Performance evaluation of DHT11 using ESP32

Rakhi Yadav[a] and Geetam Shukla[b*]

[a,b]Centre for Internet of Things, Madhav Institute of Science & Technology, Gwalior, Madhya Pradesh (India)

*Corresponding Author: geetamrichhariya@gmail.com (Geetam Shukla)

Rakhi Yadav

yadav.rakhi4321@gmail.com

1. Introduction

Espressif Systems, a Chinese company founded in 2008, focuses on developing affordable and high-performance microcontrollers for the growing Internet of Things (IoT) market. Their popular products are the ESP8266 and ESP32, which are widely used in IoT applications [1]. The company gained widespread recognition with the release of the ESP8266 in 2014, a low-cost Wi-Fi-enabled microcontroller that became popular for IoT projects. Building on this success, Espressif introduced the ESP32 in 2016, a significant upgrade featuring dual-core processing, integrated Wi-Fi, Bluetooth Low Energy (BLE), more I/O options, and low-power modes, making it ideal for a wide range of IoT applications [2].

Over the following years, the ESP32's ecosystem expanded with tools like the ESP32-DevKit and support for platforms like Arduino. In 2020, Espressif launched the ESP32-S2, a single-core variant with enhanced security features, followed by the ESP32-S3 in 2021, which introduced hardware accelerators for AI and machine learning tasks. The ESP32-C3, released in 2021, embraced the RISC-V architecture, offering improved efficiency at a lower cost, while the ESP32-H2 added support for Zigbee and Thread for smart home applications. In 2023, Espressif unveiled the ESP32-P4, focusing on high-performance computing, moving beyond wireless communication and into edge computing and AI [3].

The ESP32, designed by Espressif, is a microcontroller. A microcontroller is a small integrated circuit capable of running programs. It acts as a self-contained Wi-Fi networking solution, meaning it can connect to Wi-Fi and handle tasks independently, or it can work alongside other microcontrollers to enable Wi-Fi connectivity. ESP32 is a low-cost MCU with integrated Wi-Fi and BLE. Various modules and development boards-based ESP32 are available to build

Internet-of-Things (IoT) applications easily. Wi-Fi and BLE are common network stacks in IoT applications. These network modules can fulfil the business and project needs, providing cost-effective benefits. It also supports Bluetooth Low Energy (BLE), making it versatile for many wireless communication needs [4].

The ESP32 is available in two forms: chip form and module form. Both have different sizes and pin configurations. Choosing between them depends on project design, as the size and pin count may affect how you use the ESP32 in your custom PCBs (Printed Circuit Boards) [5].

Espressif has also released a development board called the "ESP32-DevKit" to make the ESP32 more accessible to users. This board is easy to use with a breadboard, includes a micro USB port for power and programming, and has two buttons: Enable and Boot. These buttons allow user to load or flash new programs onto the ESP32 [6]. Since the ESP32 is primarily a Wi-Fi device, user will need to configure it to connect to a network. This involves setting up basic information like the network name (SSID) and password (SSID Password). If you want the ESP32 to act as a Wi-Fi access point (creating its own network) or load custom applications onto it, there are additional configurations and steps involved.

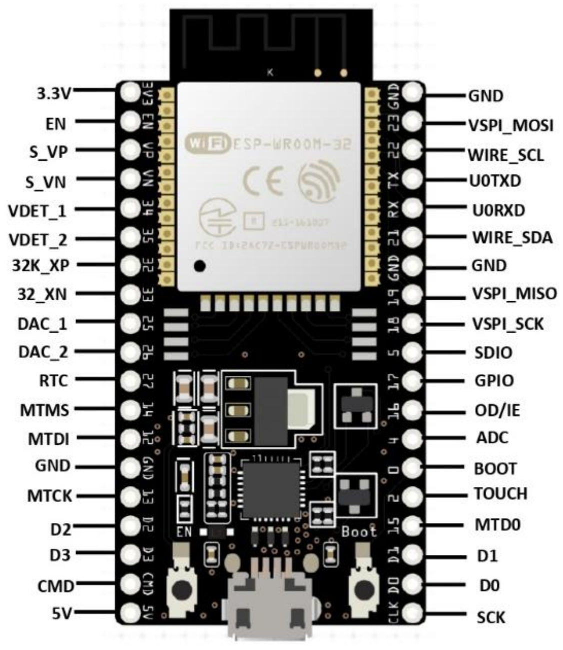


Fig.1. Pin configuration diagram of ESP32

Table 1

Advantages and Disadvantages of ESP32

Parameters	Advantages of ESP32	Disadvantages of ESP32
Performance	ESP32 features a dual-core that allows for efficient multitasking.	It can introduce complexity in programming & resource management.
Low Power Consumption	It has several sleep modes to minimize power usage.	It can perform limited tasks in low power modes.
Connectivity	Supports both Wi-Fi and Bluetooth.	With more Wi-Fi networks, it can experience interference, that affects the performance.
GPIO and Peripheral Support	Up to 34 GPIO pins with various functionalities (ADC, DAC, PWM, etc.)	Some pins have multiple functions, which can complicate designs if not managed carefully.
Development and Programming	Supports multiple programming environments like Arduino IDE, ESP-IDF, and MicroPython, making it accessible for both beginners and advanced users.	Its development and programming can be complex due to multitasking, a steep learning curve, debugging challenges, and resource constraints.
Cost	ESP32 is relatively inexpensive compared to other microcontrollers.	The low cost can sometimes mean lower-quality.
Size and Form Factor	The ESP32 module is small and suitable for compact applications.	The small size can lead to challenges in thermal management, component accessibility, and overall design complexity.
Memory	With 520 KB of SRAM and up to 16 MB of external flash, it can handle complex IoT applications and communication protocols.	520 KB may not be enough for memory-intensive applications, leading to overflow issues.
Rich Peripheral Support	It supports a variety of peripherals, including SPI, I2C, UART, ADC, and DAC, facilitating easy integration with sensors and other devices.	ESP32's peripheral support may have some challenges such as pin multiplexing, resource contention, inconsistent driver support, and complexity in development.
Security Vulnerabilities	The security features of the ESP32, including secure boot, flash encryption, etc. make it a strong choice for IoT applications where data integrity and protection are essential.	ESP32 is susceptible to security vulnerabilities, requiring careful consideration of security measures.
Comparative Analysis	The ESP32 excels in multitasking, connectivity, and peripheral support, making it versatile for IoT applications.	The complexity, power consumption, and limited RAM can be drawbacks compared to simpler or more specialized microcontrollers.

2. Platforms Overview

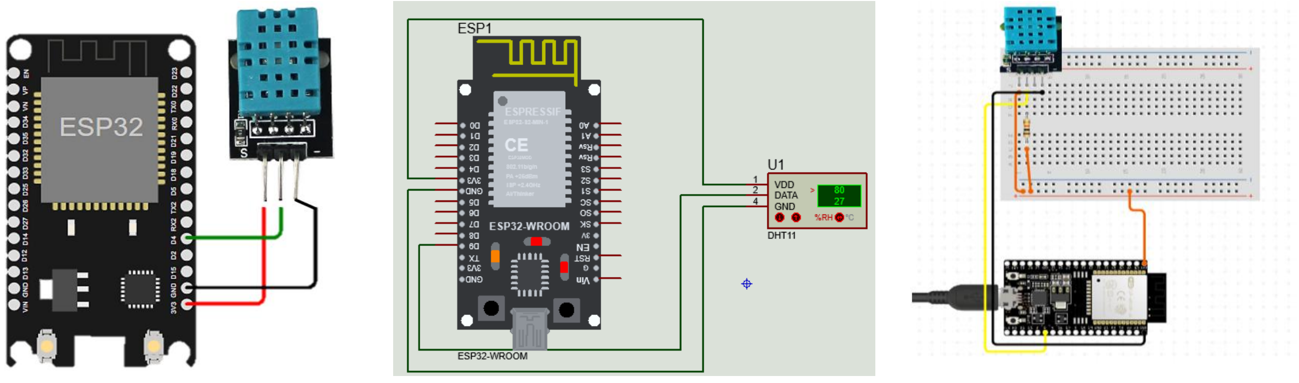
There are various platforms available for designing circuit diagrams and simulating electronic systems, enabling users to visualize and test their circuits before implementation. Wokwi, Proteus, and Circuit.io are some of them that can be used for the simulation of the circuit diagram. These all offer unique features tailored to different audiences within the electronics community. Wokwi is optimal for rapid prototyping and simple simulations, making it ideal for beginners and those looking to validate concepts quickly. Proteus offers a powerful environment for more complex designs, suitable for engineers and advanced users who need detailed simulations and PCB design tools. Circuit.io excels in educational settings, offering an accessible platform for students and hobbyists to learn and explore basic electronics and microcontroller projects.

Wokwi is an innovative online simulator specifically designed for microcontroller projects, including popular platforms such as the ESP32, Arduino, and a variety of sensors like the DHT11. One of its standout features is its user-friendly interface, which requires no installation; users can access the platform directly through their web browser. This makes Wokwi particularly appealing for rapid prototyping, allowing users to quickly test ideas and concepts without the need for physical components. Wokwi's free tier is highly accessible, enabling users to run basic simulations without incurring any costs. Users can write Arduino or C code and simulate the behavior of their circuits in real-time, facilitating an iterative design process where modifications can be made instantaneously.

However, while Wokwi excels in simplicity and ease of use, it has limitations. The platform primarily focuses on basic simulations and does not offer advanced features such as printed circuit board (PCB) design or the ability to work with highly specialized or complex components. This makes it less suitable for professional engineers or projects that require in-depth circuit analysis or extensive component support.

Proteus is a professional-grade desktop software that stands out for its comprehensive set of simulation tools designed for engineers and advanced users. It provides powerful features for simulating microcontrollers, including the ESP32, and offers extensive circuit simulation capabilities, PCB design functionalities, and a vast library of electronic components. This versatility allows users to not only simulate the behavior of their circuits but also design and layout PCBs within the same environment, streamlining the development process. One of Proteus's key strengths is its ability to handle complex projects, making it suitable for professionals working on sophisticated designs in fields such as robotics, automation, and embedded systems. The software supports various programming languages and can simulate both analog and digital circuits, providing users with a realistic representation of their designs before physical implementation. Proteus requires significant system resources, necessitating a robust computer for optimal performance. Despite these drawbacks, Proteus is an invaluable tool for engineers and advanced users who require a comprehensive and professional solution for circuit design and simulation.

Circuit.io, part of Tinkercad Circuits, is a web-based platform designed to simplify the process of breadboard circuit design and microcontroller projects, making it particularly suited for beginners and educational purposes. Its intuitive interface allows users to drag and drop components onto a virtual breadboard, making it easy to visualize and construct circuits without the need for prior electronics experience. The platform also provides a straightforward way to program microcontrollers, facilitating an accessible entry point for those new to electronics and IoT.



(a) Wokwi **(b)Proteus** **(c)Circuit.io**

Fig.2. Various Simulation Platforms

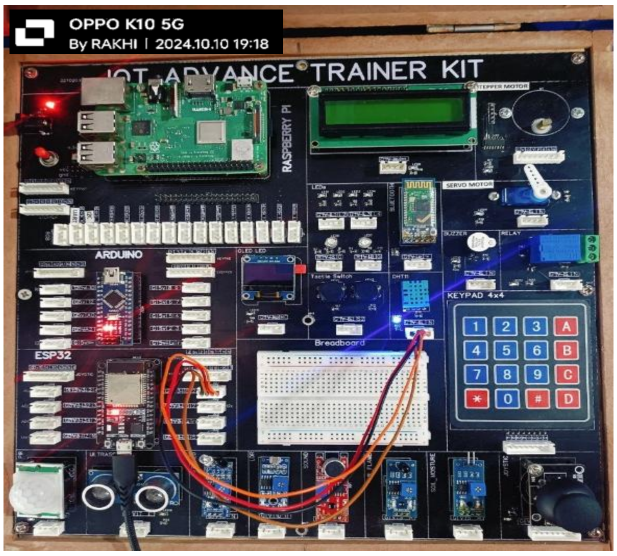


Fig. 3. IoT Trainer Kit

Selecting the right platform depends on the user's specific needs, experience level, and project requirements. By understanding the strengths and limitations of each tool, users can make informed decisions to enhance their learning and development in the ever-evolving field of electronics and IoT.

Table 2

Advantages & Disadvantages of Simulating Platforms

Wokwi	Proteus	Circuit.io
Wokwi is an online simulator.	Proteus is a desktop-based simulation software.	Circuit.io is an online tool.
It primarily focused on microcontroller-based projects like Arduino, ESP32, and ESP8266.	It offers a full suite for circuit simulation, microcontroller testing, and PCB design.	It focuses on simple circuit design and simulation, aimed at beginners.
No installation needed; runs in the browser.	Highly accurate simulations for circuits, including microcontrollers like ESP32.	Easy to access without installation.
Supports ESP32 and a variety of sensors, including the DHT11.	Supports circuit design, PCB layout, and analysis in one software.	Very intuitive for beginners, ideal for hobbyists or educational purposes.
Simple interface, easy to set up simulations.	Includes thousands of electronic components and sensors.	Focuses on breadboard layouts, allowing users to simulate wiring of components and microcontrollers.
You can explore and clone various public projects.	Better visualization of real-world scenarios, such as how components behave under different conditions.	Offers a 3D view for components, which is useful for understanding real-world designs.
Free tier available with many features, with a paid tier for advanced features.	It’s a paid software, and the pricing can be high for individual users.	Not as detailed or realistic as Proteus, especially when it comes to complex simulations.
Allows users to directly run the Arduino/C code, making it great for debugging.	Steeper learning curve compared to platforms like Wokwi and circuit.io.	It is more Arduino-centric and lacks full support for ESP32.
Not all sensors or modules are available.	Community support is not as widespread as web-based platforms.	The component library, while good for basic projects, is more limited than Proteus or Wokwi.
It focuses on circuit simulation, with no support for PCB design or layout.	Requires installation and is resource-intensive.	Not as detailed or realistic as Proteus, especially when it comes to complex simulations.
Limited when compared to full desktop applications like Proteus in terms of accuracy and detail.	Professional use for PCB design and microcontroller programming.	Prototyping hobby projects with limited complexity.

3. Research Methodology

This research paper focuses on evaluating the performance of the DHT11 sensor in measuring temperature and humidity. The DHT11 is a low-cost, digital sensor commonly used in IoT projects and environmental monitoring systems to measure the temperature and humidity of surroundings. It operates within a temperature range of 0°C to 50°C and a humidity range of 20% to 90% RH. The sensor is composed of a thermistor that measures temperature and a capacitive humidity sensor to detect humidity. One of its advantages is that it communicates through a single-wire protocol, which simplifies its integration with microcontrollers like the ESP32. Its operating voltage is between 3.3V and 5.5V, and it takes readings once per second. Its compact design, low power consumption, and easy-to-use communication protocol make it a popular choice for low-cost and simple projects.

However, the DHT11 has some limitations, such as lower accuracy, a narrow measurement range, and a slower response time compared to more advanced sensors. It is less accurate than other sensors like the DHT22 and BME280, but it is more affordable. Despite its limitations, the DHT11 is widely used in applications like home automation, weather stations, agricultural monitoring, and greenhouse control, where basic environmental measurements are sufficient. While it's not suitable for tasks requiring high accuracy or fast data collection, its simplicity and low cost make it an excellent option for beginners and projects with basic requirements.

The ESP32 microcontroller is used to interface with the DHT11 sensor in this research. The ESP32 is a powerful and versatile microcontroller developed by Espressif Systems. It is widely used in IoT projects and other applications requiring wireless connectivity because it has built-in Wi-Fi and Bluetooth. The ESP32 is powered by a dual-core processor that runs up to 240 MHz, making it capable of handling multiple tasks like data collection and communication at the same time. It has 520 KB of SRAM, which gives it enough memory to manage complex tasks efficiently.

One of the standout features of the ESP32 is its ability to connect to Wi-Fi, enabling devices to communicate over the internet. Its Bluetooth capability, which includes Bluetooth Low Energy (BLE), makes it useful for short-range, low-power communication, especially for wearable devices or personal area networks. The low power consumption of the ESP32 is another key feature, as it includes modes like deep sleep, where the power consumption can be reduced to microamperes (µA). This makes the ESP32 a suitable choice for battery-powered or remote monitoring systems, where minimizing power usage is crucial for extended operation without frequent recharging.

The ESP32 also supports a wide range of peripherals. It has GPIO (General Purpose Input/Output) pins that allow it to interface with sensors, motors, and other components. It also includes ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), PWM (Pulse Width Modulation), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), and UART (Universal Asynchronous Receiver-Transmitter) communication protocols, making it flexible for various projects. Additionally, it has built-in touch sensors that can be used for touch-based interfaces.

The ESP32 is also flexible when it comes to programming. It can be programmed using platforms like the Arduino IDE, which is easy to use for beginners, or the more advanced ESP-IDF (Espressif IoT Development Framework). It also supports MicroPython, which allows for Python-based scripting, and PlatformIO, a more professional environment for embedded systems development. This versatility makes the ESP32 suitable for both beginners and experienced developers.

Due to its powerful features, the ESP32 is used in a variety of projects, including IoT devices, smart home automation, wearable technology, and environmental monitoring systems. Its combination of processing power, wireless connectivity, low power consumption, and extensive peripheral support makes it one of the most popular microcontrollers for IoT, embedded systems, and wireless communication projects. Although there is a learning curve, especially for beginners, the ESP32's capabilities and flexibility make it a preferred choice for developers.

The Arduino IDE (Integrated Development Environment) is a popular open-source platform used for writing, compiling, and uploading code to microcontroller boards such as Arduino, ESP32, and others. Designed to be user-friendly, the Arduino IDE is an excellent choice for beginners as well as experienced developers working on embedded systems and IoT projects. It provides a simple and clean interface that supports C and C++ programming languages, along with a vast collection of libraries and pre-built functions that simplify complex tasks such as handling sensors, motors, and communication protocols. The IDE includes an easy-to-navigate code editor with features like syntax highlighting and automatic formatting, making the coding experience intuitive. It also features a built-in compiler and uploader, which compiles the code and uploads it directly to the connected hardware via a USB cable. The Arduino IDE supports a wide variety of Arduino boards and third-party microcontrollers like ESP32 through board manager extensions, allowing users to switch between different hardware seamlessly. Additionally, the IDE has a serial monitor that enables real-time debugging and communication with the microcontroller, which is crucial for testing and troubleshooting during development.

One of the key strengths of the Arduino IDE is its large and active community, providing an extensive range of tutorials, examples, and libraries, making it easier to build and customize projects. This simplicity, combined with flexibility and community support, has made the Arduino IDE a cornerstone for IoT, robotics, and embedded systems development worldwide. In this research, some IoT sensors are used to collect data, along with different software tools to set up the environment for performing the experiments. The tools and technologies that are going to be used are discussed, including Arduino IDE for programming and other software tools to develop the circuit diagrams. The main tools we used for building circuit diagrams are Wokwi, Circuit.io, and Proteus. Proteus is an offline software that allows to create circuit diagrams on computer without needing an internet connection. On the other hand, Wokwi and Circuit.io are online platforms. While Circuit.io can be used to design a wide range of circuits, Wokwi is specifically focused on ESP32 and NodeMCU boards, making it ideal for the project. Now, let's go through the steps to set up Arduino IDE for programming.

To set up the Arduino IDE, the first step is to download and install the IDE from the official Arduino website. Once installed, then connect to ESP32 or other microcontroller to the computer using a USB cable. In the IDE, select the correct board (such as ESP32) and port from the Tools menu. For ESP32, install the ESP32 board library through the Board Manager by searching for it and clicking install. After setting up the board and port, start writing code in the Arduino IDE. Once the code is ready, click the Upload button to compile and send it to the microcontroller. The serial monitor can be used to debug and view sensor data in real time. This simple setup enables to quickly start developing and testing IoT projects.

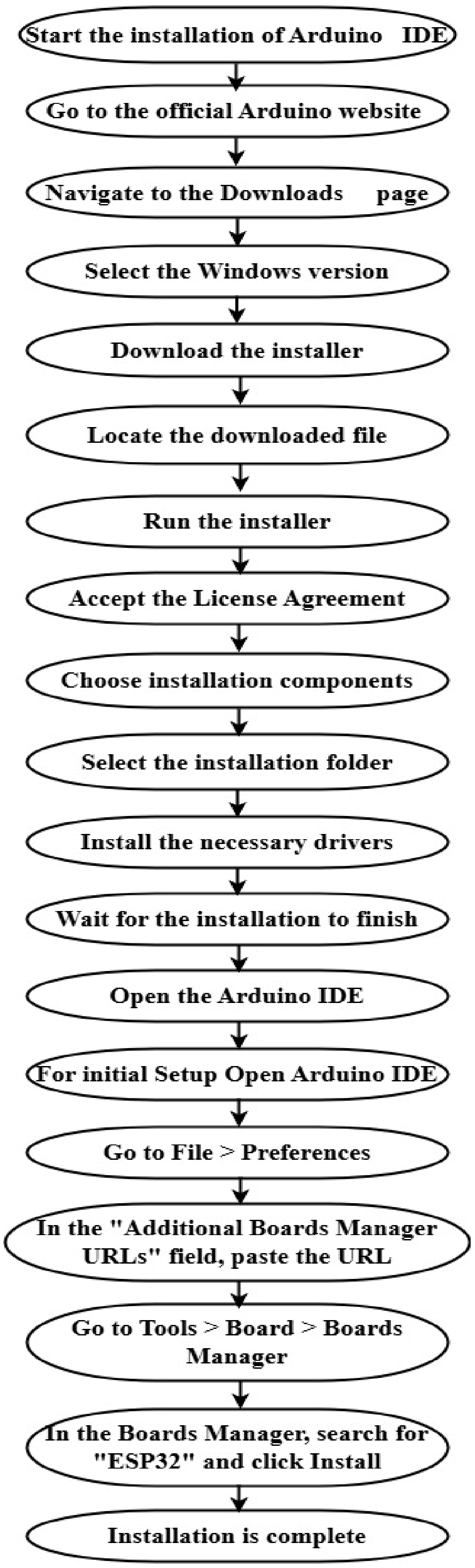
To install Arduino IDE on a Windows system and to set up ESP32 in the Arduino IDE, follow these step-by-step instructions:

Steps to Download & Install Arduino IDE in windows:

Step 1: Go to the official Arduino website at www.arduino.cc and download the Arduino IDE for your operating system (Windows, macOS, or Linux).

Step 2: On the homepage, hover over the Software tab in the menu, and click on Downloads.

Step 3: Scroll down to find the Arduino IDE version for Windows.



Start the installation of Arduino IDE
Go to the official Arduino website
Navigate to the Downloads page
Select the Windows version
Download the installer
Locate the downloaded file
Run the installer
Accept the License Agreement
Choose installation components
Select the installation folder
Install the necessary drivers
Wait for the installation to finish
Open the Arduino IDE
For initial Setup Open Arduino IDE
Go to File > Preferences
In the "Additional Boards Manager URLs" field, paste the URL
Go to Tools > Board > Boards Manager
In the Boards Manager, search for "ESP32" and click Install
Installation is complete

Fig. 4. Flowchart for the installation of Arduino IDE

Step 4: You will find options for both the Windows installer (.exe) and a zip package. It is recommended to download the installer version for ease of installation.

Step 5: Click the Windows Installer option, and the download will begin.

Step 6: After the download is complete, locate the Arduino IDE installer (usually in your Downloads folder).

Step 7: Double-click the installer file to start the installation process.

Step 8: The installation wizard will open. Then first accept the License Agreement by clicking on I Agree.

Step 9: The next window allows to select components to install. It is recommended to leave all the options checked (including the drivers) and click Next.

Step 10: Choose the folder to install Arduino IDE. By default, it will install in C:\Program Files (x86)\Arduino.

Step 11: Click Install to proceed.

Step 12: During installation, Arduino may prompt you to install drivers (especially if it's first time). Accept the installation of all necessary drivers by clicking Install when prompted.

Step 13: The installation process will take a few minutes. Once completed, click Close to exit the installer.

Step 14: After installation, launch the Arduino IDE by double-clicking Arduino icon on your desktop or searching for Arduino IDE in the Start Menu.

Step 15: The IDE will open with a blank sketch (programming window).

Step 16: Now IDE is ready to start programming with Arduino.

Once the installation is complete, the IDE will be ready to use for programming ESP32 boards. With these steps, Arduino IDE will be successfully installed and ready to use on Windows machine.

Once the setup for Arduino IDE is completed, the environment for ESP32 need to be setup. To setup ESP32 in Arduino IDE some steps needs to be followed:

Steps to install and setup ESP32 in the Arduino IDE:

Step 1: Launch the Arduino IDE application after installation.

Step 2: In the top menu, click on File and then select Preferences (or Arduino > Preferences on macOS).

Step 3: This will open the Preferences window.

Step 4: In the Preferences window, find the field labeled Additional Boards Manager URLs.

Step 5: Paste the following URL into the field: https://dl.espressif.com/dl/package_esp32_index.json

Step 6: If you already have other URLs in the field, add a comma, and paste the ESP32 URL next to it.

Step 7: After adding the URL, click OK to save the settings and close the Preferences window.

Step 8: In the top menu, go to Tools > Board > Boards Manager.

Step 9: In the Boards Manager window, type ESP32 into the search bar.

Step 10: A result titled ESP32 by Espressif Systems will be visible. Click the Install button next to it. This will download and install the required files for ESP32 support in the Arduino IDE.

Step 11: Wait for the installation to complete. It may take a few minutes depending on the internet speed.

Step 12: After installing the ESP32 package select the specific ESP32 board.

Step 13: Go to Tools > Board, In the Tools menu, hover over Board.

Step 14: In the list of boards, scroll down until the ESP32 Arduino section is visible. Choose the specific ESP32 board (for example, ESP32 Dev Module).

Step 15: If the ESP32 board doesn't automatically connect to the computer, the installation of drivers may require, a USB-to-UART driver.

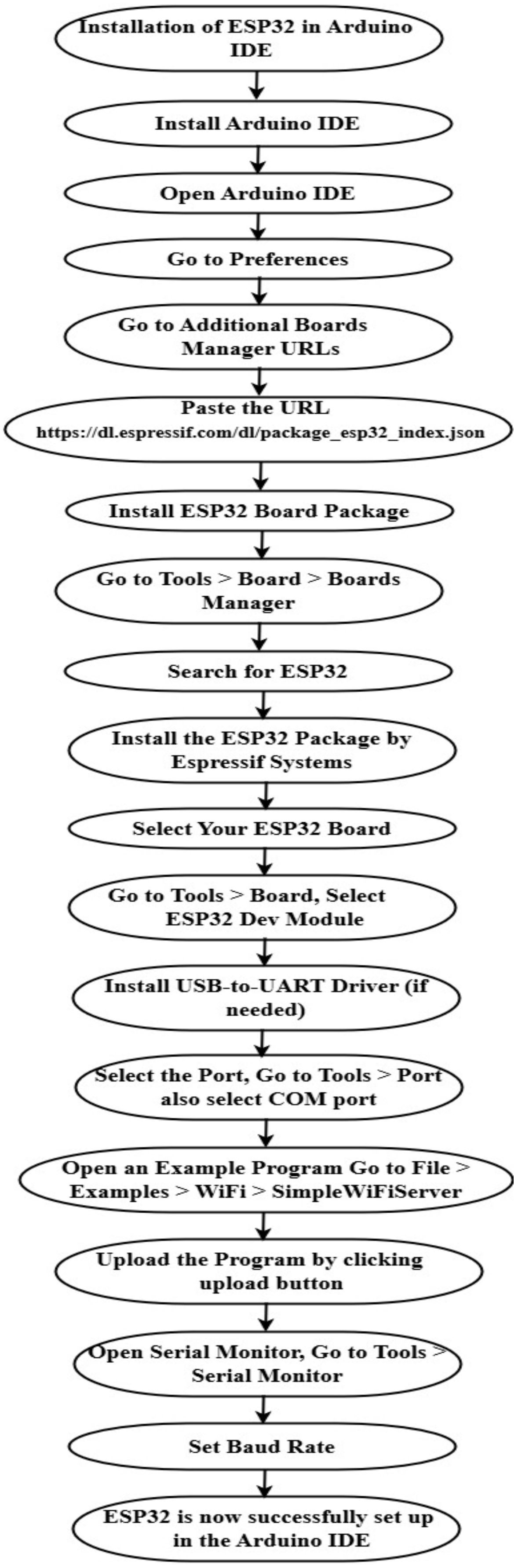

Installation of ESP32 in Arduino IDE
Install Arduino IDE
Open Arduino IDE
Go to Preferences
Go to Additional Boards Manager URLs
Paste the URL https://dl.espressif.com/dl/package_esp32_index.json
Install ESP32 Board Package
Go to Tools > Board > Boards Manager
Search for ESP32
Install the ESP32 Package by Espressif Systems
Select Your ESP32 Board
Go to Tools > Board, Select ESP32 Dev Module
Install USB-to-UART Driver (if needed)
Select the Port, Go to Tools > Port also select COM port
Open an Example Program Go to File > Examples > WiFi > SimpleWiFiServer
Upload the Program by clicking upload button
Open Serial Monitor, Go to Tools > Serial Monitor
Set Baud Rate
ESP32 is now successfully set up in the Arduino IDE

Fig. 5. Flowchart for ESP32 Setup in Arduino IDE

Step 16: Many ESP32 boards use the CP2102 or CH340 USB chips for communication, and install the appropriate drivers:

- **CP2102 Driver**: Available at Silicon Labs.
- **CH340 Driver**: Available at WCH.

Step 17: After connecting the ESP32 board to the computer via a USB cable, go to Tools > Port and select the correct COM port (usually something like COM3 or COM4).

Step 18: Start with an example program to test the board.

Step 19: In the Arduino IDE, go to File > Examples > WiFi > SimpleWiFiServer.

Step 20: Click the Upload button (right arrow) to compile the code and upload it to the ESP32.

Step 21: The status of the upload will be shown at the bottom of the IDE. If everything is set up correctly, it will say Done uploading.

Step 22: After uploading the code, open the **Serial Monitor** to see any output from the ESP32.

Step 23: In the Arduino IDE, go to Tools > Serial Monitor.

Step 24: Set the baud rate to 115200 (or whatever rate your program specifies) in the Serial Monitor.

Step 25: Now everything is ready to start working on ESP32 in Arduino IDE.

With these steps, your ESP32 is now successfully set up in the Arduino IDE, and you can start programming it for your projects!

The more refined and structured step-by-step approach for working on this research project are as follows:

Step 1: Begin by studying the vast field of IoT (Internet of Things), focusing on the most recent advancements and innovations. IoT is a broad domain that enables devices to connect and communicate through the internet, collecting and exchanging data to improve automation, efficiency, and decision-making processes.

Step 2: Examine previous research papers, technical articles, and case studies related to IoT, particularly projects that involve environmental monitoring using sensors and microcontrollers like the DHT11 and ESP32. By understanding existing methods and technologies, pinpoint areas of improvement.

Step 3: Through this exploration, the recognisation is that ESP32 is an emerging technology in the IoT field, known for its built-in Wi-Fi and Bluetooth capabilities, making it an ideal microcontroller for data transmission to the cloud.

Step 4: Identify the key components necessary for the project, particularly sensors that can measure environmental conditions like temperature and humidity.

Step 5: A low-cost digital sensor that measures both temperature and humidity. While it has limitations in accuracy and response time, it is highly suitable for budget-friendly and introductory IoT projects.

Step 6: Conduct a comparative analysis of various sensors, such as DHT22, BME280, and BMP180, based on criteria like accuracy, cost, measurement range, and response time. The DHT11 is selected for this project as it provides an optimal balance between cost and basic performance.

Step 7: Known for its superior Wi-Fi and Bluetooth capabilities, the ESP32 is chosen for this project because it allows for real-time data transmission to cloud platforms, a critical feature for IoT-based data monitoring and analysis.

Step 8: Similarly, different microcontroller platforms such as Arduino Uno, ESP8266, and ESP32. The ESP32 stands out due to its powerful processing capabilities and built-in Wi-Fi, which makes it suitable for cloud-based IoT applications.

Step 9: Study and compare various platforms used for circuit simulation and design. Tools such as Wokwi, Circuit.io, and Proteus are considered.

- Wokwi: An online platform specifically tailored for simulating ESP32 and NodeMCU-based circuits.

- Circuit.io: A versatile platform that can simulate various microcontrollers and components.
- Proteus: A robust offline software used to create detailed and highly accurate circuit diagrams and simulate real-world hardware performance.

Step 10: Choose these platforms based on their suitability for the project. Wokwi is ideal for ESP32-specific circuit simulation, whereas Proteus offers a more detailed, offline simulation experience for real-time performance.

Step 11: Design the circuit virtually using the selected platforms. The circuit involves connecting the DHT11 sensor to the ESP32 microcontroller through its GPIO pins.

Step 12: Design the wiring for the sensor's VCC, GND, and DATA pins, ensuring proper power and signal flow between the components.

Step 13: Ensure that all hardware components are correctly integrated and simulated on platforms like Wokwi and Proteus.

Step 14: Simulated the circuit design to test the connectivity and functionality of the components. This step ensures that the circuit works as intended before moving on to the physical hardware implementation. During simulation, monitor the sensor's data output to verify its accuracy.

Step 15: After successful simulation, physically set up the hardware. Connect the DHT11 sensor to the ESP32, ensuring proper pin connections:

- Connect the VCC pin of the DHT11 to the 3.3V pin of the ESP32.
- Connect the GND pin to the ground.
- Connect the DATA pin to one of the GPIO pins for communication.

Step 16: Write the code for the ESP32 in the Arduino IDE. The code is designed to:

- Initialize the DHT11 sensor.
- Read temperature and humidity data from the sensor at regular intervals.
- Process and display the sensor data on the serial monitor for debugging purposes.

Step 17: Install necessary libraries to perform the task. Libraries such as DHT.h etc, need to be installed in Arduino IDE, to work with DHT11 sensor.

Step 18: Modify the code to include Wi-Fi connectivity so the ESP32 can send data to the cloud. Establish the necessary Wi-Fi credentials and connect to the network.

Step 19: Test the setup by running the code and collecting sensor readings on the serial monitor of the Arduino IDE.

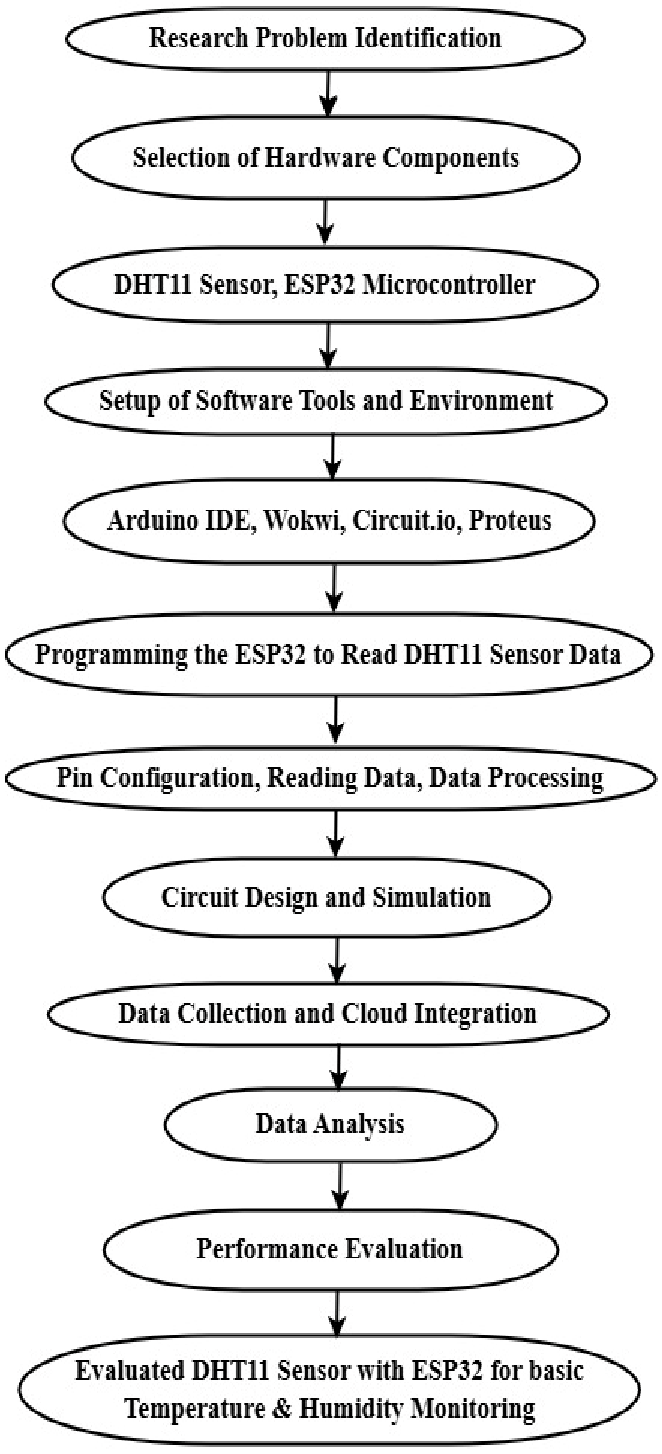

Research Problem Identification
Selection of Hardware Components
DHT11 Sensor, ESP32 Microcontroller
Setup of Software Tools and Environment
Arduino IDE, Wokwi, Circuit.io, Proteus
Programming the ESP32 to Read DHT11 Sensor Data
Pin Configuration, Reading Data, Data Processing
Circuit Design and Simulation
Data Collection and Cloud Integration
Data Analysis
Performance Evaluation
Evaluated DHT11 Sensor with ESP32 for basic Temperature & Humidity Monitoring

Fig.6. Flowchart of whole Research

Step 20: Verify that the sensor is providing accurate data under different environmental conditions.

Step 21: Integrate the hardware with a cloud platform such as ThingSpeak for real-time data visualization.

Step 22: Set up a ThingSpeak account and create a channel to store the temperature and humidity data.

Step 23: Modify the ESP32 code to upload the collected data to the ThingSpeak platform at regular intervals using HTTP requests.

Step 24: Plot the collected data on ThingSpeak, displaying it in the form of graphs for temperature and humidity over time. Analyze how the environment changes based on the data trends captured by the DHT11 sensor.

Step 25: Analyze the collected data to evaluate the performance of the DHT11 sensor. Key factors to assess include:

- Accuracy: Compare the data with more accurate sensors (if available) to evaluate the DHT11's reliability.
- Response Time: Monitor how quickly the sensor responds to changes in environmental conditions.
- Consistency: Evaluate the long-term performance of the sensor to assess its reliability over time.

Step 26: Based on the data analysis, summarize the findings. Conclude whether the DHT11 sensor, despite its limitations, is suitable for basic IoT applications.

This step-by-step approach provides a detailed and structured roadmap for the entire research, ensuring each phase is methodically planned, executed, and evaluated.

4. Results & Discussions

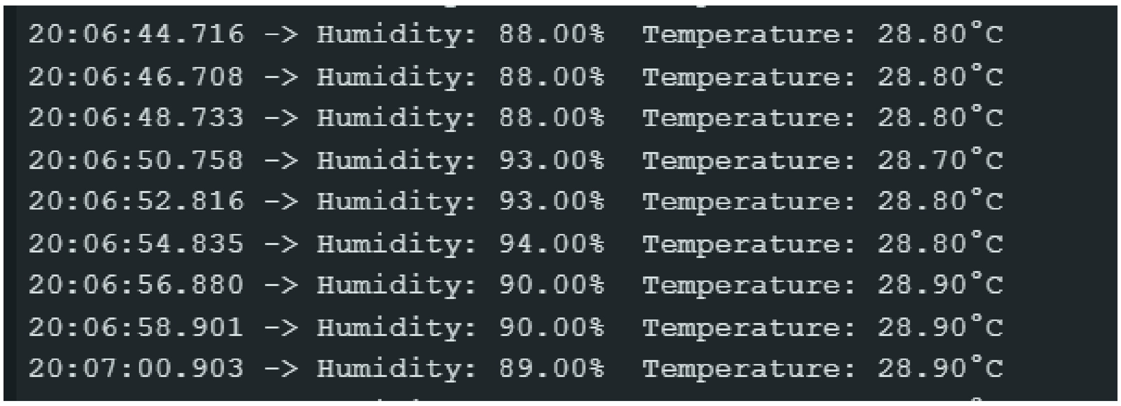
Traditionally, temperature and humidity were measured using analog methods, such as mercury thermometers and hygrometers. These traditional instruments, while simple and effective for basic applications, have notable limitations. Thermistors, for example, there are resistors that change resistance with temperature changes and are more advanced than mercury thermometers. While thermistors offer higher sensitivity and accuracy, they still require external analog-to-digital converters for integration with modern digital systems, making them less convenient for IoT and automated environments. Humidity sensors such as wet-bulb/dry-bulb hygrometers were also commonly used for humidity measurement but required manual readings and were prone to human error [7].

Arduino boards, such as the Arduino Uno and Arduino Nano, have been widely used in DIY electronics and IoT projects for temperature and humidity measurement. When paired with digital sensors like the DHT11 or DHT22, Arduino microcontrollers are easy to program using the Arduino IDE, which has a large library of sensor drivers available. Arduino boards offer a great balance between ease of use, affordability, and versatility, making them highly accessible for beginners. With 16 MHz processing power and several analog-to-digital pins, Arduino provides a reliable platform for real-time data collection from sensors [8].

This research focused on using the ESP32 microcontroller to measure temperature and humidity using the DHT11 sensor. The ESP32 stands out compared to traditional Arduinos due to its integrated Wi-Fi and Bluetooth capabilities, higher processing power, and energy efficiency. The ESP32 is equipped with a dual-core processor running at up to 240 MHz and 520 KB of SRAM, which allows it to handle more complex tasks, such as wireless data transmission and cloud integration, without external modules. The ESP32 also offers deep-sleep modes for energy-efficient operations, making it ideal for remote or battery-powered applications [9].

In this research, the ESP32 captured temperature and humidity data from the DHT11 (Temperature & Humidity sensor). This data can be accessed using the Serial Monitor feature in the Arduino IDE, which allows for real-time viewing of sensor outputs and debugging information. This built-in tool is essential for developers, as it provides immediate feedback on the data being collected from the DHT11 sensor, including temperature and humidity readings. By using the Serial Monitor, it is easy to verify that the sensor is functioning correctly and that the data being transmitted to the ThingSpeak platform is accurate.

The Serial Monitor displays readings that change over time. To print the temperature and humidity readings on the Serial Monitor, a simple code for the DHT11 sensor and ESP32 has been written, including the Serial.begin command to enable data visualization on the Serial Monitor. Additionally, the DHT sensor library has been installed to capture data from the DHT11 sensor. The Serial Plotter, another built-in feature of the Arduino IDE, has also been used to display graphs of temperature and humidity, allowing us to observe changes as they occur.



```
20:06:44.716 -> Humidity: 88.00%  Temperature: 28.80°C
20:06:46.708 -> Humidity: 88.00%  Temperature: 28.80°C
20:06:48.733 -> Humidity: 88.00%  Temperature: 28.80°C
20:06:50.758 -> Humidity: 93.00%  Temperature: 28.70°C
20:06:52.816 -> Humidity: 93.00%  Temperature: 28.80°C
20:06:54.835 -> Humidity: 94.00%  Temperature: 28.80°C
20:06:56.880 -> Humidity: 90.00%  Temperature: 28.90°C
20:06:58.901 -> Humidity: 90.00%  Temperature: 28.90°C
20:07:00.903 -> Humidity: 89.00%  Temperature: 28.90°C
```

Fig. 7. Measurement of Temperature & Humidity using Serial Monitor

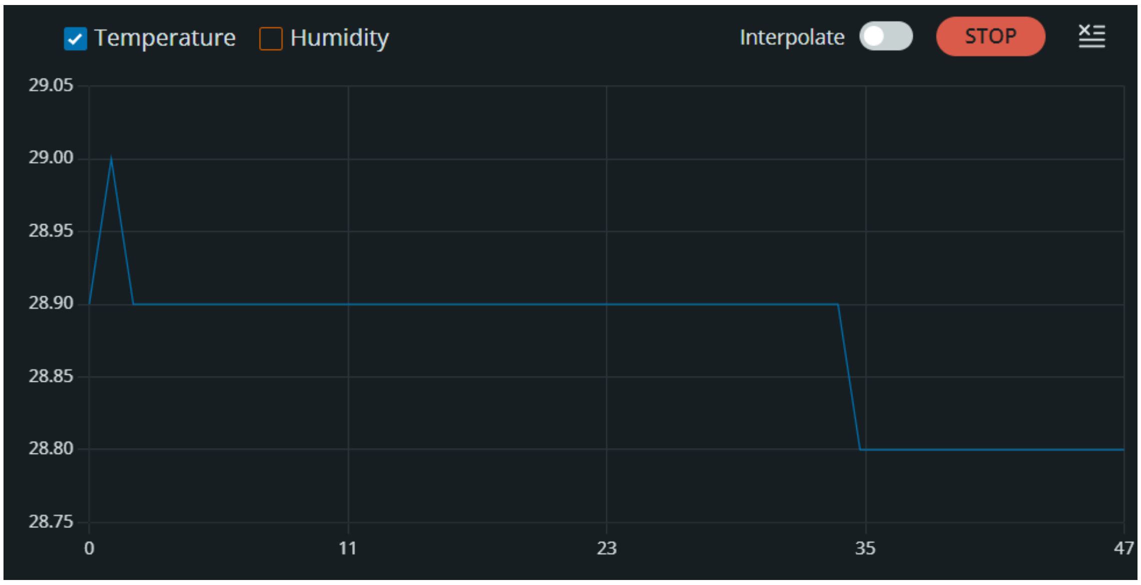


Fig.8. Measurement of temperature using Serial Plotter

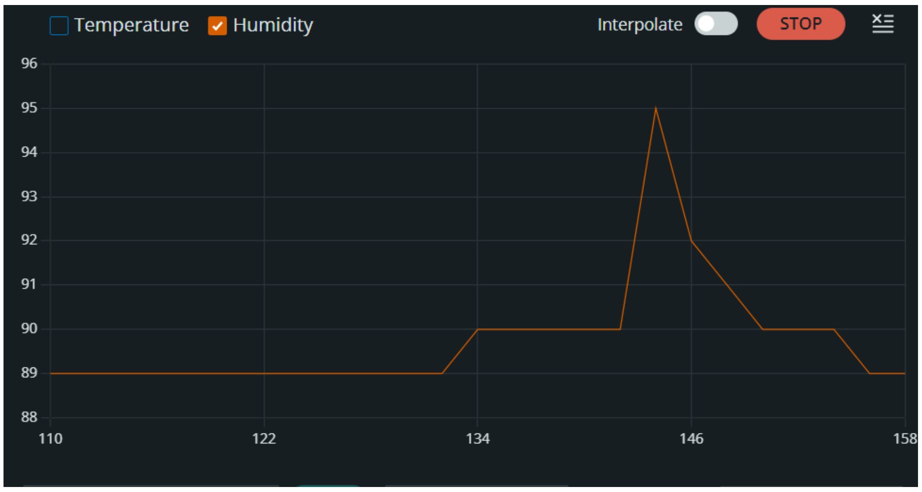


Fig.9. Measurement of humidity using Serial Plotter

Apart from the Serial Monitor, the data can also be displayed on a web page via the ESP32's Wi-Fi feature. Some modifications were made in the Arduino IDE, such as entering the Wi-Fi details for connectivity. The code was also updated to print the IP address of the Wi-Fi connection. The Wi-Fi library was installed to facilitate this connectivity. After uploading the code, the ESP32 connects to the Wi-Fi, generating the respective IP address. By pasting this IP address into a web browser (like Google Chrome), the temperature and humidity graphs become visible. Two separate graphs for temperature and humidity have been plotted to track the data over time, and variations can be detected in the displayed graphs.

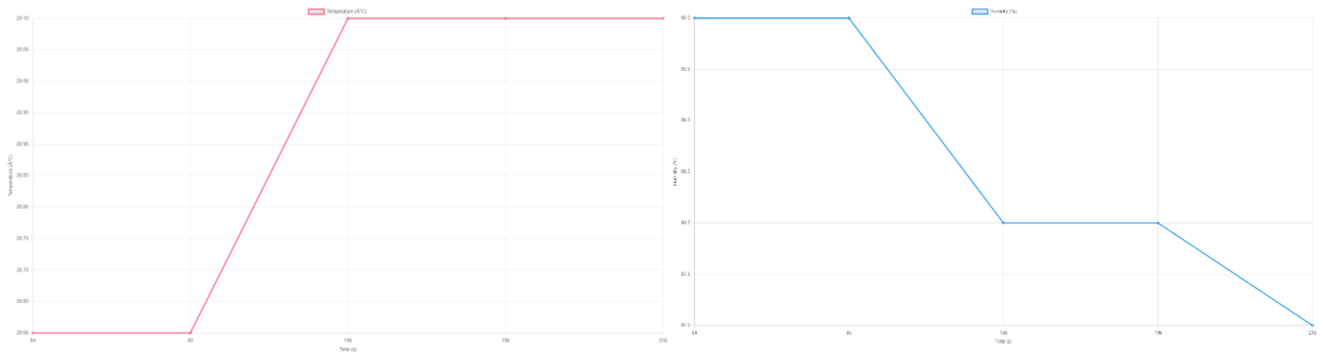
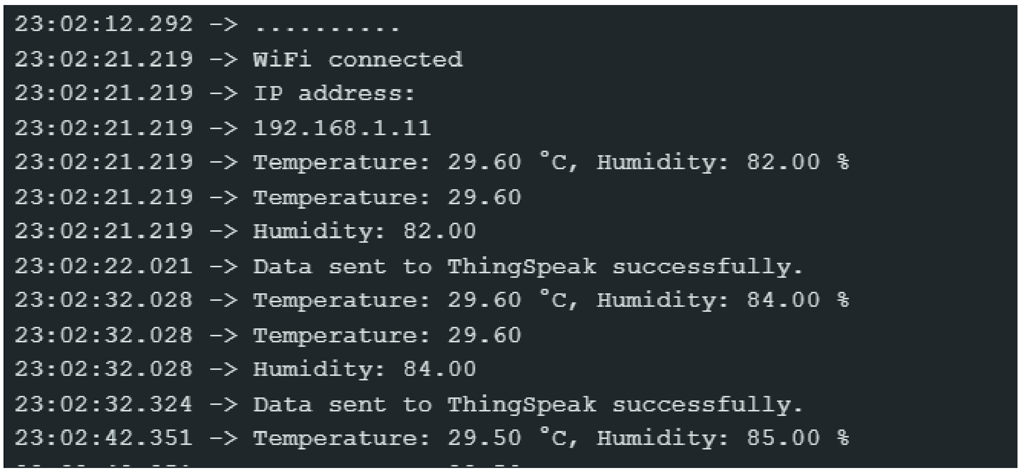


Fig.10. Graphs for Temperature & Humidity on a Web Page



```
23:02:12.292 -> ..........
23:02:21.219 -> WiFi connected
23:02:21.219 -> IP address:
23:02:21.219 -> 192.168.1.11
23:02:21.219 -> Temperature: 29.60 °C, Humidity: 82.00 %
23:02:21.219 -> Temperature: 29.60
23:02:21.219 -> Humidity: 82.00
23:02:22.021 -> Data sent to ThingSpeak successfully.
23:02:32.028 -> Temperature: 29.60 °C, Humidity: 84.00 %
23:02:32.028 -> Temperature: 29.60
23:02:32.028 -> Humidity: 84.00
23:02:32.324 -> Data sent to ThingSpeak successfully.
23:02:42.351 -> Temperature: 29.50 °C, Humidity: 85.00 %
```

Fig.11. IP Address Generated on Serial Monitor

Additionally, the data measured by the ESP32 for temperature and humidity from the DHT11 sensor has been transmitted wirelessly to the ThingSpeak cloud platform for real-time visualization and data analysis. For this purpose, modifications were made in the IDE code, including updates to specific commands and the installation of additional libraries such as Wi-Fi and HTTPClient to connect to the cloud platform. The write key generated on ThingSpeak was incorporated into the code to facilitate this connection. After these updates, the code was ready to upload. Once uploaded, the ESP32 connected to the Wi-Fi, and a connection to the cloud platform was established. The variation in the data can then be observed on the graphs plotted on the ThingSpeak platform. The humidity graph updates every second, while the temperature graph updates every minute, clearly demonstrating the variations in the temperature and humidity data. The graphs of the temperature and humidity captured from the ThingSpeak platform are as follows:

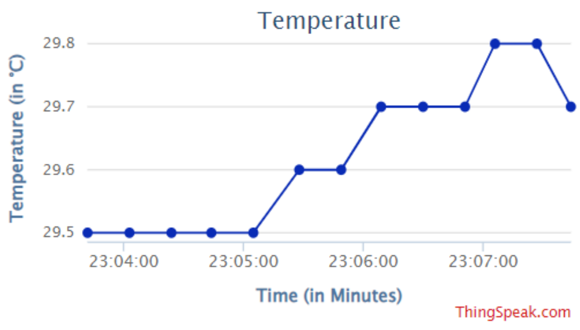


Fig.12. Temperature Graph on ThingSpeak

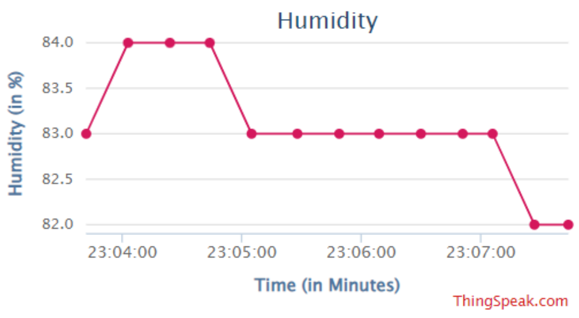


Fig.13. Humidity Graph on ThingSpeak

The integration of Wi-Fi and the ability to connect directly to cloud platforms like ThingSpeak without the need for additional hardware or modules distinguishes the ESP32 from traditional microcontrollers like Arduino. This capability makes the ESP32 particularly suitable for IoT applications that require remote data monitoring and storage.

In this research, several software tools were utilized to facilitate programming, circuit diagram development, and data visualization, each playing a crucial role in different stages of the project. One of the main tools used for programming the ESP32 microcontroller was the Arduino IDE. The Arduino Integrated Development Environment (IDE) is a widely recognized platform for coding, compiling, and uploading programs to various microcontrollers, including the ESP32, Arduino Uno, and other boards. It provides a simple and user-friendly interface that is especially suited for beginners, but also powerful enough for advanced users [10].

For the development of circuit diagrams, two online platforms—Wokwi and Circuit.io and an offline platform Proteus were used extensively. These platforms are particularly useful because they offer virtual simulation environments, allowing users to design and test their circuits before implementing them on physical hardware.

Together, these tools Arduino IDE for programming, Wokwi and Circuit.io for online circuit simulations, and Proteus for offline circuit development played a significant role in the success of this research by enabling efficient design, testing, and execution of the project at various stages. Each tool was selected based on its strengths, allowing the researcher to streamline the process and ensure the project's functionality before final implementation on hardware [11].

ThingSpeak is a popular open-source IoT platform developed by MathWorks, designed to facilitate real-time data collection, analysis, and visualization from connected devices like sensors, microcontrollers, and IoT-enabled systems. It provides a cloud-based solution for logging sensor data and displaying it through interactive charts and graphs. ThingSpeak is highly valued for its integration with MATLAB, making it a powerful tool for users who need advanced data processing and mathematical analysis, particularly in the fields of engineering, research, and hobbyist projects.

One of the key benefits of ThingSpeak is its ability to handle real-time data logging and visualization, which is crucial for IoT projects that require continuous monitoring. Another important feature of ThingSpeak is its cloud storage. All sensor data collected by devices like microcontrollers (ESP32, Arduino, etc.) is securely stored in the cloud, which can be accessed ThingSpeak's flexibility for small projects is another attractive feature. The platform offers free accounts with basic functionality, which are sufficient for smaller-scale IoT projects. Users can create a limited number of channels (which store sensor data) and monitor up to eight fields per channel. For more advanced users, there are premium features available, including higher data limits and faster update rates, but the free plan is adequate for most hobbyist projects and educational purposes.

To get started with ThingSpeak, users need to create an account on the platform. This process begins by visiting the ThingSpeak website and signing up, which redirects users to the MathWorks site (since ThingSpeak is owned by MathWorks).

Once logged in, users gain access to the ThingSpeak dashboard, where they can create channels to store sensor data. A channel in ThingSpeak can store data in up to eight fields, each corresponding to different sensor inputs. After setting up the channel, users are provided with API keys—one for writing data (sending sensor readings to ThingSpeak) and one for reading data (accessing logged data from the cloud) [13].

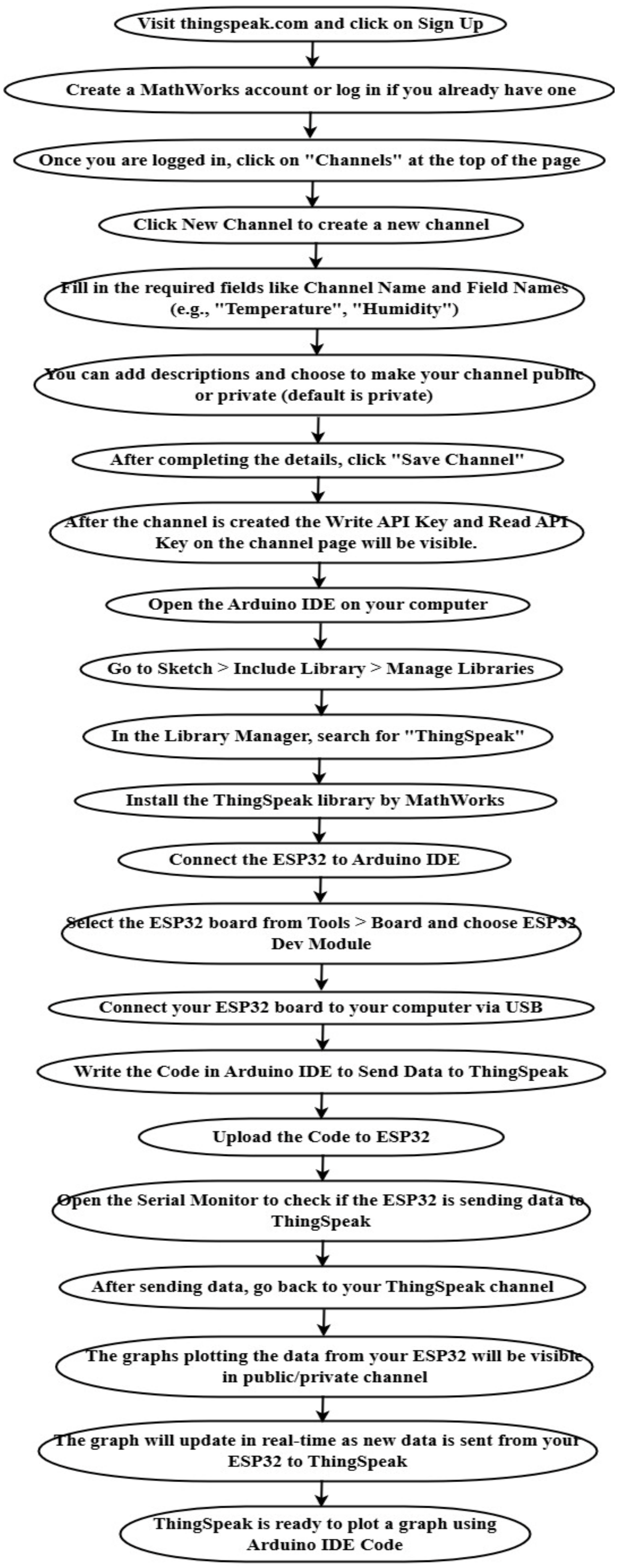

Visit thingspeak.com and click on Sign Up
Create a MathWorks account or log in if you already have one
Once you are logged in, click on "Channels" at the top of the page
Click New Channel to create a new channel
Fill in the required fields like Channel Name and Field Names (e.g., "Temperature", "Humidity")
You can add descriptions and choose to make your channel public or private (default is private)
After completing the details, click "Save Channel"
After the channel is created the Write API Key and Read API Key on the channel page will be visible.
Open the Arduino IDE on your computer
Go to Sketch > Include Library > Manage Libraries
In the Library Manager, search for "ThingSpeak"
Install the ThingSpeak library by MathWorks
Connect the ESP32 to Arduino IDE
Select the ESP32 board from Tools > Board and choose ESP32 Dev Module
Connect your ESP32 board to your computer via USB
Write the Code in Arduino IDE to Send Data to ThingSpeak
Upload the Code to ESP32
Open the Serial Monitor to check if the ESP32 is sending data to ThingSpeak
After sending data, go back to your ThingSpeak channel
The graphs plotting the data from your ESP32 will be visible in public/private channel
The graph will update in real-time as new data is sent from your ESP32 to ThingSpeak
ThingSpeak is ready to plot a graph using Arduino IDE Code

Fig. 14. Flowchart for ThingSPeak Setup & to Plot Graphs

To send data to ThingSpeak, users configure their IoT device, such as an ESP32 or Arduino, to communicate with ThingSpeak's servers using HTTP requests. For example, when working with a DHT11 sensor to measure temperature and humidity, the ESP32 can send the collected data to ThingSpeak by making a simple HTTP GET request using the provided API key. Once data is being sent to the platform, users can visualize the data using ThingSpeak's built-in graphing tools. Charts are automatically created for each field in a channel, allowing users to view real-time updates as new data is received. Users can customize these charts by adjusting the time ranges, changing the plot styles (e.g., line, scatter), and labeling the axes for better data interpretation.

In conclusion, ThingSpeak is a comprehensive, user-friendly platform that offers cloud-based data collection, real-time visualization, and advanced analysis tools, making it an ideal solution for IoT projects ranging from hobbyist experiments to professional research applications.

By following the steps, mentioned in the flowchart the collected data from sensors connected to your ESP32 can be visualize on ThingSpeak through real-time graphs, providing you with an easy way to monitor IoT data remotely.

5. Conclusion

In summary, this research highlights the significant role IoT technologies, particularly the combination of the DHT11 sensor and ESP32 microcontroller, can play in creating efficient, automated environmental monitoring systems. By incorporating a range of software platforms—such as Arduino IDE for programming, Wokwi and Circuit.io for simulation, Proteus for advanced circuit design, and ThingSpeak for cloud-based data visualization—this research demonstrated how to effectively integrate sensors, microcontrollers, and cloud platforms into a cohesive IoT system. This structured approach offers a solid foundation for future work in IoT development, and the insights gained from this research pave the way for further innovations in sensor technology, cloud computing, and automated monitoring systems. Looking ahead, this research encourages continued exploration of advanced sensors, microcontrollers, and software tools. Future work could explore the integration of machine learning algorithms with IoT systems for predictive analytics, enabling smarter decision-making in environmental monitoring and home automation. The findings presented here reflect the vast potential of IoT solutions to address real-world challenges in climate and environmental monitoring, showcasing the power and promise of these technologies in our increasingly connected world.

6. References

[1] M. J. E. Gavira, A. A. Perez, J. C. P. Salas, J. M. S.Fernandez, P. R. Carmona, J. J. G. de-la-Rosa, (2024). Characterization and Performance Evaluation of ESP32 for Real-time Synchronized Sensor Networks. Procedia Computer Sci., 237, 261-268.

[2] C. N. Oton and M. T. Iqbal, (2021). Low-Cost Open Source IoT-Based SCADA System for a BTS Site Using ESP32 and Arduino IoT Cloud, IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 0681-0685.

[3] P. Macheso, S. Chisale, C. Daka, N. Dzupire, J. Mlatho and D. Mukanyirigira, (2021). Design of Standalone Asynchronous ESP32 Web-Server for Temperature and Humidity Monitoring, 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 635-638.

[4] S. Zakaria, P. Mativenga, E. A. R. E. Ariff, (2023). An Investigation of Energy Consumption in Fused Deposition Modelling using ESP32 IoT Monitoring System, Procedia CIRP, 116, 263-268.

[5] N. Sharma, P. Sharma, D. Irwin and P. Shenoy, (2011). Predicting solar generation from weather forecasts using machine learning, IEEE International Conference on Smart Grid Communications (SmartGridComm), Brussels, Belgium, 528-533.

[6] Priyamvada, & Wadhvani, R. (2017). Review on various models for time series forecasting Intern Conf. on Inventive Computing and Informatics (ICICI).

[7] B. Sun, G. Xu, X Ji, Z. Yang, C. Guan, S. Chen, X. Chen, Y. Ma, Y. Yu, J. F. (2024). A strain-resistant flexible thermistor sensor array based on CNT/MXene hybrid materials for lithium-ion battery and human temperature monitoring, Sensors and Actuators A: Physical, 368, 115059.

[8] P. F. Pereira, N. M. M. Ramos, (2022). Low-cost Arduino-based temperature, relative humidity and CO2 sensors - An assessment of their suitability for indoor built environments, Journal of Building Engg., 60, 105151.

[9] J. Doshi, T. Patel, S. K. Bharti, (2019). Smart Farming using IoT, a solution for optimally monitoring farming conditions, Procedia Computer Sci., 160, 746-751.

[10] P. Anuradha, K. Vasanth, G. Renuka, A. R. Rao, (2023). IoT based enabling home automation system for individuals with diverse disabilities, e-Prime - Advances in Electrical Engg., Electronics and Energy, 6, 100366.

[11] P. Pawar, P. Pawale, T. Nagthane, M. Thakre, N. Jangale,(2021). Performance enhancement of dual axis solar tracker system for solar panels using proteus ISIS 7.6 software package, Global Transitions Proceedings, 2, 2, 455-460

[12] M. W. Hasan, (2023). Building an IoT temperature and humidity forecasting model based on long short-term memory (LSTM) with improved whale optimization algorithm, Memories - Materials, Devices, Circuits and Systems, 6, 100086

[13] Chandrasekaran, G. Kumar, N.S., A, C., V, G., Priyadarshi, N., Khan, B. (2023). IoT enabled smart solar water heater system using real time ThingSpeak IoT platform. IET Renew. Power Gener. 00, 1–13.