IoT Based Smart Egg Incubator

Group Members

Minhal Waheed	19I-0761
Khuzaima Ahmed	19I-0762
Wardah Malik	19I-0765

Project Supervisor

Dr. Ata ul Aziz Ikram

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Islamabad

2023

Developer’s Submission

“This report is being submitted to the Department of Electrical Engineering of the National University of Computer and Emerging Sciences in partial fulfillment of the requirements for the degree of BS in Electrical Engineering.”

Developer's Declaration

"We accept full responsibility for the completion of all work for the "**IoT Based Smart Egg Incubator**" Final Year Project (FYP). We solemnly declare that the project work presented in the FYP report is done solely by us with no significant help from any other person; however, small help wherever taken is duly acknowledged. Additionally, we wrote the entire FYP report by ourselves. Moreover, we have never submitted any portion of the thesis or this FYP to another degree-awarding school in Pakistan or abroad (or substantially equivalent project work).

We are aware that the NUCES management has a no-plagiarism policy in the department of electrical engineering. As a result, we, as the authors of the FYP report, hereby solemnly declare that no part of our report contains any passages that were taken verbatim from other sources and that all such passages were correctly cited. Additionally, the work that is provided in the report is our own work, and we have explicitly distinguished it from the pertinent work of the other projects while still citing their related work.

We agree that the university has a complete right to withdraw our BS degree if any member is found guilty of any plagiarism. We also understand that the university holds the right to publish our names on the official website to keep a record of students who have submitted plagiarized FYP reports. "

_____	_____	_____
Minhal Waheed	Khuzaima Ahmed	Wardah Malik
BS(EE) 2019-0761	BS(EE) 2019-0762	BS(EE) 2019-0765

_____ _____

Certified by Supervisor Verified by Plagiarism Cell Officer

Dated: _____

Developer's Declaration

IoT Based Smart Egg Incubator

Sustainable Development Goals

(Please tick the relevant SDG(s) linked with FYDP)

SDG No	Description of SDG	SDG No	Description of SDG
SDG 1	No Poverty	SDG 9	Industry, Innovation, and Infrastructure
✓ **SDG 2**	**Zero Hunger**	SDG 10	Reduced Inequalities
✓ **SDG 3**	**Good Health and Well Being**	SDG 11	Sustainable Cities and Communities
SDG 4	Quality Education	SDG 12	Responsible Consumption and Production
SDG 5	Gender Equality	SDG 13	Climate Change
SDG 6	Clean Water and Sanitation	SDG 14	Life Below Water
SDG 7	Affordable and Clean Energy	SDG 15	Life on Land
✓ **SDG 8**	**Decent Work and Economic Growth**	SDG 16	Peace, Justice and Strong Institutions
		SDG 17	Partnerships for the Goals

Developer's Declaration

Range of Complex Problem Solving			
	Attribute	**Complex Problem**	
1	Range of conflicting requirements	Involve wide-ranging or conflicting technical, engineering and other issues.	
2	Depth of analysis required	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.	✓
3	Depth of knowledge required	Requires research-based knowledge much of which is at, or informed by, the forefront of the professional discipline and which allows a fundamentals-based, first principles analytical approach.	✓
4	Familiarity of issues	Involve infrequently encountered issues	✓
5	Extent of applicable codes	Are outside problems encompassed by standards and codes of practice for professional engineering.	✓
6	Extent of stakeholder involvement and level of conflicting requirements	Involve diverse groups of stakeholders with widely varying needs.	
7	Consequences	Have significant consequences in a range of contexts.	✓
8	Interdependence	Are high level problems including many component parts or sub-problems	✓
Range of Complex Problem Activities			
	Attribute	**Complex Activities**	
1	Range of resources	Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies).	✓
2	Level of interaction	Require resolution of significant problems arising from interactions between wide ranging and conflicting technical, engineering or other issues.	✓
3	Innovation	Involve creative use of engineering principles and research-based knowledge in novel ways.	✓
4	Consequences to society and the environment	Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation.	✓
5	Familiarity	Can extend beyond previous experiences by applying principles-based approaches.	

Abstract

In Pakistan, where the poultry sector plays a vital role in the country's economy and food security, egg incubators have provided numerous benefits. They have had a transformative impact on the poultry industry in Pakistan. The introduction of egg incubators has revolutionized the industry by significantly increasing hatch rates and improving overall efficiency. Traditionally, egg incubation required constant manual monitoring and adjustment of environmental conditions. With the IoT-based smart egg incubator, these processes are automated and can be conveniently managed through a smartphone or web application. Our “Smart Egg Incubator” incorporates IoT technology and is equipped with various sensors to gather real-time data on temperature, humidity, and air quality, ensuring the ideal conditions for egg development.

Acknowledgments

Dr. Ata ul Aziz Ikram has our sincere gratitude for providing invaluable guidance, comments, and suggestions throughout the course of this project. His efforts towards providing us assistance with technical guidance throughout the course of the project are highly appreciated.

Table of Contents

Table of Contents

List of Figures

Chapter 1. Introduction

The expansion of our poultry industry is strongly correlated with the rise of the population in Pakistan. In contrast to industrialized nations where 41kg of meat and 300 eggs are available annually, Pakistan has a per capita availability of poultry meat and eggs of 5kg and 51 eggs annually. Our annual consumption of eggs is approximately 15 billion on average but only 10 billion eggs are produced in Pakistan annually.

Our product aims to target the 1.7K imports of incubators as Pakistan is one of the largest importers. Our product is quite cost efficient and has the following enhanced features:

- Completely automatic incubators with ceramic heaters and cooler to control extreme weather conditions.
- Self-designed controller that can store run time data in history. This prevents the initial conditions of incubator from resetting in case of power shortage.
- Adjustable rotating trays that can be adjusted according to the size of egg placed thus allowing us the hatching of different species as well.

Egg incubators have played a significant role in revolutionizing the poultry industry by offering numerous benefits and advancements. Overall, egg incubators have transformed the poultry industry by improving hatch rates, optimizing resource utilization, enhancing disease control and reducing labor intensity. These advancements have contributed to increased productivity, improved profitability, and sustainable growth in the poultry sector.

1.1.Motivation

A single hen hatches approximately 200-250 eggs each year whereas egg incubators can hatch around 200-300 eggs in a single batch. Usage of egg incubators also results in lower diseases rate as compared to open shed method of hatching, and as we are all aware of the fact that Pakistan's poultry industry has suffered greatly due to disease outbreak and spread of viruses that caused a great decline in their production. According to WHO, our annual production should be approximately 3 trillion to provide proper nutrition to our generation.

1.2.Problem Statement

"Design an IoT-enabled environmentally controlled egg incubation system."

Poultry industry plays a vital role in our national economy and in meeting the nutrition needs for our population at large. Improper management of environmental conditions such as temperature and humidity greatly affect our production rate. Egg Incubators create an environment with ideal conditions for chicks to grow, performing the role of a broody hen. Our project will increase the production of A-grade chicks in a more efficient and economical way as well as allow us to remotely monitor them.

1.3. Literature Review

A LOW-COST SMART EGG INCUBATOR:

This research looks into the creation and use of a low-cost, automated incubator for nearby chicken producers. Its goal is to create a low-cost smart incubator that will guarantee the preservation of the ideal environmental conditions required for egg hatching. [1]

- Only the cost factor was handled with the addition of no new features in this incubator.

DEVELOPMENT OF AN AUTOMATIC ELECTRIC EGG INCUBATOR:

Using materials found locally, an electric-powered incubator was created and put to the test on hatchable hen eggs. The goal was to boost day-old chick production for small and medium-sized poultry farmers and create a low-cost incubator. [2]

- This is a battery-powered incubator but in order to keep the cost low wasn't an efficient one. Poor monitoring of environmental conditions.

DESIGN AND IMPLEMENTATION OF AUTOMATIC FIXED FACTORS EGG INCUBATOR:

This project uses a programmed microcontroller to activate and deactivate components. Well-lagged rectangular box material is used in order to prevent surrounding air enter into the box. [3]

- Conditions controlled and monitored but don't incorporate a cooler to handle extreme weather conditions. Any enhanced feature also missing.

1.4.Report Outline

This report consists of the chapters listed below.

In Chapter 2, the proposed solution is discussed in detail. It includes details of the Block diagram, a Flow chart of the process, and the interfacing of the sensors with the processing module. It also includes hardware and software specifications.

Experimental results are explained in Chapter 3. It further discusses the conclusions drawn from the obtained results and the recommendations/future work that is proposed for further enhancements. The project budget, milestones achieved, SDG mapping,, and lifelong learning are discussed here as well.

Chapter 2. Solution Design & Implementation

The design of the whole system is discussed in this section. The block diagram and module specifications are explained in section 2.1. The details of the flowchart are given in section 2.2 whereas; Section 2.3 discusses the software implementation. Section 2.4 discusses the hardware implementation of the modules.

2.1. Block Diagram

The block diagram of the design implemented is presented in this section. Technical specifications of each part are discussed below.

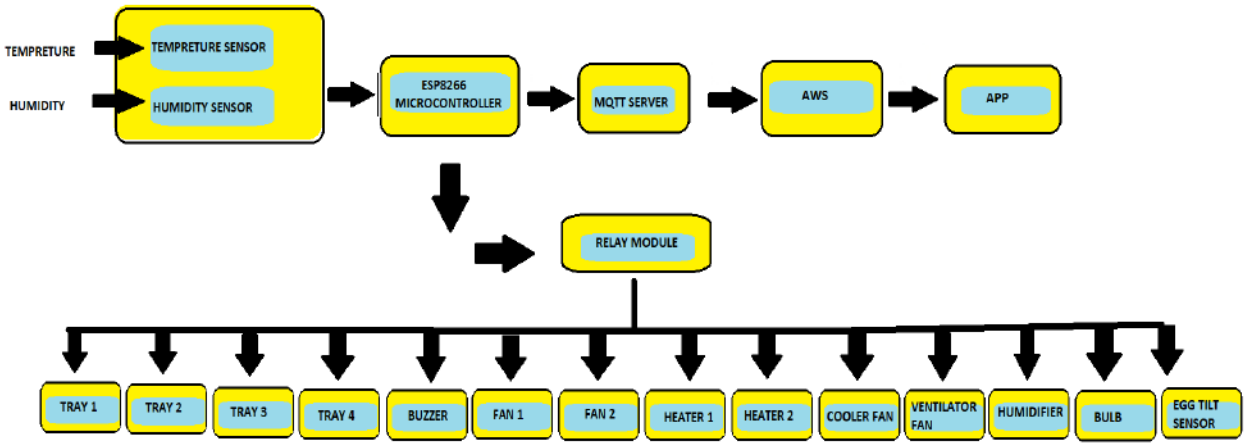

Figure 2.1-a Block diagram of the project

- **Node Specifications**

The NodeMCU ESP8266 microcontroller is utilized in a system designed to create a controlled environment based on input values from temperature and humidity sensors. This system aims to control various components such as fans, a humidifier, a ceramic heater, a cooler, and a buzzer to establish the desired environmental conditions. Additionally, the system incorporates a tray rotation mechanism based on built-in timer codes and provides real-time information on temperature, humidity, and tray states. Furthermore, a tilt sensor is employed to detect the tilt of eggs during rotation, and all data is saved in a database for monitoring purposes. The system allows for monitoring survival and mortality rates, and the relays are directly controlled through an application.

- **Environmental Control System**

The main objective of the system is to create a controlled environment by actively managing temperature and humidity. The NodeMcu ESP8266 microcontroller acts as the central processing unit, receiving input values from temperature and humidity sensors. These sensors provide real-time data on the current environmental conditions. Based on this data, the microcontroller controls the operation of various components to maintain the desired conditions. The fans, humidifier, ceramic heater, cooler, and buzzer are connected to the microcontroller and are activated or deactivated according to the predefined set points.

- **Tray Rotation Mechanism**

In addition to environmental control, the system incorporates a tray rotation mechanism. This mechanism ensures that the eggs are evenly exposed to the controlled environment for optimal incubation conditions. The rotation is accomplished using built-in timer codes that trigger the rotation mechanism at regular intervals. By rotating the trays, the system minimizes the potential for temperature and humidity variations within the incubator. This rotation process ensures consistent heat distribution and increases the chances of successful hatching.

- **Real-Time Monitoring and Data Storage**

To facilitate monitoring and analysis, the system includes a database for storing all relevant data. The microcontroller collects and saves information such as temperature, humidity, tray states, and tilt sensor readings. This data can be accessed and analyzed later to monitor the survival and mortality rates of the eggs during the incubation process. By tracking these rates, the system provides valuable insights into the overall health and success of the incubation process.

- **Control via Application**

The system allows for convenient control and management through an application. The application serves as an interface for the user to monitor and adjust the environmental parameters as needed. The user can remotely control the relays connected to the fans, humidifier, ceramic heater, cooler, and buzzer through the application. This feature provides flexibility and ease of use, enabling the user to fine-tune the incubation environment in real-time. In summary, the NodeMcu ESP8266 microcontroller-based system is designed to create a controlled environment for incubating eggs. By actively managing temperature and humidity and incorporating a tray rotation mechanism, the system aims to optimize the hatching process. Real-time monitoring, data storage, and remote-control capabilities enhance the user experience and enable effective management of the incubation process.

2.2. Flow Chart

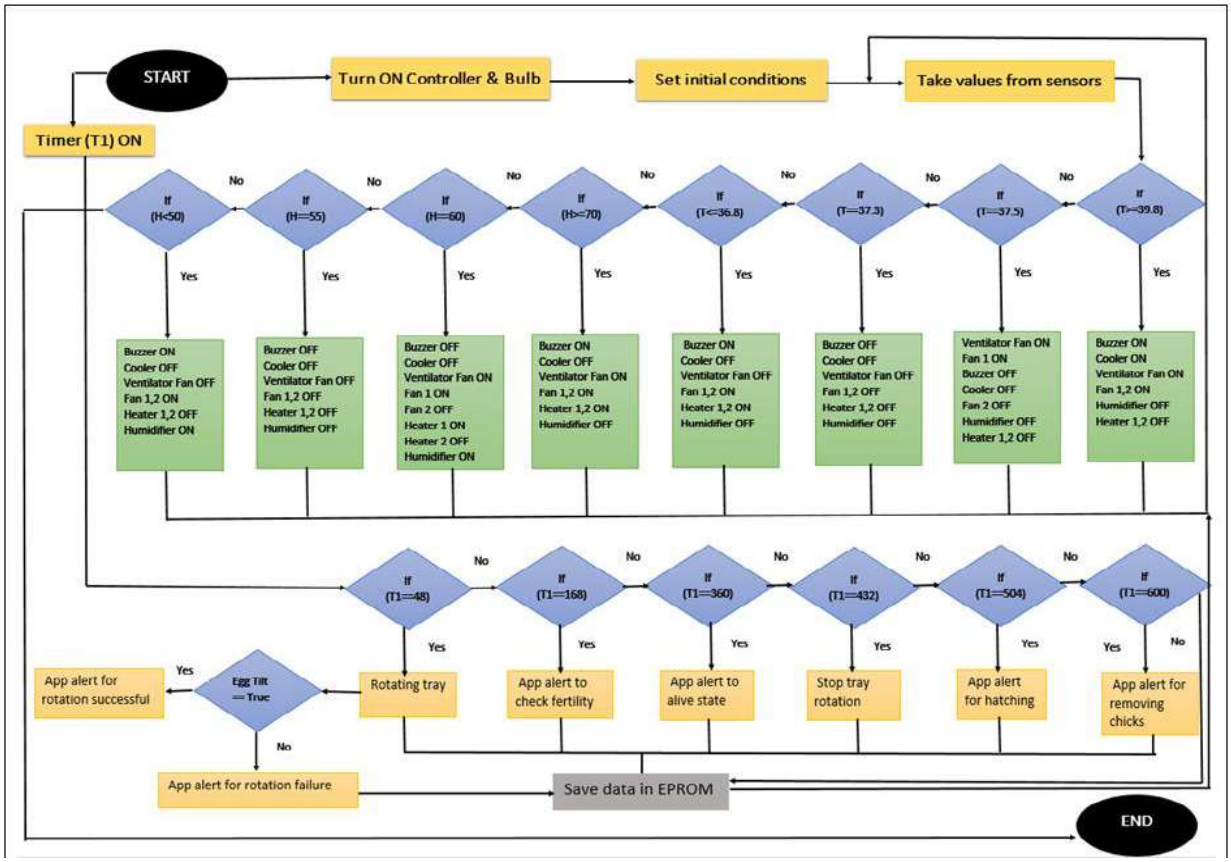

Figure 2.2-a Flow chart of project

Flowchart Overview

The flowchart provides an overview of the working conditions and actions taken by the controller to maintain the desired environmental conditions. It illustrates the decision-making process and the control of various loads to achieve the desired outcomes. The flowchart serves as a visual representation of the system's logic and control flow.

- **Load Control and Environmental Conditions**

This section of the flowchart focuses on load control and the maintenance of environmental conditions. It outlines the decision-making process based on input values from temperature and humidity sensors. The flowchart depicts how the microcontroller determines whether to turn on or off specific loads such as fans, humidifier, ceramic heater, cooler, and buzzer based on the measured environmental conditions. The conditions for activating or deactivating each load are clearly defined, ensuring the system maintains the desired temperature and humidity levels.

Temperature Conditions:

There are 4 conditions applied on temperature on the basis of which load will be controlled.

1. **If (Temp >= 39.8 degrees)**
   After reading the temperature from the sensor, the controller will check whether the temperature is greater than 39.8 degrees and if the condition is satisfied, it will turn on the buzzer to intimate the user of the extreme condition, a cooler fan to provide cool air inside the incubators body to reduce temperature, ventilator fan be on to take out hot air, circulating fans 1, 2 will be on to circulate cool air inside to lower the temperature, humidifier and heaters will remain off. Then again sensor will check the temperature for further actions.

2. **If (Temp >= 37.5 degrees)**
   After reading the temperature from the sensor, the controller will check whether the temperature is greater than or equal to 37.5 degrees and if the condition is satisfied then it will keep the buzzer, cooler fan, humidifier, fan 1, and heaters off. Only ventilator fan and fan 1 will be on to balance the conditions. Then again sensor will check the temperature for further actions.

3. **If (Temp == 37.3 degrees)**
   After reading the temperature from the sensor, the controller will check whether the temperature is equal to 37.3 degrees and if the condition is satisfied then it will keep all the loads off to balance conditions. Then again sensor will check the temperature for further actions.

4. **If (Temp <= 36.8 degrees)**
   After reading the temperature from the sensor, the controller will check whether the temperature is less than or equal to 36.8 degrees and if the condition is satisfied then it will turn on the buzzer, circulating fans, and the heaters. Remaining loads will remain off. Then again sensor will check the temperature for further actions.

Humidity Conditions:

There are 4 conditions applied on humidity on the basis of which load will be controlled.

1. **If (Humid >= 70)**
   After reading humidity from the sensor if it satisfies the above condition then the controller will turn on the buzzer, ventilator fan, and heaters as the temperature have an inverse relation with humidity. Circulating fans, humidifier and the cooler will remain off. Then again sensor will check the humidity for further actions.

2. **If (Humid == 60)**
   After reading humidity from the sensor if it satisfies the above condition then the controller will turn on the ventilator fan and heater 1 as the temperature has an inverse relation with humidity. Circulating fan 1 and the remaining loads will be off. Then again sensor will check the humidity for further actions.

3. **If (Humid == 55)**
   After reading humidity from the sensor if it satisfies the above condition then the controller will keep all the loads off to maintain balanced conditions. Then again sensor will check the humidity for further actions.

4. **If (Humid <= 50)**
   After reading humidity from the sensor if it satisfies the above condition then the controller will turn on the buzzer, humidifier and circulating fans. All the other loads will remain off to maintain balanced conditions. Then again sensor will check the humidity for further actions.

- **Data Storage and Cloud Integration**

The flowchart includes a segment dedicated to data storage and cloud integration. This section illustrates the process of saving the data received by the cloud into a database. The flowchart highlights how the microcontroller communicates with the cloud platform and sends the received data for storage. This stored data becomes part of the backend for the user interface application, enabling further analysis and visualization of the environmental conditions.

- **User Interface Application**

The user interface application section of the flowchart demonstrates how the stored data is presented to the user in an interactive manner. It showcases the connection between the backend database and the user interface, highlighting how the data is retrieved and displayed to the user. The flowchart may illustrate the various components of the user interface, such as graphs, charts, or real-time updates, to enhance the user experience and provide a comprehensive view of the incubation environment.

- **Tray Control and MongoDB**

This section of the flowchart focuses on the control of trays and the storage of live status data. The flowchart outlines how the built-in timers control the rotation of trays at specific intervals. Additionally, it illustrates the process of saving the live status data, including tray positions and other relevant information, in MongoDB. This storage mechanism ensures that the last saved states are retained in case of any power breakdown, allowing the system to resume operation seamlessly.

In summary, the flowchart provides a detailed visualization of the decision-making process and control flow within the system. It encompasses load control, data storage, cloud integration, user interface application, tray control, and AWS storage. By dividing the flowchart into specific

headings, the different aspects of the system's functionality are clearly depicted and easy to understand.

2.3. Software Implementation

2.3.1. ESP8266 NODEMCU Implementation

- **Species Selection and Incubation Conditions**

The application developed for the system allows users to select the specific type of eggs they wish to place inside the incubator. This feature enables customization of the incubation conditions based on the requirements of the selected species. The application provides a user-friendly interface where users can input their preferences and select the appropriate species from a list or input it manually.

Once the species is selected, the application provides options to set the desired incubation conditions. These conditions may include temperature, humidity, and other environmental parameters specific to the chosen species. The user can input the preferred temperature range, humidity level, and any other specific requirements for successful egg incubation.

The selected conditions are then communicated to the NodeMcu ESP8266 microcontroller, which acts as the central control unit for the incubator system. The microcontroller receives the user-defined incubation conditions from the application and adjusts the operation of various components accordingly. For example, if a user selects a species that requires a higher temperature range, the microcontroller will activate the ceramic heater to increase the temperature within the incubator. Similarly, if the chosen species requires higher humidity levels, the microcontroller will activate the humidifier to raise the humidity to the desired level.

The microcontroller continuously monitors the environmental conditions using temperature and humidity sensors. It compares the measured values with the user-defined set points and takes appropriate actions to maintain the conditions within the desired range. If the temperature or humidity deviates from the set points, the microcontroller adjusts the operation of the relevant loads to bring the conditions back to the desired levels.

Throughout the incubation process, the application provides real-time feedback on the current environmental conditions inside the incubator. The user can monitor the temperature and humidity values through the application's interface, ensuring that the selected species' requirements are being met.

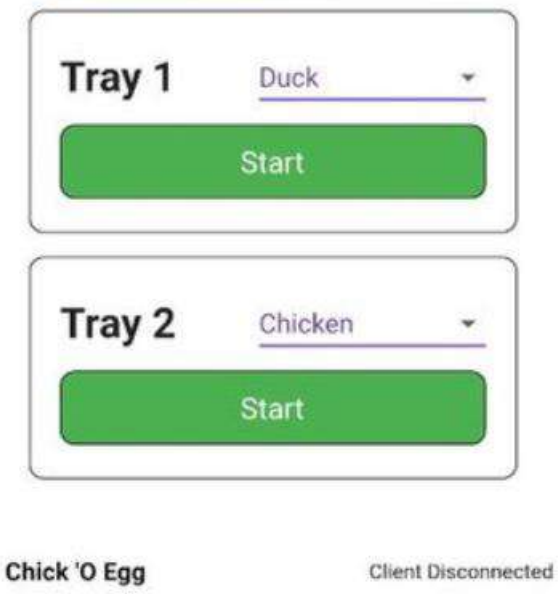

Figure 2.3.1-a Specie Selection

- **DHT22 Sensors Placement and Communication:**

In the incubation system, three DHT22 sensors are strategically placed in different corners to ensure accurate monitoring and efficient balancing of the environmental conditions inside. This placement allows for comprehensive coverage of the incubator space, minimizing any variations or inconsistencies in temperature and humidity levels.

The DHT22 sensors continuously measure temperature and humidity readings at their respective locations. The sensors are connected to the NodeMcu ESP8266 microcontroller, which acts as the central processing unit for the system. The microcontroller receives data from each sensor after every 5 seconds, providing real-time updates on the current conditions in the incubator.

The communication between the DHT22 sensors and the ESP8266 microcontroller is achieved through appropriate wiring and protocols. The microcontroller is programmed to receive the sensor data and process it to determine the necessary actions for maintaining the desired conditions.

Load Operation Based on Conditions:

Once the microcontroller receives the temperature and humidity readings from the DHT22 sensors, it compares these values with the user-defined set points for the selected species. The microcontroller employs decision-making algorithms to analyze the data and determine the appropriate operation of the loads, such as fans, humidifier, ceramic heater, cooler, and buzzer.

For example, if the temperature reading from one of the sensors indicates that the temperature has exceeded the upper threshold, the microcontroller triggers the cooling mechanism by activating the cooler. This helps to bring down the temperature to the desired range.

Similarly, if the humidity reading from another sensor falls below the lower threshold, the microcontroller signals the activation of the humidifier to increase the humidity level within the incubator.

The microcontroller continuously monitors the readings from all three sensors and adjusts the operation of the loads based on the specific conditions detected. This dynamic control mechanism ensures that the environmental conditions remain within the defined range, providing an optimal incubation environment for the eggs.

The periodic communication between the sensors and the microcontroller, along with the intelligent load operation based on the conditions, contributes to the efficient balancing of temperature and humidity within the incubation system. This balance is essential for the successful incubation and hatching of the selected species.

Figure 2.3.1-b DHT22 Temperature & Humidity Sensor

2.3.2. Cloud Working

- **AWS IoT Core**

AWS IoT Core, a cloud service provided by Amazon Web Services, serves as the foundation for this project. It enables seamless communication between IoT devices and the cloud. By leveraging AWS IoT Core, we can ensure secure and efficient data transmission from the incubator to the mobile app. The core functionality of AWS IoT Core involves device registration, authentication, and data exchange via MQTT.

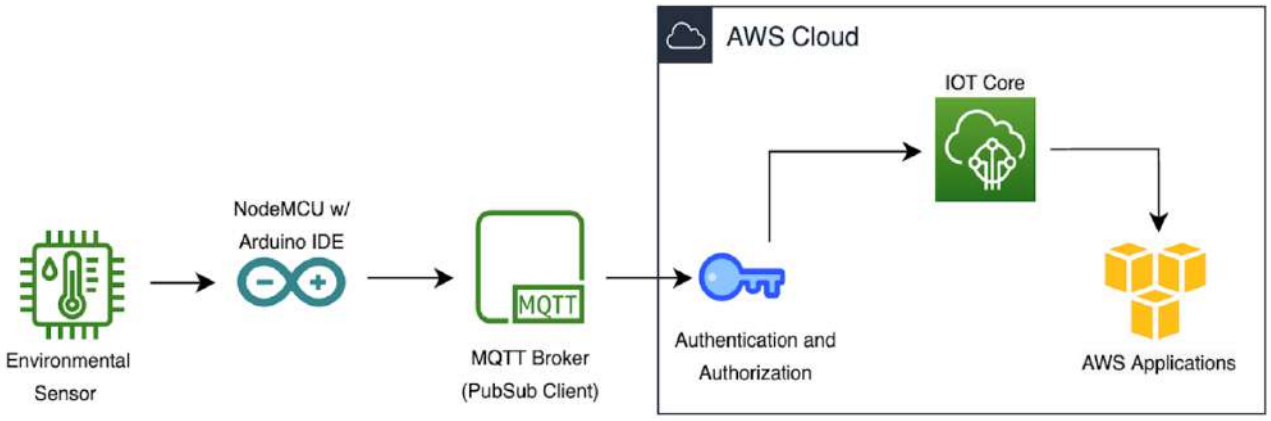

Figure 2.3.2-a AWS with NodeMcu

- **Device Configuration and Authentication**

To establish a secure connection between the IoT devices and AWS IoT Core, each device, including the NodeMcu acting as the gateway device, is registered as an IoT Thing within the AWS IoT Core platform. Device authentication is accomplished using X.509 certificates and private keys, ensuring that only authorized devices can connect to the AWS IoT Core service. These credentials are generated and securely stored on the NodeMcu, enabling it to establish a secure MQTT connection with AWS IoT Core.

- **Sensor Data Publication**

The NodeMcu, equipped with the ESP8266 Wi-Fi module, is responsible for collecting data from the sensors within the incubator. Once the data is acquired, the NodeMcu publishes it to an MQTT topic in AWS IoT Core. MQTT topics serve as logical channels for data exchange. In this project, a topic named "incubator/load_statuses" is created specifically for publishing the statuses of the loads within the incubator. Publishing the sensor data to this topic ensures that the information is readily available for retrieval by the mobile app.

- **Mobile App Integration**

To provide users with real-time access to the load statuses of the incubator, a mobile app is developed to connect with AWS IoT Core and subscribe to the "incubator/load statuses" MQTT topic. By subscribing to this topic, the app establishes a bidirectional communication channel with AWS IoT Core, allowing it to receive updates whenever new data is published. This integration empowers the app to display the load statuses in a user-friendly format, providing end users with valuable insights and control over the incubator's conditions.

- **Data Processing and Storage**

AWS IoT Core provides seamless integration options with other AWS services, enabling additional functionalities for data processing and storage. For instance, AWS IoT Rules can be configured to process incoming sensor data and trigger actions based on predefined conditions. This project may leverage AWS Lambda functions to perform further processing of the data or store it in an Amazon DynamoDB database for historical analysis. These capabilities enhance the overall functionality of the system, enabling advanced analytics and decision-making based on the collected data.

By employing AWS IoT Core as the communication backbone, this project establishes a robust and secure system for monitoring and controlling an incubator's loads. The integration of sensor devices, NodeMcu as the gateway device, AWS IoT Core, and a mobile app ensures real-time data visualization, empowering users with the ability to make informed decisions regarding the incubator's load statuses.

- **MQTT Protocol Procedure**

The MQTT (Message Queuing Telemetry Transport) protocol plays a crucial role in the transmission of live load statuses, temperature, and humidity values from the incubator to AWS IoT Core. The NodeMcu, acting as the gateway device, utilizes MQTT to establish a lightweight and efficient communication channel. Upon successful authentication with AWS IoT Core, the NodeMcu subscribes to specific MQTT topics to receive commands or configuration updates, if applicable. Simultaneously, it publishes the live load statuses, temperature, and humidity values to the designated MQTT topic "incubator/load statuses." The MQTT publish operation encapsulates the data within a payload and assigns it a Quality of Service (QoS) level. The QoS level determines the level of reliability and guarantees of message delivery. In this project, QoS level 1, which ensures at least once delivery, is employed to prioritize data integrity. AWS IoT Core, acting as an MQTT broker, receives the published messages and performs necessary validations and processing. Subsequently, AWS IoT Core forwards the messages to the mobile app, which has subscribed to the "incubator/load statuses" topic. This enables the mobile app to receive live updates of the load statuses, temperature, and humidity values, ensuring real-time monitoring and control of the incubator's conditions for enhanced decision-making and operational efficiency.

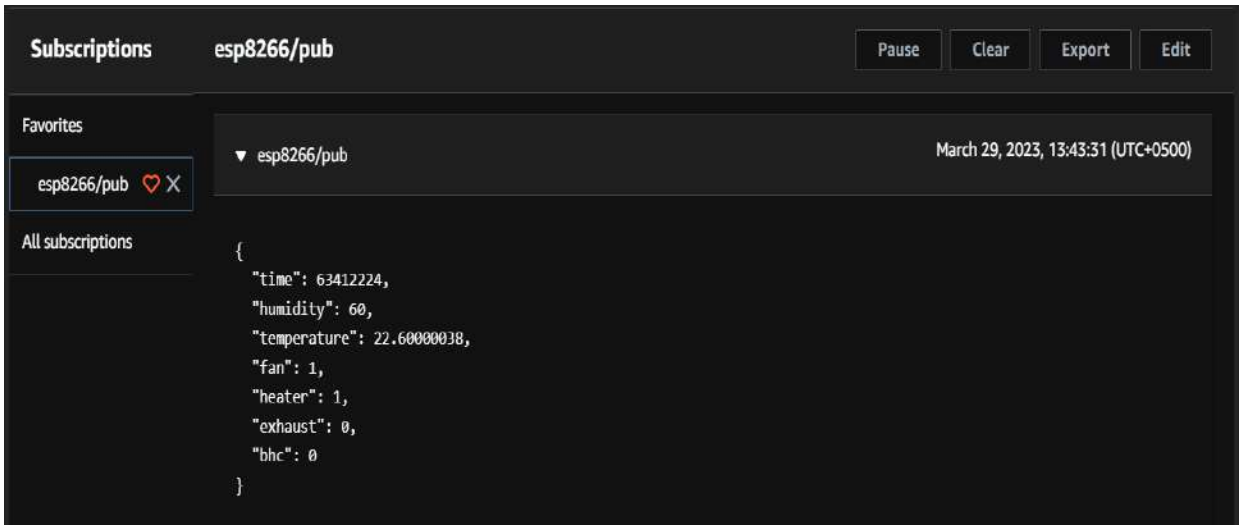

Figure 2.3.2-b Data on AWS Platform

2.3.3. Application Development

- **Application Development with Flutter in Visual Studio**

In this project, an application is developed using the Flutter framework in Visual Studio, targeting both iOS and Android platforms. The application serves as a user interface to monitor and control an incubator's conditions, leveraging data received from AWS IoT Core.

- **Displaying Live Temperature and Humidity Values**

The application integrates with AWS IoT Core by subscribing to the "incubator/load_statuses" MQTT topic. Upon receiving updates from AWS, the live temperature and humidity values are extracted from the messages and displayed in real-time on the application's user interface. The values are presented in a visually appealing and easily understandable format, providing users with immediate insights into the incubator's environmental conditions.

- **Live Statuses of Loads**

In addition to temperature and humidity, the application also receives live statuses of loads within the incubator from AWS IoT Core. These statuses can include information about the status of specific components or parameters, such as power supply, ventilation, or any other relevant load within the incubator. The application dynamically updates and displays the statuses, allowing users to promptly identify any issues or abnormalities and take appropriate actions.

- **Tilt Detection Status of Trays**

To ensure the safety and integrity of the incubator, trays inside it are equipped with tilt sensors. These sensors detect any tilting or movement of the trays and transmit the information to AWS IoT Core. The application subscribes to the corresponding MQTT topic to receive updates on the tilt detection status. The real-time status is then reflected on the application's interface, alerting users in case of tray displacement or any tilting occurrences that may affect the incubation process.

- **Mortality Rate Calculator**

The application incorporates a mortality rate calculator feature to provide users with insights into the success of the incubation process. Users can input the total number of eggs placed inside the incubator, and the application calculates the mortality rate by subtracting the number of hatched eggs from the total, and then multiplying the result by one hundred. The calculated mortality rate is displayed to users, enabling them to evaluate the efficiency of the incubation process and make informed decisions for improvements if needed.

- **Tray Species Selection**

Considering the diverse needs of incubator management, the application allows users to select specific species or type of tray being used. Users can choose from a predefined list of tray species or add custom specifications. This selection helps the application to provide species-specific recommendations, alerts, and optimized settings for temperature, humidity, and other environmental parameters.

- **Graph Display of Temperature and Humidity**

To offer a comprehensive view of environmental trends, the application features graphical representations of temperature and humidity data collected over time. The app retrieves historical data from AWS IoT Core and presents it using line charts or other visualizations. This graph display allows users to identify patterns, trends, or anomalies in the incubator's conditions, facilitating better decision-making and proactive management of the incubation process. By developing the application using Flutter in Visual Studio, users gain a user-friendly interface to monitor and control the incubator. The application displays live temperature and humidity values, statuses of loads, tilts detection status of trays, mortality rate calculations, species selection, and visualizes historical data using graphs. These features enhance the user experience, provide valuable insights, and enable efficient management of the incubation process.

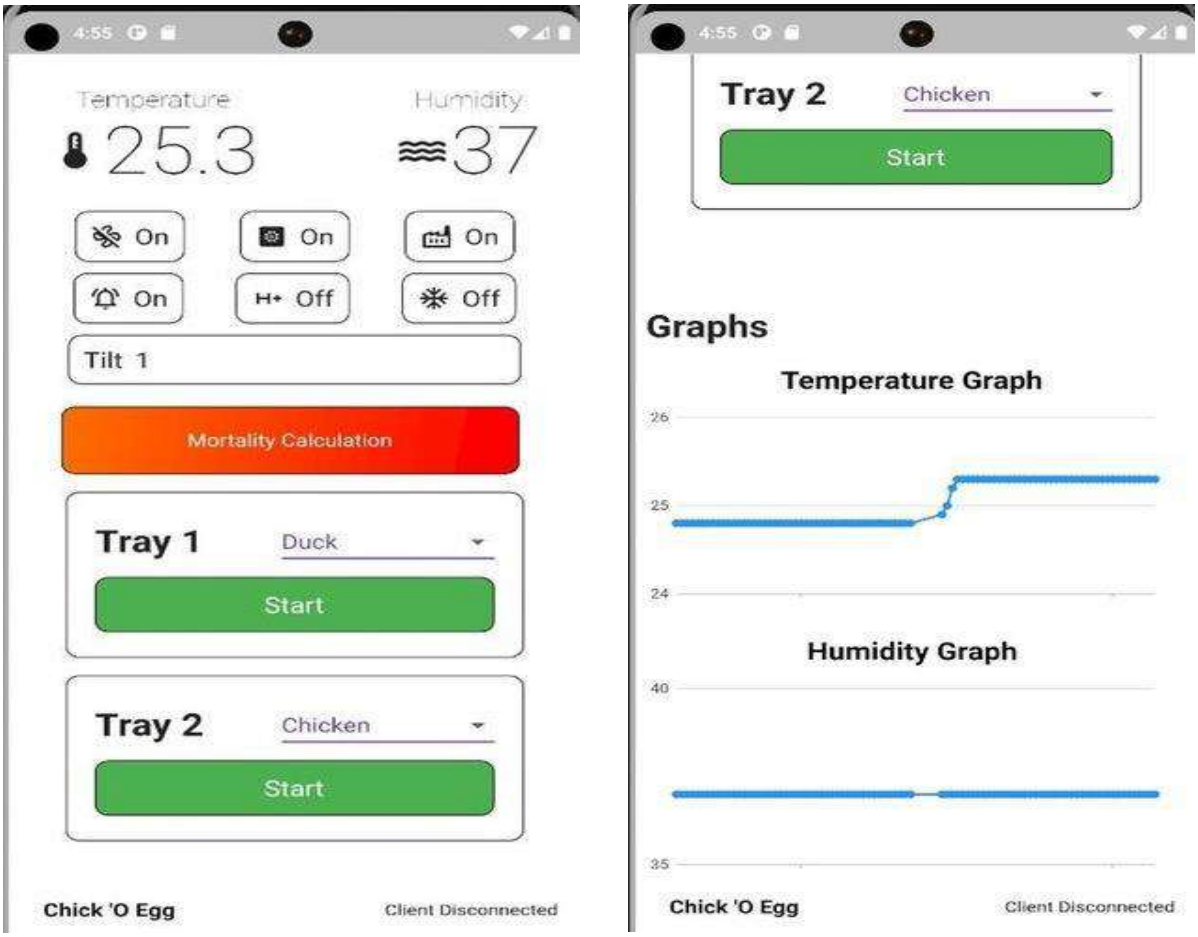

Figure 2.3.3-a Application

2.3.4. Login and Sign-Up Functionality

The Flutter application includes a login and sign-up feature to provide user authentication and access control. The login functionality allows registered users to securely log into the application using their credentials, while the sign-up functionality enables new users to create an account.

- **Backend Database: MongoDB**

To store user credentials securely, the application integrates with a MongoDB database at the backend. MongoDB is a popular NoSQL database that offers flexibility, scalability, and robust document-based storage. The database is hosted on a server and provides a reliable and efficient way to store user information.

- **Sign-Up Process**

When a new user wants to create an account, they navigate to the sign-up screen in the Flutter application. The user provides their desired username, email address, and password. Upon submission, the application sends an HTTP request to the backend server, which handles the request and interacts with the MongoDB database.

- **Storing User Credentials in MongoDB**

The backend server receives the sign-up request and validates the provided information. It checks if the username or email address is already registered to prevent duplicates. If the information is valid, the server securely stores the user's credentials in MongoDB. The password is typically hashed and salted using encryption techniques to enhance security.

- **Login Process**

For existing users, the login process begins when they enter their username or email address and password on the login screen of the Flutter application. The application then sends an HTTP request to the backend server to verify the credentials.

- **Credential Verification**

The backend server receives the login request and retrieves the user's stored credentials from MongoDB based on the provided username or email address. It then compares the hashed and salted password stored in the database with the password provided during login. If the credentials match, the server generates a secure authentication token, which is returned as a response to the application.

- **Token-Based Authentication**

Upon successful login, the Flutter application receives the authentication token from the backend server. This token serves as proof of the user's identity and is securely stored locally on the device. The token is used for subsequent API requests to the backend, allowing the application to authenticate the user's actions without requiring them to re-enter their credentials for each request.

- **Error Handling and Security**

Throughout the login and sign-up processes, the application and backend server implement error handling mechanisms to address potential issues, such as invalid credentials, network errors, or database connection failures. Additionally, security measures, such as encryption, hashing, and token-based authentication, are employed to protect user credentials and ensure secure communication between the Flutter application and the MongoDB database.

By implementing a login and sign-up functionality in the Flutter application and storing user credentials in a MongoDB database at the backend, users can securely create accounts, log in, and access the application's features. This approach ensures data integrity, confidentiality, and a seamless user experience.

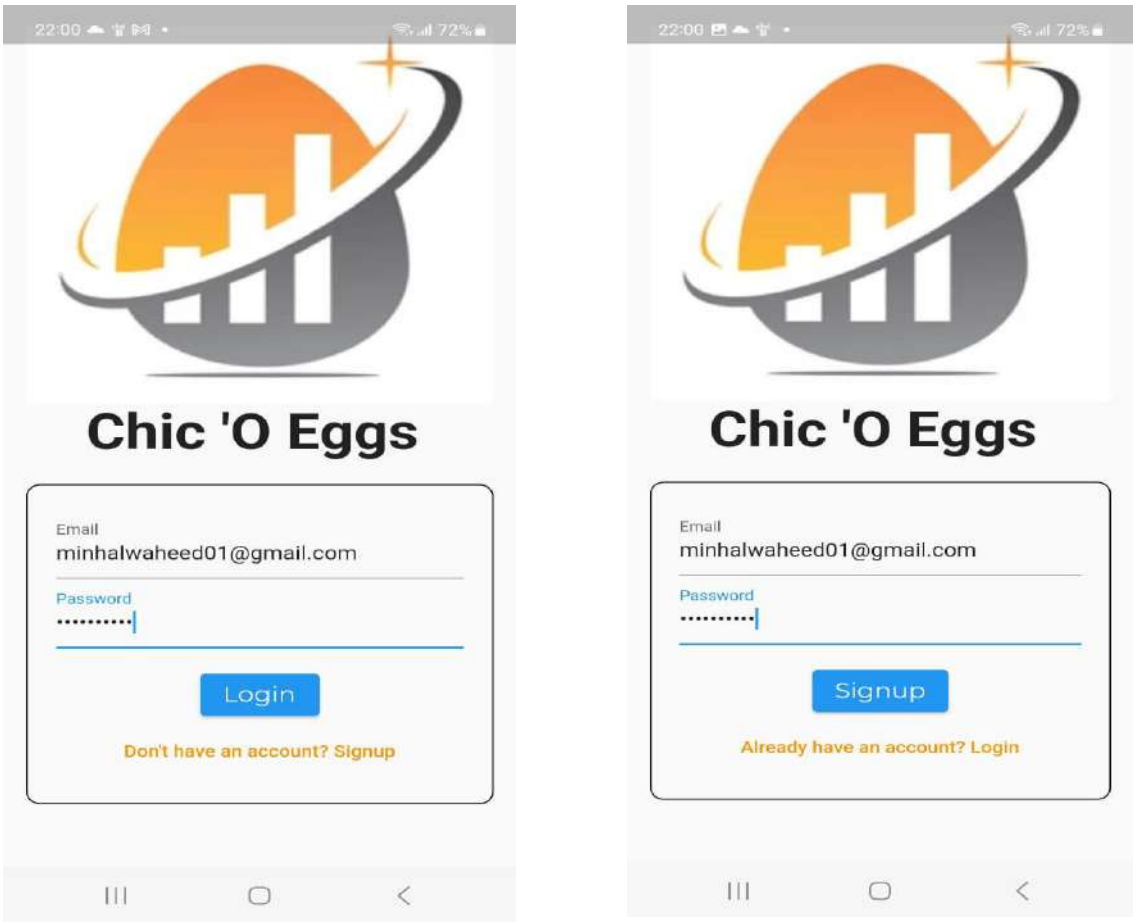

Figure 2.3.4-a Application Login & Sign up

2.4. Hardware Implementation

2.4.1. Structure

The structure was first designed on software. As visible from the diagram the outer structure frame is made of wood. The wooden structure is covered with PVC sheets for better environmental control. Inside the PVC sheets, we installed thermocol sheets. This multilayered specific structure can control temperature and humidity quite efficiently.

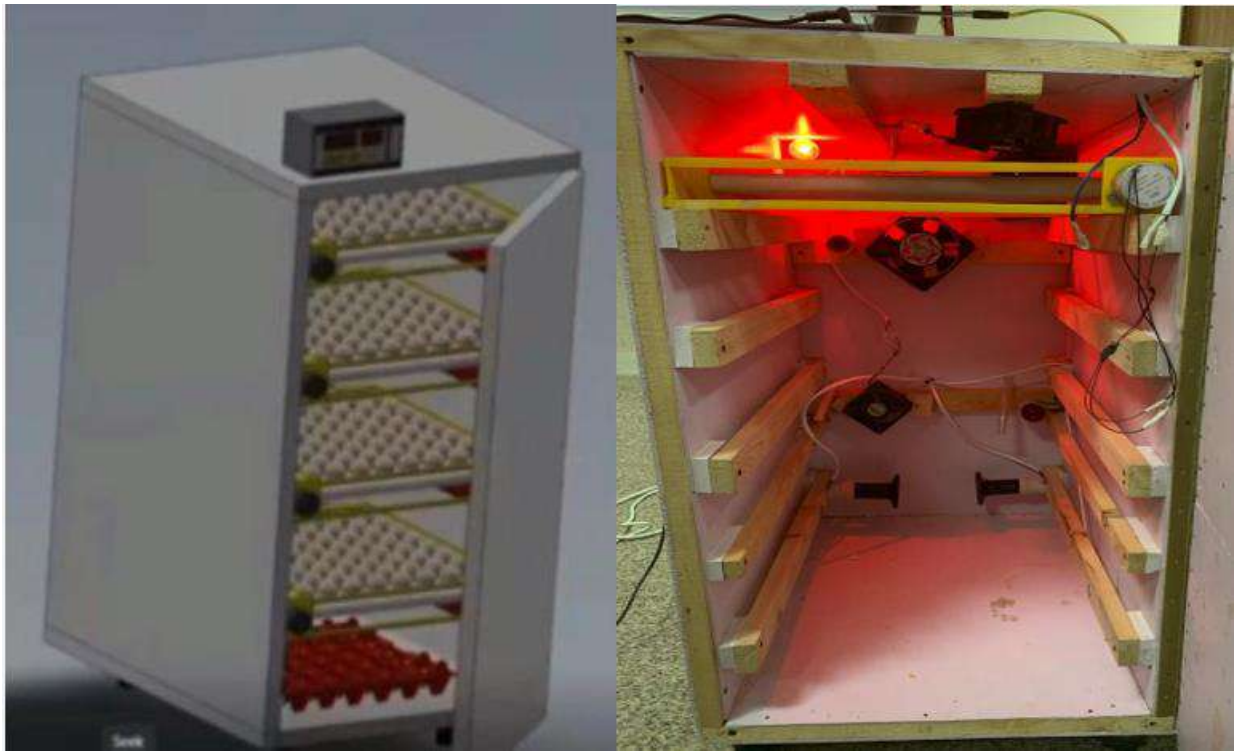

Figure 2.4.1-a Hardware Structure

2.4.2. Controller

The hardware includes a node and a single gateway. The node includes a microcontroller, sensors, fans (DC), ceramic heaters, rotating stepper motor trays, a cooler, a humidifier and a buzzer.

ESP8266 (NODEMCU)

The reason for using this microcontroller is the built-in Wi-Fi module and its different power consumption for different modes of operation.

- **High Durability**
  ESP8266EX is capable of functioning consistently in industrial environments, due to its wide operating temperature range. With highly-integrated on-chip features and minimal external discrete component count, the chip offers reliability, compactness, and robustness.
- **Power-Saving Architecture**
  Engineered for mobile devices, wearable electronics, and IoT applications, ESP8266EX achieves low power consumption with a combination of several proprietary technologies. The power-saving architecture features three modes of operation: active mode, sleep mode, and deep sleep mode. This allows battery-powered designs to run longer.
- **Compactness**

ESP8266EX is integrated with a 32-bit Tensilica processor, standard digital peripheral interfaces, antenna switches, RF balun, power amplifier; low noise receives amplifier, filters and power management modules. All of them are included in one small package, ESP8266EX.

- **32-bit Tensilica Processor**
  The ESP8266EX microcontroller integrates a Tensilica L106 32-bit RISC processor, which achieves extra-low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow about 80% of the processing power to be available for user application programming and development.

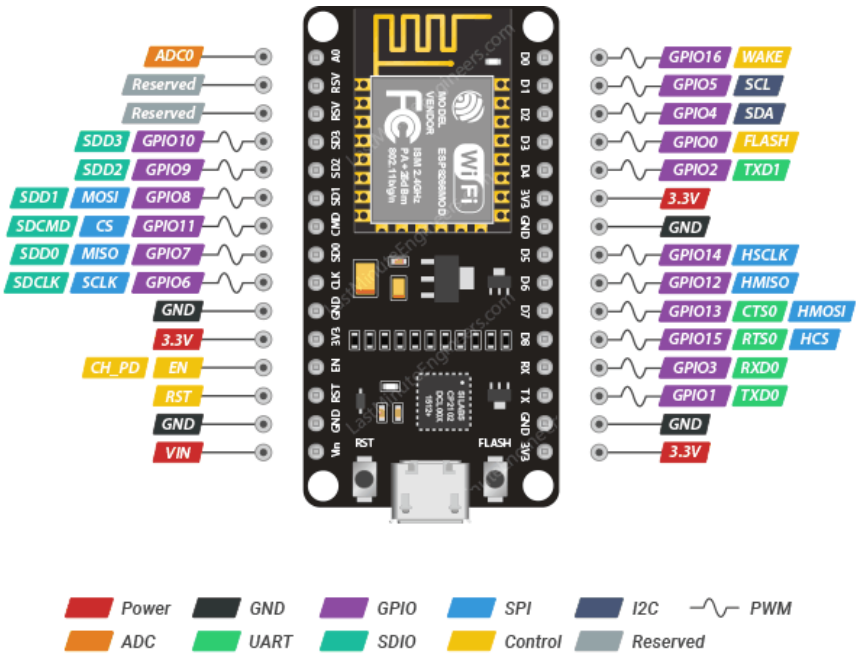

Figure 2.4.2-a ESP8266

Arduino

Arduino is an open-source electronics platform for creating interactive projects. It consists of a microcontroller board and software for programming and controlling various electronic components. It is widely used for prototyping and DIY electronics projects.

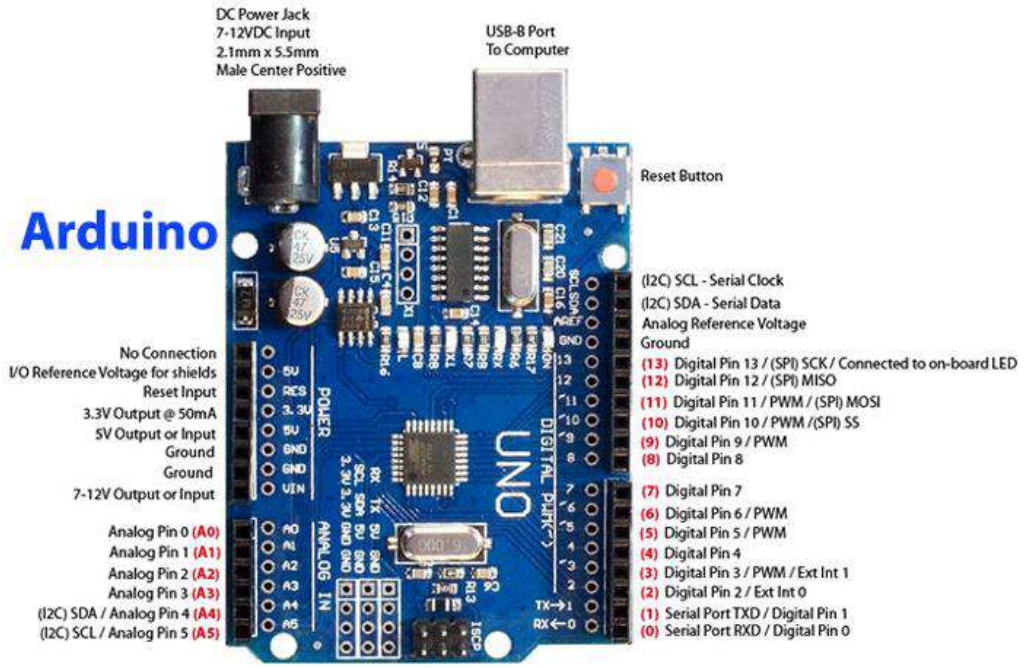

Figure 2.4.2-b Arduino Uno

Relay Module

Relay module is used to control the load attached to the controller. It's an electrical switch that is operated by an electromagnet. The electromagnet is activated by a separate low-power signal from a microcontroller. When activated, the electromagnet pulls to either open or close an electrical circuit.

Figure 2.4.2-c Relay Module

Egg Rotating Trays

Egg rotating stepper motor-based tray is used that will rotate the tray with an angle of 45 degrees four times to give the egg a tilt of 180 degrees after every 6 hours.

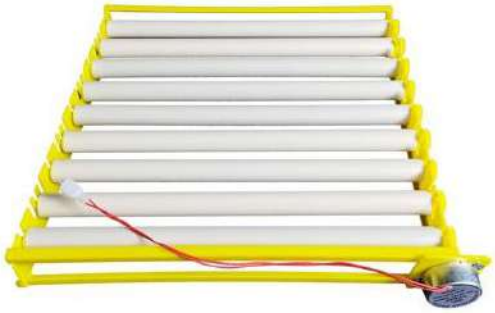

Figure 2.4.2-d Egg Rotating Tray

DC Fan

Dc fans are used for the purpose of cooling, ventilation and circulation of air inside the incubators body.

Figure 2.4.2-e DC Fan

Ceramic Heaters

Ceramic heaters are used to produce heat inside the incubator body to provide the ideal temperature for chick development inside the egg shell.

Figure 2.4.2-f Ceramic Heaters

Egg Tilt Sensor

It is used to monitor the rotation of the rotating tray. It will be connected with NodeMcu to receive the signal notifying if the stepper motor has rotated after every 6 hours or not.

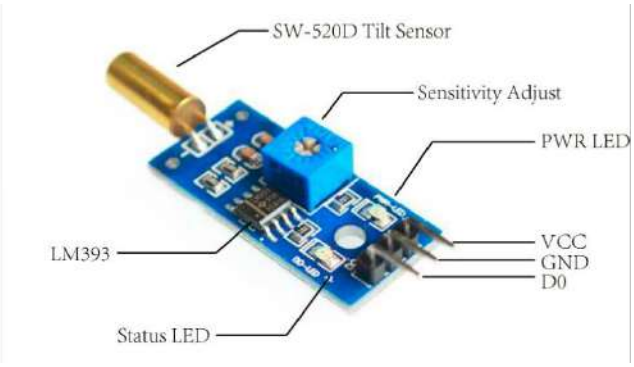

Figure 2.4.2-g Egg Tilt Sensor

Humidifier

It is used to provide a humid environment for proper nourishment of the chick inside the eggshell.

Figure 2.4.2-h Humidifier

Cooler

A cooling system is attached as a module to the incubator's body so for places with extremely high temperatures can use it for the purpose of decreasing the temperature inside the body of the incubator if it's exceeding 39.8 degrees.

Figure 2.4.2-i Cooler

Bulb

A 10Watt bulb to light up the inside of the incubator when it is needed.

Figure 2.4.2-j Bulb

2.4.3. Solar Panel

Below is mentioned the calculation of unit consumption of loads, battery calculation, and panels required to operate the incubator in case of power breakdown. For proper operation, we require a battery, inverter, and controller.

UNIT CONSUMPTION OF LOADS

One batch of eggs takes 25 days in Incubator

- **Fans**

Total kWh = (12.2 Watts)x(24 Hrs)x(25 Days) = 7320 Wh
Total Units = 7320/1000 = 7.32 units

- **Exhaust Fan**

Total kWh = (21 Watts)x(24 Hrs)x(25 Days) = 12600 Wh
Total Units = 12600/1000 = 12.6 units

- **Ceramic Heaters**

Total kWh = (50 Watts)x(24 Hrs)x(25 Days) = 30000 Wh
Total Units = 30000/1000 = 30 units

- **Cooler**

Total kWh = (7 Watts)x(24 Hrs)x(25 Days) = 4200 Wh
Total Units = 4200/1000 = 4.2 units

- **Humidifier**

Total kWh = (30 Watts)x(24 Hrs)x(25 Days) = 18000 Wh
Total Units = 18000/1000 = 18 units

- **Tray (Motor rotates after 6 hours for 5 mins, total time=20 mins/day)**

Total kWh = (4 Watts)x(0.33 Hrs)x(25 Days) = 33 Wh
Total Units = 33/1000 = 0.033 units

Total unit consumption = 7.32+12.6+30+4.2+18+0.033 = 72 units.
Daily energy consumption = 72/25 = 2.88 units per day
All loads not operating 24 hours, Approximately 1.5 Units
Solar panel system = Daily energy consumption/Peak sunlight hours
= (1.5 kWh) / (5 Hrs) = 0.3 kWh -->1 unit=1kWh

BATTERY CALCULATION

For 1 hour backup = (2 kWh) x (1/24 Day) = 0.083 kWh
Battery capacity = (330 W)x(5 hrs)x(1.3)/24V = 90 Ah
We have used a 12V battery for the basic functioning of the incubator.

Figure 2.4.3-a 12V Battery

PANELS REQUIRED

One panel is 330W, so we require one such panel to have an output of 0.3 KWh. A boost inverter is required for the operation of all loads. It would step up the voltage from 12V. We have used a 120W panel for the basic functioning of the incubator.

Figure 2.4.3-b 120W Solar Panel

- **Controller**

A solar panel controller regulates the flow of electric current between solar panels, battery, and load. It prevents overcharging, over-discharging, and maximizes charging efficiency. We have used a controller without an inverter connection to show the basic functioning of the incubator.

Figure 2.4.3-c Solar Panel Controller

Chapter 3. Results & Recommendations

We tested our project in several phases to improve the project and to ensure we were meeting our milestones with reliable results. Each individual module was tested separately prior to integration. Based on the results of testing we made improvements and changes to our design and approach as well. We have also discussed the budget of the project and the milestones achieved in this chapter.

3.1. Project Results

- **Load Status Based on Environmental Conditions**
  The Flutter application integrates with the AWS IoT Core service to receive real-time data from the ESP8266 device, which measures environmental conditions such as temperature and humidity within the incubator. The application implements logic to determine the load status based on the measured conditions.

Figure 3.1-a Load Statuses

- **Defined Activation and Deactivation Criteria**
  Clear criteria are established to activate or deactivate each load based on the desired temperature and humidity levels. For instance, if the measured temperature exceeds a predefined threshold, the ceramic heater load may be activated. Conversely, if the temperature falls below a specified threshold, the heater load is deactivated. Similar criteria are defined for other loads, such as fans, humidifiers, coolers, and buzzers, based on their respective roles in maintaining the desired incubator environment.

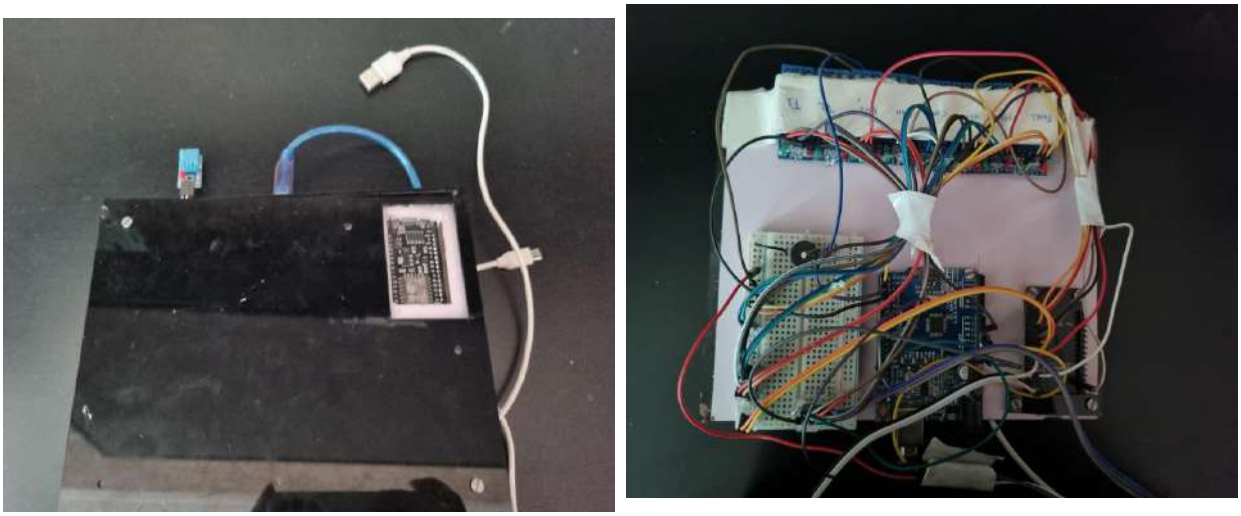

Figure 3.1-b Load Control

- **Data Transmission from ESP8266 to AWS IoT Core**
  The ESP8266 device, equipped with sensors to measure temperature and humidity, establishes a connection with AWS IoT Core using MQTT. It publishes the measured data to a specific MQTT topic, for example, "incubator/environmental_data." The ESP8266 periodically sends the data to AWS IoT Core, ensuring that the application receives updated measurements for analysis and load status determination.

- **AWS IoT Core Integration**
  The Flutter application subscribes to the incubator/environmental data MQTT topic on AWS IoT Core. It receives the published environmental data, including temperature and humidity values, in real-time. The application then applies the defined criteria to determine the status of each load.

- **Displaying Load Status on the App**
  The Flutter application provides a user-friendly interface that displays the status of each load based on the received environmental data. It visualizes the status using intuitive icons or indicators, such as "on/off" or color-coded representations. Users can easily monitor the status of fans, humidifiers, ceramic heaters, coolers, and buzzers on the app's interface, allowing them to stay informed about the current state of each load and the overall environmental conditions in the incubator.

- **Real-time Load Control**
  Additionally, the Flutter application can provide users with the ability to manually control the loads based on their preferences or specific requirements. Users can interact

with the app's interface to manually activate or deactivate loads, overriding the automated load control based on the measured environmental conditions. This feature empowers users to fine-tune the incubator's settings according to their needs.

By integrating with AWS IoT Core and receiving environmental data from the ESP8266 device, the Flutter application enables real-time monitoring and control of the loads within the incubator. Users can easily visualize the load status based on the measured temperature and humidity, ensuring the system maintains the desired conditions for successful incubation.

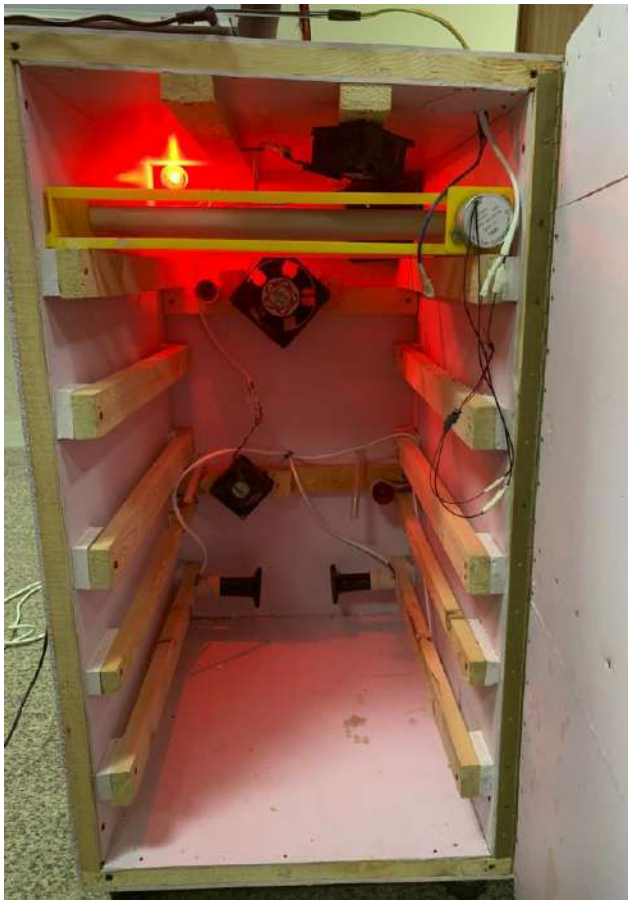

Figure 3.1-c Balanced Incubator Environment

3.2. Budget

S.No.	Items	Price/Unit	Unit	Price
1.	ESP8266 (Node MCU)	700	1	700
2.	16 Relay Module	1500	1	1500
3.	Buzzer	100	1	100
4.	Wooden Panels	6.67 per inch	150 inches	1000
5.	Thermocol Sheets	0.67 per inch	420 inches	280
6.	PVC Sheets	4.63 per inch	540 inches	2500
7.	DC Fans	200	4	800
8.	Ceramic Heaters	700	2	1400
9.	Cooler System	1000	1	1000
10.	Humidifier	3000	1	3000
11.	Rotating Trays	1700	4	6800
12.	Temperature & Humidity Sensor	350	1	350
13.	Egg Tilt Sensor	250	4	1000
14.	Bulb	300	1	300
15.	Solar Panel	25000	1	25000
16.	Battery	2000	1	2000
17.	Controller	2000	1	2000
18.	Overhead	10%		4973
	Total cost of Project			54703 ≈ 55000

3.3. Gantt Chart

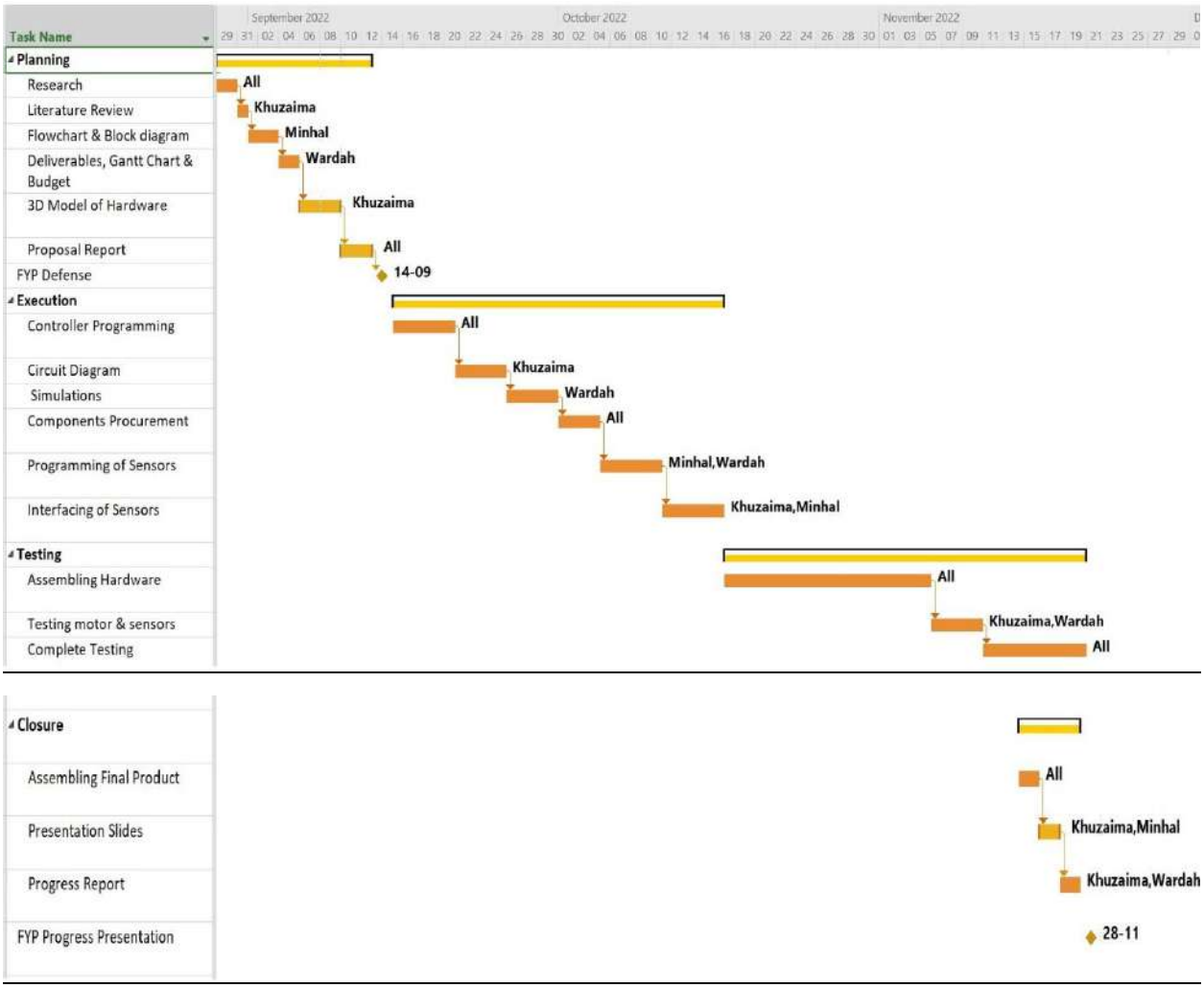

Figure 3.3-a FYP I Gantt chart

NOTE: The work distribution of FYP-I is divided into four major milestones:

- **Planning:** This part includes all of the research and preparation we had to conduct for our initial defence.
- **Execution:** This part includes the designing of the controller as well as the interfacing of sensors.
- **Testing:** This part includes the setup of hardware as well as testing the controller in real conditions.
- **Closure:** This part includes assembling the incubator's hardware as well as preparation for the final defense.

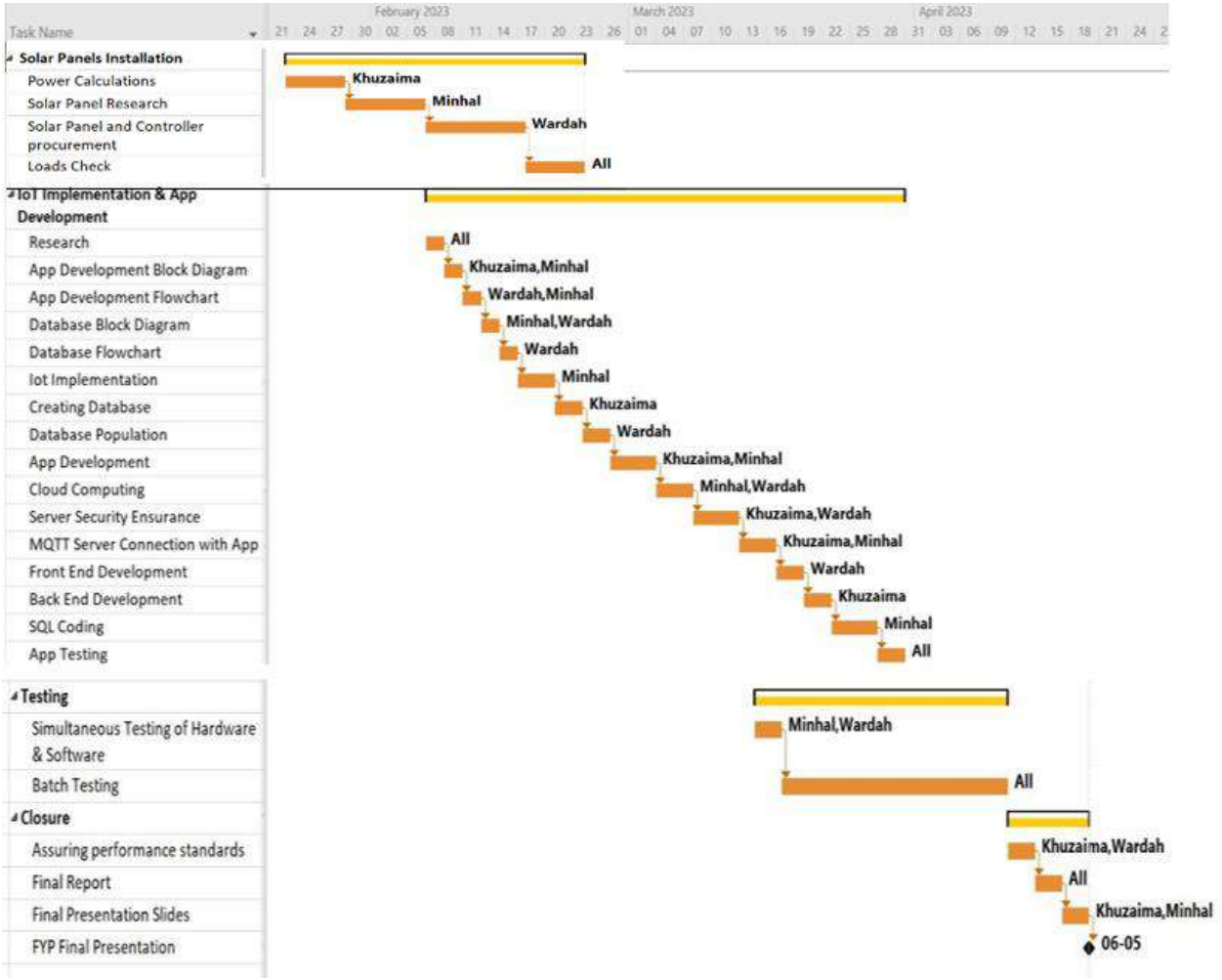

Figure 3.3-b FYP II Gantt Chart

NOTE: The work distribution of FYP-II is also divided into four major milestones:

- **Installation of Solar Panels:** This part includes all the research work, calculation and installation of solar panels.
- **IoT implementation & app development:** This part includes appropriate research work and understanding of IoT as well as app development. Then we will use the same knowledge to achieve this milestone.
- **Testing:** This part includes testing hardware and software simultaneously.
- **Closure:** This part includes assuring the performance standards of the incubator as well as the preparation for the final FYP-II presentation.

3.4. Milestones Achieved

Our Egg incubator creates ideal conditions to enhance production by having:

- Remote Monitoring
- Efficient balancing of the environment
- Mobile Application
- Operational for different bird species
- Calculation of survival & mortality rates

3.5. SDG Mapping

- **Goal 2: Zero Hunger**

Day by day, the shortage of food is increasing. Through efficient egg-hatching techniques used by our incubator, we can increase poultry production. This can empower farmers as well as provide access to nutritious protein sources.

Figure 3.5-a Goal 2

- **Goal 3: Good Health & Well Being**

In this world, not everyone has access to grade-A and hygienic eggs. Our cost-efficient incubator can be brought by all classes of people. In this way, they can have access to a healthy, safe and nutritious lifestyle.

Figure 3.5-b Goal 3

- **Goal 8: Decent Work & Economic Growth**

The poultry industry is a very large industry and targets a bigger market. This project can create opportunities for farmers, entrepreneurs and anyone related to the poultry sector. It can create job opportunities for many educated young generations.

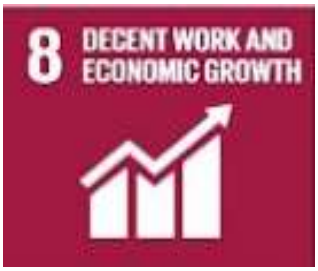

Figure 3.5-c Goal 8

3.6. Conclusions

The egg incubator we have developed incorporates several features that create optimal conditions for enhanced production. Firstly, the incubator is equipped with remote monitoring capabilities, allowing users to keep track of crucial parameters and make necessary adjustments from a distance. This feature ensures convenience and minimizes the need for constant physical presence.

Efficient balancing of the environment is another key aspect of our incubator. It is designed to maintain precise and stable temperature, humidity, and ventilation levels, providing an ideal habitat for successful egg incubation. By regulating these factors, we ensure that the eggs receive the necessary conditions for proper development and hatching.

To further enhance user experience and accessibility, we have integrated a mobile application with our incubator system. This application enables users to remotely control and monitor the incubator using their smart phones or tablets, providing real-time updates on crucial parameters and allowing for prompt adjustments if needed.

Our incubator is also versatile in its functionality, as it is designed to be operational for different bird species. This adaptability allows users to incubate eggs from various bird species without the need for separate incubation units, streamlining the process and maximizing efficiency.

Furthermore, our incubator incorporates advanced algorithms for the calculation of survival and mortality rates. By analyzing data such as temperature fluctuations, humidity levels, and other relevant factors, the system can provide valuable insights into the health and development of the eggs, enabling users to make informed decisions and optimize incubation success rates.

Lastly, our incubator features dual tray operation, which allows for the simultaneous incubation of two separate batches of eggs. This capability increases productivity by enabling users to maximize the utilization of the incubator's capacity and effectively manage different incubation cycles.

Overall, our egg incubator combines remote monitoring, efficient environmental balancing, and mobile application integration, compatibility with various bird species, survival and mortality rate calculations, and dual tray operation to create an advanced and versatile solution for optimizing egg incubation processes.

3.7. Recommendations / Future Work

- **Dual Tray Operation**
  Our incubator features could further in cooperate dual tray operation, which allows the simultaneous incubation of two separate batches of eggs. This capability increases productivity by enabling users to maximize the utilization of the incubator's capacity and effectively manage different incubation cycles.

- **Egg Candling**
  Egg candling is a process in which we use a bright light source to make the egg's internal structure visible. Egg candling is a valuable technique from which the quality of the egg, fertility and development progress can be evaluated without breaking the egg. This technique can be implemented with the help of image processing techniques.

- **Brooder**
  A brooder is a close and controlled environment which provides warmth to newly hatched chicks. Chicks require a proper heat source so they don’t die right after they are hatched. An incubator can also be used as a brooder by lowering the temperature day by day until they are adapted to room temperature.

- **Automatic Egg Detection**
  Automatic egg detection utilizes computer vision and machine learning to identify and locate eggs automatically, eliminating the need for human contact. This efficient system improves productivity and accuracy including egg production, sorting, and hatchery operations.

- **Egg Rotation Synchronization**
  Egg rotation synchronization involves periodically turning eggs during incubation to promote even development and improve hatching success. Automated methods are commonly employed to ensure consistent rotation, resulting in higher hatch rates and healthier chicks.

Integrated Egg Disinfection System

The Integrated Egg Disinfection System is an advanced automated setup designed for the efficient sanitization of eggs. It incorporates specialized equipment to effectively reduce microbial and harmful contamination. This system ensures enhancing the safety and quality of eggs.

Appendix-A: Project Codes

A-1 ESP8266 & AWS Code

```
////////////////////////////////////////////////////////////////////////////// Libraries
//////////////////////////////////////////////////////////////////////
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <time.h>
#include "secrets.h"
#include "DHT.h"

////////////////////////////////////////////////////////////////////////////  Decleratio
n //////////////////////////////////////////////////////////////////////

#define DHTPIN 12         // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11    // DHT 11
//const char *TIME_ZONE = "UTC+5";

DHT dht(DHTPIN, DHTTYPE);

float h ;
float t;
unsigned long lastMillis = 0;
unsigned long previousMillis = 0;
const long interval = 5000;

////////////////////////////////////////////////////////////////////////   Defining
RELAYS PINS ////////////////////////////////////////////////////
const int FAN_RELAY = 16;// D0
const int HEATER_RELAY = 5;// D1
const int HUMIDIFIER_RELAY = 4;// D2
const int COOLER_RELAY = 0 ;// D3
const int EXHAUST_RELAY = 2; //D4
const int BUZZER = 14;//D5
const int TRAY1_RELAY = 13;//D7

////////////////////////////////////////////////////////////////////    Toggle States
for RELAYS ////////////////////////////////////////////////////
int toggleState_1 = 1; //Define integer to remember the toggle state for relay
1
int toggleState_2 = 1; //Define integer to remember the toggle state for relay
2
int toggleState_3 = 1; //Define integer to remember the toggle state for relay
3
```

```
int toggleState_4 = 1; //Define integer to remember the toggle state for relay
4
int toggleState_5 = 1; //Define integer to remember the toggle state for relay
5
int toggleState_6 = 1; //Define integer to remember the toggle state for relay
6

////////////////////////////////////////////////////////////     Defining
Subscribing variables    ////////////////////////////////////////////////////
#define sub1 "Specie_Selection_switch"

////////////////////////////////////////////////////////////    Defining
Publishing varaiables    ////////////////////////////////////////////////////

#define pub1 "FAN_switch_status"
#define pub2 "HEATER_switch_status"
#define pub3 "EXHAUST_FAN_switch_status"
#define pub4 "BUZZER_switch_status"
#define pub5 "HUMIDIFIER_switch_status"
#define pub6 "COOLER_switch_status"

////////////////////////////////////////////////////////////    AWS TOPICS
////////////////////////////////////////////////////////////

#define AWS_IOT_PUBLISH_TOPIC   "esp8266/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"

WiFiClientSecure net;

BearSSL::X509List cert(AWS_CERT_CA);
BearSSL::X509List client_crt(AWS_CERT_CRT);
BearSSL::PrivateKey key(AWS_CERT_PRIVATE);

PubSubClient client(net);

time_t now;
time_t nowish = 1510592825;


void NTPConnect(void)
{
  Serial.print("Setting time using SNTP");
  configTime(5 * 3600, 0 * 3600, "pool.ntp.org", "time.nist.gov");
  now = time(nullptr);
  while (now < nowish)
```

Appendix-A: Project Codes

```
  {
    delay(500);
    Serial.print(".");
    now = time(nullptr);
  }
  Serial.println("done!");
  struct tm timeinfo;
  gmtime_r(&now, &timeinfo);
  Serial.print("Current time: ");
  Serial.print(asctime(&timeinfo));
}


////////////////////////////////////////////////////////////////////  Message
recieved from AWS
////////////////////////////////////////////////////////////////////
void messageReceived(char *topic, byte *payload, unsigned int length)
{
  Serial.print("Received [");
  Serial.print(topic);
  Serial.print("]: ");
 if (strstr(topic, sub1))
  {
    for (int i = 0; i < length; i++) {
      Serial.print((char)payload[i]);
    }
    Serial.println();
    // Switch on the LED if an 1 was received as first character

    if ((char)payload[0] == '0') {
      Serial.println("Chicken Egg is Placed!");
      Chicken_func();
    } else if ((char)payload[0] == '1')
      Serial.println("Duck Egg is Placed!");
      Duck_func();
    }else{
      Serial.println("Recieving Invalid Values from App");
      Serial.println("unsubscribed topic");
    }


  }

////////////////////////////////////////////////////////////////////    AWS
Connection Function
////////////////////////////////////////////////////////////////////
```

Appendix-A: Project Codes

```
void connectAWS()
{
  delay(3000);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.println(String("Attempting to connect to SSID: ") +
String(WIFI_SSID));

  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(1000);
  }

  NTPConnect();
  Serial.println("WiFi Connected");
  net.setTrustAnchors(&cert);
  net.setClientRSACert(&client_crt, &key);

  client.setServer(MQTT_HOST, 8883);
  client.setCallback(messageReceived);


  Serial.println("Connecting to AWS IOT");

  while (!client.connect(THINGNAME))
  {
    Serial.print(".");
    delay(1000);
  }

  if (!client.connected()) {
    Serial.println("AWS IoT Timeout!");
    return;
  }
  // Subscribe to a topic
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);

  Serial.println("AWS IoT Connected!");
  Serial.println("Loads Live Statuses: ");
  client.subscribe(sub1);//subscribing for Specie selection

}
```

Appendix-A: Project Codes

```
////////////////////////////////////////////////////////////////   Publishing
Message to AWS ////////////////////////////////////////////////////////////////
void publishMessage()
{
  StaticJsonDocument<200> doc;
  doc["time"] = millis();
  doc["humidity"] = h;
  doc["fan"] = toggleState_1;
  doc["heater"] = toggleState_2;
  doc["exhaust"] = toggleState_3;
  doc["bhc"] = toggleState_4*100+toggleState_5*10+toggleState_6;
  char jsonBuffer[512];
  serializeJson(doc, jsonBuffer); // print to client


}


////////////////////////////////////////////////////////////////////////// Humidity
and Tempreture Condition function for
Chicken  ////////////////////////////////
void Humidity_Conditions(float humid)
{
     float upperbound = 70.00;
     float lowerbound = 55.00;

    if (isnan(humid))
    {
      Serial.println("Failed to read from DHT sensor!");

      return;
    }
    else if(humid>= upperbound)
    {
      Serial.print("In extreme condition of humidity\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, LOW); // turn heaters  on
      Serial.print("HEATERS are ON\n");
      toggleState_2 = 1;
      client.publish(pub2, "1");
      digitalWrite(EXHAUST_RELAY , LOW); // turn exhaust fan  on
```

Appendix-A: Project Codes

```
    Serial.print("EXHAUST FAN  is ON\n");
    toggleState_3 = 1;
    client.publish(pub3, "1");
    digitalWrite(BUZZER, LOW); // Heater 2 stays on
    Serial.print("BUZZER is ON\n");
    toggleState_4 = 1;
    client.publish(pub4, "1");
    digitalWrite(HUMIDIFIER_RELAY, HIGH);
    Serial.print("HUMIDIFIER is OFF\n");
    toggleState_5 = 0;
    client.publish(pub5, "0");
  }
  else if(humid>=lowerbound && humid <upperbound)
  {
    Serial.print("In normal condition of humidity\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
    toggleState_3 = 0;
    client.publish(pub3, "0");
    digitalWrite(BUZZER, HIGH); // Heater 2 stays on
    Serial.print("BUZZER is OFF\n");
    toggleState_4 = 0;
    client.publish(pub4, "0");
    digitalWrite(HUMIDIFIER_RELAY, HIGH);
    Serial.print("HUMIDIFIER is OFF\n");
    toggleState_5 = 0;
    client.publish(pub5, "0");
  }
  else if(humid<lowerbound)
  {
    Serial.print("In extreme lowest condition of humidity\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
    toggleState_3 = 0;
    client.publish(pub3, "0");
    digitalWrite(BUZZER, LOW); // Heater 2 stays on
    Serial.print("BUZZER is ON\n");
```

Appendix-A: Project Codes

```
      toggleState_4 = 1;
      client.publish(pub4, "1");
      digitalWrite(HUMIDIFIER_RELAY, LOW);
      Serial.print("HUMIDIFIER is ON\n");
      toggleState_5 = 1;
      client.publish(pub5, "1");
    }
    else
    {
      Serial.print("In else condition of humidity\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
      Serial.print("HEATERS are OFF\n");
      toggleState_2 =0 ;
      client.publish(pub2, "0");
      digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
      Serial.print("EXHAUST FAN  is OFF\n");
      toggleState_3 = 0;
      client.publish(pub3, "0");
      digitalWrite(BUZZER, HIGH); // BUZZER on
      Serial.print("BUZZER is OFF\n");
      toggleState_4 = 0;
      client.publish(pub4, "0");
      digitalWrite(HUMIDIFIER_RELAY, HIGH);
      Serial.print("HUMIDIFIER is OFF\n");
      toggleState_5 = 0;
      client.publish(pub5, "0");
      digitalWrite(COOLER_RELAY, HIGH);
      Serial.print("COOLER is OFF\n");
      toggleState_6 = 0;
      client.publish(pub6, "0");

    }
    delay(100);
    }
// Temp condition for chicken
void Temperature_Conditions(float temp)
{
   float upperbound = 38.80;
   float lowerbound = 36.80;
    // Check if any reads failed and exit early (to try again).
    if (isnan(temp))
```

Appendix-A: Project Codes

```
    {
      Serial.println("Failed to read from DHT sensor!");

      return;
    }
else if ( temp >= upperbound )
    {
      Serial.print("In extreme condition of Temperature\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
      Serial.print("HEATERS are OFF\n");
      toggleState_2 = 0;
      client.publish(pub2, "0");
      digitalWrite(EXHAUST_RELAY , LOW); // turn exhaust fan  on
      Serial.print("EXHAUST FAN  is ON\n");
      toggleState_3 = 1;
      client.publish(pub3, "1");
      digitalWrite(BUZZER, LOW); // Heater 2 stays on
      Serial.print("BUZZER is ON\n");
      toggleState_4 = 1;
      client.publish(pub4, "1");
      digitalWrite(COOLER_RELAY, LOW);
      Serial.print("COOLER is ON\n");
      toggleState_6 = 1;
      client.publish(pub6, "1");
      // Trays function is independent of tempreture and humidity

    }
    else if  (temp >lowerbound && temp < upperbound)
    {
      Serial.print("In normal condition of Temperature\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
      Serial.print("HEATERS are OFF\n");
      toggleState_2 = 0;
      client.publish(pub2, "0");
      digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaist fan  on
      Serial.print("EXHAUST FAN  is OFF\n");
      toggleState_3 = 0;
```

```
    client.publish(pub3, "0");
    digitalWrite(BUZZER, HIGH); // Heater 2 stays on
    Serial.print("BUZZER is OFF\n");
    toggleState_4 = 0;
    client.publish(pub4, "0");
    digitalWrite(COOLER_RELAY, HIGH);
    Serial.print("COOLER is OFF\n");
    toggleState_6 = 0;
    client.publish(pub6, "0");
    }
  else if(temp<lowerBound)
  {
    Serial.print("In extreme lowest condition of Temperature\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(HEATER_RELAY, LOW); // turn heaters  on
    Serial.print("HEATERS are ON\n");
    toggleState_2 = 1;
    client.publish(pub2, "1");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
    toggleState_3 = 0;
    client.publish(pub3, "0");
    digitalWrite(BUZZER, LOW); // BUZZER on
    Serial.print("BUZZER is ON\n");
    toggleState_4 = 1;
    client.publish(pub4, "1");
    digitalWrite(COOLER_RELAY, HIGH);
    Serial.print("COOLER is OFF\n");
    toggleState_6 = 0;
    client.publish(pub6, "0");
  }
  else
  {
    Serial.print("In else condition of Temperature\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
    Serial.print("HEATERS are OFF\n");
    toggleState_2 =0 ;
    client.publish(pub2, "0");
```

Appendix-A: Project Codes

```
      digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
      Serial.print("EXHAUST FAN  is OFF\n");
      toggleState_3 = 0;
      client.publish(pub3, "0");
      digitalWrite(BUZZER, HIGH); // BUZZER on
      Serial.print("BUZZER is OFF\n");
      toggleState_4 = 0;
      client.publish(pub4, "0");
      digitalWrite(COOLER_RELAY, HIGH);
      Serial.print("COOLER is OFF\n");
      toggleState_6 = 0;
      client.publish(pub6, "0");

    }
    delay(100);
}

////////////////////////////////////////////////////////////////  Humidity
and Tempreture Condition for Duck  ////////////////////////////////////

void Humidity_Conditions_Duck(float humid)
{
     float upperbound = 75.00;
     float lowerbound = 55.00;

    if (isnan(humid))
    {
      Serial.println("Failed to read from DHT sensor!");

      return;
    }
    else if(humid>= upperbound)
    {
      Serial.print("In extreme condition of humidity\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, LOW); // turn heaters  on
      Serial.print("HEATERS are ON\n");
      toggleState_2 = 1;
      client.publish(pub2, "1");
      digitalWrite(EXHAUST_RELAY , LOW); // turn exhaust fan  on
      Serial.print("EXHAUST FAN  is ON\n");
      toggleState_3 = 1;
```

```
    client.publish(pub3, "1");
    digitalWrite(BUZZER, LOW); // Heater 2 stays on
    Serial.print("BUZZER is ON\n");
    toggleState_4 = 1;
    client.publish(pub4, "1");
    digitalWrite(HUMIDIFIER_RELAY, HIGH);
    Serial.print("HUMIDIFIER is OFF\n");
    toggleState_5 = 0;
    client.publish(pub5, "0");
  }
  else if(humid>=lowerbound && humid <upperbound)
  {
    Serial.print("In normal condition of humidity\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
    toggleState_3 = 0;
    client.publish(pub3, "0");
    digitalWrite(BUZZER, HIGH); // Heater 2 stays on
    Serial.print("BUZZER is OFF\n");
    toggleState_4 = 0;
    client.publish(pub4, "0");
    digitalWrite(HUMIDIFIER_RELAY, HIGH);
    Serial.print("HUMIDIFIER is OFF\n");
    toggleState_5 = 0;
    client.publish(pub5, "0");
  }
  else if(humid<lowerbound)
  {
    Serial.print("In extreme lowest condition of humidity\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
    toggleState_3 = 0;
    client.publish(pub3, "0");
    digitalWrite(BUZZER, LOW); // Heater 2 stays on
    Serial.print("BUZZER is ON\n");
    toggleState_4 = 1;
    client.publish(pub4, "1");
```

```
      digitalWrite(HUMIDIFIER_RELAY, LOW);
      Serial.print("HUMIDIFIER is ON\n");
      toggleState_5 = 1;
      client.publish(pub5, "1");
    }
    else
    {
      Serial.print("In else condition of humidity\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
      Serial.print("HEATERS are OFF\n");
      toggleState_2 =0 ;
      client.publish(pub2, "0");
      digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
      Serial.print("EXHAUST FAN  is OFF\n");
      toggleState_3 = 0;
      client.publish(pub3, "0");
      digitalWrite(BUZZER, HIGH); // BUZZER on
      Serial.print("BUZZER is OFF\n");
      toggleState_4 = 0;
      client.publish(pub4, "0");
      digitalWrite(HUMIDIFIER_RELAY, HIGH);
      Serial.print("HUMIDIFIER is OFF\n");
      toggleState_5 = 0;
      client.publish(pub5, "0");
      digitalWrite(COOLER_RELAY, HIGH);
      Serial.print("COOLER is OFF\n");
      toggleState_6 = 0;
      client.publish(pub6, "0");

    }
    delay(100);
    }
// Temp condition for chicken
void Temperature_Conditions_Duck(float temp)
{
   float upperbound = 39.80;
   float lowerbound = 35.80;
    // Check if any reads failed and exit early (to try again).
    if (isnan(temp))
    {
      Serial.println("Failed to read from DHT sensor!");
```

```
      return;
    }
else if ( temp >= upperbound )
    {
      Serial.print("In extreme condition of Temperature\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
      Serial.print("HEATERS are OFF\n");
      toggleState_2 = 0;
      client.publish(pub2, "0");
      digitalWrite(EXHAUST_RELAY , LOW); // turn exhaust fan  on
      Serial.print("EXHAUST FAN  is ON\n");
      toggleState_3 = 1;
      client.publish(pub3, "1");
      digitalWrite(BUZZER, LOW); // Heater 2 stays on
      Serial.print("BUZZER is ON\n");
      toggleState_4 = 1;
      client.publish(pub4, "1");
      digitalWrite(COOLER_RELAY, LOW);
      Serial.print("COOLER is ON\n");
      toggleState_6 = 1;
      client.publish(pub6, "1");
      // Trays function is independent of tempreture and humidity

    }
    else if  (temp >lowerbound && temp < upperbound)
    {
      Serial.print("In normal condition of Temperature\n");
      digitalWrite(FAN_RELAY, LOW); // turn FANs on
      Serial.print("FANS are ON\n");
      toggleState_1 = 1;
      client.publish(pub1, "1");
      digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
      Serial.print("HEATERS are OFF\n");
      toggleState_2 = 0;
      client.publish(pub2, "0");
      digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaist fan  on
      Serial.print("EXHAUST FAN  is OFF\n");
      toggleState_3 = 0;
      client.publish(pub3, "0");
      digitalWrite(BUZZER, HIGH); // Heater 2 stays on
```

```
    Serial.print("BUZZER is OFF\n");
    toggleState_4 = 0;
    client.publish(pub4, "0");
    digitalWrite(COOLER_RELAY, HIGH);
    Serial.print("COOLER is OFF\n");
    toggleState_6 = 0;
    client.publish(pub6, "0");
    }
  else if(temp<lowerbound)
  {
    Serial.print("In extreme lowest condition of Temperature\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(HEATER_RELAY, LOW); // turn heaters  on
    Serial.print("HEATERS are ON\n");
    toggleState_2 = 1;
    client.publish(pub2, "1");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
    toggleState_3 = 0;
    client.publish(pub3, "0");
    digitalWrite(BUZZER, LOW); // BUZZER on
    Serial.print("BUZZER is ON\n");
    toggleState_4 = 1;
    client.publish(pub4, "1");
    digitalWrite(COOLER_RELAY, HIGH);
    Serial.print("COOLER is OFF\n");
    toggleState_6 = 0;
    client.publish(pub6, "0");
  }
  else
  {
    Serial.print("In else condition of Temperature\n");
    digitalWrite(FAN_RELAY, LOW); // turn FANs on
    Serial.print("FANS are ON\n");
    toggleState_1 = 1;
    client.publish(pub1, "1");
    digitalWrite(HEATER_RELAY, HIGH); // turn heaters  on
    Serial.print("HEATERS are OFF\n");
    toggleState_2 =0 ;
    client.publish(pub2, "0");
    digitalWrite(EXHAUST_RELAY , HIGH); // turn exhaust fan  on
    Serial.print("EXHAUST FAN  is OFF\n");
```

Appendix-A: Project Codes

```
      toggleState_3 = 0;
      client.publish(pub3, "0");
      digitalWrite(BUZZER, HIGH); // BUZZER on
      Serial.print("BUZZER is OFF\n");
      toggleState_4 = 0;
      client.publish(pub4, "0");
      digitalWrite(COOLER_RELAY, HIGH);
      Serial.print("COOLER is OFF\n");
      toggleState_6 = 0;
      client.publish(pub6, "0");

    }
    delay(100);
}


////////////////////////////////////////////////////////////////////  Calling
Functions  ////////////////////////////////////////////////////////////////////
void Chicken_func(){
// Tempareture function called
  Serial.print("Temperature Conditions Check: \n");
  Temperature_Conditions(t);
  //Humdity Function called
  Humidity_Conditions(h);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.println(F("°C "));
  delay(2000);
}

void Duck_func(){
  // Tempareture function called
  Serial.print("Temperature Conditions Check: \n");
  Temperature_Conditions_Duck(t);
  //Humdity Function called
  Humidity_Conditions_Duck(h);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.println(F("°C "));
  delay(2000);
}
```

Appendix-A: Project Codes

```
////////////////////////////////////////////////////////////////////  SetUp   ///
//////////////////////////////////////////////////////////////////////////////////
void setup()
{
  Serial.begin(115200);
  pinMode(DHTPIN,INPUT);
  pinMode(FAN_RELAY, OUTPUT);
  pinMode(HEATER_RELAY, OUTPUT);
  pinMode(HUMIDIFIER_RELAY, OUTPUT);
  pinMode(EXHAUST_RELAY, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  connectAWS();
  dht.begin();
  Serial.println("Device Started");
  Serial.println("-------------------------------------");
  Serial.println("Running DHT!");
  Serial.println("-------------------------------------");
}

////////////////////////////////////////////////////////////////////    Loop  ////
//////////////////////////////////////////////////////////////////////////////////
/
void loop()
{
  h = dht.readHumidity();
  t = dht.readTemperature();

  if (isnan(h) || isnan(t) )  // Check if any reads failed and exit early (to
try again).
  {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  now = time(nullptr);

  if (!client.connected())
  {
    connectAWS(); // This function contains the subscription value on which
tempreture n humidity values will be adjusted according to the specie
  }
  else
  {
    client.loop();

    if (millis() - lastMillis > 5000)
    {
      lastMillis = millis();
      publishMessage();
    }
  }
}
```

A-2 Timer & Tilt Sensor Code

```
#define TILT 8 // pin 8 for tilt sensor 1
#define TILT 9 // pin 9 for tilt sensor 2
#define LED 13 // pin 13 for LED
int signal = 2; // pin 2 for motor 1 //Tray 1
int signal1 = 3; // pin 3 for motor 2 //Tray 2
unsigned long event1=90000;    // 1.5MIN-->2days
unsigned long event2=600000;   // 10MIN-->18days

const int LEDpin = 3;
const long onDuration = 30000;// ON time for LED (0.5min-->6.5min)
const long offDuration = 60000;// OFF time for LED (1MIN-->6hrs)
int LEDState = LOW;// initial state of LED
int RelayState = HIGH;

long rememberTime=5000;// this is used by the code

void setup()
{
  Serial.begin(115200);
  pinMode(TILT, INPUT); //define Data input pin input pin
  pinMode(LED,OUTPUT); //define LED pint  pin as output
  pinMode(LEDpin,OUTPUT);// define LEDpin as output
  pinMode(signal,OUTPUT);
  pinMode(signal1,OUTPUT);
  digitalWrite(LEDpin,LEDState);// set initial state
  digitalWrite(signal,RelayState);
  digitalWrite(signal1,RelayState);
  delay(500);
}
// Tilt sensor function
void Tilt_sensor()
{
 int TILT_SENSED = digitalRead(TILT); //read TILT sensor
  if(TILT_SENSED == HIGH) //if tilt isn't sensed
  {
    Serial.println("      0");
    digitalWrite(LED,LOW); //set the LED pin LOW
  }else if (TILT_SENSED == LOW) //if tilt is sensed
  {
    Serial.println("      1");
    digitalWrite(LED,HIGH); //Set the LED pin HIGH
  }else
  {
    Serial.println("End");
  }
}
// Timer function
  void timer()
{
  //Serial.println(millis());

  if(millis()>=event1 && millis()<=event2)
```

Appendix-A: Project Codes

```
  {
    //Serial.println("DAY 2 START");
    if( LEDState == LOW )
  {
    //Serial.println("lED OFF");
    if( (millis()- rememberTime) >= offDuration)
    {
    //Serial.println("Led ON");
    LEDState = HIGH; //change the state of LED
    RelayState = LOW; //Turn the motor ON
    //Tilt_sensor();
    rememberTime=millis();// remember Current millis() time
    }
  }
  else
  {
    //Serial.println("LED ON");
    if( (millis()- rememberTime) >= onDuration)
    {
    LEDState = LOW;// change the state of LED
    RelayState = HIGH;
    rememberTime=millis();// remember Current millis() time
    //Serial.println("LED OFF");
    }
  }
  digitalWrite(LEDpin,LEDState);// turn the LED ON or OFF
  digitalWrite(signal,RelayState);
  digitalWrite(signal1,RelayState);
  }
}
void loop()
{
  Tilt_sensor();
  timer();
}
```

A-3 Flutter Code

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:http/http.dart' as http;
import 'package:iotproject/main.dart';

class LoginSignupPage extends StatefulWidget {
  @override
  _LoginSignupPageState createState() => _LoginSignupPageState();
}

class _LoginSignupPageState extends State<LoginSignupPage> {

  bool _isLogin = true; // Toggle between login and signup
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>(); // For form
validation

  String _email = '';
  String _password = '';
  String _errorMessage = '';

  void _submit() async {
  if (!_formKey.currentState!.validate()) return;

  _formKey.currentState?.save();
  setState(() {
    _errorMessage = '';
  });

  // API URL
  String apiUrl = 'http://3.85.112.57';
  String endpoint = _isLogin ? '/login' : '/signup';

  final response = await http.post(
    Uri.parse(apiUrl + endpoint),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
      'username': _email,
      'password': _password,
    }),
```

```
    );

    if (response.statusCode == 200) {
      // Successfully logged in or signed up
      if (_isLogin) {
        print('Logged in successfully');
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => MQTTClient()),
        );
      } else {
        print('User created successfully');
      }
    } else {
      // Error handling
      final responseData = json.decode(response.body);
      setState(() {
        _errorMessage = responseData['error'] ?? 'An error occurred';
      });
    }
  }

  void _toggleForm() {
    setState(() {
      _isLogin = !_isLogin;
      _errorMessage = '';
    });
  }

  @override
  Widget build(BuildContext context) {
    var size = MediaQuery.of(context).size;
    return Scaffold(
      body: SingleChildScrollView(
        child: Container(
          height: size.height*0.95,
          padding: EdgeInsets.all(16),
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.end,
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Image.asset('./assets/logo.jpg'),
                const Padding(
```

Appendix-A: Project Codes

```
                padding: EdgeInsets.only(bottom:20.0, right: 10),
                child: Text("Chic 'O Eggs",
                 textAlign: TextAlign.start,
                  style: TextStyle(
                    fontSize: 42,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
              Container(
                padding: EdgeInsets.all(20),
                decoration: BoxDecoration(
                                  borderRadius: BorderRadius.circular(10.0),
                                  border: Border.all(
                                    color: Colors.black,
                                    width: 1.0,
                                  ),
                                ),
                child: Column(
                  children: [
                    // Email input field
                    TextFormField(
                      keyboardType: TextInputType.emailAddress,
                      decoration: InputDecoration(
                        labelText: 'Email',
                      ),
                      validator: (value) {
                        if (value!.isEmpty) return 'Email is required.';
                        if (!RegExp(r'\S+@\S+\.\S+').hasMatch(value!)) return
'Invalid email.';
                        return null;
                      },
                      onSaved: (value) {
                        _email = value!.trim();
                      },
                    ),
                    // Password input field
                    TextFormField(
                      obscureText: true,
                      decoration: InputDecoration(
                        labelText: 'Password',
                      ),
                      validator: (value) {
                        if (value!.isEmpty) return 'Password is required.';
```

Appendix-A: Project Codes

```
                    if (value.length < 6) return 'Password must be at least
6 characters.';
                    return null;
                  },
                  onSaved: ( value) {
                    _password = value!;
                  },
                ),
                // Error message
                if (_errorMessage.isNotEmpty)
                  Padding(
                    padding: EdgeInsets.only(top: 8),
                    child: Text(
                      _errorMessage,
                                  style: TextStyle(
                    color: Colors.red,
                    fontSize: 14,
                    ),
                  ),
                ),
              // Submit button
              Padding(
                padding: EdgeInsets.only(top: 16),
                child: ElevatedButton(
                  onPressed: _submit,

                  child: Text(
                    _isLogin ? 'Login' : 'Signup',
                    style: GoogleFonts.montserrat(fontSize: 18),
                  ),
                ),
              ),
              // Toggle form type
              Padding(
                padding: EdgeInsets.only(top: 0),
                child: TextButton(
                  onPressed: _toggleForm,
                  child: Text(
                    _isLogin
                        ? 'Don\'t have an account? Signup'
                        : 'Already have an account? Login',
                    style: TextStyle(
                      fontSize: 14,
                      color: Color.fromARGB(255, 247, 148, 0),
                    ),
```

```
                    ),
                  ),
                ),
                    ],
                  ),
              ),
            ],
          ),
      ),
    ),
  ),
);
}

  // Add your code here
}
```

```
// ignore_for_file: prefer_const_constructors

import 'dart:ffi';
import 'dart:io';
import 'dart:math';
import 'dart:typed_data';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:iotproject/loginsignup.dart';
import 'package:timezone/data/latest.dart' as tz;
import 'package:timezone/timezone.dart' as tz;
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:mqtt_client/mqtt_client.dart';
import 'package:mqtt_client/mqtt_server_client.dart';
import 'package:ndialog/ndialog.dart';
import 'dart:convert';
import 'package:charts_flutter/flutter.dart' as charts;
import 'dart:convert';
import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

void main() {
  WidgetsFlutterBinding.ensureInitialized();
```

```
  tz.initializeTimeZones();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Egg Hatch',
      themeMode: ThemeMode.light,
      // home: MQTTClient(),
      home: LoginSignupPage()
    );
  }
}

Future<void> scheduleNotificationOnSpecificDay(
    int id, String title, String body, DateTime scheduledDate) async {
  // Create an instance of FlutterLocalNotificationsPlugin
  FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
      FlutterLocalNotificationsPlugin();

  // Initialize the plugin with settings for Android and iOS
  var initializationSettingsAndroid =
      AndroidInitializationSettings('@mipmap/ic_launcher');
  var initializationSettingsIOS = IOSInitializationSettings();
  var initializationSettings = InitializationSettings(
      android: initializationSettingsAndroid, iOS: initializationSettingsIOS);
  await flutterLocalNotificationsPlugin.initialize(initializationSettings);

  // Define notification details for Android
  var androidPlatformChannelSpecifics = AndroidNotificationDetails(
      'channel_id', 'channel_name',
      importance: Importance.max, priority: Priority.high, showWhen: false);

  // Define notification details for iOS
  var iOSPlatformChannelSpecifics = IOSNotificationDetails();

  // Combine the notification details for both platforms
  var platformChannelSpecifics = NotificationDetails(
      android: androidPlatformChannelSpecifics,
      iOS: iOSPlatformChannelSpecifics);

  // Convert the scheduledDate to the local timezone
```

```
  tz.TZDateTime scheduledDateInLocalTimezone =
      tz.TZDateTime.from(scheduledDate, tz.local);

  // Schedule the notification
  try {
      await flutterLocalNotificationsPlugin.zonedSchedule(
      id,
      title,
      body,
      scheduledDateInLocalTimezone,
      platformChannelSpecifics,
      androidAllowWhileIdle: true,
      uiLocalNotificationDateInterpretation:
          UILocalNotificationDateInterpretation.absoluteTime);
  } catch (e) {

  }

}

class ChartData {
  final DateTime time;
  final double value;

  ChartData(this.time, this.value);
}

class MQTTClient extends StatefulWidget {
  const MQTTClient({Key? key}) : super(key: key);

  @override
  _MQTTClientState createState() => _MQTTClientState();
}

class _MQTTClientState extends State<MQTTClient> {
  String statusText = "Status Text";
  /*
    Status text to let you know the status of the client.
    Lisit Chart stores the data for the chart two Lists are made to store
    Temperatur
    and
    Humidity
    over time.
   */
  // ignore: prefer final fields
```

Appendix-A: Project Codes

```
  List<charts.Series<ChartData, DateTime>> _seriesList = [
    charts.Series<ChartData, DateTime>(
      id: 'Value',
      colorFn: (_, __) => charts.MaterialPalette.blue.shadeDefault,
      domainFn: (ChartData data, _) => data.time,
      measureFn: (ChartData data, _) => data.value,
      data: [],
    ),
  ];

  List<charts.Series<ChartData, DateTime>> _seriesList2 = [
    charts.Series<ChartData, DateTime>(
      id: 'Value',
      colorFn: (_, __) => charts.MaterialPalette.blue.shadeDefault,
      domainFn: (ChartData data, _) => data.time,
      measureFn: (ChartData data, _) => data.value,
      data: [],
    ),
  ];

  late bool _animate = true;

  Future<void> sendStartRequest(String tray, String date, String specie) async
{
  final apiUrl = 'http://3.85.112.57/start'; // Replace with your Flask API URL

  final response = await http.post(
    Uri.parse(apiUrl),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
      'tray': tray,
      'date': date,
      'specie': specie,
    }),
  );

  if (response.statusCode == 200) {
    print('Request successful');
    // Handle the response data if needed
  } else {
    print('Request failed with status: ${response.statusCode}.');
  }
}

  void addData(ChartData data) {
```

```
    /*
      Add Chart Data to the list of charts for temperature
    */
    _seriesList[0].data.add(data);
  }

  void addData2(ChartData data) {
    /*
      Add Chart Data to the list of charts for humidity
    */
    _seriesList2[0].data.add(data);
  }

  Future<bool> isNotificationScheduled(int notificationId) async {
    // Create an instance of FlutterLocalNotificationsPlugin
    FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
        FlutterLocalNotificationsPlugin();

    // Get the list of pending notifications
    List<PendingNotificationRequest> pendingNotifications =
        await flutterLocalNotificationsPlugin.pendingNotificationRequests();

    // Check if a notification with the specified ID is in the list
    bool isScheduled = pendingNotifications
        .any((notification) => notification.id == notificationId);

    return isScheduled;
  }

  void _notification() async{
    var dda = await fetchData();
    int notificationId = 0; // Replace with the ID you want to check
    for (var i = 0; i < dda.length; i++) {
      if(dda[i]['status'] == false){
        for (var j = 0; j < dda[i]['events'].length; j++) {

                    notificationId = (((i+1) * 10) + j);
                    bool scheduled = await
isNotificationScheduled(notificationId);
                    String inputDateString = dda[i]['events'][j]['date'];
                    String reformattedDateString =
inputDateString.replaceAll('-', ' ');
                    reformattedDateString =
reformattedDateString.replaceAll(':', ' ');
                    List<int> numbers = reformattedDateString
```

Appendix-A: Project Codes

```
                                        .split(' ')
                                        .map((numberString) =>
int.parse(numberString))
                                        .toList();
                    DateTime specificDay = DateTime(numbers[2], numbers[1],
numbers[0], numbers[3], numbers[4]);
                    scheduleNotificationOnSpecificDay(
                        notificationId, dda[i]['events'][j]['event'],
dda[i]['events'][j]['message'], specificDay);
          }
        }
      }
    }
    @override
    initState() {
      _notification();
      _connect();

    }

    void clearData() {
      setState(() {
        _seriesList[0].data.clear();
      });
    }
    List<String> _selectedOption = ["Chicken", "Chicken"];
    bool isConnected = false;

    TextEditingController idTextController = TextEditingController();

    final MqttServerClient client =
        MqttServerClient('a300u8hke8ocee-ats.iot.us-east-1.amazonaws.com', '');

    @override
    void dispose() {
      idTextController.dispose();
      super.dispose();
    }
    List<dynamic> dd = [{},{}];
    fetchData() async {
    final url = Uri.parse('http://3.85.112.57/getspe');
    final response = await http.get(url);

    if (response.statusCode == 200) {
      List<dynamic> data = jsonDecode(response.body);
```

Appendix-A: Project Codes

```
    dd = data;
    setState(() {
      _selectedOption[0] = dd[0]['specie'];
      _selectedOption[1] = dd[1]['specie'];
    });

    return data;
  } else {
    print('Failed to fetch data: ${response.statusCode}');
  }
}

@override
Widget build(BuildContext context) {
  var width = MediaQuery.of(context).size.width;
  final bool hasShortWidth = width < 600;

  return Scaffold(
    body: Column(
      children: [body(hasShortWidth), footer()],
    ),
  );
}

Widget header() {
  /* Header for Making Top of the App Currently not in use */
  return Expanded(
    // ignore: avoid_unnecessary_containers, sort_child_properties_last
    child: Container(
      // ignore: prefer_const_constructors
      child: Center(
        child: Text(
          "AWS IoT - Chick 'O Egg",
          style: TextStyle(
              fontSize: 22,
              color: Colors.orange,
              fontWeight: FontWeight.bold),
          textAlign: TextAlign.center,
        ),
      ),
    ),
    flex: 3,
  );
}
```

Appendix-A: Project Codes

```
  Widget body(bool hasShortWidth) {
    /* Main Body where all the menu and Stream is created  */
    return Expanded(
      // ignore: sort_child_properties_last
      child: Padding(
        padding: const EdgeInsets.only(top: 40.0),
        child: Container(
          child: hasShortWidth
              ? Column(
                  children: [bodyMenu(), Expanded(child: bodySteam())],
                )
              : Row(
                  children: [
                    Expanded(
                      child: bodyMenu(),
                      flex: 2,
                    ),
                    Expanded(
                      child: bodySteam(),
                      flex: 8,
                    )
                  ],
                ),
        ),
      ),
      flex: 20,
    );
  }

  Widget bodyMenu() {
    return Container(
      color: Color.fromARGB(66, 255, 255, 255),
      child: Column(
        children: [
          // Padding(
          //   padding: const EdgeInsets.symmetric(horizontal: 16, vertical:
8),
          //   child: TextFormField(
          //       enabled: !isConnected,
          //       controller: idTextController,
          //       decoration: InputDecoration(
          //           border: UnderlineInputBorder(),
          //           labelText: 'MQTT Client Id',
          //           labelStyle: TextStyle(fontSize: 10),
          //           suffixIcon: IconButton(
```

```
          //             icon: Icon(Icons.subdirectory_arrow_left),
          //             onPressed: _connect,
          //           ))),
          // ),
          // isConnected
          //     ? TextButton(onPressed: _disconnect, child:
Text('Disconnect'))
          //     : Container()
        ],
      ),
    );
  }

  Widget bodySteam() {
    /*
        The code uses the StreamBuilder widget to listen for updates from the
MQTT client
        and update the UI accordingly. When a new message is received, it
decodes the payload
        from the MQTT message into a JSON string, parses it, and extracts the
relevant data such as
        the time, temperature, and humidity. It then updates the data in the
chart using the addData
        and addData2 functions, which are not shown in the code snippet.
        The UI displays the time, temperature, humidity, and status of various
appliances
        using the Text widget within Row widgets. The color of the text is set
to Colors.amber
        and the font size is set to 20. If the status of an appliance is 0, it
is displayed as
        "OFF", otherwise it is displayed as "ON".
        The UI is contained within a Container widget with a white background
color.
        The entire widget is scrollable using the SingleChildScrollView widget,
which
        is useful for displaying a large amount of data on a small screen. The
height
        of the container is set to 70% of the screen height using size.height *
0.7.
        Overall, the code provides a simple and efficient way to display real-
time data
        from a MQTT client in a Flutter app.
     */
    var size = MediaQuery.of(context).size;
    return Container(
```

Appendix-A: Project Codes

```
      color: Colors.white,
      child: StreamBuilder(
        stream: client.updates,
        builder: (context, snapshot) {
          if (!snapshot.hasData)
            return Center(
              child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(Colors.white),
              ),
            );
          else {
            final mqttReceivedMessages =
                snapshot.data as List<MqttReceivedMessage<MqttMessage?>>?;

            final recMess =
                mqttReceivedMessages![0].payload as MqttPublishMessage;

            String result = utf8.decode(recMess.payload.message);
            var hs = json.decode(result);
            ChartData data = ChartData(
                DateTime.fromMillisecondsSinceEpoch(hs['time']),
                hs['tempreture'].toDouble());
            addData(data);
            ChartData data2 = ChartData(
                DateTime.fromMillisecondsSinceEpoch(hs['time']),
                hs['humidity'].toDouble());
            addData2(data2);
            String input = hs['tilt'];
            RegExp regExp = RegExp(r'\d+');
            Iterable<Match> matches = regExp.allMatches(input);
            int xxnum =0;
            if (matches.isNotEmpty) {
               xxnum = int.parse(matches.first.group(0) ?? '0');
              // prints: 1
            }
            // ignore: sized_box_for_whitespace
            return Container(
              color: Colors.grey,
              width: size.width,
              height: size.height * 0.7,
              child: SingleChildScrollView(
                child: Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Column(
                    children: [
```

Appendix-A: Project Codes

```
Container(
  padding: EdgeInsets.all(20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Column(
        children:  [
          Text(
            'Temperature',
            style: const TextStyle(
                color: Colors.orange,
                fontWeight: FontWeight.w200,
                fontSize: 20,
                ),
          ),
          Row(
           children:  [
             Icon(Icons.thermostat, size: 40,),

             Text(hs['tempreture'].toStringAsFixed(1),
             style:TextStyle(
               color: Colors.orange,
               fontWeight: FontWeight.w200,
               fontSize: 60,
               ),
             ),
           ],
          ),
        ],
      ),
      Column(
        children:  [
           Text(
            'Humidity',
            style: const TextStyle(
                color: Colors.orange,
                fontWeight: FontWeight.w200,
                fontSize: 20,
                ),
          ),
          Row(
           children:  [
             Icon(Icons.water, size: 40,),
             Text(hs['humidity'].toString(),
               style: const TextStyle(
```

Appendix-A: Project Codes

```
                        color: Colors.orange,
                        fontWeight: FontWeight.w200,
                        fontSize: 60,
                        ),
                      ),
                    ],
                  ),
                ],
              ),
            ],
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(left:20.0, right: 20),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
              Container(
                padding: EdgeInsets.all(10),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10.0),
                  border: Border.all(
                    color: Colors.orange,
                    width: 1.0,
                  ),
                ),
                child: Row(
                  children: [
                    Padding(
                      padding: const EdgeInsets.only(right:8.0),
                      child: Icon(Icons.mode_fan_off_outlined,),
                    ),
                    Text(
                      hs['fan'] == 0 ? "Off" : "On",
                      style:TextStyle(
                        fontSize: 20,
                      ),
                    )
                  ],
                ),
              ),
              Container(
                padding: EdgeInsets.all(10),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10.0),
```

```
                    border: Border.all(
                      color: Colors.orange,
                      width: 1.0,
                    ),
                  ),
                  child: Row(
                    children: [
                      Padding(
                        padding: const EdgeInsets.only(right:8.0),
                        child: Icon(Icons.heat_pump_rounded,),
                      ),
                      Text(
                        hs['heater'] == 0 ? "Off" : "On",
                        style:TextStyle(
                          fontSize: 20,
                        ),
                      )
                    ],
                  ),
                ),
                Container(
                  padding: EdgeInsets.all(10),
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(
                      color: Colors.orange,
                      width: 1.0,
                    ),
                  ),
                  child: Row(
                    children: [
                      Padding(
                        padding: const EdgeInsets.only(right:8.0),
                        child: Icon(Icons.factory_outlined,),
                      ),
                      Text(
                        hs['exhaust'] == 0 ? "Off" : "On",
                        style:TextStyle(
                          fontSize: 20,
                        ),
                      )
                    ],
                  ),
                )
              ],
```

```
                    ),
                  ),
                  Padding(
                    padding: const EdgeInsets.only(top:10,left:20.0, right:
20),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.spaceAround,
                      children: [
                        Container(
                          padding: EdgeInsets.all(10),
                          decoration: BoxDecoration(
                            borderRadius: BorderRadius.circular(10.0),
                            border: Border.all(
                              color: Colors.orange,
                              width: 1.0,
                            ),
                          ),
                          child: Row(
                            children: [
                              Padding(
                                padding: const EdgeInsets.only(right:8.0),
                                child:
Icon(Icons.notifications_on_outlined,),
                              ),
                              Text(
                                hs['bhc'] == 0
                              ? "Off"
                              : hs['bhc'] == 010 ||
                                      hs['bhc'] == 011 ||
                                      hs['bhc'] == 001
                                  ? "Off"
                                  : "On",
                                  style:TextStyle(
                                    fontSize: 20,
                                  ),
                              )
                            ],
                          ),
                        ),

                        Container(
                          padding: EdgeInsets.all(10),
                          decoration: BoxDecoration(
                            borderRadius: BorderRadius.circular(10.0),
                            border: Border.all(
```

```
                color: Colors.orange,
                width: 1.0,
              ),
            ),
            child: Row(
              children: [
                Padding(
                  padding: const EdgeInsets.only(right:8.0),
                  child: Icon(Icons.h_plus_mobiledata,),
                ),
                Text(
                 hs['bhc'] == 0
                ? "Off"
                : hs['bhc'] == 010 ||
                      hs['bhc'] == 110 ||
                      hs['bhc'] == 011
                  ? "On"
                  : "Off",
                  style:TextStyle(
                    fontSize: 20,
                  ),
                )
              ],
            ),
          ),


      Container(
            padding: EdgeInsets.all(10),
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(10.0),
              border: Border.all(
                color: Colors.orange,
                width: 1.0,
              ),
            ),
            child: Row(
              children: [
                Padding(
                  padding: const EdgeInsets.only(right:8.0),
                  child: Icon(Icons.ac_unit_outlined,),
                ),
                Text(
                 hs['bhc'] == 0
                ? "Off"
```

```
                                : hs['bhc'] == 001 ||
                                      hs['bhc'] == 101 ||
                                      hs['bhc'] == 011
                                  ? "On"
                                  : "Off",
                                  style:TextStyle(
                                    fontSize: 20,
                                  ),
                                )
                              ],
                            ),
                          ),
                        ],
                      ),
                    ),
                      Padding(
                        padding: const EdgeInsets.only(top:10,left:35.0,
right: 35),
                        child: Container(
                         width: size.width* 0.90,
                             padding: EdgeInsets.all(10),
                             decoration: BoxDecoration(
                               borderRadius: BorderRadius.circular(10.0),
                               border: Border.all(
                                 color: Colors.orange,
                                 width: 1.0,
                               ),
                             ),
                             child: Row(
                               children: [
                                 Padding(
                                   padding: const
EdgeInsets.only(right:8.0),
                                   child: Text(
                                  "Tilt",
                                   style:TextStyle(
                                     fontSize: 20,
                                   ),
                                 ),
                                 ),
                                 Text(
                                  xxnum.toString(),
                                   style:TextStyle(
                                     fontSize: 20,
                                   ),
```

```
                                    )
                                  ],
                                ),
                              ),
                            ),
                          Padding(
                            padding: EdgeInsets.only(left: 30, top: 20),
                            child: Row(
                              children: [
                                GestureDetector(
                                  onTap: (){
                                    _showMortalityRatePopup(context);
                                  },
                                  child: Container(
                                    width: size.width*0.8,
                                    padding: EdgeInsets.all(20),
                                      decoration: BoxDecoration(
                                      borderRadius: BorderRadius.circular(10.0),

                                      border: Border.all(
                                        color: Colors.orange,
                                        width: 1.0,
                                      ),
                                      gradient: LinearGradient(
                                        begin: Alignment.topLeft,
                                        end: Alignment.bottomRight,
                                        colors: [Color.fromARGB(255, 255, 111, 0),
Color.fromARGB(255, 255, 0, 0)],
                                      ),
                                    ),
                                    child: Column(
                                      children: [
                                        Text("Mortality Calculation", style:
TextStyle(fontSize: 16,color: Colors.white),),
                                      ],
                                    )
                                  ),
                                )
                              ],
                            ),
                          ),
                          SingleChildScrollView(
                            // ignore: sized_box_for_whitespace
                            child: Container(
                              height: size.height*0.5,
```

Appendix-A: Project Codes

```
                        child: Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Column(
                            children: [
                              Padding(
                                padding: const EdgeInsets.all(8.0),
                                child: Container(
                                    width: size.width*0.8,
                                    height: 150,
                                    padding: EdgeInsets.all(20),
                                      decoration: BoxDecoration(
                                      borderRadius:
BorderRadius.circular(10.0),

                                      border: Border.all(
                                        color: Colors.orange,
                                        width: 1.0,
                                      ),

                                    ),
                                    child: Column(
                                      children: [
                                        Row(
                                          mainAxisAlignment:
MainAxisAlignment.spaceBetween,

                                          crossAxisAlignment:
CrossAxisAlignment.center,

                                          children: [
                                            Text("Tray 1", style:
TextStyle(fontSize: 28, fontWeight: FontWeight.bold),),
                                            DropdownButton<String>(
                                              hint: Text('Select an option'),
                                            icon:

Icon(Icons.arrow_drop_down),

                                            iconSize: 24,
                                            elevation: 16,
                                            style: TextStyle(
                                              color: Colors.deepPurple,
                                              fontSize: 18,
                                            ),
                                            underline: Container(
                                              height: 2,
                                              color: Colors.deepPurpleAccent,
                                            ),
                                            value:  selectedOption[0],
```

Appendix-A: Project Codes

```
                                                items: const [
                                                  DropdownMenuItem(
                                                    child: Text('Chicken'),
                                                    value: 'Chicken',
                                                  ),
                                                  DropdownMenuItem(
                                                    child: Text('Duck'),
                                                    value: 'Duck',
                                                  ),
                                                ],
                                                onChanged: (value){
                                                  setState(() {
                                                    if(dd[0]['status'] == true){
                                                      _selectedOption[0] =
value!;
                                                    }

                                                  });
                                                },
                                              ),
                                              ],
                                            ),
                                            Padding(
                                              padding: const
EdgeInsets.only(top:8.0),

                                              child: GestureDetector(
                                                onTap: (){
                                                  if(dd[0]['status'] == true){
                                                    DateTime now =
DateTime.now();
                                                    DateFormat format =
DateFormat('dd-MM-yyyy HH:mm');
                                                    if(_selectedOption[0] ==
"Duck"){
                                                      publishMessage('Specie_Se
lection_switch', '{"0":"1"}');
                                                    }
                                                    else{
                                                      publishMessage('Specie_Se
lection_switch', '{"0":"0"}');
                                                    }
                                                    sendStartRequest("1",
format.format(now), _selectedOption[0]);
                                                    setState(() {
```

Appendix-A: Project Codes

```
                                        dd[0]['date'] =
format.format(now);
                                      });
                                    }

                                  },
                                  child: Container(
                                    height: 50,
                                    decoration: BoxDecoration(
                                    borderRadius:
BorderRadius.circular(10.0),
                                    color: dd[0]['status'] == true?
Colors.green: Colors.grey,
                                    border: Border.all(
                                      color: Colors.orange,
                                      width: 1.0,
                                    ),


                                  ),
                                    child: Center(child:
Text(dd[0]['status'] == true?'Start':'In progress since '+ dd[0]['date'],
style: TextStyle(fontSize: dd[0]['status'] == true?20:10,color:
Colors.white),),),
                                  ),
                                ),
                              )
                            ],
                          )
                        ),
                      )
                      ,Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Container(
                            width: size.width*0.8,
                            height: 150,
                            padding: EdgeInsets.all(20),
                              decoration: BoxDecoration(
                              borderRadius:
BorderRadius.circular(10.0),

                              border: Border.all(
                                color: Colors.orange,
                                width: 1.0,
                              ),
```

Appendix-A: Project Codes

```
                                    ),
                                    child: Column(
                                      children: [
                                        Row(
                                          mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                                          crossAxisAlignment:
CrossAxisAlignment.center,
                                          children: [
                                            Text("Tray 2", style:
TextStyle(fontSize: 28, fontWeight: FontWeight.bold),),
                                            DropdownButton<String>(
                                              hint: Text('Select an option'),
                                            icon:
Icon(Icons.arrow_drop_down),
                                            iconSize: 24,
                                            elevation: 16,
                                            style: TextStyle(
                                              color: Colors.deepPurple,
                                              fontSize: 18,
                                            ),
                                            underline: Container(
                                              height: 2,
                                              color: Colors.deepPurpleAccent,
                                            ),
                                            value: _selectedOption[1],
                                            items: const [
                                              DropdownMenuItem(
                                                child: Text('Chicken'),
                                                value: 'Chicken',
                                              ),
                                              DropdownMenuItem(
                                                child: Text('Duck'),
                                                value: 'Duck',
                                              ),
                                            ],
                                            onChanged: (value){
                                              setState(() {
                                               if(dd[1]['status'] == true){
                                                    _selectedOption[1] =
value!;
                                               }
                                              });
                                            },
```

```
                                                    ),
                                                    ],
                                                  ),
                                                  Padding(
                                                    padding: const
EdgeInsets.only(top:8.0),
                                                    child: GestureDetector(
                                                      onTap: (){
                                                        if(dd[1]['status'] == true){
                                                          DateTime now =
DateTime.now();
                                                          DateFormat format =
DateFormat('dd-MM-yyyy HH:mm');
                                                          if(_selectedOption[1] ==
"Duck"){
                                                              publishMessage('Specie_Se
lection_switch', '{"0":"1"}');
                                                          }
                                                          else{
                                                              publishMessage('Specie_Se
lection_switch', '{"0":"0"}');
                                                          }
                                                          sendStartRequest("2",
format.format(now), _selectedOption[1]);
                                                          setState(() {
                                                            dd[1]['status'] = false;
                                                            dd[1]['date'] =
format.format(now);
                                                          });
                                                        }
                                                      },
                                                      child: Container(
                                                        height: 50,
                                                        decoration: BoxDecoration(
                                                        borderRadius:
BorderRadius.circular(10.0),
                                                        color: dd[1]['status'] == true?
Colors.green: Colors.grey,
                                                        border: Border.all(
                                                          color: Colors.orange,
                                                          width: 1.0,
```

Appendix-A: Project Codes

```
                                        ),
                                          child: Center(child:
Text(dd[1]['status'] == true?'Start':'In progress since '+ dd[1]['date'],
style: TextStyle(fontSize: dd[1]['status'] == true?20:10,color:
Colors.white),),),
                                        ),
                                      ),
                                    )
                                  ],
                                )
                              ),
                            )

                          ],
                        ),
                      ),
                    ),
                  ),
                ),
                Align
                (
                  alignment: Alignment.centerLeft,
                  child: Text("Graphs", style: TextStyle(fontSize: 28,
fontWeight: FontWeight.bold,),)),
                const Padding(
                  padding: EdgeInsets.only(top: 20.0),
                  child: Text(
                    "Temperature Graph ",
                    // hs['time'].toString(),
                    style: TextStyle(
                      color: Colors.orange,
                      fontSize: 22,
                      fontWeight: FontWeight.w600,
                    ),
                    textAlign: TextAlign.left,
                  ),
                ),
                // ignore: sized_box_for_whitespace
                Container(
                  height: 200,
                  width: size.width * 0.95,
                  child: charts.TimeSeriesChart(
                    _seriesList,
                    animate: _animate,
```

Appendix-A: Project Codes

```
      defaultRenderer:
          charts.LineRendererConfig(includePoints: true),
      primaryMeasureAxis: charts.NumericAxisSpec(
        tickProviderSpec:
            charts.BasicNumericTickProviderSpec(
                zeroBound: false),
      ),
      domainAxis: charts.DateTimeAxisSpec(
        renderSpec: charts.SmallTickRendererSpec(
          labelStyle: charts.TextStyleSpec(
              fontSize: 14,
              color: charts.MaterialPalette.white),
          lineStyle: charts.LineStyleSpec(
              color: charts.MaterialPalette.gray.shade400),
        ),
      ),
    ),
  ),
  const Padding(
    padding: EdgeInsets.only(top: 20.0),
    child: Text(
      "Humidity Graph ",
      // hs['time'].toString(),
      style: TextStyle(
        color: Colors.orange,
        fontSize: 22,
        fontWeight: FontWeight.w600,
      ),
      textAlign: TextAlign.left,
    ),
  ),
  // ignore: sized_box_for_whitespace
  Container(
    height: 200,
    width: size.width * 0.95,
    child: charts.TimeSeriesChart(
      _seriesList2,
      animate: _animate,
      defaultRenderer:
          charts.LineRendererConfig(includePoints: true),
      primaryMeasureAxis: charts.NumericAxisSpec(
        tickProviderSpec:
            charts.BasicNumericTickProviderSpec(
                zeroBound: false),
      \
```

Appendix-A: Project Codes

```
                        domainAxis: charts.DateTimeAxisSpec(
                          renderSpec: charts.SmallTickRendererSpec(
                            labelStyle: charts.TextStyleSpec(
                                fontSize: 14,
                                color: charts.MaterialPalette.white),
                            lineStyle: charts.LineStyleSpec(
                                color: charts.MaterialPalette.gray.shade400),
                          ),
                        ),
                      ),
                    )

                  ],
                ),
              ),
            ),
          );
        }
      },
    ),
  );
}

Widget footer() {
  return Expanded(
    // ignore: sort_child_properties_last
    child: Container(
      color: Colors.grey,
      alignment: Alignment.centerRight,
      padding: const EdgeInsets.only(right: 16),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Padding(
            padding: const EdgeInsets.only(left:25.0, bottom: 20, ),
            child: Text(
              "Chick 'O Egg",
              style: TextStyle(
                  fontWeight: FontWeight.w700,
                  fontSize: 17,
                  color: Colors.orange),
            ),
          ),
```

Appendix-A: Project Codes

```
            padding: const EdgeInsets.only( bottom: 20),
            child: Text(
              statusText,
              style: TextStyle(
                  fontWeight: FontWeight.normal,
                  color: Colors.orange),
            ),
          ),
        ],
      ),
    ),
    flex: 1,
  );
}

_connect() async {
  ProgressDialog progressDialog = ProgressDialog(context,
      blur: 0,
      dialogTransitionType: DialogTransitionType.Shrink,
      dismissable: false);
  progressDialog.setLoadingWidget(CircularProgressIndicator(
    valueColor: AlwaysStoppedAnimation(Colors.red),
  ));
  // progressDialog
  //     .setMessage(Text("Please Wait, Connecting to AWS IoT MQTT Broker"));
  // progressDialog.setTitle(Text("Connecting"));
  // progressDialog.show();

  isConnected = await mqttConnect('esp8266/pub');
  // progressDialog.dismiss();
}

_disconnect() {
  client.disconnect();
}

void publishMessage(String topic, String message) {
final MqttClientPayloadBuilder builder = MqttClientPayloadBuilder();
builder.addString(message);
client.publishMessage(topic, MqttQos.atMostOnce, builder.payload!);
}


Future<bool> mqttConnect(String uniqueId) async {
  setStatus("Connecting MQTT Broker");
```

```
    /*
        The function mqttConnect connects to the MQTT broker using AWS IoT
certificates
        that are loaded from the assets/cert directory. It sets up the client
object
        with necessary parameters such as the keep-alive period, port,
security, and
        callback functions for various events such as connection success,
disconnection,
        and pong response.
        After establishing a connection, the function subscribes to an MQTT
topic with Quality
        of Service (QoS) set to at most once.
        The setStatus, onConnected, onDisconnected, and pong functions are
helper functions that
        update the state of the widget (if this code is running in a Flutter
application) and
        provide callbacks for the MQTT client events.
     */
    // After adding your certificates to the pubspec.yaml, you can use Security
Context.

    ByteData rootCA = await rootBundle.load('assets/cert/AmazonRootCA1.pem');
    ByteData deviceCert = await rootBundle.load('assets/cert/cert.pem.crt');
    ByteData privateKey = await rootBundle.load('assets/cert/pri.pem.key');

    SecurityContext context = SecurityContext.defaultContext;
    context.setClientAuthoritiesBytes(rootCA.buffer.asUint8List());
    context.useCertificateChainBytes(deviceCert.buffer.asUint8List());
    context.usePrivateKeyBytes(privateKey.buffer.asUint8List());

    client.securityContext = context;

    client.logging(on: true);
    client.keepAlivePeriod = 20;
    client.port = 8883;
    client.secure = true;
    client.onConnected = onConnected;
    client.onDisconnected = onDisconnected;
    client.pongCallback = pong;

    final MqttConnectMessage connMess =
        MqttConnectMessage().withClientIdentifier(uniqueId).startClean();
    client.connectionMessage = connMess;
```

```
    await client.connect();
    if (client.connectionStatus!.state == MqttConnectionState.connected) {
      print("Connected to AWS Successfully!");
    } else {
      return false;
    }

    const topic = 'esp8266/pub';
    client.subscribe(topic, MqttQos.atMostOnce);

    return true;
  }

  void setStatus(String content) {
    setState(() {
      statusText = content;
    });
  }

  void onConnected() {
    setStatus("Client connection was successful");
  }

  void onDisconnected() {
    setStatus("Client Disconnected");
    isConnected = false;
  }

  void pong() {
    print('Ping response client callback invoked');
  }
}



Future<void> _showMortalityRatePopup(BuildContext context) async {
  TextEditingController eggsController = TextEditingController();
  TextEditingController chicksController = TextEditingController();

  return showDialog<void>(
    context: context,
    barrierDismissible: true,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Mortality Rate Calculator'),
```

Appendix-A: Project Codes

Bibliography

[1] Benjamin Kommey, Daniel Akudbilla, Godfred Doe, Clifford Owusu Amponsah, "A low-cost smart egg-incubator", Department of Computer Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, Vol. 4, No. 1, February 2022, pp.22-33.

[2] Adegbulugbe T. A, Atere A.O, Fasanmi O.G, "Development of an Automatic Electric Egg Incubator", International Journal of Scientific & Engineering Research, Volume 4, Issue 9, September 2013.

[3] IDOKO E, OGBEH G.O, IKULE F.T, "Design and implementation of automatic fixed factors", Electrical and Electronics Department. Agricultural and Environmental Engineering Department. College of Engineering, University of Agriculture, Makurdi, Volume - 5, Issue - 6, June – 2019.

[4] Mariani MJP, Wacas RU, Padre RJ, Soriano GT, Elveña VB, Sarne JC (2021), "Design modification of a cost-efficient microcontroller-based egg incubator", Indian Journal of Science and Technology, vol. 2, no. 2, pp. 51-56, 2021.