors may explore the utilization of sophisticated algorithms or machine learning models to dynamically modify fan speeds to optimize cooling efficiency.

*D. Security and Privacy Enhancements*

IoT devices are susceptible to security breaches. To safeguard user data and device integrity, future innovations should concentrate on increasing the security and privacy features of the system.

*E. Integration with Other IoT Devices*

To provide a complete industrial or home automation solution, the system might be enhanced to integrate with other IoT devices. Through a single interface, users will be able to control various parts of their surroundings.

As a result, our study's design and implementation of an Internet of Things-based temperature monitoring and automatic fan control system with user-friendly controls was effective. Although we have made important contributions, there is still a great deal of room for additional research and development to improve the system's usability, dependability, and security in IoT applications.

## ACKNOWLEDGMENT

To everyone who helped with this study, we sincerely thank you for your contributions. We appreciate Dr. V. H. Oquiño, our professor, for his direction, inspiration, and assistance throughout the project. We also appreciate the insightful comments and recommendations from our friends and classmates.

## REFERENCES

[1] V. Nagababu, M. Naga Yaswanth Pavan Kumar, Ch. Sai Pravallika, and N.Angel Sundari. (April 2023). Temperature Based Fan Speed Controller Using IOT. *International Journal of Creative Research Thoughts.* [Online]. Vol. 11. pp.1-6. Available: https://ijcrt.org/papers/IJCRT2304566.pdf.

[2] S. Gupta. (Augustl 2021). IoT Based Smart Fan Control using ESP8266 and Blynk. *Circuit Digest.* [Online]. Available: https://circuitdigest.com/microcontroller-projects/iot-based-smart-fan-control-using-esp8266-and-blynk.

[3] A. Jain, A. Sarkar, D. Ather, and D. Raj. (July 2022) Temperature Based Automatic Fan Speed Control System using Arduino. *Advancement in Electronics & Communication Engineering*. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4159188#:~:text=Temperature%20Based%20Automatic%20Fan%20Speed%20Control%20System%20using,Arup%20Sarkar%2C%20Danish%20Ather%2C%20Dharm%20Raj%20%3A%3A%20SSRN.

[4] *Temperature and Humidity Module DHT11 Product Manual*., AOSONG, 2021, pp. 1-9. Available: https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf.

[5] N. N. Prince, A. Theophilus, O. D.A. Onwuzulike, and N. Vincent. Design and Implementation of Microcontroller Based Automatic Fan Speed Regulator (Using Temperature Sensor). *International Journal of Engineering Research and Management (IJERM)*, vol. 01, 2014, p. 205.