Smart IoT-Based Broiler Room Controller: Design, Implementation, Performance Evaluation, and Optimization

Shamsu Sabo[1,*]
Abubakar Muhammad Umaru[2]
Lawan Abdullahi Yusuf[3]

Received: 16/10/2024
Revised: 7/11/2024
Accepted: 16/12/2024

[1] Department of Computer Science, National Open University of Nigeria, Fagge Study Center. Nigeria.
[2] Department of Computer Science, Yusuf Maitama Sule University Kano, Nigeria. Email: Amumaru@yumsuk.edu.ng
[3] Directorate of Information and Communication Technology, National Open University of Nigeria, Fagge Study Center. Nigeria, Email: lyusuf@noun.edu.ng
* Corresponding Author address: 1440msusabo@gmail.com

Smart IoT-Based Broiler Room Controller: Design, Implementation, Performance Evaluation, and Optimization

Shamsu Sabo
Department of Computer Science, National Open University of Nigeria, Fagge Study Center, Nigeria
1440msusabo@gmail.com

Abubakar Muhammad Umaru
Department of Computer Science, Yusuf Maitama Sule University Kano.
Nigeria
Amumaru@yumsuk.edu.ng

Lawan Abdullahi Yusuf
Directorate of Information and Communication Technology, National Open University of Nigeria, Fagge Study Center,
Nigeria
lyusuf@noun.edu.ng

Abstract— This study presents the creation and performance assessment of an Internet of Things (IoT) smart broiler room controller. In order to monitor and regulate vital environmental parameters like temperature and humidity and guarantee ideal circumstances for broiler development, the system combines an ESP32 microcontroller with the Arduino platform. The system creates the perfect broiler-growing environment by dynamically adjusting fan, heater, and light settings based on real-time data using sensors like the DHT11 and LDR. We conducted a comprehensive performance study over seven weeks, comparing sensor values (temperature from the DHT11 and thermometer, and humidity from the DHT11 and hygrometer) with reference standards. With an average temperature of 29.99°C and a humidity level of approximately 39.95%, the results demonstrated steady temperature regulation and consistent, albeit marginally poor, humidity control. We found problems with the system's humidity management mechanism, which relied on a fan and heater and lacked sufficient accuracy. The absence of a real-time clock (RTC) module also impacted scheduling consistency, and the remote interface's lack of security features increased the risk of unwanted access. Adding an RTC module for improved scheduling, incorporating a humidifier for more accurate humidity control, and implementing a secure login mechanism to improve remote interface security were some of the suggestions offered to optimize the system. Additionally, using platforms like PlatformIO and MicroPython instead of the Arduino IDE could streamline code management and boost system performance. Additionally, the report recommends power consumption optimization techniques, such as using MQTT communication protocols and integrating solar power. In order to suggest strategies for system optimization, this study reuses elements of the experimental data, methodology, figures, tables, and implementation from our previous work [1], along with additional text and code included in the methodology. Through increased accuracy, sustainability, and efficiency, the suggested techniques seek to strengthen the system's performance as a reliable broiler farming solution.

Keywords— Smart Broiler Room Controller, IoT Integration, ESP32 Microcontroller, DHT11 Sensor, LDR Sensor.

I. INTRODUCTION

The adoption of Internet of Things (IoT)-based broiler room control represents a major breakthrough in the quickly changing poultry industry, as it helps overcome the drawbacks of both antiquated automation systems and conventional methods. These creative solutions, which make use of cutting-edge technologies, provide intelligent control over vital environmental elements like temperature, humidity, and lighting—all of which are crucial to the broiler production process. IoT-based systems improve sustainability, efficiency, and precision, while traditional methods find it difficult to adjust to the changing demands of contemporary broiler farming. This marks the beginning of a new era of innovation in the sector [2].

Providing ideal conditions for poultry at various growth stages requires careful environmental management. Temperatures between 20°C and 25°C, a healthy relative humidity range of 50% to 70%, and suitable lighting are all ideal for three-week-old broilers. The ideal temperature range for older broilers and laying hens is 26°C to 27°C, while the ideal range for laying hens is 18°C to 23.9°C. Furthermore, you should carefully regulate temperatures between 32°C and 35°C during the first week of life for delicate chicks, lowering them by roughly 3°C each week [3].

II. LITERATURE REVIEW

The review of literature explores current temperature control systems used in different industries, with an emphasis on performance evaluation and optimisation techniques. One system that is noteworthy is the one described in [4], which used an LM35 sensor, an ATmega8535 microcontroller, and a GSM modem to monitor server room temperature. Although it worked, the fact that it needed human intervention highlighted how crucial optimisation is to support autonomous functionality and increase system efficiency.

Similarly, using an Arduino, LM35 sensors, and light bulbs, [5] demonstrated an effective way to keep food warming at a steady 60°C. The system exhibits effectiveness and offers opportunities for optimisation to improve energy efficiency, scalability, and performance even more.
Another pertinent example is a system described in [6] that included an LM35, LCD, ADC, and relay to enable flexible temperature control between -4°C and 130°C. The primary objective of optimisation efforts may be to improve

responsiveness, accuracy, and adaptability to shifting environmental conditions.

Furthermore, a temperature distribution control system for a baby incubator was designed to prevent baby deaths caused by bacteria due to poor temperature management. [7] The on/off control system used an Arduino AT2560 as a microcontroller to automatically control the temperature. We used DHT and LM35 sensors to detect the temperature, and programmed MATLAB/Simulink to determine the correct and desired temperature distribution of 36°C in the baby's incubator. The fan would automatically turn on when the temperature exceeded 36°C and turn off when it fell below 36°C, while the heater would turn on. This system was highly accurate but consumed power and provided fairly adequate control. Another study [8] aimed to provide automation for small-scale poultry farms by designing an automatic monitoring and controlling system for a broiler house. The system uses water.

A PIC microcontroller interfaces with a pump, water level sensor, temperature sensor, humidity sensor, and NH3 sensor to provide automatic poultry management.

In another instance, researchers developed an electronic jacket using Arduino technology to control weather conditions for workers in extreme environments. The system featured a temperature sensor, fan, heater, and GSM module to enhance safety. However, it did not include capabilities for detecting vital signs, toxic gases, or obstacles [9].

In a unique investigation, a temperature control system using a mobile application interface was designed to control temperature using a mobile application interface and Bluetooth communication [10]. It used an Arduino development board and the MIT inventor programming environment to develop the app's interface. This system provided a low-cost alternative for controlling and monitoring the temperature but only worked to lower the room temperature.

Additionally, an automatic temperature control system, designed for enclosed areas, employed an LM35 sensor and Arduino Uno ATmega328p microcontroller to regulate temperature [11]. The system activated cooling or heating devices based on temperature variations from the preset value and could issue alerts for temperature deviations. However, its lack of remote control and limited scalability might constrain its suitability for complex or remote temperature control applications.

While some systems showed promise in terms of efficacy, scalability, connectivity, or functional limitations, there are numerous opportunities for optimisation in terms of energy efficiency, responsiveness, scalability, and environmental adaptability.

Improvements in smart poultry house monitoring [12] urbanized chicken coop monitoring [13] and IoT- based temperature and humidity monitoring in broiler farms [14] show promise but might require optimization efforts to overcome infrastructure, scalability, and connectivity challenges.

The evaluated publications are imprecise in a number of important areas: they don't provide a weekly schedule of temperature and humidity, and the majority only address heating or cooling systems, with very few combining both features. Furthermore, the majority only provide remote monitoring features; they lack the ability to manually operate devices remotely or on-site in order to handle emergency situations. The proposed system fills these gaps by incorporating a thorough weekly environmental schedule, supporting both heating and cooling operations, and enabling human management from a distance as well as up close.

The performance of the created system was also assessed, and recommendations for improvement were made in order to improve its sustainability, dependability, and efficiency in broiler production methods. By these contributions, the system opens the door for improvements in agricultural technology and tackles important issues facing the poultry sector.

III. SYSTEM PROPOSED

This system, which is intended to monitor and regulate environmental parameters in a broiler room, is made up of both hardware and software components. A thorough plan was created to evaluate and improve the functionality of the system, with rigorous testing, benchmarking, and performance analysis used to guarantee effectiveness and efficiency. The system's architecture is illustrated in the block diagram in Figure 1, which visually represents the relationships between the key components.

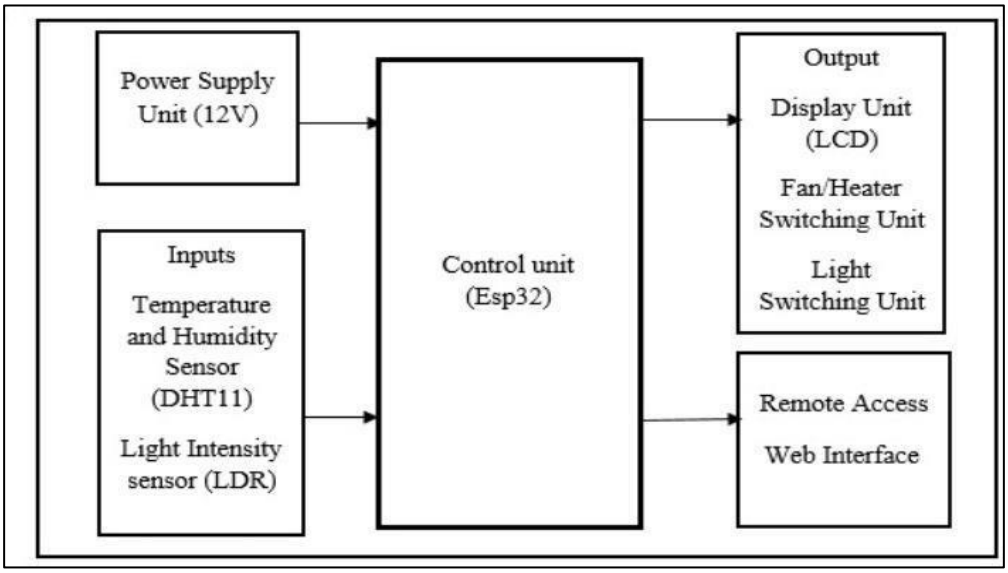

Fig. 1. System Architecture Block Diagram

A. Hardware Implementation

The fundamental elements, design factors, and implementation techniques that make up the produced solution's framework are all included in the hardware section of this system. There are two subsections within it the development consideration and components and the hardware design and implementation.

1) Development Considerations and Components: This subsection details the key hardware components used in the system and the factors considered in their selection and integration.

a) ESP32 Development Board: The ESP32 microcontroller functioned as the central processing unit, managing peripheral communication, sensor data collection, and control logic. Its dual-core processor, built-in Wi-Fi, and Bluetooth capabilities were important considerations in the selection process because they allowed for future expansion and smooth functioning. In order to improve energy usage throughout development—a crucial aspect of a battery-operated system—its low-power features were employed. The GPIO flexibility of the ESP32 was crucial for integrating with a variety of sensors and actuators.

b) Sensors: Sensors were used in the system to measure the temperature, humidity and the light level in the control room. These sensors are:

c) Display Unit: Real-time data, including temperature, humidity, and system status, were shown on a 16x2 Liquid Crystal Display (LCD). By integrating an I2C module, the system's small design was compatible with reduced GPIO utilization and easier wiring. The backlit screen made it possible to read in low light, which is typical in broiler settings. To prolong the display's useful life and protect it from dust and moisture, a protective cover was added.

d) Actuators: Fan and heater (where a red bulb was used) were used to control the temperature and humidity in the broilers room.

i) *Fan:* By controlling airflow, the fan maintained the right humidity and temperature. It was managed by a 12V relay and functioned according to thresholds established by the DHT11 sensor. Energy efficiency, robustness, and the capacity to provide enough airflow for the size of the space were given top priority while choosing the fan. In order to guarantee dependability, current needs and compatibility with the relay's switching capacity were also taken into account.

ii) *Red Bulb:* For demonstration purposes, a red light bulb was utilized as the heating element. When turned on, it produced heat and visual feedback, demonstrating the system's capacity to control temperature. Based on data from the DHT11 sensor, the ESP32 sent signals to a 12V relay, which in turn regulated the bulb's operation. The bulb was appropriate for demonstration because to its low cost and simplicity of integration, and safety measures were implemented to prevent overheating.

e) LED for Lighting: Because of their lifetime, low heat production, and energy efficiency, LEDs were used to provide extra lighting. Relays were used to control the LEDs in accordance with the LDR readings, guaranteeing proper illumination levels during various phases of the broiler's development. LED positioning was carefully considered during development to ensure consistent illumination free from glare or shadows. The power needs of the LEDs were matched with the relay's switching capacity.

f) Power Supply Unit: A 12V battery configuration was used to power the system, guaranteeing uninterrupted functioning even in the event of a power loss. To power low-voltage components like the ESP32 and sensors, a voltage regulation circuit with the LM7805 regulator IC was used to step down the 12V supply to 5V. Because of its ease of use and consistent performance under various load scenarios, the LM7805 was chosen. A heat sink was connected to the regulator to control heat dissipation, guaranteeing steady functioning.

g) Electronic Components: Different electronic components were use for specific function of the system they include:

i) *Resistors (1kΩ and 10kΩ):* Pull-up configurations for digital inputs and the voltage divider circuit for the LDR both made use of resistors. Because of their fine tolerance and low noise, metal film resistors were chosen to improve signal stability and guarantee precise readings.

ii) *LM7805 Voltage Regulator IC:* The 12V battery output was controlled by the LM7805 to produce a steady 5V supply. This was essential to ensuring that delicate parts like the ESP32 continued to function properly. A heat sink was included to minimize thermal overload during extended operation, and heat dissipation and current needs were taken into account throughout the selection process.

iii) *Relay (12V) and Transistor (BC547):* The BC547 transistor served as a driver, boosting control signals from the ESP32, while relays managed high-power components including the fan, heating bulb, and LEDs. In order to ensure compatibility with the attached components, the relays were selected based on their coil voltage and switching capacity. To safeguard the circuit from voltage spikes produced during relay switching, fly back diodes were incorporated. Because of its dependability in low-power switching applications, the BC547 transistor was chosen.

2) Hardware Components and Integration: The broiler room control system's central processing unit is the ESP32 microcontroller, which runs on a 12V battery. To guarantee dependable and effective operation, it synchronizes the activities of several sensors and actuators. Real-time ambient light level measurement is made possible via the ESP32's analog input interface with the Light Dependent Resistor (LDR) sensor. The DHT11 sensor, which detects temperature and humidity, communicates with the ESP32 via a digital protocol. It provides current environmental data that is necessary to maintain optimal conditions in the broiler chamber. The ESP32 and additional components for voltage regulation and stability are powered by an LM7805 regulator integrated circuit, which ensures stable operation even when the battery's input voltage fluctuates. Based on preset thresholds, the Control Unit's ESP32 regulates the ambient conditions. This unit has resistors, a BC547 transistor to regulate motor speed, and relays to control a variety of appliances, including fans and heaters. By modifying the LDR's sensitivity, the potentiometer provides exact control over light-dependent behaviors. Lastly, a 16*2 LCD (Liquid Crystal Display) on the Output Unit serves as a user interface and displays important information such as temperature, humidity, and system status. The ESP32 regulates the display of this data, giving consumers the most recent details regarding the broiler room's conditions. Additionally, the ESP32 system's web server interface allows users to remotely monitor and control it. Connectors and interface components facilitate the seamless integration of the LCD and other components to guarantee seamless operation and user-friendly interaction.

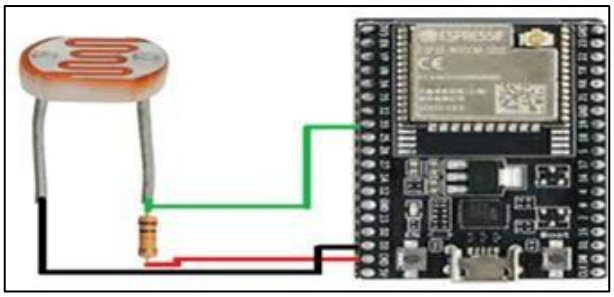

Fig. 2. LDR Interfaced with ESP32

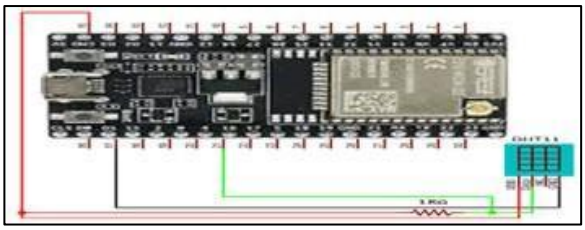

Fig. 3. DHT11 Interfaced with ESP32

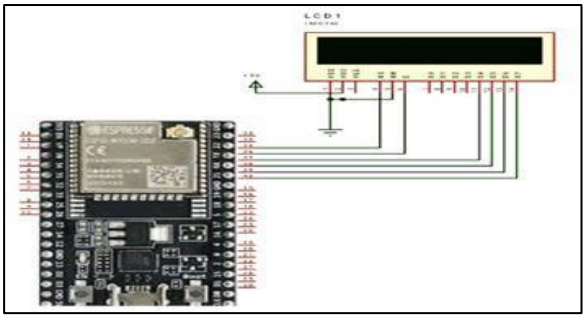

Fig. 4. 4:16*2 LCD interfaced with ESP32

By integrating this input, control, and output modules, the system efficiently gathers environmental data, processes it

with the ESP32, controls conditions, and provides users with pertinent information through the LCD interface and remote access functions. A comprehensive overview of the system architecture is given in Figure 5, which displays the hardware integration and the connections between the ESP32, sensors, actuators, and power components.

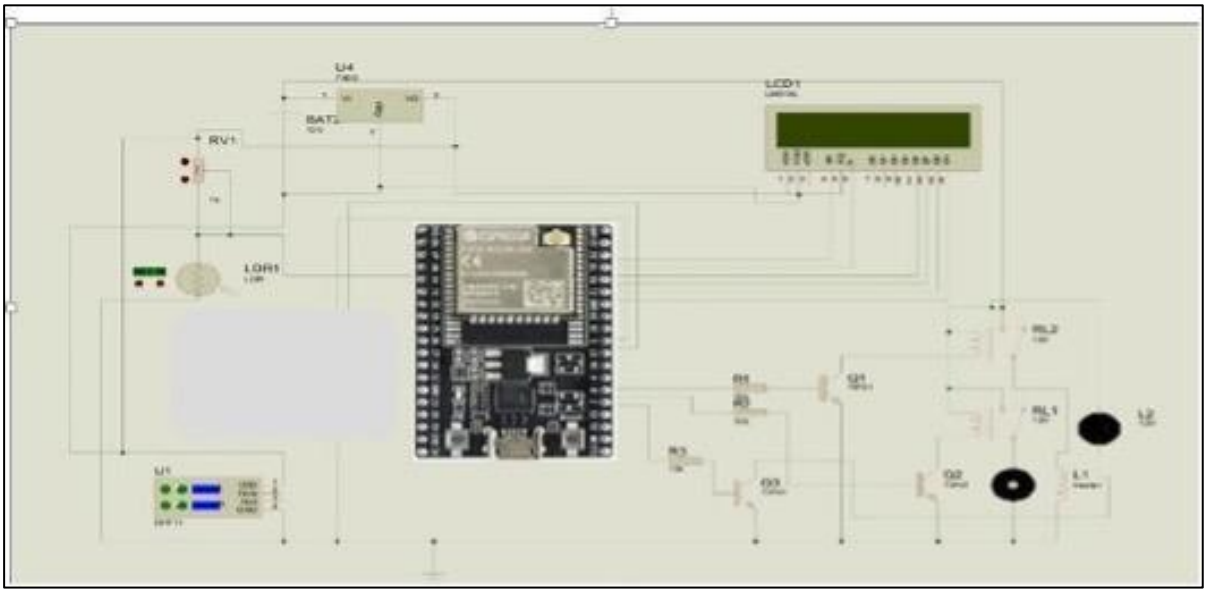

Fig. 5. Shows the suggested systems circuit diagram

B. Software Implementation

C++ was used in the Arduino IDE to create the software for the broiler environmental control system. It incorporates a number of essential elements, such as sensor interface, Wi-Fi connectivity, and control logic for environmental regulation.

1) *Wi-Fi Connectivity:* This component's goal is to make remote control and monitoring possible over the network. After establishing a Wi-Fi connection, the system configures an AsyncWebServer to process HTTP requests. This enables users to engage with the system via a web interface and get real-time data. It improves accessibility and enables users to monitor and modify environmental factors from any place, which has a substantial impact on system performance. This enhances productivity and reaction times, ensuring swift system modifications in response to user input. The software dependencies for Wi-Fi connectivity include the WiFi.h, AsyncTCP.h, and ESPAsyncWebServer.h libraries, which are necessary for setting up and managing the web server and handling asynchronous TCP/IP connections.
2) *Interface with Sensors:* The main goal of the sensor is to monitor the ambient factors (temperature and humidity) that are essential for broiler growth. At regular intervals, the system reads the temperature and humidity using a DHT sensor (DHT11). The web interface receives these data and displays them locally on an LCD. This ongoing monitoring ensures the maintenance of environmental conditions within ideal ranges. This ongoing monitoring affects system performance by aiding in the maintenance of an environment that promotes broiler health and growth, a crucial aspect of effectiveness and output. To communicate with the DHT sensor and obtain precise environmental data, the software uses the DHT.h library.
3) *Both Automatic and Manual Control Logic*: By using pre-established schedules and human input, the control logic seeks to govern environmental conditions. Both manual and automatic control options are available. In manual mode, users can utilize the web interface to change the fan, heater, and light as necessary, overriding the automated settings. When operating in automatic mode, the system modifies the surroundings according to preset timetables and sensor data. This dual mode ensures ideal conditions for broiler farming by striking a balance between automated modifications and user control. This dual mode significantly impacts the system's performance, minimising manual intervention, boosting energy efficiency, and preserving stable environmental conditions—all crucial for the production and well-being of broiler chickens. The TimeLib.h library facilitates time-based operations used by the algorithms in the control logic.
4) *Web-Based Interface:* The web interface offers a user-friendly platform for remote control and monitoring. Users can utilize a web browser to access status updates and manage environmental parameters through an AsyncWeb server. This involves manually operating appliances such as the fan, heater, and light, as well as viewing sensor data. Users can effectively monitor and

operate the system, which greatly impacts system performance and enables prompt reactions to environmental changes. The program makes use of the ESPAsyncWebServer.h library to control the web server configuration and requests and the AsyncTCP.h library to handle asynchronous TCP/IP interactions.

5) *Decision-Making and Control Logic:* Algorithms for automatic adjustments depending on preset schedules and sensor readings are part of the control logic. It creates a climate that is appropriate for the broiler growth phases by adjusting the temperature and humidity levels. Users can also control the system settings by using the manual override options. Through the maintenance of ideal environmental conditions for broiler growth and health, this dual approach improves system performance. By keeping the environment appropriate, it improves productivity, health, and efficiency. The program makes use of the EEPROM.h library to store and retrieve schedules and configuration information, which are essential for the system's long-term functionality and flexibility in response to changing circumstances. Table. 1 the code example showcasing the system software.

TABLE 1. EXAMPLE CODE

```
//libraries
#include <WiFi.h> #include <AsyncTCP.h>
#include <ESPAsyncWebServer.h> #include <LiquidCrystal.h> #include <DHT.h>
#include <EEPROM.h> #include <TimeLib.h>

// Wi-Fi credentials
const char* ssid = "U.kairi";
const char* password = "12345678";
// Temperature and humidity schedule for broiler growth (in weeks)
const int temp_schedule[] = {32, 30, 28, 26, 24, 22, 20};  // Target temperatures
(°C)
const int humidity_schedule[] = {60, 55, 50, 45, 40, 35, 30}; // Target humidity
levels (%)

// DHT sensor setup
#define DHTPIN 4  // Pin connected to the DHT sensor #define DHTTYPE DHT11 // DHT
11 sensor type
DHT dht(DHTPIN, DHTTYPE);

// LCD setup (16x2 display)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
// Web server setup
AsyncWebServer server(80);
lcd.clear();
lcd.print("WiFi Connected");

// Set up web server routes
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) { request->send(200,
"text/plain", "Welcome to Broiler Control System");
});

server.on("/status", HTTP_GET, [](AsyncWebServerRequest *request) {

// Send sensor readings to the web interface float currentTemperature =
readTemperature(); float currentHumidity = readHumidity();
String status = "Temperature: " + String(currentTemperature) + " C\n" +
"Humidity: " + String(currentHumidity) + " %";
request->send(200, "text/plain", status);
});

server.on("/manual-control", HTTP_GET, [](AsyncWebServerRequest *request) { if
(request->hasParam("fan")) {
```

```
manualFanControl = request->getParam("fan")->value() == "1";
}


if (request->hasParam("heater")) {

manualHeaterControl = request->getParam("heater")->value() == "1";

}

if (request->hasParam("light")) {

manualLightControl = request->getParam("light")->value() == "1";

}

request->send(200, "text/plain", "Manual Control Updated");

});

// Start server
server.begin();

void loop() {
}
// Current time
unsigned long currentTime = millis();

// Read environmental parameters every 2 seconds
if (currentTime - previousReadTime >= readInterval) { previousReadTime =
currentTime;
float currentTemperature = readTemperature(); float currentHumidity =
readHumidity();
// Update LCD display
lcd.clear(); lcd.setCursor(0, 0);
lcd.print("Temp: " + String(currentTemperature) + "C"); lcd.setCursor(0, 1);
lcd.print("Humidity: " + String(currentHumidity) + "%");

// Log readings for debugging
Serial.print("Temperature: ");
Serial.print(currentTemperature); Serial.print(" °C, Humidity: ");
Serial.print(currentHumidity); Serial.println(" %");
}

// Device control logic: Manual or Automatic
if (manualFanControl || manualHeaterControl || manualLightControl) {

// Manual control: User overrides automatic settings
if (manualFanControl) { handleManualFanControl();
}

if (manualHeaterControl) { handleManualHeaterControl();
}

if (manualLightControl) { handleManualLightControl();
}

} else {
```

```
// Automatic control: Adjust devices based on sensor readings and schedules
adjustFan();
adjustHeater(); adjustLight();
}

// Function stubs for sensor readings
float readTemperature() {
// Read and return the temperature from the DHT sensor return
dht.readTemperature();
}

float readHumidity() {

// Read and return the humidity from the DHT sensor return dht.readHumidity();
}

// Functions for automatic adjustments
void adjustFan() {
// Logic for fan adjustment based on temperature
}

void adjustHeater() {
// Logic for heater adjustment based on temperature
}

void adjustLight() {
// Logic for automatic light adjustment
}

// Functions for manual control
void handleManualFanControl() {
// Logic for manual fan control
}

void handleManualHeaterControl() {
// Logic for manual heater control
}

void handleManualLightControl() {
// Logic for manual light control
}
```

C. Functionality and Display of the System: By activating the heater when the temperature dropped below the threshold and the fan when it increased above it, the system showed that it could independently control itself. It also adjusted the lighting, turning it on when ambient levels fell below the 500-lux threshold and off when they increased above it. The device also followed a routine, adjusting the temperature and humidity once a week. The humidity levels were adjusted to 65%, 60%, 55%, 50%, 45%, 40%, and 35%, respectively, while the temperature was precisely fixed at 32°C, 30°C, 28°C, 26°C, 24°C, 22°C, and 20°C for each succeeding week [15] The customization of the system ensured optimal circumstances for the growth of broilers.

Fig. 6. Illustration of the fully assembled system

The remote interface is accessed by entering the IP address of the ESP32 into the browser search bar. This allows users to monitor and control the ESP32 remotely. Figure 7 illustrates the remote monitoring interface.

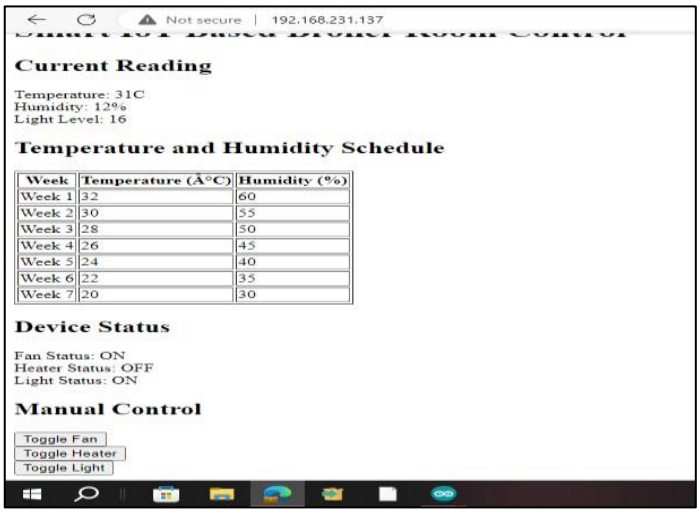

Fig. 7. Illustration of the remote interface

IV. RESULTS AND DISCUSSION

A. Sensor Measurements and Performance Evaluation:

A weekly average for each environmental element was calculated by comparing the two weekly readings from the system's temperature (DHT, Thermometer) and humidity (DHT, Hygrometer) sensors with reference standards, as shown in Table 2. These measurements were examined in order to evaluate the system's performance.

1) Temperature Readings (DHT(°C) and Thermometer(°C)): The average temperature readings from both the DHT sensor and the thermometer sensor ranged between 29.25°C and 30.58°C, with an overall average of 29.99°C and 29.75°C, respectively. These values indicate relatively stable temperature regulation within the system.

2) *Humidity Readings (DHT(%) and Hygrometer(%)):* Humidity readings from the DHT sensor and hygrometer ranged from 37.67% to 45.38%, with overall averages of 39.95% and 38.51%, respectively. These readings suggest that humidity control was consistent but slightly below ideal conditions.

TABLE 2. COMPARISON OF TEMPERATURE AND HUMIDITY MEASUREMENTS

Time	DHT (°C)	Thermometer (°C)	DHT2 (% Humidity)	Hygrometer (% Humidity)
Week 1	30.58	30.33	38.00	36.00
Week 2	29.54	29.32	45.38	44.38
Week 3	30.21	29.21	37.67	36.42
Week 4	30.46	30.23	38.50	37.13
Week 5	30.38	30.13	39.29	37.88
Week 6	29.25	28.99	40.33	38.79
Week 7	29.54	29.24	40.46	39.00
Average	29.99	29.75	39.95	38.51

Our technology differs from previous studies in several important aspects, including accuracy, efficiency, reliability, sustainability, and power usage. Inaccurate environmental control resulted from traditional systems' frequent reliance on static temperature thresholds that were unable to adjust to the changing needs of broilers at various growth stages. Our method, in contrast, ensures accuracy and ideal conditions for the birds by dynamically adjusting temperature and humidity thresholds once a week to correspond with each growth phase.

Another drawback of earlier systems, which ran *2)* continuously without taking demand into account in real time, was their energy efficiency. Our solution dramatically lowers energy usage by integrating light management and only turning on heating or ventilation systems when required, *3)* improving sustainability and efficiency.

Remote monitoring and control functions, which enable real-time modifications and prompt responses to environmental changes—a benefit over systems without such features—further increase reliability. This guarantees reliable and constant performance.

Our design places a strong emphasis on sustainability. The *4)* system minimizes its influence on the environment while fostering the health and well-being of birds by combining temperature, humidity, and light regulation. Dynamic control optimizes power use by shutting off devices when not in use and cutting down on wasted energy. This method increases the system's cost-effectiveness and sustainability.

By providing accurate environmental control, maximizing energy utilization, and guaranteeing sustainability and dependability, our system overcomes the drawbacks of previous methods. It is now a complete and effective solution for contemporary chicken husbandry thanks to these developments.

B. Challenges:

The study has effectively identified the following challenge that needs to be considered for system optimization.

1) Humidity Control Mechanism: The system's accuracy and efficacy were limited because it relied on a heater and fan to control humidity. This approach lacked the agility required for precise humidity control, which led to irregular fluctuations and less-than-optimal broiler development conditions.

2) Lack of Real-Time Clock (RTC): The absence of a real-time clock module in the system made timekeeping and scheduling challenging. When the system restarted, thescheduling process reset to week one rather than seamlessly carrying over from the previous recorded week.

3) Remote Interface Security and Management: Due to inadequate security measures, there was a chance of unauthorized access because the remote interface could only be accessed by IP address and required no login information. Furthermore, the monitoring and control of numerous devices is made more difficult by the lack of centralized management functions.

4) Dependency on Arduino IDE Libraries: Because of the Arduino IDE's extensive reliance on libraries, coding became more challenging and resulted in complex code with management problems.

C. Enhancing Accuracy, Sustainability, Reliability, and Efficiency:

The study has point out accuracy, sustainability and efficiency improvements in the developed system and how they can be improved.

1) Improving Accuracy: The study developed an accurate temperature and humidity measurement system using the DHT11 sensor. This method ensures a consistent

environment that closely mimics ideal conditions for broiler health by comparing these readings to reference standards. This exact control lessens temperature and humidity swings, which are crucial for maintaining comfortable and stress-free broiler habitats. The system initially struggled with proper humidity control due to limitations in the heater and fan configuration, potentially leading to unexpected effects on temperature management. A performance evaluation found that adding a humidifier might significantly enhance humidity level control without compromising temperature stability. This improvement will reduce volatility and establish a stable environment that fosters broiler growth.

2) Enhancing Sustainability: In order to minimise the impact on the environment and improve the sustainability of the system, the study suggests integrating solar power, using microcontroller low-power modes, and implementing communication protocols like MQTT. We also advise integrating an RTC module to maintain accurate scheduling and timekeeping, ensuring consistent system functioning even after restarts, thereby enhancing overall system sustainability.

3) Increasing Efficiency: The study found that the system's code became more complicated and less effective due to its significant reliance on Arduino IDE. Switching to platforms like PlatformIO and MicroPython, which offer more efficient development environments, can facilitate easier maintenance and expansion of the system. These changes increase system performance, decrease library dependencies, and write more efficient code. These platforms, such as Arduino Core for ESP32, Zephyr, and ChibiOS, offer characteristics such as cross-platform interoperability, real-time operating system (RTOS) capabilities, and lower power consumption. These platforms offer more possibilities for performance improvement and adaptability, which makes it possible to create broiler farming systems that are more dependable and efficient.

4) Power Consumption Optimisation: Although the system used less power than in earlier versions, there is still room for Integration of solar power, use of microcontroller low-power modes, and implementation can greatly reduce power consumption.MQTT (Message Queuing Telemetry Transport) is an effective communication protocol. These improvements encourage sustainability and increase system autonomy, guaranteeing continuous operation with low energy usage.

5) Enhancing Reliability: Integrating a real-time clock (RTC) for consistent scheduling, improving security with secure login credentials and centralised management, upgrading humidity control with precise humidifiers, implementing effective development platforms like PlatformIO to simplify code, and optimising power consumption through low-power microcontroller modes, solar power integration, and effective communication protocols like MQTT are all ways to increase the system's reliability.

V. Conclusion

The developed IoT-based broiler room controller system successfully addressed significant environmental management gaps in broiler farming. It demonstrated reliable temperature regulation and constant humidity control, with evaluation results highlighting both its benefits and drawbacks. By putting suggestions like installing a humidifier, integrating an RTC module, bolstering security measures, and using more efficient development platforms into practice, the system can achieve improved accuracy, sustainability, reliability, power consumption, and operational efficiency. With these advancements, the system is well-positioned to use IoT technology to significantly modernize broiler production. By addressing evaluation-identified problems and offering practical solutions, this study provides a roadmap for developing more reliable and adaptable agricultural technology systems, encouraging higher production and sustainability in broiler farming operations.

VI. References

[1] M. Umaru, S. Sabo, J. A. Ibrahim, and A. T. Sulaiman, "Design and implementation of smart IoT-based broiler room controller," *Journal of Science Technology and Education*, vol. 12, no. 2, 2024. [Online]. Available: www.atbuftejoste.com.ng.

[2] F. U. Rekwot, G. Z. Owoshagba, O. B. Ahmed, and Atiku, "Challenges of poultry production in Nigeria: A review Anosike," 2018.

[3] S. Bhawa, J. C. Morêki, and J. B. Machete, "Poultry management strategies to alleviate heat stress in hot climates: A review," J. Worlds Poult. Res., vol. 13, no. 1, pp. 1–19, Mar. 2023, doi: 10.36380/jwpr.2023.1.

[4] T. Wellem and B. Setiawan, "A microcontroller-based room temperature monitoring system," 2012.

[5] A. Enriko, R. A. Putra, and Estananto, "Automatic temperature control system on smart poultry farm using PID method," Green Intelligent Systems and Applications, vol. 1, no. 1, pp. 37–43, Nov. 2021, doi: 10.53623/gisa.v1i1.40.

[6] Y. B. Lawal, J. Dada, and F. Ahmed-Ade, "Construction of an automatic temperature controller for monitoring and cooling systems characterizing weather variability for radio communication," 2016. [Online]. Available: https://www.researchgate.net/publication/321105840

[7] W. Widhiada, "Temperature distribution control for baby incubator system using Arduino AT Mega 2560," 2018.

[8] R. D. Ramdurge and M. S. Patil, "Automatic monitoring and controlling system for broiler house," Int. J. Innov. Res. Electr. Electron. Instrum. Control Eng., vol. 4, pp. 2321–5526, 2016, doi: 10.17148/IJIREEICE.2016.4708.

[9] A. Brad and M. Brad, "Development of a smart clothing product using an Arduino platform," Int. J. Adv. Stat. IT&C Econ. Life Sci., vol. 11, no. 1, pp. 38–61, 2021, doi: 10.2478/ijasitels-2021-0002.

[10] C. M. M. Castro, A. Mestria, and M. Mestria, "Temperature control system using mobile application interface," Eur. J. Formal Sci. Eng., 2022, doi: 10.26417/729pbt84.

[11] E. A. Nyiekaa and C. A. Francis, "Design and construction of an automatic temperature control system," J. Sci. Eng. Res., vol. 84, no. 6, pp. 84–91. [Online]. Available: www.jsaer.com

[12] G. M. Debele and X. Qian, "Automatic room temperature control system using Arduino UNO R3 and DHT11 sensor," in 2020 17th Int. Comput. Conf. Wavelet Active Media Technol. Inf. Process. (ICCWAMTIP), Dec. 2020, pp. 428–432, doi: 10.1109/ICCWAMTIP51612.2020.9317307.

[13] S. Adsule, S. Mohite, R. Patil, and N. Dhawas, "Automatic temperature-based fan speed controller using Arduino," in Int. Conf. Commun. Inf. Process., 2020. [Online]. Available: https://ssrn.com/abstract=3645388

[14] R. Sasirekha, R. Kaviya, G. Saranya, A. Mohamed, and U. Iroda, "Smart poultry house monitoring system using IoT," in E3S Web Conf., Jul. 2023, doi: 10.1051/e3sconf/202339904055.

[15] A. Enriko, R. A. Putra, and Estananto, "Automatic temperature control system on smart poultry farm using PID method," Green Intelligent Systems and Applications, vol. 1, no. 1, pp. 37–43, Nov. 2021, doi: 10.53623/gisa.v1i1.40.