VIETNAM – KOREA UNIVERSITY OF INFORMATION & COMUNICATION TECHNOLOGY

**FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS**



**DESIGN AND IMPLEMENT A SMARTGREENHOUSE SYSTEM**

**SUMMARY OF GRADUATION THESIS**

**MAJORING IN COMPUTER ENGINEERING**

|  |  |
| --- | --- |
| **Student** | **: Le Thien Nhan** |
| **Student ID** | **: 19CE030** |
| **Major** | **: Computer Engineering Technology** |
| **School year** | **: 2019 - 2024** |
| **Instructor** | **: MSc. Nguyen Thi Huyen Trang** |

Da Nang – January 2024

VIETNAM – KOREA UNIVERSITY OF INFORMATION & COMUNICATION TECHNOLOGY

**FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS**



**DESIGN AND IMPLEMENT A SMARTGREENHOUSE SYSTEM**

**SUMMARY OF GRADUATION THESIS**

**MAJORING IN COMPUTER ENGINEERING**

|  |  |
| --- | --- |
| **Student** | **: Le Thien Nhan** |
| **Student ID** | **: 19CE030** |
| **Major** | **: Computer Engineering Technology** |
| **School year** | **: 2019 - 2024** |
| **Instructor** | **: MSc. Nguyen Thi Huyen Trang** |

Da Nang – January 2024

VIETNAM – KOREA UNIVERSITY OF INFORMATION & COMUNICATION TECHNOLOGY

**FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS**



**DESIGN AND IMPLEMENT A SMARTGREENHOUSE SYSTEM**

**SUMMARY OF GRADUATION THESIS**

**MAJORING IN COMPUTER ENGINEERING**

|  |  |
| --- | --- |
| **Student** | **: Le Thien Nhan** |
| **Student ID** | **: 19CE030** |
| **Major** | **: Computer Engineering Technology** |
| **School year** | **: 2019 - 2024** |
| **Instructor** | **: MSc. Nguyen Thi Huyen Trang** |

Da Nang – January 2024

# **PREAMBLE**

1. **Urgency of the topic**

In today's context, designing and implementing a smart greenhouse system is crucial for advancing agriculture and technology. This project addresses key aspects, including modernizing agriculture, efficient data management, integrating new technologies like MongoDB and Outsystems, fostering professional development in embedded programming and IoT. The system's automatic environment adjustments help farmers navigate complex climate conditions. By combining practical application with technology, this project contributes to global development and represents a personal challenge for self-improvement.

1. **Methods of implementation**

Theoretically

Research project challenges, configure devices, transfer data over LoRa connections, learn how to build servers and integrate with MongoDB with Outsystem technology.

Experimentally

Process and display data on the user interface, use MongoDB to store data, install Outsystem to design applications on Android and iOS.

1. **Subject and scope of study**

**Research subjects**

Smart greenhouse system, sensors and microcontroller.

IoT technology, especially data communication and LoRa waves.

Data storage and server

Outsystem technology for developing environmental monitoring applications.

**Sesearch scope**

* Details of the structure and operation of the greenhouse system.
* Operating principle of sensor and microcontroller.
* Data communication method using LoRa waves.
* Server data and settings
* Applying Outsystem technology in developing environmental monitoring applications.

1. **Research content**

**Hardware**

* Component options include honeycombs, pumps, motors, relays
* Use modules like L298, LM2596, RFID, LoRa, Uno R3, ESP8266 to take advantage of flexibility.
* Sensors such as rain, light, DHT11, soil moisture and devices such as fans, lights, LCDs, pumps, mist pipes, doors and roofs..
* Use materials such as wooden panels, mica, carpet, iron frames for mechanical construction and other accessories

**Software**

* Draw model diagrams to understand the structure and interactions between parts.
* Deploy the hardware and mount the model.
* Transmits data from 2 greenhouse node sensors and connects to the central node.
* Program a NodeJS server to connect to MongoDB.
* Programming and connecting to the central ESP8266 network, linking to the server and MongoDB.
* Connect MongoDB to the Outsystem system.
* Build Outsystem Android and iOS applications
* Test, evaluate and adjust the system to ensure stable and effective results.

1. **Scientific and practical significance of the topic**

**Scientific significance:**

* Enhance your understanding
* Applying new technology
* Challenge and overcome
* Contribution to the research community

**Practical Significance**

* Enhance agricultural efficiency
* Automatic and effective management
* Highly practical application
* Save energy and resources

In summary, the project bridges theory and practice, offering valuable contributions to both science and practical applications in agriculture.

1. **Topic layout**

* Acknowledgments
* Table of contents
* List of symbols and abbreviations
* List of tables
* Preamble
* Chapter 1: Theoretical basis
* Chapter 2: System design
* Chapter 3: System implement
* Conclusion and development direction
* References

# **CHAPTER 1: OVERVIEW**

1. **GREENHOUSE SYSTEM**

A smart greenhouse system optimizes agricultural efficiency by integrating sensors, automation, and IoT connectivity. It monitors and controls environmental factors, utilizes data analytics, and incorporates advanced technologies like AI and blockchain. This approach enhances crop performance, reduces water and energy consumption, and provides sustainable solutions for food production.

1. **DATA TRANSMISSION METHOD**
2. **LoRa**

LoRa, or Long Range Radio, developed by Cycleo and later acquired by Semtech, facilitates energy-efficient data transmission over extended distances. Ideal for IoT devices like sensor networks, LoRa enables signals to cover several kilometers without a power amplifier, making it well-suited for applications requiring long-range data collection.

1. **LoRaWAN**

LPWAN, like LoRaWAN, is a low-power wireless technology for IoT, balancing cost-effectiveness and energy efficiency. Components include sensors (end devices), gateways, network server, and application servers. LPWAN offers extended coverage, low bandwidth, and prolonged battery life, making it ideal for IoT devices requiring long-range connectivity with minimal power consumption.

1. **HTTP connection protocol**

HTTP is a TCP/IP-based protocol for retrieving resources such as HTML documents, text, videos, and images. It facilitates data exchange in the Client/Server model, where clients send requests to HTTP servers and receive responses. These requests and responses are simple-structured messages called HTTP Messages.

1. **HARDWARE**
2. **ESP8266**
3. **Arduino UNO R3**
4. **Data transceiver module**
5. **RF transceiver module NRF24L01**
6. **RF transceiver circuit NRF24L01 antenna**
7. **RFID transceiver module**
8. **Sensor module**
9. **Rain sensors module**
10. **DHT11 sensor module**
11. **Soil moisture sensor module**
12. **Light sensor module**
13. **Proximity sensor module**
14. **Engine module**
15. **L298N engine control module**
16. **LM2596 voltage regulator circuit**
17. **DC Gearmotor**
18. **Honeycomb power supply**
19. **Pump motor**
20. **Other devices**
21. **LCD**
22. **Mist nozzle**
23. **Fan, light bulb**
24. **Aluminum, mica, anti-UV film**
25. **SOFTWARE AND DATABASE**
26. **Software**
27. **Arduino IDE**

Arduino IDE is tailored for programming Arduino boards like Uno R3 and ESP8266. It serves as a text editor, simplifying code writing and facilitating program development and testing on the boards.

1. **Visual Studio Code**

Visual Studio Code: Lightweight, versatile IDE for multiple programming languages including C++, C#, Java, Python, JavaScript, HTML, CSS, and more.

1. **Altium Designer**

Altium Designer is a versatile tool for designing electronic circuits, specializing in drawing schematics and PCBs. Widely employed in the electronics industry.

1. **Outsystem**

OutSystems is a low-code platform that expedites flexible development, supporting cross-platform integration for web and mobile applications. It includes tools for application performance monitoring, management, and efficient error handling.

1. **Postman**

Postman is a computer application that helps develop, test, and interact with APIs .

1. **Database**

MongoDB is a NoSQL (Not Only SQL) database management system (DBMS), designed to store and retrieve data in a flexible and scalable way.

# **CHAPTER 2: SYSTEM DESIGN**

1. **NECESSARY REQUIREMENTS**

Design a flexible and safe greenhouse system with seamless hardware-software interaction.Sensor data integration, energy optimization, and easy maintenance. Ensure robust connectivity with technologies like OutSystem and NodeJS for remote monitoring and user-friendly applications.

1. **SYSTEM STRUCTURE**

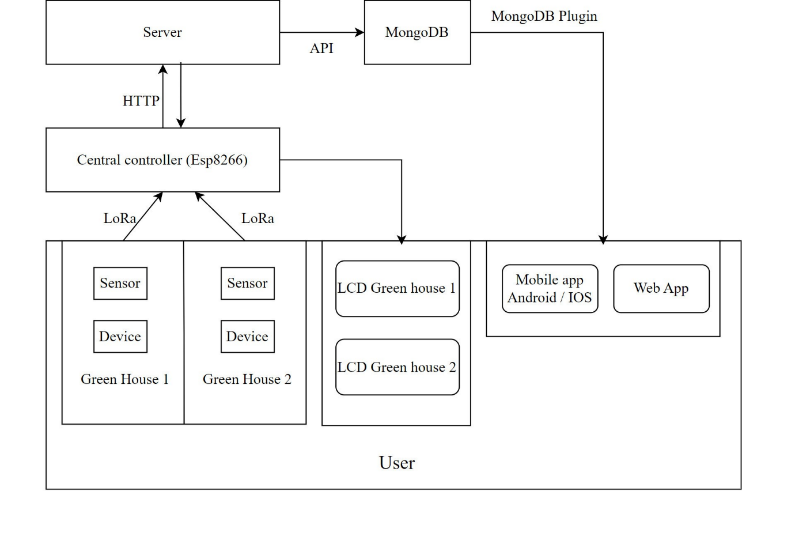


Figure 1.1: System structure

The project employs automation and manual control.

* Automation:

Sensors trigger actions, and data is sent via LoRa waves.

* Manual:

Central device receives and sends sensor values to the server.

Server saves data to MongoDB, displayed on an LCD.

Mobile/web apps, through Outsystem, connect to MongoDB for real-time monitoring.

1. **ALGORITHM FLOWCHART**
2. **Diagram of centeral control alogrithm**

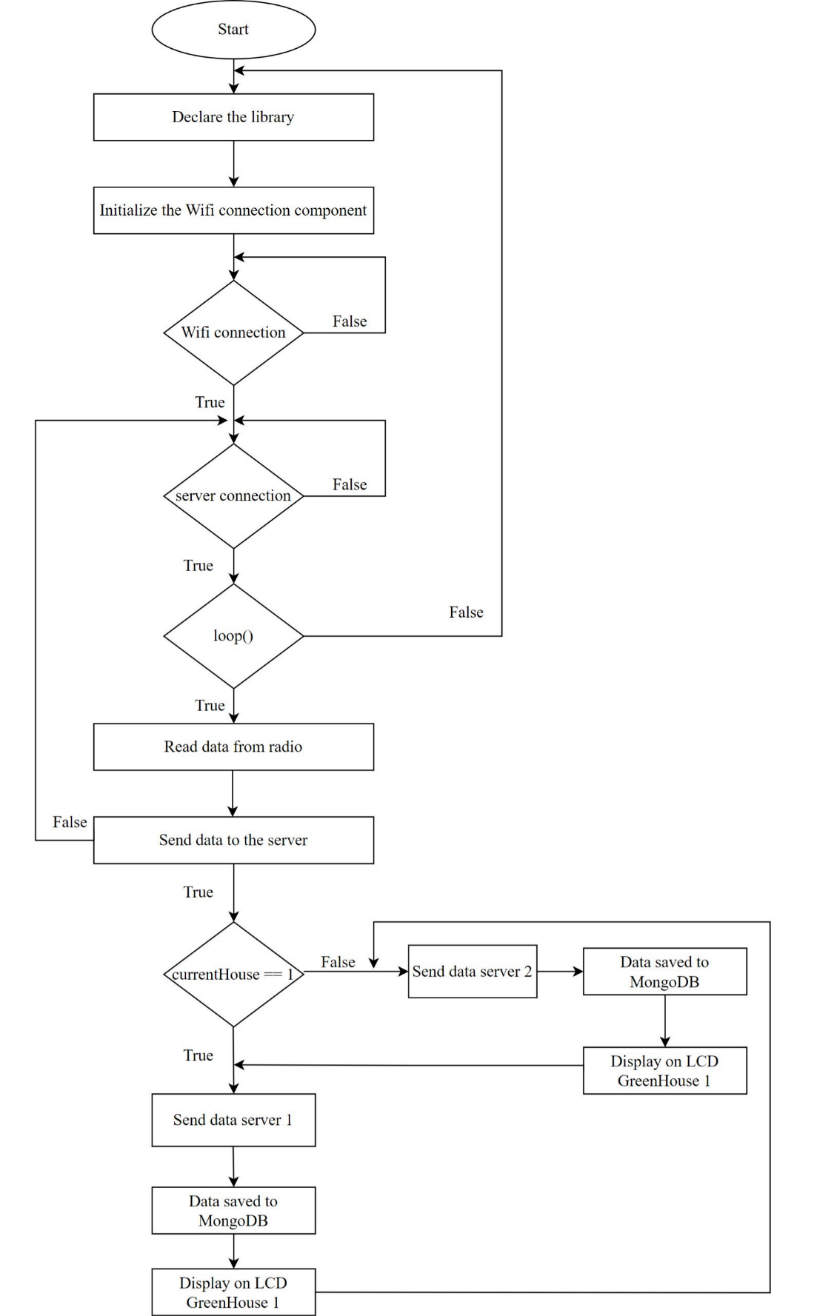
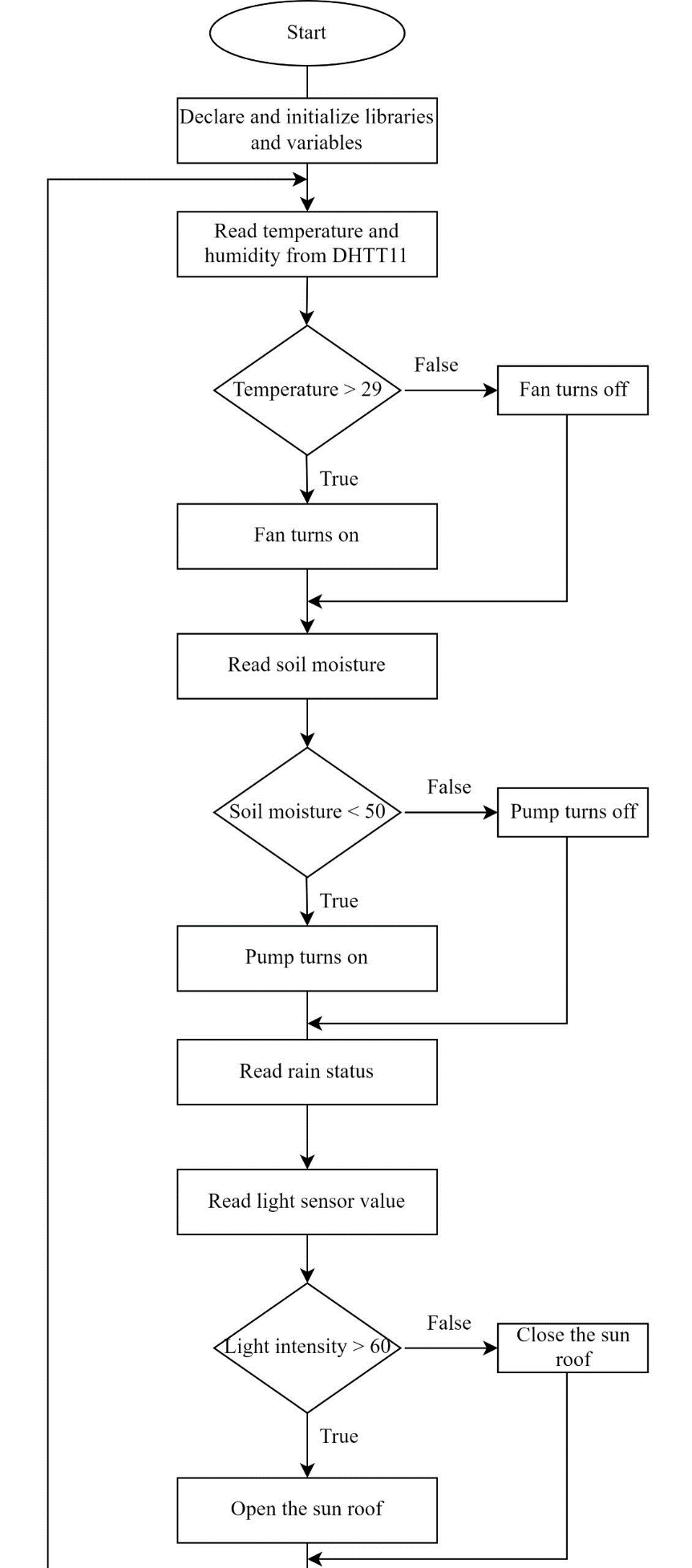


Figure 2.1: Diagram of central control algorithm

1. **Greenhouse algorithm flow**

****

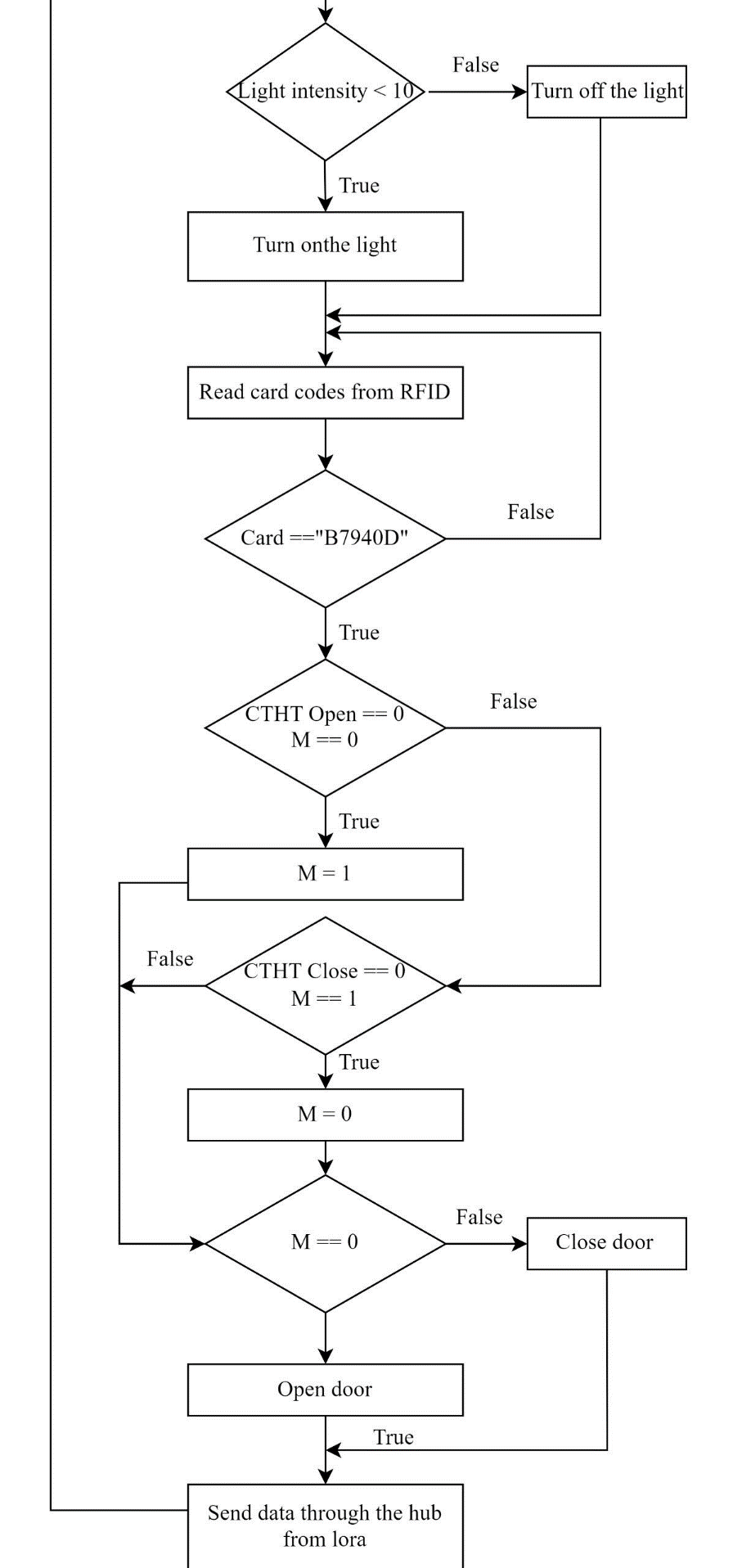
****

Figure 2.2 : Greenhouse algorithm flow diagram 1 và 2

The algorithm diagram for greenhouses 1 and 2 are similar, the only difference is that house number 1 is treated with additional roof equipment.

1. **CIRCUIT DESIGN**
2. **Greenhouse principle circuit**

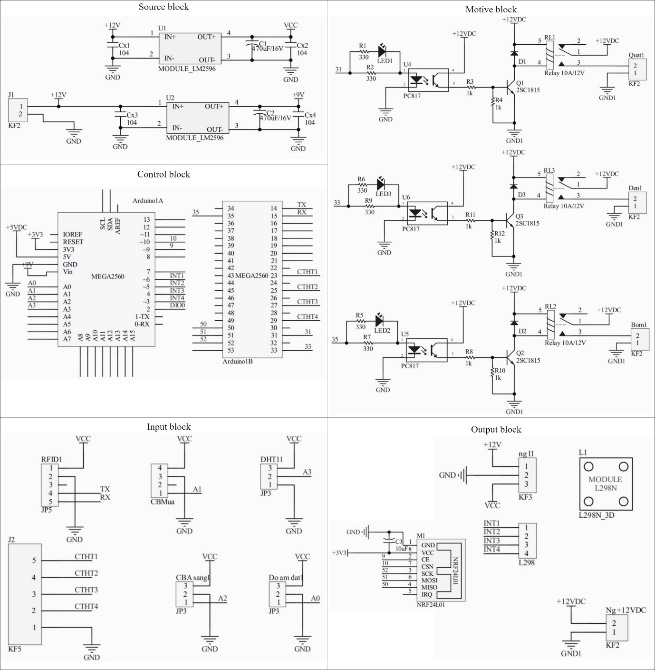


Figure 2.3: Greenhouse principle circuit

1. **Greenhouse central monitoring principle circuit**

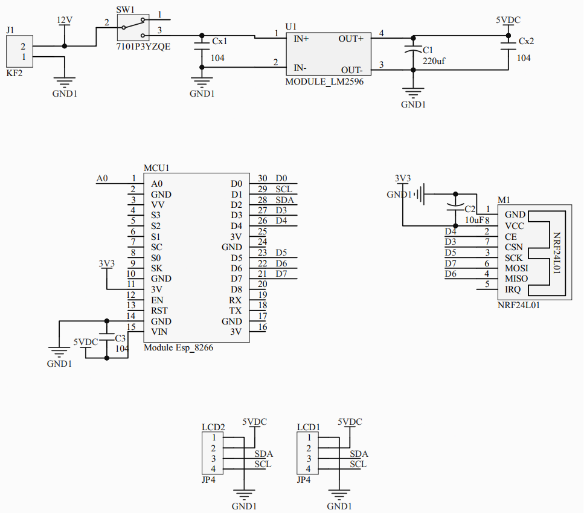


Figure 2.4: Greenhouse central monitoring principle circuit

1. **PCB circuit diagram**

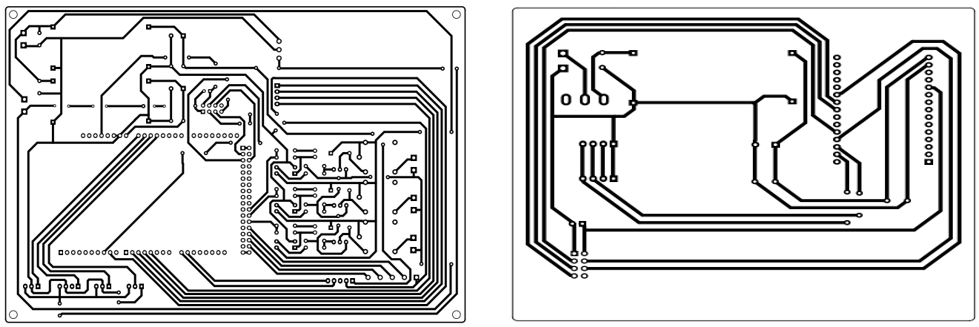
****

Figure 2.5: Greenhouse 1,2 and central system PCB circuit

1. **DESIGN OF SERVER STRUCTURE**

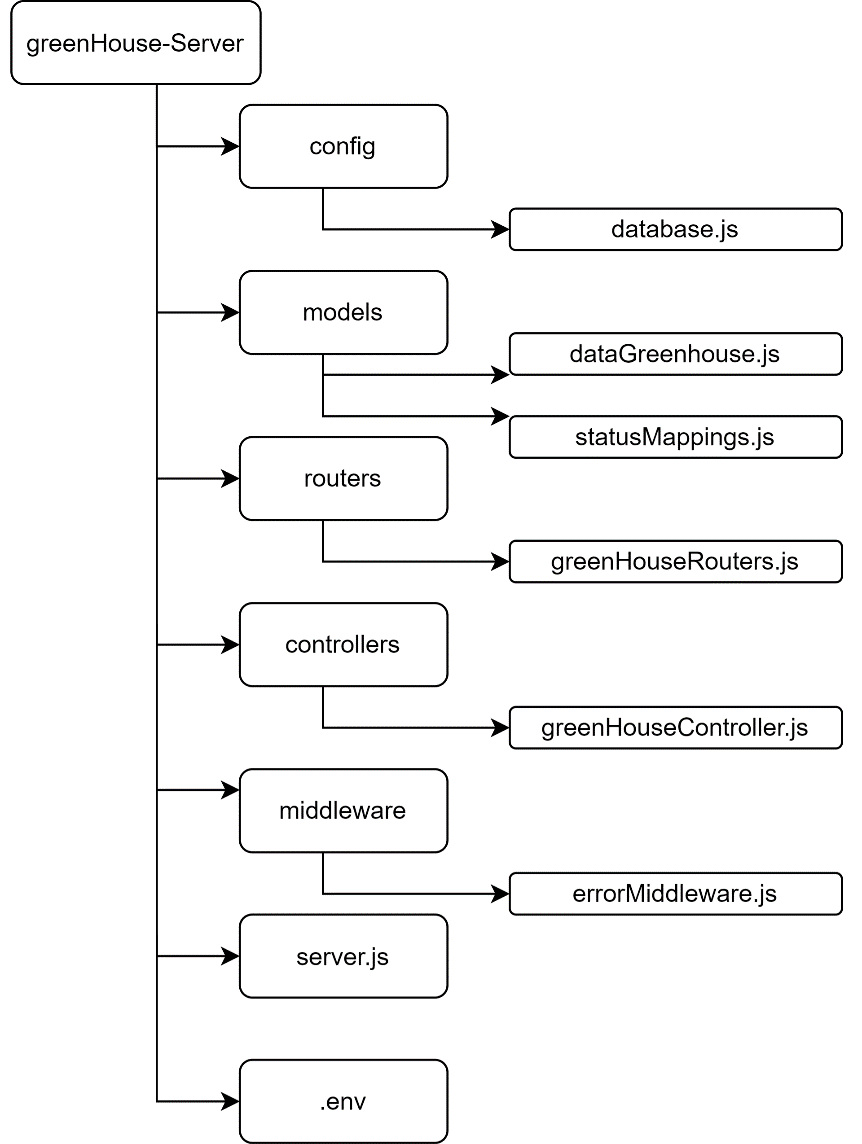
****

Figure 2.6: Server structure diagram

# **CHAPTER 3: IMPLEMENTATION**

1. **Implementation process**

The system will basically consist of 5 main parts

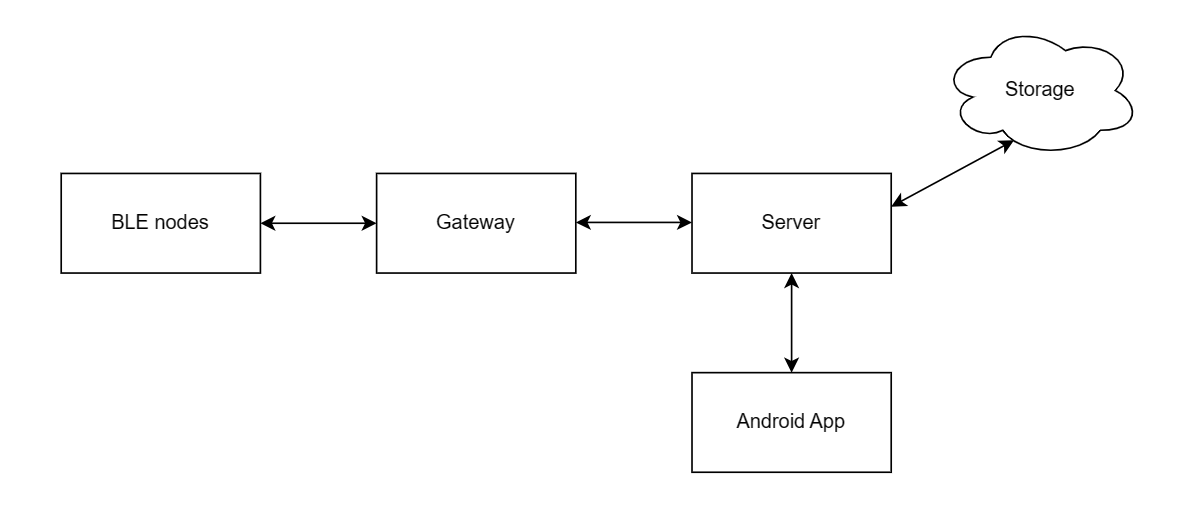


Figure 3. 1 System block

1. **Gateway setup**
2. **Environment setup**

Libraries used in the project:

* + **Bleak**
  + **Asyncio**
  + **Aiohttp**
  + **Requests**
  + **Firebase admin**

1. **Implement**
2. **Make file is a service**

During operation, we may encounter situations such as not being able to connect to the BLE server, power outage, etc. Therefore, we should create a service file so that we can easily restart it when encountering unexpected problems.

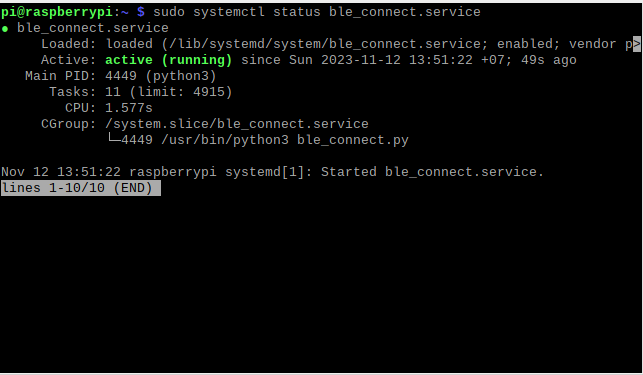


Figure 3. 2 Initialize and run Service

1. **BLE node**

The system uses ESP32 as 3 nodes, 1 server node and 2 client nodes.

1. **Server node**

A BLE Server is also known as a BLE Peripheral. With a BLE Server, we will expose one or more services where each service has one or more characteristics and each characteristic may have zero or more descriptors.

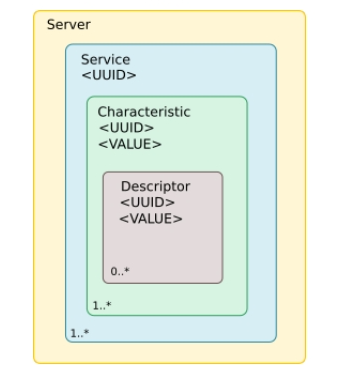


Figure 3. 3 Structure of BLE server

In ESP32, we create the concept of classes that represent these items.

* BLEServer – Models a server.
* BLEService – Models a service. Owned by a BLEServer.
* BLECharacteristic – Models a characteristic. Owned by a BLEService.
* BLEDescriptor – Models a descriptor. Owned by a BLECharacteristic.
* BLEAdvertising – Models advertising. Owned by a BLEServer to let others know of server existence.

1. **Client node**

In this part, ESP32 doesn't host services but instead wishes to be a consumer of services hosted elsewhere. The story can further be broken down into two sub-stories namely scanning and interaction.

If assume that ESP32 starts up and wishes to be a client of a remote BLE server then it has to connect to that server. In order to connect to the server, we need to know the address of the target server. An address is a 6 byte value commonly written in the form:

nn:nn:nn:nn:nn

While in principle this can be hard-coded into application or manually entered, this is not the common practice. Instead, we need perform a procedure known as a scan. Scanning is the idea of helping client nodes actively listen to the BLE radio frequencies of servers that are advertising their existence.

When perform a BLE scan, client nodes receive small sized records of data that always contain the address of the advertiser and sometimes additional information such as the services they provide or other descriptive items. This information arrives at nodes passively.

1. **Server**

In this project, I use Java as the programming language and Spring boot Framework.

The server undertakes the following functions:

* Create devices corresponding to each hardware.
* Set up words to easily control each device.
* Monitoring, measurement, statistics and analysis.
* Set up notifications.
* Schedule status updates for each device.
* Registration, login and user related settings.
* For management users, it is possible to update user status and manage the system easily.
* There are charts that make observing values easier.

1. **API**

* Device API: are APIs that perform operations with Devices, each device will correspond to a real device.
* Device Monitor API: is used to store monitor data for analysis and measurement.
* Device Timer API: is used to store timers for control type devices, we can easily schedule devices to turn on and off quickly and easily.
* Device Token API: is used to store device tokens so that notifications can be received.
* Speech Data API: is used to store words so we can use voice to control devices such as turning on, turning off...
* Notification API: is used to store notifications sent to users.
* News API: is used as a place to store user activities, such as turning on and off, controlling, saving, deleting,...
* User API: is used to manage, create, assign user rights...

1. **Android App**

Android application written in Kotlin language. It can be used in 2 situations:

* In case the user is near BLE devices and the internet signal is lost, the device can be observed through the application with BLE mode. The mobile application will be considered a node and can easily read and write to other nodes.
* In case the user is not near the BLE devices and the internet signal is active, the device can be observed with internet mode.

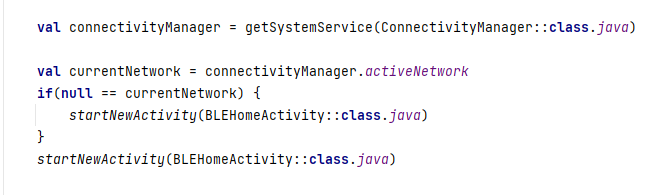


Figure 3. 4 Check the connection to switch between 2 modes of Android devices

The application includes functions:

* Displays the device list.
* Display details of each device.
* If the device is control type, it can be turned on and off, create timers, and control on and off by voice. And if it is a monitor type, we can see the temperature difference every minute of the day, month and year. We can also create alerts so that when the temperature reaches a threshold we set it. will send a warning to your phone.
* Setup words that can be used to control the device by voice.
* Connects to BLE devices and can view lists, control, and monitor.

1. **Deploy**

Server system and mongoDB database are deployed on VPS. For the java server I used Docker so I could easily deploy it quickly.

In the server, I config docker using Jib library, it handles all steps of packaging your application into a container image.



Figure 3. 5 configuration of the jib library in the XML file

Once the image is built, it will automatically be pushed to Docker Hub. On VPS, we just need to create a docker-compose file to create a container from the image we just pulled.

1. **Conclusion**

In this chapter, I have outlined the system's implementation process

* Setting up a Gateway as an intermediary so that data can communicate and transmit easily.
* Initialize BLE nodes with a server node and 2 client nodes (one device is used to measure temperature and humidity sent to the server and one device is used to receive control signals from the server to control the device)
* Shows the steps to set up a BLE node on the ESP32 module
* Install the server, initialize APIs so clients can communicate with it, create a system management website
* Install android applications, with two modes: BLE mode and Internet mode so users can easily use it anywhere.
* Perform VPS configuration, install docker to launch the server, and install the database.

# **RESULT AND DEVELOPMENT ORIENTATION**

**These achievements**

1. Understand the ble protocol.
2. Write a program for Node Server and Client on ESP32 to be able to communicate with each other
3. Gateway configuration to be able to get BLE signals from the nodes and send it to the server for storage.
4. Understand how to design applications on Android.
5. Perform the connection and exchange of control data between the device running Android operating system and the ESP32 module via Bluetooth Low Energy.
6. Perform data on the server for storage through gateway.
7. Set two control modes on Android devices to help users conveniently control.
8. Understand how to configure a VPS to install the database and run the server.
9. Understand how to create a docker image and run the container docker on the server.
10. Create utilities for users to easily check, monitor and control equipment.

**The shortcomings have not been resolved**

1. Voice control cannot be done within the application.
2. The transmission of dynamic uuid to each device has not been done but must be assigned hard on the code on each device.
3. On Android devices, it is not possible to automatically load the Service and Characteristics lists that can be connected but can only be assigned hard on the code to connect.

**Development orientation**

1. Research the methods so that the device can automatically load the Characteristics.
2. Improve the design of the server and Android application.
3. Increase security for data transmitted between nodes and from node to server.

# **REFERENCES**

1. Mohammad Afaneh (2018), Intro to Bluetooth Low Energy: The easiest way to learn BLE, Independently pulished.
2. Craig Walls (2022), Spring in Action, Sixth Edition, Manning publications.
3. Chendong Liu (2021), “A Comprehensive Study of Bluetooth Low Energy”, Jounal of Physics: Conference Series.
4. Carles Gomez, Joaquim Oller, Josep Paradells (2012), “Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology”, Sensors.
5. Nigel Poulton (2017), Docker Deep Dive, Independently published.

**Reference websites**

1. <https://developer.android.com>.
2. <https://docs.spring.io>.
3. <https://docs.docker.com>.