VIETNAM – KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS

GRADUATION THESIS

DESIGN AND IMPLEMENTATION OF A SMART GREENHOUSE SYSTEM

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

Da Nang, December 2023

VIETNAM – KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS

GRADUATION THESIS

DESIGN AND IMPLEMENTATION OF A SMART GREENHOUSE SYSTEM

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

Da Nang, December 2023

ACKNOWLEDGMENTS

Success is never the achievement of a single individual but always reflects the consensus, support, and help from those around. During the journey of completing the graduation project at the end of course 19, I had the opportunity to study and receive great help from teachers and people who had gone through the previous course. Valuable lessons and dedicated guidance have helped me accumulate valuable experiences.

I would like to express my deep gratitude to the teachers of the Faculty of Computer Engineering and Electronics, VietNam - Korea University of Information Communication Technology for sharing their valuable knowledge and helping me during the past 4 years. At the same time, I would like to sincerely thank the valuable contributions of former students who supported me in completing this graduation project.

In particular, I would like to express my deep gratitude to MSc. Nguyen Thi Huyen Trang, who spent her time and effort guiding me step by step to complete the project. Thanks to her guidance and instructions, I have achieved the results I have today. However, I realized that my project still had many shortcomings due to limited time and knowledge. I hope to receive sincere suggestions from teachers and friends to help me improve further.

Sincerely, thank.

Student

Le Thien Nhan

TABLE OF CONTENT

ABBREVIATIONS

ABBREVIATIONS	MEANING
IoT	Internet of Things
AI	Artificial Intelligence
LoRa	Long Range
LPWAN	Low-Power Wide-Area Network
RF	Radio Frequency
RFID	Radio-Frequency Identification
AES	Advanced Encryption Standard
HTTP	HyperText Transfer Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
UART	Universal Asynchronous Receiver/Transmitter
IC	Integrated Circuit
AVR	Alf and Vegard's RISC processor
ROM	Read-Only Memory
RAM	Random Access Memory
SPI	Serial Peripheral Interface
TWI	Two-Wire Interface
LCD	Liquid Crystal Display
IDE	Integrated Development Environment
PCB	Printed Circuit Board
API	Application Programming Interface
NoSQL	Not Only SQL
JSON	JavaScript Object Notation
SPI	Serial Peripheral Interface
CDS	Cadmium Sulphide
URI	Uniform Resource Identifier
DNS	Domain Name System

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1. Urgency of the topic

In today's context, the topic "Design and implementation of a smart greenhouse systems" is not only a technical dream but also an important step forward for the development of agriculture and technology. The urgency of this topic includes several important aspects:

- Modernizing agriculture: Smart green house promise to bring modernization to agriculture, optimize management and increase output.
- Effectively manage data: The ability to collect and manage data through apps and the web not only provides an overview but also a powerful decision tool.
- Connecting new technology: The combination of MongoDB and Outsystem is not only a breakthrough in data storage but also an opportunity to expand the application of technology into many new fields.
- Professional development: By setting learning objectives to supplement the fundamental knowledge and applications of embedded programming and IoT, this topic also provides opportunities to develop professional knowledge and technical skills.
- Adaptation to climate change: The system has the ability to automatically adjust the environment, helping farmers work less and more effectively in increasingly complex climate change conditions.
- Actual control: Testing and actual use of electronic components, sensors, and Android applications not only proves feasibility but also brings important experiences in applying technology to life daily.

Realizing the above issues, this project not only meets practical needs in agriculture but is also an important step towards modernization and technological integration, thereby contributing to global development. So I decided to learn, design, build a greenhouse system and challenge myself to surpass myself.

2. Methods of implementation

- *Theoretically*
  - Learn overview of the practical challenges faced by the project, helping to clearly define specific methods and goals.
  - Learn the operating principles of each device variant and module to get an overview of the project. configuration system.

- Learn how to transmit data via Lora connection, collect information from sensors in each greenhouse and transmit it to the central system.
- Study how the research center system transmits data to the NodeJs intermediate Server and stores information at the MongoDB database.
- Research how to set up the environment and research connections to bring data from MongoDB to the new Outsystem technology.

- *Experimentally*
  - Design and build applications on Android and iOS platforms to conveniently manage the system.
  - Perform data processing and display them on the user interface, while monitoring and automating system functions.
  - Create and use a MongoDB database, along with setting up key functions for effective communication and connectivity.
  - Create an account, set up the environment and system so that Outsystem can proceed with the build and deployment process and connect to MongoDB.
  - Combine devices into one system, perform installation and testing to ensure compatibility.

In short, this method emphasizes the balance between theory and practice, ensuring that every step from research to experiment is approached in an organized and effective manner.

3. Object and scope of the study

- *Research subjects*
  - Detailed research on the structure, operation and features of smart greenhouse systems, focusing on monitoring and automation capabilities.
  - Research and learn about electronic components used in the system, especially the operating principles of sensors and technical specifications.
  - Survey of automation control solutions and methods.
  - Learn and use LoRa waves to transmit and receive system data.
  - Learn about data transmission methods between smart home systems and cyberspace, especially protocols and techniques related to the Internet of Things (IoT).
  - Research MongoDB database to store system data information..

- Research system monitoring applications using new Outsystem technology on Android and iOS platforms.

- *Research scope*

  - Develop a smart greenhouse system model based on knowledge of embedded systems, microcontrollers, and IoT connection protocols.
  - Research and apply effective data communication methods between smart home systems and cyberspace, including the use of LoRa waves.
  - Use the new technology platform Outsystem to build an environmental monitoring application for both Android and iOS operating systems.

This research scope will focus on building detailed knowledge and practical applications of smart greenhouse systems, from hardware to software and the combination of embedded programming and IoT. In particular, applying new Outsystem technology to the system.

4. Research content

- *Hardware*

  - Component selection: Identify and use components such as honeycomb sources, pumps, geared motors, relays, and dominoes to build the system's operating infrastructure.
  - Using modules and ICs: Combine modules such as L298, LM2596, RFID, LoRa, Uno R3, ESP8266 to take advantage of their flexibility and interoperability.
  - Sensor and device integration: Connect and integrate sensors such as rain sensors, light sensors, DHT11 sensors, soil moisture sensors, proximity sensors, along with other devices such as fans, light bulbs, LCDs, water pumps, and mist lines.
  - Mechanical implement and accessories: Use materials such as boards, mica, carpet, and iron frames to build mechanics and accessories to support the system.

- *Software*

  - Draw a diagram of the overall model to clearly understand the structure and interaction between parts.
  - Carry out the hardware implement process and install the smart greenhouse system model.
  - Program and set up data transmission lines from the sensors of 2 greenhouse nodes and connect to the central node.

- Program and setup NodeJS server to connect to MongoDB database and deploy to the web.
- Programming, setting up and connecting the center's ESP8266 network, linking the server and MongoDB.
- Program and establish MongoDB data connection with Outsystem system.
- Programming and building applications using Outsystem language, automating sensor devices.
- Perform testing, evaluation, and adjustment of the system to ensure stability and efficiency.

5. Scientific and practical significance of the topic

- *Scientific significance*
  - Enhance your understanding: Expand your knowledge of embedded systems, IoT technology, and sensor operating principles, providing an in-depth understanding of the field of modern engineering and technology.
  - Applying new technology: Integrating MongoDB and Outsystem is not only a step forward in data storage and management but also an opportunity to introduce new technologies, contributing to innovation and development in the industry .
  - Challenge and overcome: Setting big challenges such as connecting between systems, handling big data, and using Outsystem is an opportunity to overcome limits, develop thinking and problem-solving skills.
  - Contribution to the research community: Providing new information and experiences, enriching the general knowledge base in the field of technology, creating conditions for the development of the research community.
- *Practical significance*
  - Enhance agricultural efficiency: Applying smart greenhouse systems helps improve efficiency and quality in agricultural production, contributing to the sustainability and development of the industry.
  - Automatic and effective management: The automatic monitoring and management system via mobile application helps farmers have a comprehensive view of the greenhouse environment, thereby optimizing the management process and making decisions. effective.

- Highly practical application: The topic not only stops at the theoretical level but also includes practical applications, from device integration to building control applications, ensuring high usefulness and application in the field. reality.
- Save energy and resources: Efficiency in managing the greenhouse environment helps save energy and resources, leading to a sustainable and environmentally friendly choice.
- Industry 4.0 development: Modeling and automation are challenging topics that help shape the development of Industry 4.0, creating new opportunities for research and innovation.

In short, the project is not only a harmony between theory and practice but also brings important contributions to both the scientific community and the agricultural production environment.

6. Topic layout

- Acknowledgments
- Table of contents
- Abbreviations
- List of figures
- List of tables
- Introduction
- Chapter 1: Theoretical basis
- Chapter 2: System design
- Chapter 3: System implementation
- Conclusion and development direction
- References

CHAPTER 1. THEORETICAL BASIS

1.1 GREENHOUSE SYSTEM

A smart greenhouse system is a technology application in the agricultural sector, where automation components and smart systems are integrated to create an efficient and energy-saving growing environment. The goal of the smart greenhouse system is to improve the ability to manage and monitor the growing process, from monitoring environmental conditions to controlling factors such as light, temperature, humidity and nutrients.

***Figure 1.1*: Greenhouse system**

The main features of the smart greenhouse system include:

- Sensors and monitoring: Use sensors to measure and monitor environmental factors such as temperature, humidity, brightness, CO2, and other data important to the growing process.
- Automation: Use automatic control system to perform functions such as water replenishment, cooling, brightness and temperature adjustment to optimize environmental conditions.
- Internet of things: Uses IoT connectivity to transmit data from sensors and control devices, allowing remote monitoring and management via the internet.
- Data analytics: Use data analytics to understand and evaluate crop performance, expect water and nutrient needs, and optimize the growing process.
- Technology application: Use applications and user interfaces to manage and monitor smart greenhouse systems, providing detailed and operational information for managers.

- Technology integration: Links technologies such as artificial intelligence (AI), machine learning and blockchain to improve performance and integrate smart elements into the growing process.

Smart greenhouse systems help improve agricultural productivity, reduce water and energy consumption, and provide sustainable solutions for food production.

1.2 DATA TRANSMISSION METHODS

1.2.1 LoRa

LoRa stands for Long Range Radio, developed by Cycleo and acquired by Semtech in 2012. Lora allows data transmission over distances of several tens of kilometers without the need for a power amplifier circuit. This saves significant energy when transmitting/receiving data. Therefore, signals can travel long distances, even through buildings, with very little energy. This is suitable for IoT devices with limited battery capacity. Lora can be widely used in data collection applications such as sensor networks, where sensor nodes can send measurement values to a center several kilometers away and can run for long periods on battery power.

Figure 1.2: LoRa technology

1.2.2 LoRaWAN

LPWAN stands for Low Power Wide Area Network, meaning low power wide area network. It does not refer to any one specific technology, but is a general term for any network designed to communicate wirelessly at a lower power than other networks such as cellular, satellite or WiFi networks.

LPWAN has characteristics such as large coverage, low bandwidth, small packet size and long battery life. LPWAN networks cost less than cellular networks and have a wider range than short-range wireless networks.

LPWAN provides connectivity for devices and applications with low mobility and low data transmission such as sensors and smart meters that are part of the IoT. Therefore, LPWAN will bring a new choice for IoT data transmission, developed to

meet the goals of low energy consumption, extended operating time of terminal devices, and the ability to transmit with distance up to tens of kilometers.

LoRaWAN is a low power, wide area network (LPWAN) protocol developed by the LoRa Alliance, 'active' wireless connectivity to the internet in regional, national or global networks, targeting the requirements of the Internet of Things (IoT) such as two-way communications, end-to-end security services, mobility, and localization.

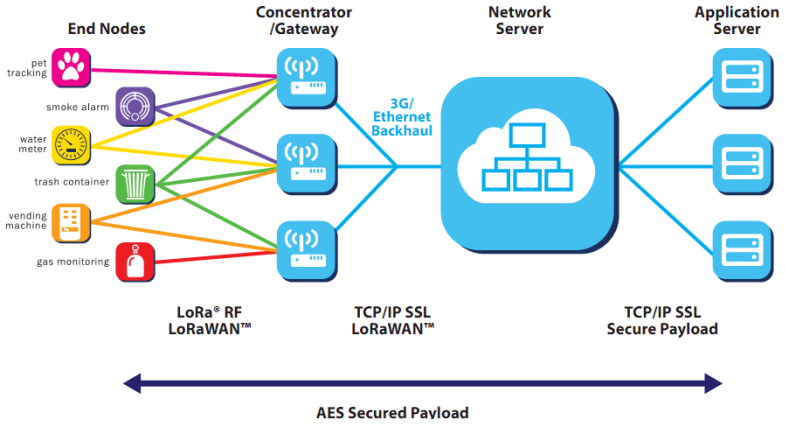


Figure 1.3: LoRaWAN system architecture

Components of the LoRaWAN network include:

End Devices (end devices) supporting LoRaWAN: Is a sensor or actuator that is wirelessly connected to the LoRaWAN network through gateways using LoRa modulation technology. These devices are mostly battery operated and perform functions of digitizing physical or environmental information such as: street lighting, door locking, water valve shutoff, leak prevention...

Gateway LoRaWAN (LoRaWAN gateway): Receives LoRa-modulated RF data from end devices and forwards this data to the server in the LoRaWAN network. Sensors are connected to the gateway through the IP backbone network, especially the same sensor can send data to multiple gateways as long as there is a connection between them. This greatly reduces the likelihood of packet errors (as the likelihood that at least one gateway will receive the notification is very high) and also reduces battery costs for mobile location-capable sensors.

Network server: Manages the entire network system, appropriate parameters to adjust the system and establish a secure 128-bit AES connection to transmit and control data. The network server ensures the authenticity of every sensor on the network and the integrity of messages, but cannot see or access application data.

Application servers: Responsible for securely processing, managing and interpreting data received from sensors, and creating a downlink payloads to endpoint devices.

1.2.3 HTTP connection protocol

HTTP (HyperText Transfer Protocol) is a hypertext transfer protocol, HTTP protocol is based on TCP/IP protocol, it allows to retrieve resources such as HTML documents, text, videos, images...

HTTP is the platform used to exchange data for applications following the Client/Server model. The requests (request or HTTP Request) created by the client are sent to the http server, the server sends back data (response or HTTP Response) for the client to receive. Requests and responses are messages with a simple structure, collectively called HTTP Messages.

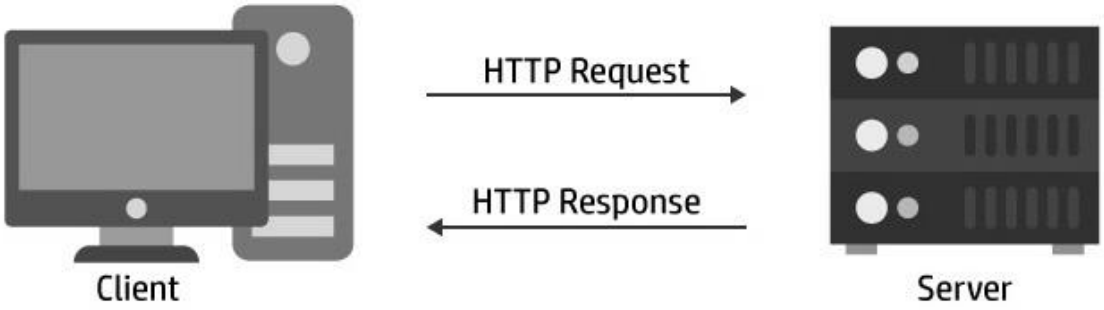


Figure 1.4: HTTP protocol operating model

1.3 HARDWARE

1.3.1 ESP8266

The ESP8266 NodeMCU Lua Wifi transceiver RF set uses the latest, most stable UART CP2102 download and transmission chip, capable of self-identifying Drivers on all Windows and Linux operating systems. Used for applications that need to connect, collect data and control via Wifi, especially applications related to IoT.

Figure 1.5: ESP8266 NodeMCU Lua CP2102

Specifications:

- Communication input port: Micro || Type-C
- Compatible with wifi standards: 802.11 b/g/n

- Support: Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated frequency multiplier, voltage stabilizer, DCXO and power management units
- +25.dBm output power in 802.11b mode
- Power down leakage current of <10uA
- Integrated low power 32-bit CPU can be used as application processor
- SDIO 1.1/2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Current consumption in Standby Mode < 1.0mW (DTIM3)

1.3.2 Arduino UNO R3

The Arduino board uses Atmel's 8-bit mega AVR microprocessor line with the two most popular chips, ATmega328 and ATmega2560. These microprocessor lines allow programming of complex control applications because they are equipped with powerful configurations with types of ROM, RAM and Flash memory, digital I/O inputs and outputs, including many capable inputs PWM signal output, analog signal reading ports and diverse communication standards such as UART, SPI, TWI (I2C).

Arduino Uno R3 DIP is applied to simple circuits such as light sensor circuits that turn lights on and off, motor control circuits, etc.

Figure 1.6: Arduino Uno R3

Specifications:

- Microcontroller: ATmega328 - 8bit
- Operating voltage: 5~12V DC (recommended)
- Operating frequency: 16 MHz
- Current consumption: About 30mA
- Limit input voltage: 19V DC
- Number of Digital I/O pins: 14 (6 PWM pins)
- Number of Analog pins: 6 (10bit resolution)
- Maximum current per I/O pin: 30 mA
- Maximum output current (5V): 500 mA
- Maximum output current (3.3V): 50 mA
- Flash memory: 32 KB (ATmega328) with 0.5KB used by bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Weight: 25 grams

1.3.3 Data transceiver module

1.3.3.1 *RF transceiver module NRF24L01*

The NRF24L01 transceiver module operates on the 2.4GHz frequency band and uses SPI communication, the maximum distance in unobstructed conditions is up to 100m.

Figure 1.7: RF transceiver module NRF24L01

Specifications:

- Operating voltage: 1.9V – 3.6V
- 2.4GHz ceramic antenna available.

- Can transmit 100m in open environment with transmission speed of 250kbps.
- Data transmission speed via radio waves: 250kbps to 2Mbps.
- Automatic handshake (Auto Acknow).
- Automatically retransmit when there is an error (auto Re-Transmit).
- Multi-function set – 6 data tubes.
- Separate data buffer for each transmit and receive channel: 32 Byte TX and RX FIFO separation.
- All IO pins can withstand 5V input voltage.
- Programmable transmission channels between 2400 MHz and 2525 MHz (125 channels selectable).
- Communication pin order: GND, VCC, CS, CSN, SCK, MOSI, MISO, IQR.

1.3.3.2 RF transceiver circuit NRF24L01 separate antenna

RF transceiver circuit NRF24L01 PA LNA 2.4Ghz separate antenna (with transceiver power amplifier) uses the main RF IC NRF24L01 + from Nordic, designed with additional PA (power amplifier) and LNA (Low Noise Amplifier) parts to be able to increased capacity and much longer transceiver distance (in ideal conditions, the manufacturer claims it can transmit 1000m).

Figure 1.8: RF transceiver circuit NRF24L01 separate antenna

Specifications:

- Main IC: NRF24L01
- Supply voltage: 3.3VDC
- GPIO communication voltage: 3.3VDC, when communicating with 5VDC circuit boards, it is necessary to connect the resistor in series or use voltage level conversion circuits.
- Communication: SPI

- Current consumption: 45mA
- Wave frequency: 2.4Ghz
- Use the same as NRF24L01 without amplification and can communicate with modules without PA and LNA amplification.
- Integrated power amplifier PA (power amplifier) and LNA (Low Noise Amplifier)
- Transceiver power: 20dBm
- Maximum transmission and reception speed: 2Mbit/s
- Standard 2×8 pins similar to NRF24L01 circuits without amplification.

1.3.3.3 RFID transceiver module

UART RDM6300 RF 125kHz RFID transceiver module is compactly designed and comes with RFID antenna, transmitting and receiving circuit via UART communication standard with fixed Baudrate parameters: 9600, N, 8, 1.

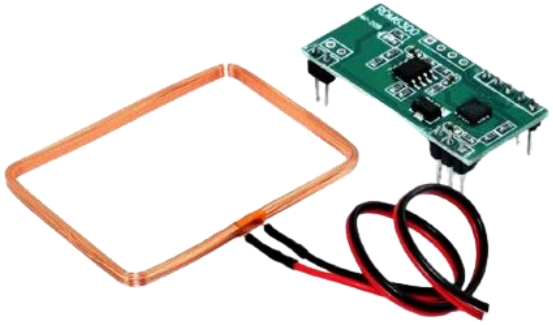


Figure 1 9: RFID transceiver module RDM6300

Specifications:

- Operating voltage: 5VDC
- Frequency: 125Khz
- Transmission speed: 9600
- Communication type: RS232 TTL format
- Working current: <50mA
- Detection distance: 20~50mm (depending on antenna, tag and surrounding environment)
- Operating temperature: -10 ~ 70 degrees Celsius
- Storage temperature: -20 ~ 80 degrees Celsius
- Coil size: 46 x 32 x 3mm

- Module size: 38.5 x 19 x 9mm
- Weight: 10g

1.3.4 Sensor module

1.3.4.1 *Rain sensors module*

Rain sensors are used to detect water levels, rain, or environments with water. The rain sensor circuit is placed outdoors to check if it is raining, thereby transmitting a control signal to turn on or off the relay.

Figure 1.10: Rain sensor module

Specifications

- Voltage: 5V
- Power indicator LED (Blue)
- Rain warning LED (Red)
- Works based on the principle: Water falling on the board will create an electrically conductive environment.
- There are 2 types of signals: Analog (AO) and Digital (DO)
- Signal format: TTL, 100mA output (Can use directly Relay, Small capacity buzzer...)
- Adjust sensitivity using potentiometer.
- Use LM358 to switch AO –> DO
- Size: 5.4*4.0mm
- 1.6mm thick

1.3.4.2 DHT11 sensor module

The DHT11 is a very popular sensor today because it is cheap and easy to get data through a one-wire communication (a single digital wire communication transmits data).

The sensor has a built-in signal preprocessor that helps receive accurate data without having to perform any calculations.

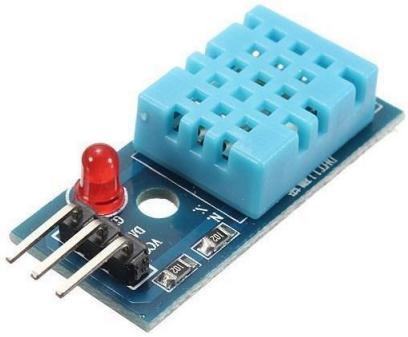


Figure 1.11: DHTT11 sensor module

Specifications:

- Operating voltage: 3V - 5V (DC)
- Operating humidity range: 20% - 90% RH, error ±5%RH
- Operating temperature range: 0°C ~ 50°C, error ±2°C
- Maximum transmission distance: 20m

1.3.4.3 Soil moisture sensor module

Soil moisture sensors are also known as soil moisture meters. It is mainly used for measuring the volumetric water content of soil, monitoring soil moisture, agricultural irrigation and forestry protection.

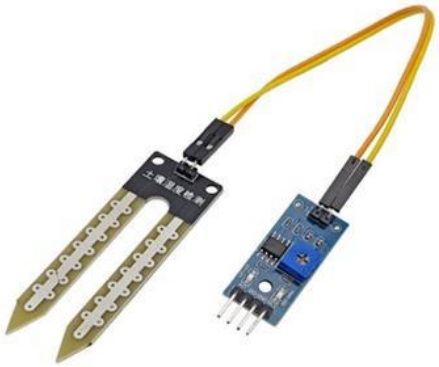


Figure 1.12: Soil moisture sensor module

Specifications:

- Operating voltage: 3.3V-5V17
- PCB size: 3cm * 1.6cm
- Red LED indicates input power, green LED indicates humidity.
- Comparator IC: LM393

- VCC: 3.3V-5V
- GND: 0V
- D0: Digital signal output
- A0: Analog output

1.3.4.4 Light sensor module

The light sensor module is an electronic component whose resistance changes and decreases with incident light. The photoconductor is made of a high-impedance semiconductor and has no junction. In the dark, the photoconductor has a resistance of several MΩ. When there is light, the resistance drops to a few hundred Ω. Accuracy: ±5%RH and ±2°.

The operation of photovoltaics is based on the photoelectric effect in the mass of matter. When a photon with enough energy hits it, it knocks electrons out of the molecule, becomes free in the substance, and causes the semiconductor to become conductive. The degree of conductivity depends on the number of photons absorbed.

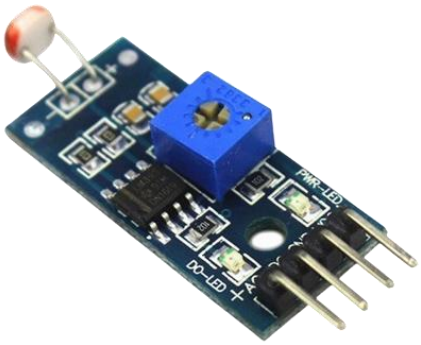


Figure 1.13: Light sensor module

Specifications:

- Operating voltage: 3.3V-5V
- PCB size: 3cm * 1.6cm
- Green LED indicates power and light
- Comparator IC: LM393
- VCC: 3.3V-5V
- GND: 0V
- DO: Digital signal output (0 and 1)
- AO: Analog output (0 and 1)

1.3.4.5 Proximity sensor module

This is a type of proximity metal sensor with the ability to detect metal at close range. Proximity sensor is highly accurate, compact and easy to install and adjust. This proximity sensor is often widely used in machine tools, metallurgy, chemicals, textiles, printing...Highly applied in the fields of automation, mechatronics, determining mechanical control sliding structure.

Figure 1.14: Proximity sensor module

Specifications:

- Model: SN04-N
- Wire Type: DC 3 wire
- Dimensions: 18 x 18 x36mm
- Output status: NPN is normally open with 3 wires
- Operating voltage: 10-30VDC
- Output current: 300mA
- Detection distance: 4mm
- Detection object: Metal
- Conversion frequency: 200Hz
- Materials: ABS plastic
- Cord length: 1.4m
- Weight: 60g

1.3.5 Motor module

1.3.5.1 L298N motor control module

The motor control module uses the L298N control IC to control 2 DC motors or 1 4-phase stepper motor, which can be controlled with a peak current of 2A. IC L298N

is attached to diodes on the board to help protect the microprocessor against induced currents from starting or shutting down the motor.

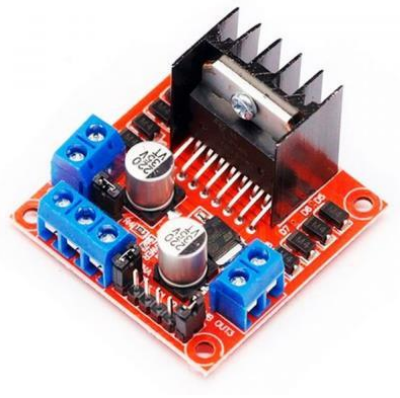


Figure 1.15: L298N motor control module

Specifications:

- Driver: L298N integrates two H-bridge circuits.
- Control voltage: +5 V ~ +12 V
- Maximum current for each H-bridge is: 2A (=>2A for each motor)
- Voltage of control signal: +5 V ~ +7 V
- Current of control signal: 0 ~ 36mA
- Power loss: 20W (when temperature T = 75 ℃)

1.3.5.2 LM2596 voltage regulator circuit

The compact LM2596 3A DC voltage reducer circuit is capable of reducing voltage from 30V to 1.5V while still achieving high efficiency (92%). Suitable for power sharing and voltage-lowering applications, it supplies devices such as cameras, motors, robots.

Figure 1.16: LM2596 voltage regulator circuit

Specifications:

- Input voltage: From 3V to 30V.

- Output voltage: Adjustable from 1.5V to 30V.
- Maximum response current is 3A.
- Efficiency: 92%
- Power: 15W
- Dimensions: 45 (length) * 20 (width) * 14 (height)

1.3.5.3 DC Gearmotor

The 12V gear reduction motor uses a powerful 399 brush motor, the motor wire material is completely made of copper, 407 steel sheets, nickel contact rings, strong magnetic magnets for the motor to have strong and durable performance. The planetary metal gearbox has a compact gearbox structure and is meticulously machined, providing large torque and high efficiency, powerful transmission, hard to wear and long life. Suitable for use in machinery serving production and services, mechanical transmissions, large robots.

Figure 1.17: DC Gearmotor GA25 399

Specifications:

- Engine: 3530 engine
- Motor diameter: 32.8 MM
- Gearbox diameter: 37MM
- Total height: 50.6 MM (without carrying)
- Output shaft: 6 MM (flat position is 5 MM)
- Output shaft length: 17.5 MM
- Weight: 174 g
- Voltage: 12 V
- Speed: 399 RPM

1.3.5.4 Honeycomb power supply

The 12V 15A honeycomb power supply is a DC power source with small size, high performance, medium cost, portability, stability and quality. Has the function of overvoltage protection, short circuit protection, overcurrent protection.

Figure 1.18: Honeycomb power supply

Specifications:

- Product size: 198*98*H42mm
- Input voltage range: AC180V-240V
- Input frequency: 50-60Hz
- Output voltage: 12VDC adjustable 5%
- Maximum output current: 15A.
- Efficiency: > 80%
- Overload protection: 25% of rated value
- Capacity: 180W
- Operating temperature: 0 ℃ ~ 40 ℃, 10% ~ 90% RH
- Storage temperature: -20 ℃ ~ 85 ℃, 10% ~ 90%

1.3.5.5 Relay

Relay SRD-12VDC-SL-A is currently very popular in equipment switching circuits, compact contacts, high durability, fast contact opening and closing, large current. Relay is quite commonly used in motor control and lighting applications.

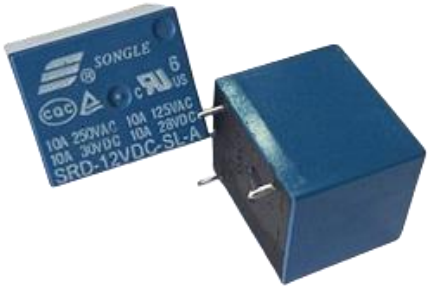


Figure 1.19: Relay

Specifications:

- Control voltage: 12V
- Maximum current: 10A
- Impact time: 10ms
- Brake release time: 5ms
- Operating temperature: -45oC ~ 75oC

1.3.5.6 Pump motor

R365 12VDC pump motor is a mini water pump motor designed with the ability to self-suction water, saving energy up to 3 liters/minute, so it is often used in non-contact hand washing machine mode, projects smart agriculture project, spraying disinfectant. system, evaporative cooling.

Figure 1.20: Pump motor

Specifications:

- Working voltage: 12VDC
- No-load current: 0.23A
- Flow: 2-3 liters/minute (12V)
- Output pressure: 1-2.5 kg
- Suction depth achieved: 1-2.5 meters
- Normal working life: 2-3 years
- Inlet and outlet diameter: outer diameter 8mm
- Weight: 111g

1.3.6 Other devices

1.3.6.1 LCD

The 2004 LCD text screen uses the HD44780 driver, is capable of displaying 4 lines with 20 characters each, the screen is highly durable, very popular, has many designs and is easy to use, suitable for beginners and workers.

Figure 1.21: LCD

Specifications:

- Operating voltage is 5 V
- Dimensions: 98 x 60 x 13.5 mm
- White text, green background
- The distance between the two connection pins is 0.1 inch, convenient when connecting to Breadboard.
- Pin names are written on the back of the LCD screen to support connection and wiring.
- There is a backlight LED, you can use a variable resistor or PWM to adjust the brightness to use less power.
- Can be controlled with 6 signal wires
- Has a built-in character set that supports English and Japanese

1.3.6.2 Mist nozzle

Misting nozzles are used to install into misting pipe systems with the main task of creating tiny water jets or spraying mist into the outside environment. Misting nozzles come in a variety of types, sizes, and materials, such as stainless steel, plastic, copper, etc., suitable for installation in many agricultural greenhouse systems, cafes,...

Figure 1.22: Mist nozzle

Specifications:

- Nozzle hole diameter: 0.1 mm, 0.15 mm, 0.2 mm, 0.3 mm, 0.4 mm, 0.5 mm, 0.6 mm, 0.7 mm,...
- Construction materials: stainless steel, copper, plastic
- Mist spray pressure: 20 - 70 kg/cm2
- - Water flow ml/min: 13 – 35 ml/min, 20 – 46 ml/min, 49 – 89 ml/min, 80 – 145 ml/min, 95 – 178 ml/min, 130 – 243 ml/min, 175 – 320 ml/min, 210 – 399 ml/min
- - Water flow l/h: 0.78 – 2.1 l/h, 1.2 – 2.76 l/h, 2.94 – 5.34 l/h, 4.8 – 8.7 l/h, 5.7 – 10.68 l/h, 7.8 – 14.58 l/h, 10.5 – 19.2 l/h, 12.6 – 23.94 l/h

1.3.6.3 Fan, light bulb

Fans have the effect of creating airflow and helping to reduce ambient temperature. They can be used to cool the air in hot, dry environments.

Light bulbs are mainly used to provide light in a room or any space that needs lighting.

Figure 1.23: Fan, light bulbs

Specifications

- Fan: 12V
- Light bulb: 220V

1.3.6.4 Aluminum, mica, anti-UV film

Aluminum is widely used in many fields, including the implementation of lightweight structures such as window and door frames.

Mica is often used in the electronics industry because of its electrical insulation and high temperature resistance. It is also used to make large models, etc.

Anti-UV film is often widely used in greenhouse systems and buildings to prevent heat from the sun, helping plants avoid severe sunburn.

Figure 1.24: Aluminum, mica, anti-UV film

1.4 SOFTWARE AND DATABASE

1.4.1 Software

1.4.1.1 *Arduino IDE*

- Arduino IDE is specially designed to develop programs for Arduino boards like Arduino Uno R3, ESP8266 and many other boards etc.
- Arduino IDE (Arduino Integrated Development Environment) is a text editor that helps write code and load it onto the board, making the program development and testing process convenient.

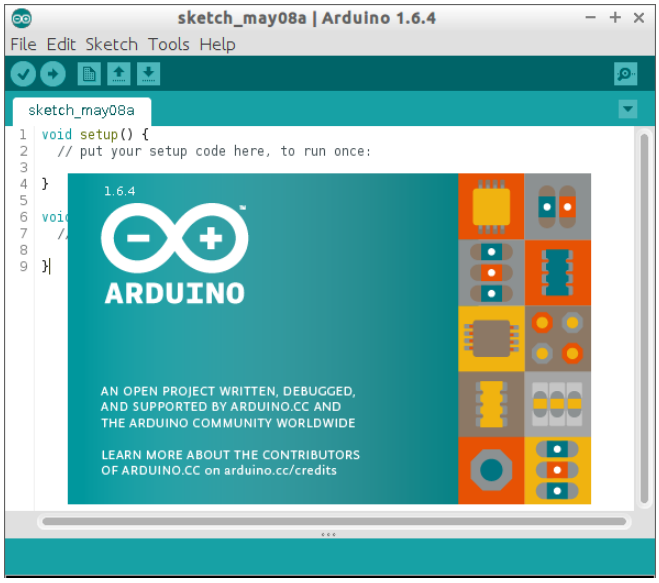


Figure 1.25: Arduino IDE software

1.4.1.2 Visual Studio Code

- Visual Studio Code is a lightweight and flexible IDE designed to support multiple programming languages.

- Visual Studio Code supports many different programming languages, including C++, C#, Java, Python, JavaScript, HTML, CSS and many others.

Figure 1.26: Visual Studio Code software

1.4.1.3 Altium Designer

- Used to design electronic circuits and specialize in drawing Schematic and PCB.
- Altium Designer is widely used in the electronics industry, from designing small circuits for consumer electronic products to complex industrial and medical systems.

Figure 1.27: Altium Designer software

1.4.1.4 Post man

- Postman is a computer application that helps develop, test, and interact with APIs (application programming interfaces).
- Postman allows users to create and send HTTP requests such as GET, POST, PUT, and DELETE to interact with APIs and test collections easily.
- Postman is available for many different operating systems, including Windows, macOS, and Linux.

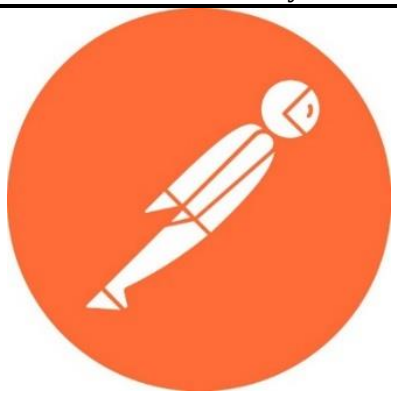


Figure 1.28: Postman Software

1.4.1.5 Outsystem

- Outsystem is a new language that uses a low-code platform, helping to create a fast and flexible development environment.
- Outsystem supports cross-platform integration, such as web and mobile, helping to reduce the implementation time and costs required for application development.
- Provides application performance monitoring and management tools, as well as error handling solutions.

Figure 1.29: Outsystem software

The most powerful and outstanding features of the Outsystem system:

- Integration Builder
  + Create connectors to establish seamless communication between your apps and external data sources.
  + Facilitate the integration of disparate systems, ensuring smooth data exchange.
- LifeTime
  + Oversee the entire lifecycle of your applications, from development to deployment.

+ Manage IT users, configure environment settings, and enforce security protocols.

- Service Center

+ Monitor and administer applications within each environment.

+ Handle error logs, manage web services, and optimize app performance using timers.

- Users

+ Access tools to view, update, or delete end users within your apps.

+ Effectively manage permission roles to control user access and permissions.

- App Feedback

+ Implement in-app feedback mechanisms to collect user suggestions and bug reports.

+ Enhance user engagement and gather valuable insights for continuous app improvement.

1.4.2 Database

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

Figure 1.30: MongoDB database

Here are some key features of MongoDB:

- NoSQL Database:

MongoDB is a NoSQL database, which means it does not use a relational model (like SQL) but instead uses more flexible structured or unstructured data storage mechanisms.

- Data is stored in a JSON-like format:

Data in MongoDB is stored as BSON (Binary JSON), a format similar to JSON, making it easy to read and understand.

- Schema-Less:

MongoDB is a schema-less database, which means you can insert data without first defining its structure.

- Indexing support:

  MongoDB supports indexes to optimize data searching and retrieval.

- Integrating Sharding and Replication:

  Provides data sharding to enhance scalability and supports replication processes to ensure data availability and security.

- Widely used in modern applications:

  MongoDB is commonly used in web and mobile applications and projects that need flexible and scalable data storage.

1.5 SUMMARY

In chapter 1, the main focus is on exploring the theoretical background related to the system. The main goal is to provide an understanding of greenhouse systems, while also providing insight into their hardware details and specifications. Additionally, this chapter introduces key theories related to software and databases, enriching the understanding of the comprehensive theoretical framework important for system design and implementation.

CHAPTER 2. SYSTEM DESIGN

2.1 NECESSARY REQUIREMENTS

- The system needs to have flexible interaction between hardware and software components to ensure flexibility in greenhouse operation and management.
- Ensure the system is designed and implemented according to safety standards, especially in the use of electrical components and equipment.
- The system needs to be able to integrate data from sensors combined with devices to optimize device management and control.
- Optimize energy use to reduce environmental impact and ensure stable energy sources.
- System design for easy maintenance and repair, reducing operating interruption time.
- Ensure strong connectivity with new technologies such as Outsystem and NodeJS to deploy and store database applications through MongoDB.
- The system needs remote connection and monitoring to manage and monitor it anywhere.
- Design applications and integration processes to ensure ease of use and integration with other systems.

2.2 SYSTEM STRUCTURE

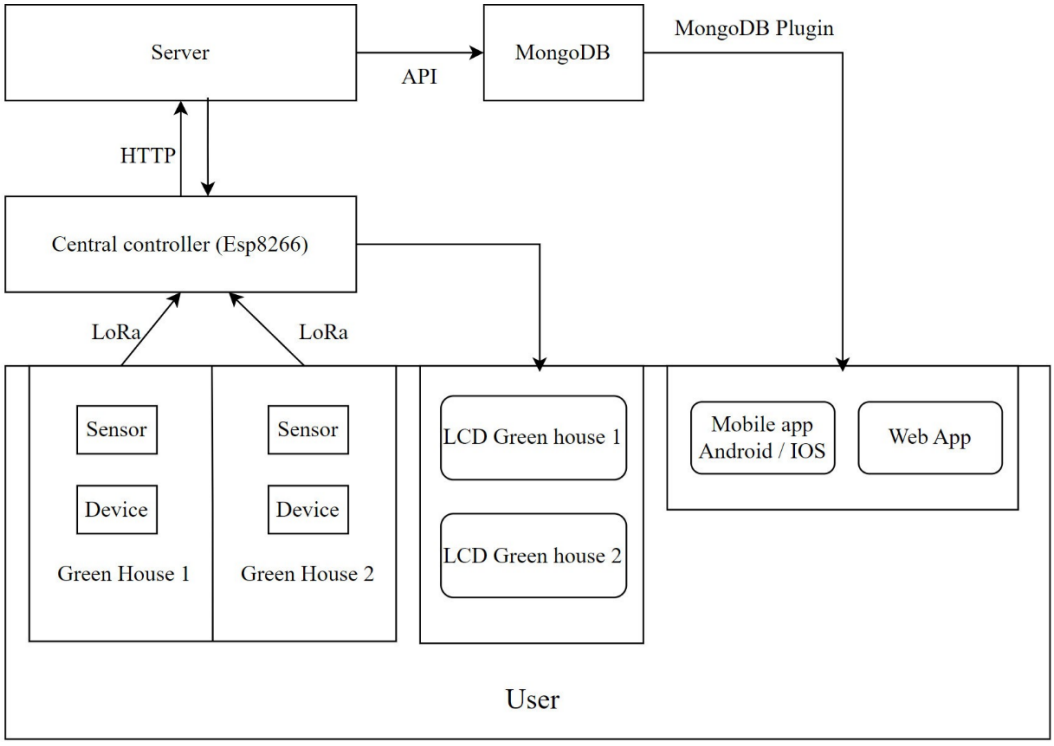


Figure 2.1: System structure

Description of activities: developed in two directions: automation and manual

The model applies design and implementation using advanced technology and is processed automatically to produce the most optimal and effective results. In addition, when unforeseen problems arise, they can be used manually to monitor and manage them immediately.

With automation mode:

The sensors receive signals and send them to the controller to process information.

- DHT11 temperature sensor measures temperature > 29 degrees Celsius when the fan runs, <29 degrees Celsius then the fan turns off.
- Photoresistive light sensor measures light > 60, the roof closes, <60, the roof opens.
- If the soil moisture sensor measures humidity < 50%, the faucet sprays; if it is > 50%, the faucet turns off.
- If the rain sensor measures rain > 510, it reports rain, if < 510, it doesn't rain.
- Light sensor gives digital signal =0 => when it is dark, the light will turn on, digital signal = 1 => when it is bright, the light will turn off.
- Arduino RFID UART 125kHz module gives digital signal = 0 => door opens, digital signal = 1 => door closes.
- Data values of each greenhouse system will be sent through the center using lora waves.

With manual mode:

- After the values of each greenhouse system are sent to the central device, the central device will receive those values and use the ESP8266 to connect to wifi and read the received sensor values. Before sending data to the Server, checking the connection and sending data, the program will check to see if it can connect to the Server or not. If the connection is successful, data will be sent to the Server via HTTP POST request.
- The server here will take on the role of intermediary to receive data from the ESP82266 and save it to the MongoDB database to store system data. At the same time, after successfully sending to the server, data for each house will also be sent and displayed on the LCD screen.
- Android or iOS Apps, Web apps will use available libraries provided from the Outsystem system, supporting connection to the MongoDB database. Once fully

setup, it will be possible to connect to the Outsystem. Proceed to retrieve MongoDB data values and proceed to build a monitoring application, collecting statistics as well as updated values every 30 seconds.

2.3 ALGORITHM FLOWCHARTS

2.3.1 Alogrithm flowchart for central controller

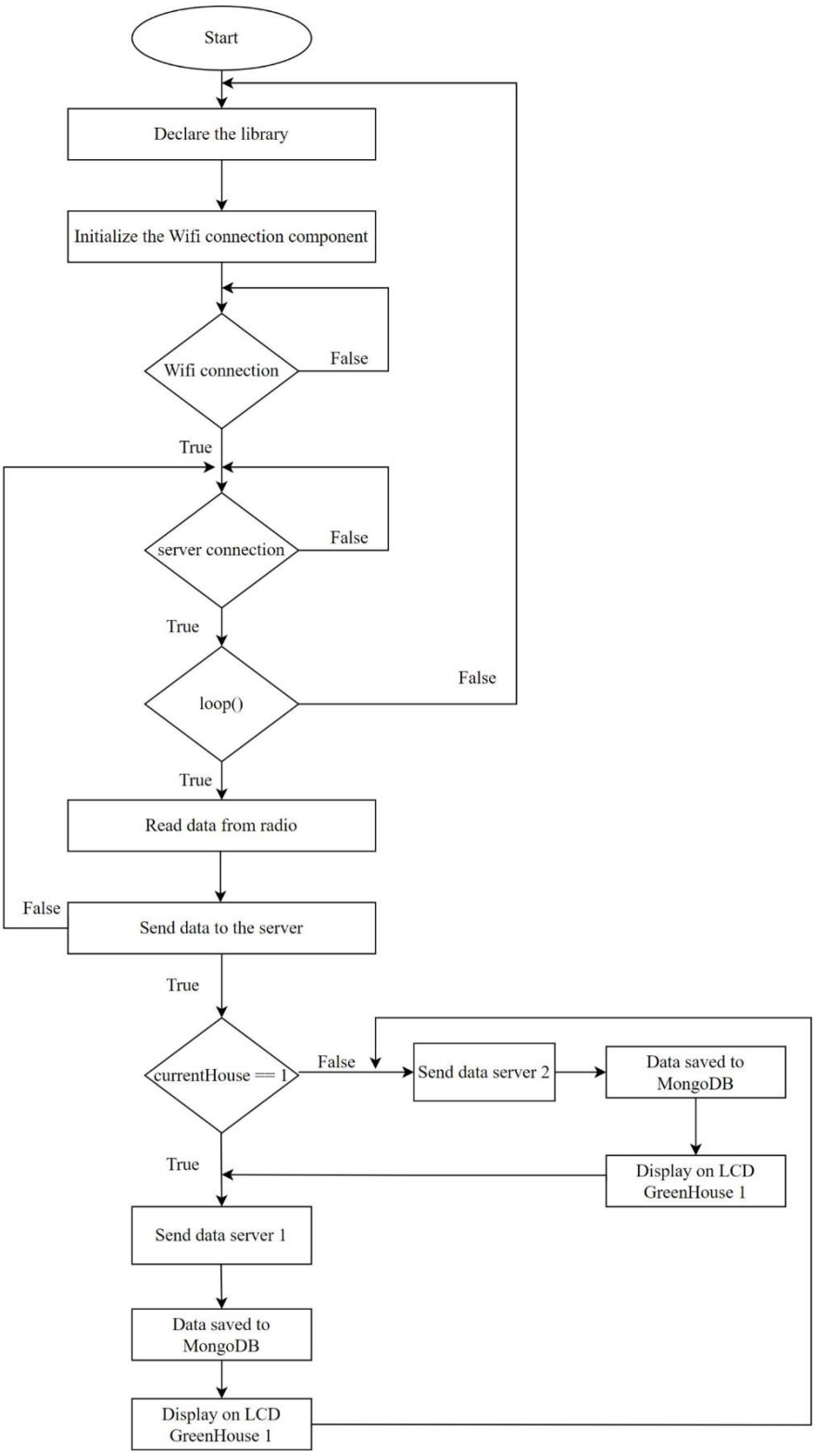


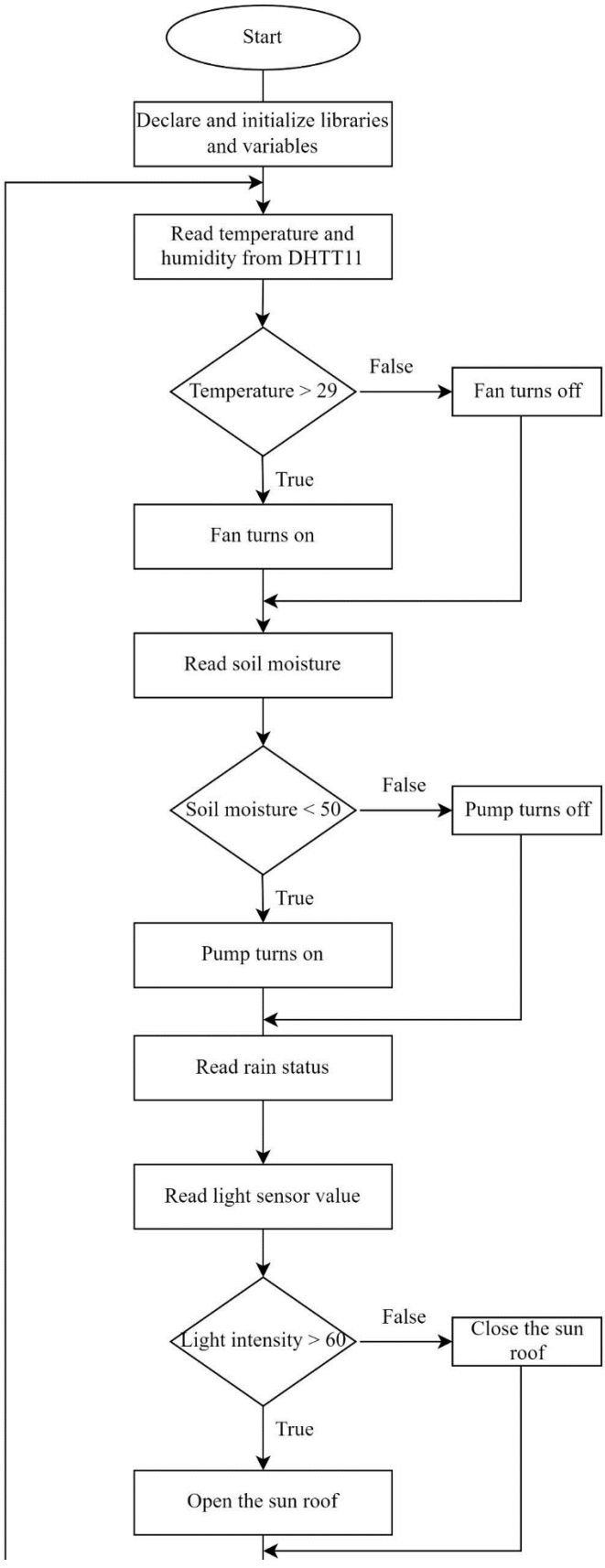
Figure 2.2: Algorithm flowchart for central controller

Activity description:

- First, declare the library and initialize necessary variables such as WiFi, radio, information LCD and variables measuring values from sensors.
- Set the address and configure the WiFi connection, wait until the connection is successful. During each cycle, check the connection status to WiFi and Server. If the connection is lost, try again to connect successfully.
- In the loop() function there will be an infinite loop, this loop will check the connection status to WiFi and Server. If the connection is lost, try again when the connection is successful. Once connected, it will read radio data sent from the greenhouse nodes to receive sensor values and status of devices in the greenhouse. After receiving the data, package it into a JSON string and send it to the Server corresponding to each greenhouse.
- Next, the ESP8266 reads the values from the sensors and the status of the devices received from the communication nodes through the greenhouse 1 and greenhouse 2 systems are sent by string characters.
- After receiving the value of each house, it will transmit the data and package the read data into a Json string. If an error is sent, the error recovery procedure is performed, which includes closing the connection and retrying the connection to the Server. If submitted successfully, testing will begin. There are 2 servers here storing information of 2 greenhouses. Check the appropriate value of the data for each house and if the data is in greenhouse 1, the information will be displayed and sent to the Server for greenhouse 1 and vice versa for greenhouse 2.
- If all data of greenhouse 1 and greenhouse 2 are sent successfully, the server will be updated and the data will be saved to the corresponding collection in MongoDB. If there is an error, it will return to the beginning to connect.

2.3.2 Algorithm flowchart for greenhouses

- Algorithm flowchart for greenhouse 1



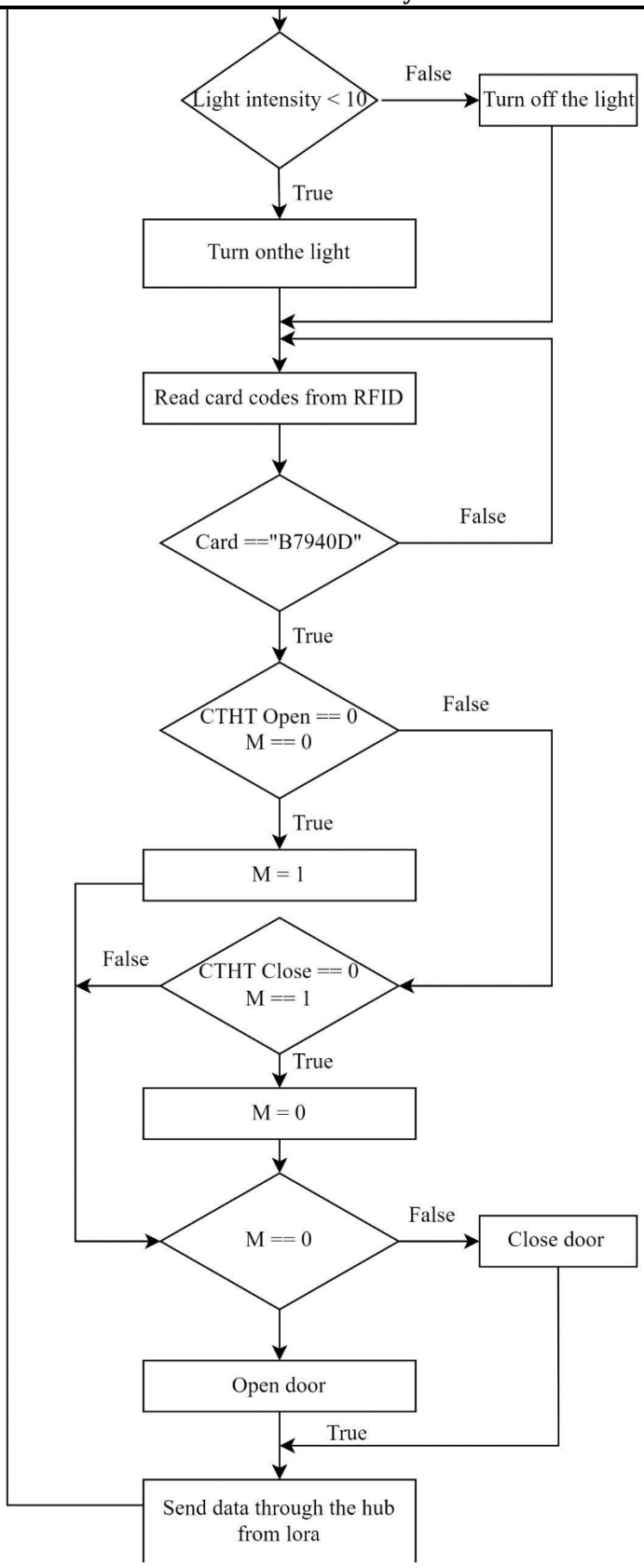
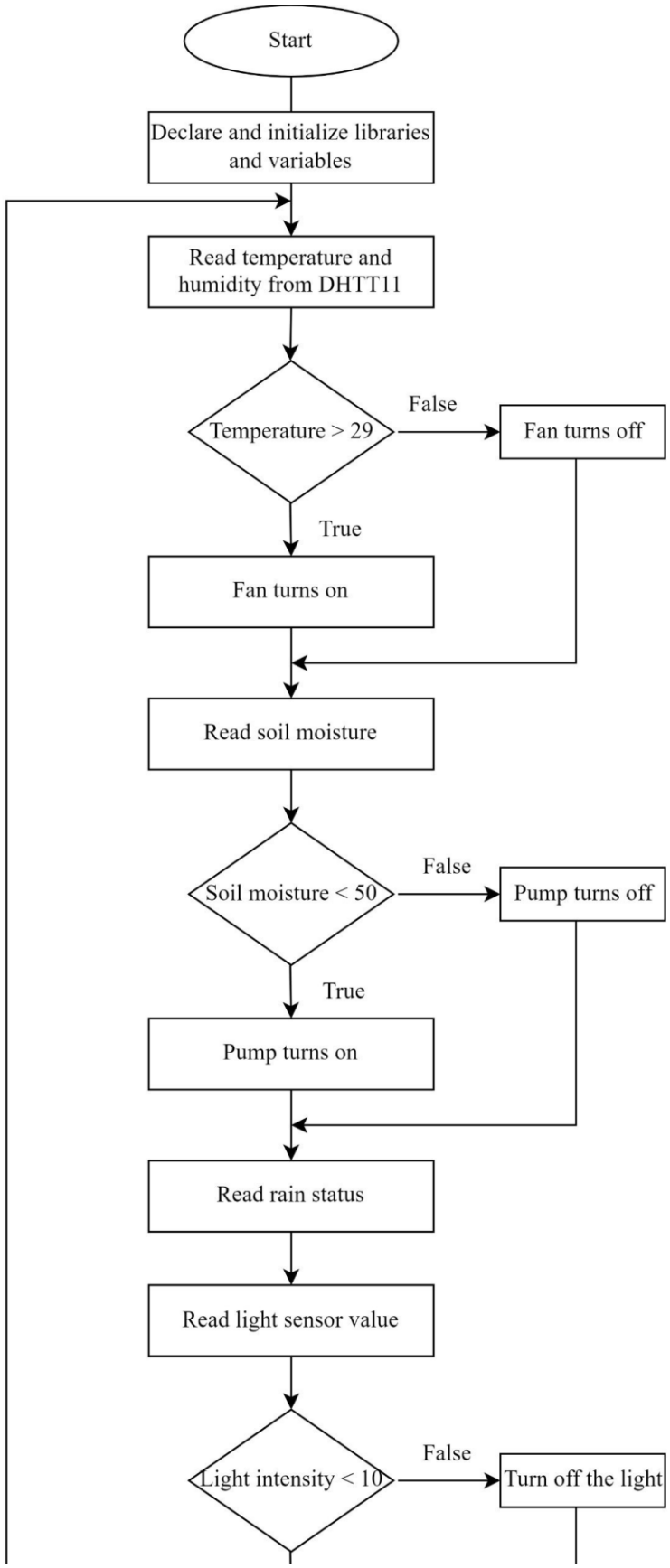


Figure 2.3: ***Algorithm flowchart for greenhouse 1***

- Algorithm flowchart for greenhouse 2



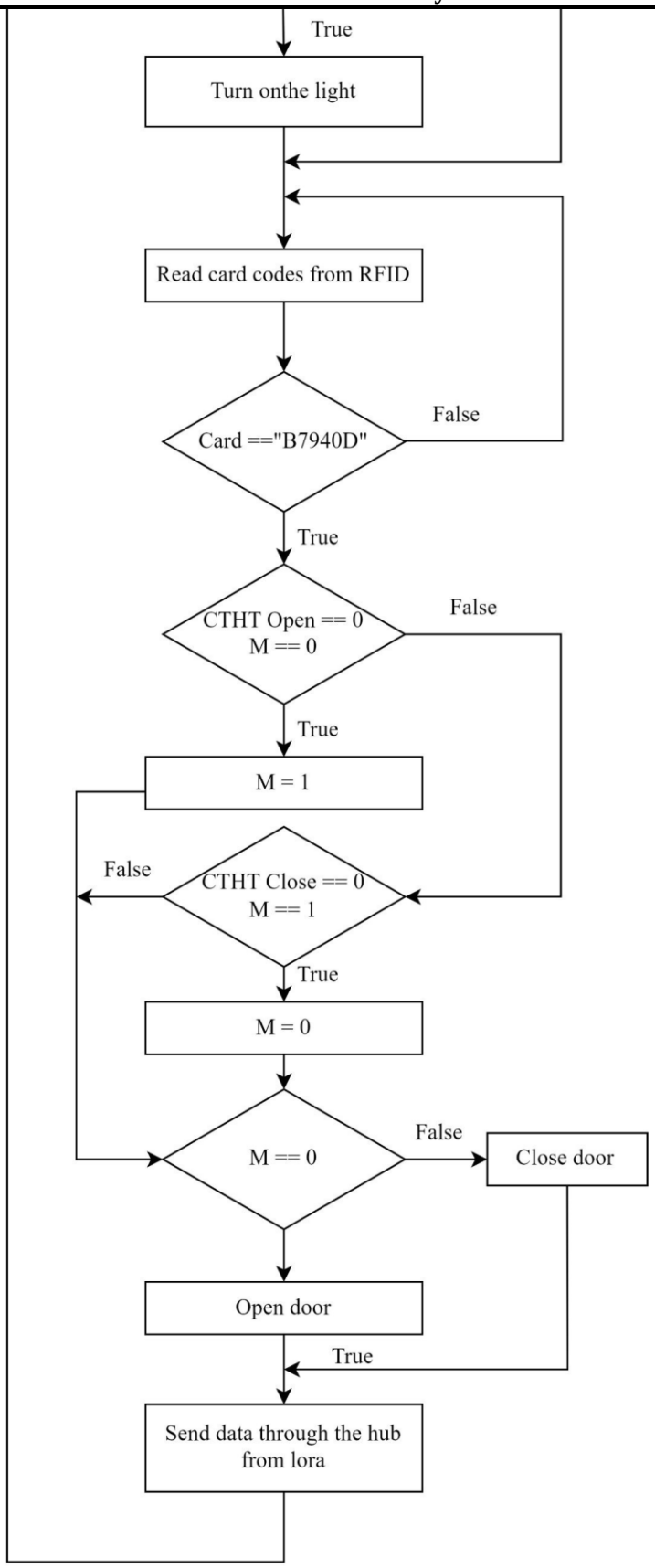


Figure 2.4: Algorithm flowchart for greenhouse 2

Activity description:

- Declare libraries, declare and initialize SPI, nRF24L01, RF24, DHT to connect and read data from devices. Declare and initialize variables and sensor pins, lights, fans, pump, RF connection ports, LoRa connection ports. Set the address for the LoRa module and start it as a transfer. Set pins and output modes for devices such as motors, electric lights, fans, pump.
- Read the value of the DHT11 temperature sensor to measure temperature > 29 degrees Celsius, then the fan runs, <29 degrees Celsius, then the fan turns off.
- Read soil moisture sensor data value, humidity < 50% nozzle, > 50% nozzle off.
- Read the rain sensor value for digital signal = 0 => when it is dark the light will turn on, digital signal = 1 => when it is light the light will turn off.
- Read the optical light sensor data value. If the light is > 80, the roof can be closed comfortably. If it is < 80, the roof is open.
- Light sensor gives digital signal = 0 => when it is dark the light will turn on, digital signal = 1 => when it is light the light will turn off.
- Read the RFID value signal module, if the string “B7940D” appears in the read data, the value will be returned. Here if variable M == 0 and door open switch == 0 then M is assigned to 1. On the contrary, if M == 1 and door close interrupt rule == 0 then M is assigned to 0. Make sure the door opens and closes alternately when there is a signal from the RFID module.
- After reading the value data of each greenhouse system, it will be sent to the center by lora waves in string form for the center to receive.

2.4 SYSTEM DESIGN

2.4.1 Greenhouse block principle circuit

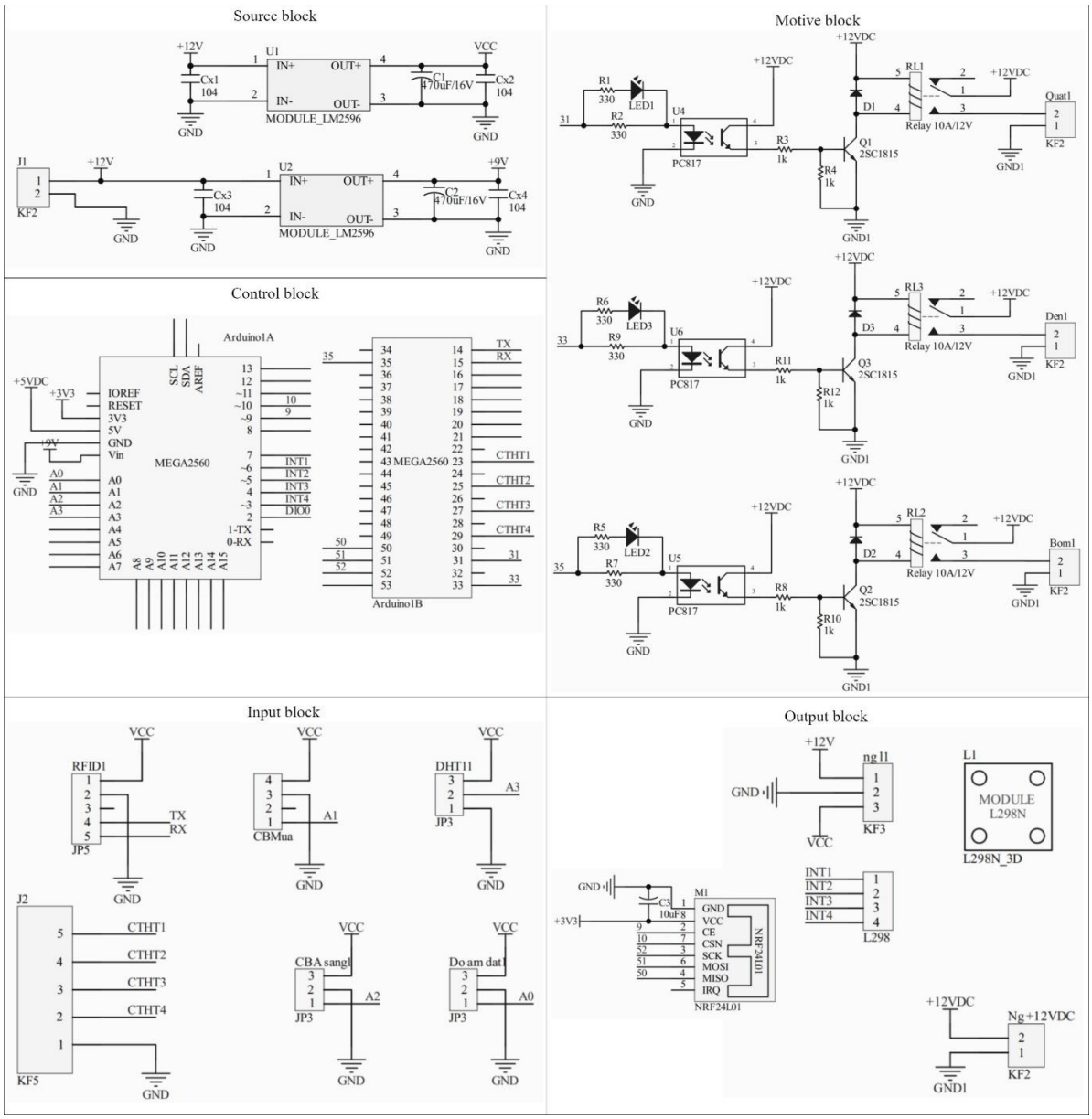


Figure 2.5: Greenhouse block principle circuit

- **Source block**

Configure the power parameters for the circuit, defining important parameters such as voltage, current, and other properties of the power source. In this case, the power block is used to integrate the LM2596 module, which reduces voltage and stabilizes the current in the circuit.

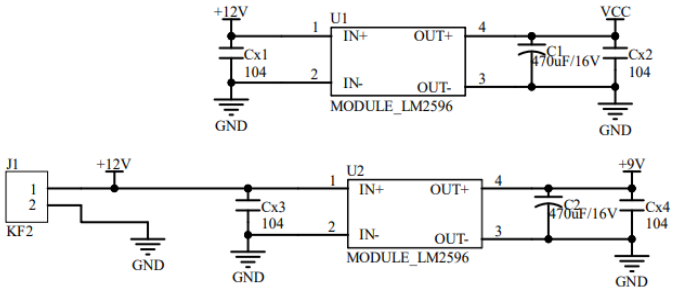


Figure 2.6: Greenhouse source block principle circuit

- **Input block**

Responsible for collecting data from the environment and converting it into control signals. The input block includes input data such as RFID, rain sensor, DHT11, light sensor, soil moisture sensor, and limit switch for the door system.

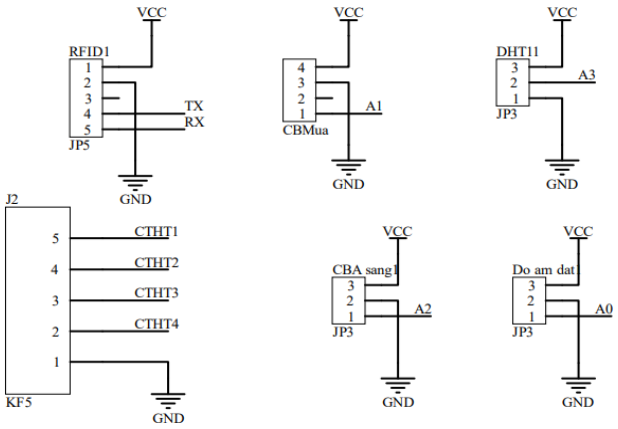


Figure 2.7: Greenhouse input block principle circuit

- **Control block**

Performs important functions in controlling and managing an automation system. Includes wireless communication, sensor management. Use control logic to perform operations such as turning on/off LEDs, lights, fans, doors, roofs, and send control commands to the motive block to initiate control actions.

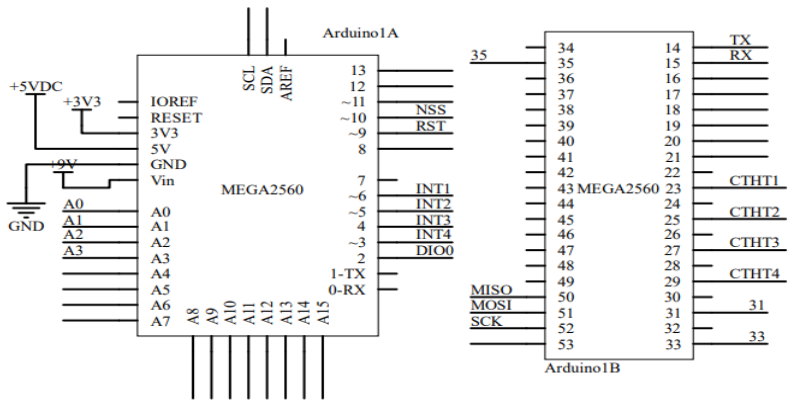


Figure 2.8: Greenhouse control block principle circuit

- **Motive block**

In the output block, receive data from the greenhouse control block and perform the corresponding control logic actions. Use realay to control fans, lights and pumps. If the value is 1 then the device will be off and if the value is 2 then the device will be on.

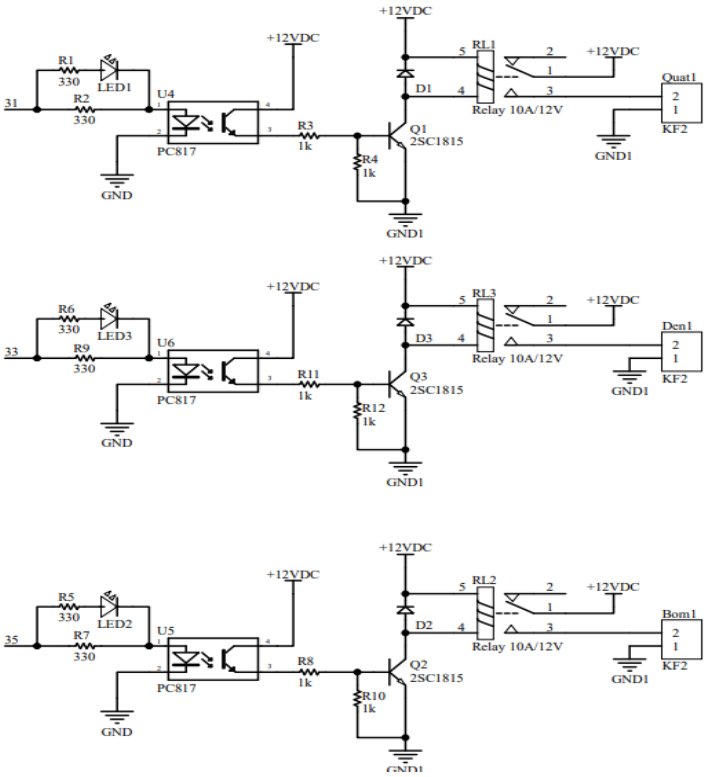


Figure 2. 9 Greenhouse motive block principle circuit

- **Output block**

In the output block, receive data from the greenhouse control block and perform the corresponding control actions. Use the LN2983 module to control devices with large currents, such as doors and curtains. Send all sensor data and device status via LoRa Module SX1278 to the central device.

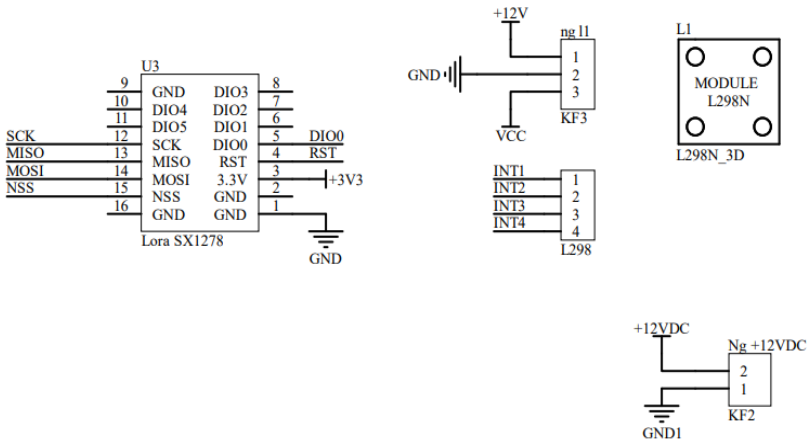


Figure 2.10: Greenhouse output block principle circuit

2.4.2 Central control block principle circuit

Use module nRF24L01 to receive data from the greenhouse in the greenhouse system. Then use ESP8266 to process the data and send data to the Server, updating information on the LCD screen corresponding to each greenhouse. In addition, use the LM2595 module to control the central on/off

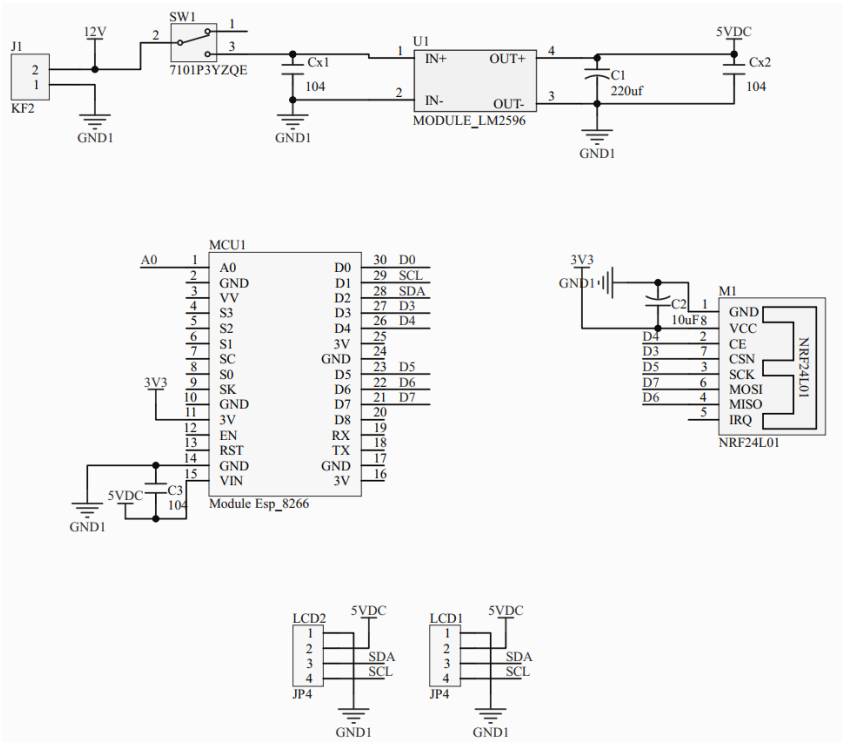


Figure 2.11: Central control block principle circuit

2.4.3 PCB circuit

- Greenhouse LoRa controller

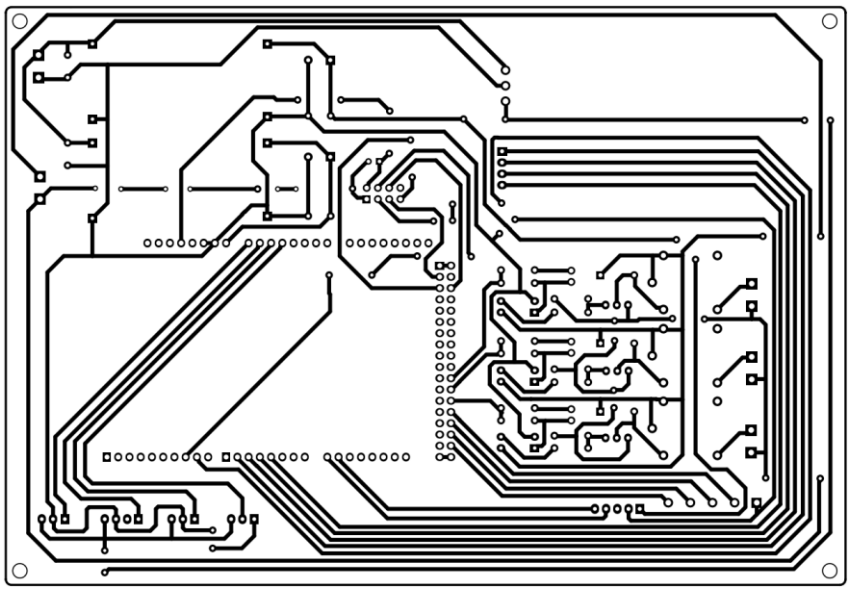


Figure 2.12: Greenhouse PCB circuit 1 and 2

- Greenhouse central controller

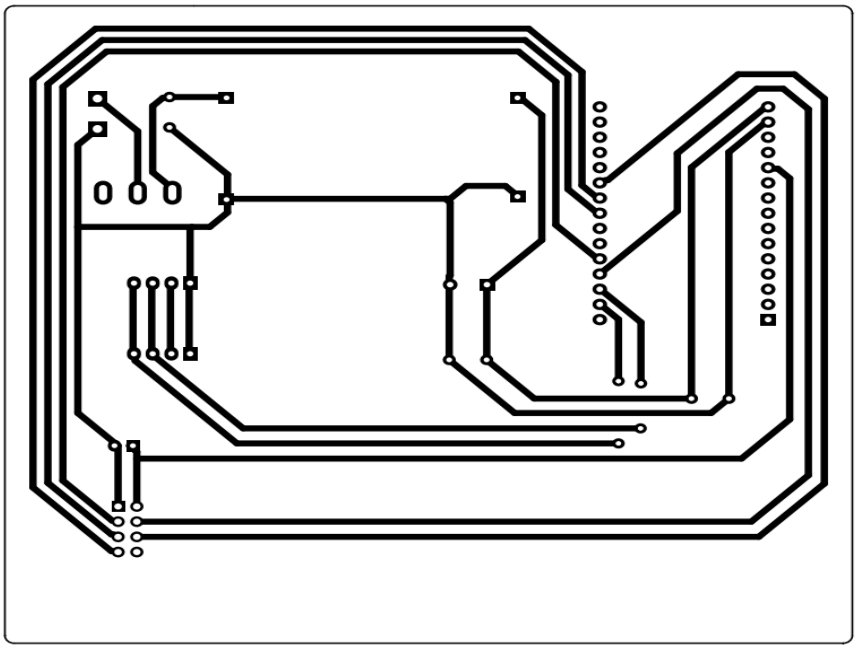


Figure 2.13: Greenhouse central system PCB circuit

2.5 DESIGN OF SERVER STRUCTURE

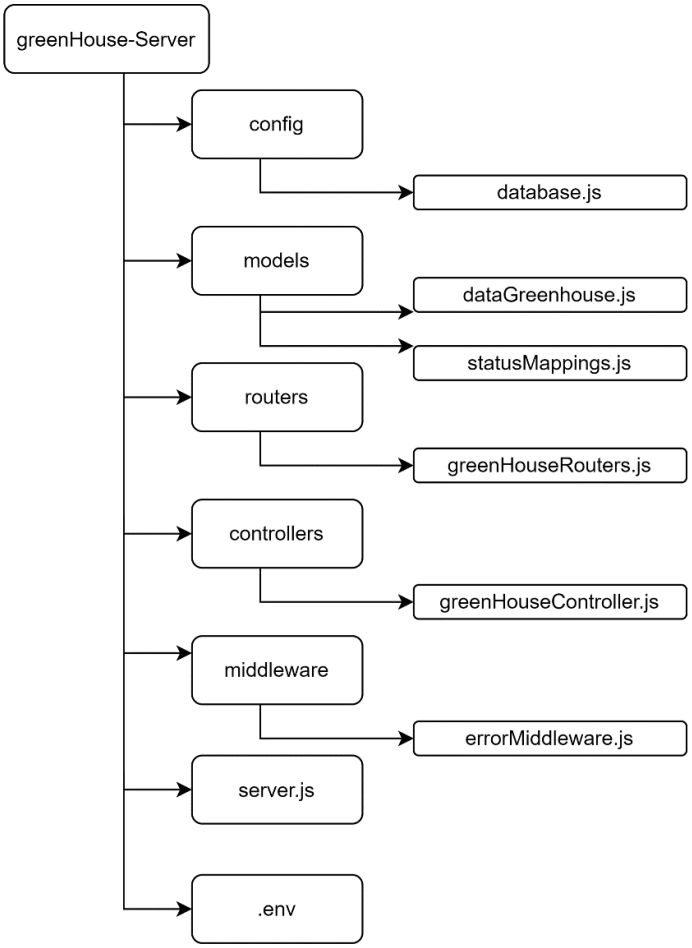


Figure 2.14: Server structure diagram

Structure description:

- config/database.js: Defines the connection configuration to the database, contains information about the database, and other settings.
- models/dataGreenhouse.js:Defines the data model for the "dataGreenhouse" object in the database and describes the structure and rules of the data the application will store and use.
- models/statusMappings.js: Defines mapping tables for statuses in the application and contains information about the states and maps them to corresponding values.
- routes/greenhouseRoutes.js: Defines API routes related to greenhouse-related operations such as getting data, adding data, updating status, and other operations.
- controllers/greenhouseController.js: Handles logic and interaction with the database based on API requests from routes and contains handler functions for API routes, performing operations such as querying the database, process input and output data.
- middlewares/errorMiddleware.js: Handle and navigate errors in the application. Contains middleware for error handling, logging, and proper response when an error occurs during request processing.
- server.js: Create and configure the server to listen for requests from clients. Includes server configuration, database connections, and necessary routes and middleware registration.
- .env: Contains environment variables for the application. Store sensitive information such as API keys MongoDB data connection account information.

2.6 SUMMARY

In Chapter 2, the focus is primarily on greenhouse system design. Clearly describe the system's operating structure and provide algorithm flowchart for the greenhouse and central controller. Each block in the greenhouse system is meticulously designed with every detail clearly described. Furthermore, this chapter advances the design of the server stucture required for efficient system deployment. This shows that chapter 2 provides a comprehensive overview of each design issue, with each component described clearly and meticulously.

CHAPTER 3. SYSTEM IMPLEMENTATION

3.1 MAIN STEPS TO BUILD THE SYSTEM

3.1.1 Hardware implementation

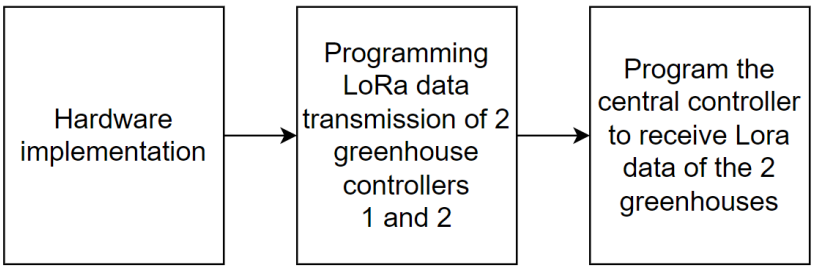


Figure 3.1: Hardware implementation steps

Description of deployment steps:

- Implement the construction of the theoretical circuit and PCB circuit.
- Program the hardware for the Arduino Uno boards of Greenhouse 1 and Greenhouse 2 to transmit sensor data and device information via LoRa waves.
- Program the hardware for the ESP8266 module in the central control unit to receive data from the two greenhouses.

3.1.2 Building Server

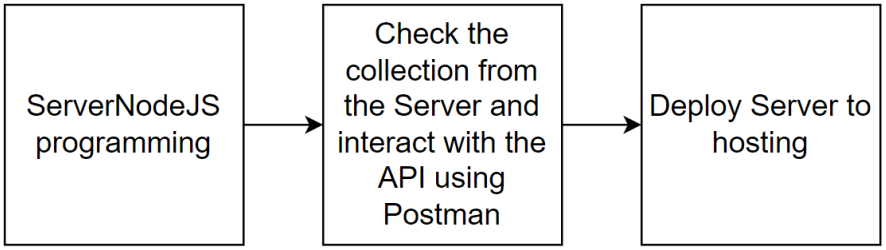


Figure 3.2: Steps to build a server

Description of deployment steps:

- Develop and program a NodeJS server to serve as an intermediary for data transmission.
- Deploy the server to hosting for easy accessibility, management, and real-time monitoring of data.
- Establish a connection between the server and the MongoDB database system to store greenhouse data.
- Test the server using Postman by performing POST and GET requests to the greenhouse's data collection through the server's URL.

3.1.3 Hardware and software connections

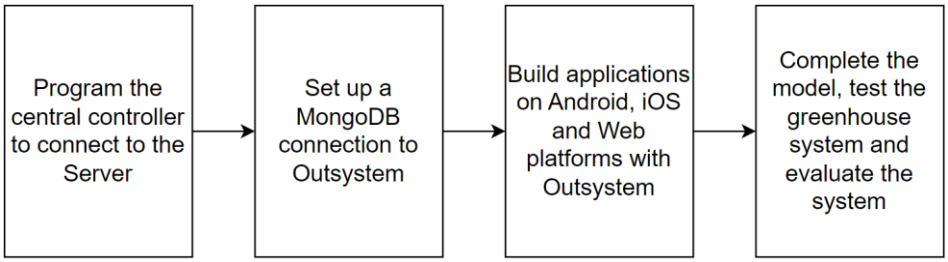


Figure 3.3: Steps to hardware and software connection

Description of deployment steps:

- Proceed to program the ESP8266 connection to send data to the server.
- Configure and set up external systems connected to the MongoDB database. Get data from 2 greenhouses corresponding to 2 collections in the MongoDB database with values from sensors and devices.
- Retrieve data from MongoDB databases for external systems and build Android, iOS, and web applications using the new OutSystems language.
- Complete the model, test, and evaluate the system.

3.2 HARDWARE PROGRAMMING

3.2.1 Hardware product

Through the process of deploying and implementing the hardware implementation of the system according to the circuit design, successful results have been achieved. Below is the hardware implementation result for the entire greenhouse system.

- Greenhouse circuit design results 1

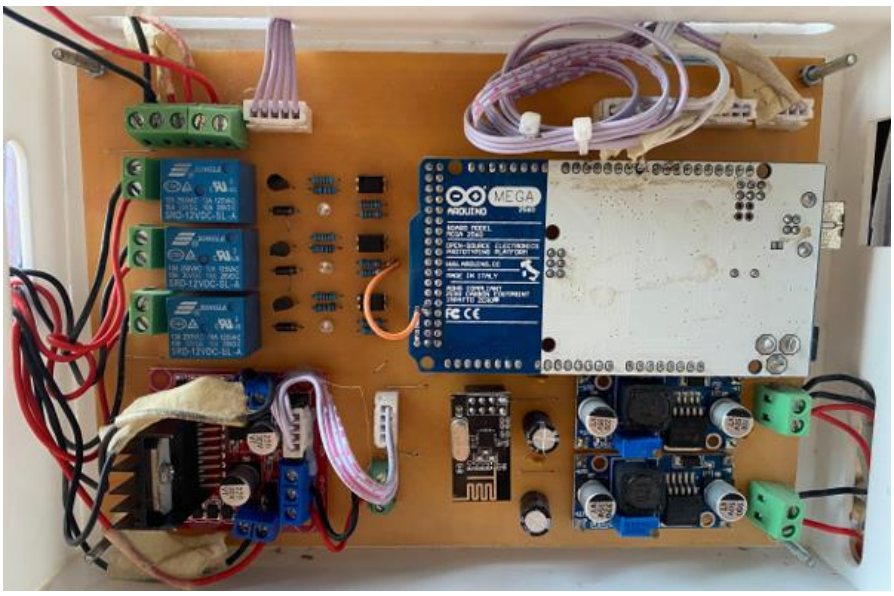


Figure 3.4: Greenhouse circuit 1

- Greenhouse circuit design results 2

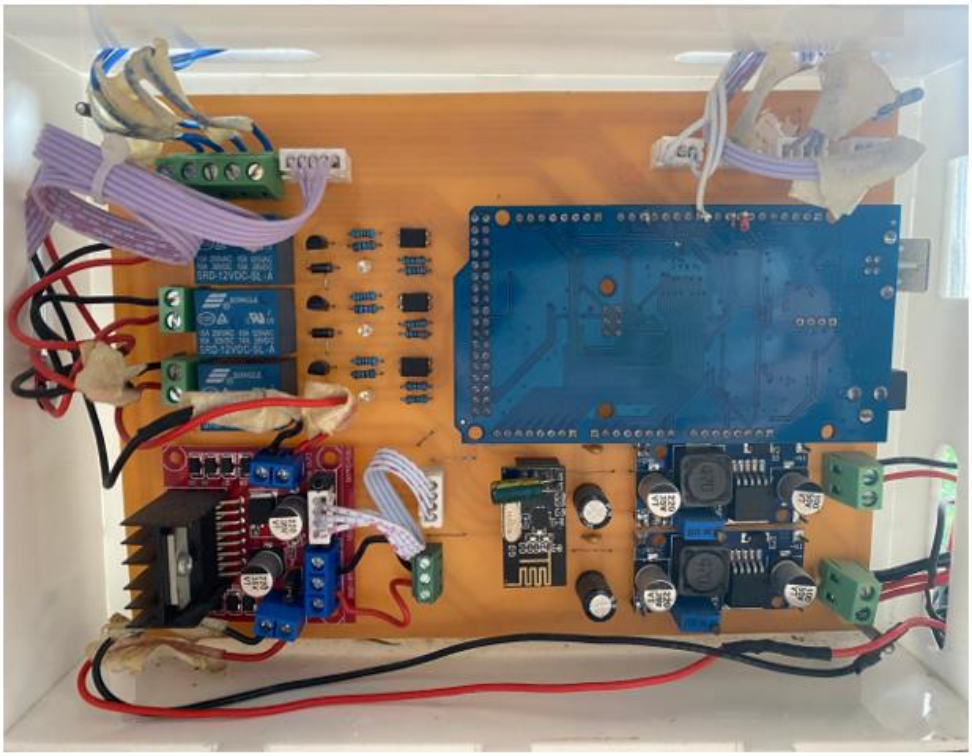


Figure 3.5: Greenhouse circuit 2

- Greenhouse central circuit design results

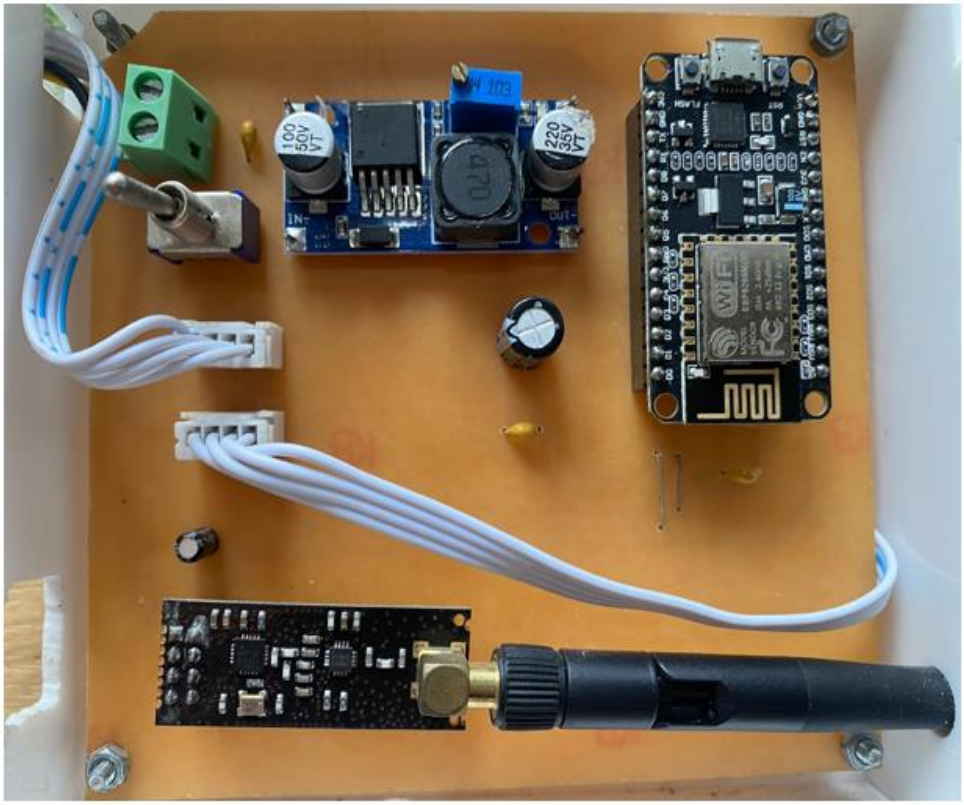


Figure 3.6: Greenhouse central circuit

3.2.2 Programming for greenhouses

- Declare the SPI, nRF24L01, and RF24 libraries for wireless communication and the DHT library to read data from temperature and humidity sensors.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <DHT.h>
```

Figure 3.7: Declare the greenhouse system library

- Define connection pins and devices such as lights, pumps, fans, doors, awnings, and switches.

```
#define cs 10
#define rst 9
#define dio0 2
#define rain_sensor A1
#define light_sensor A2
#define soil_moisture_sensor A0

#define led 33
#define pump 35
#define fan 31

#define open_door 5
#define close_door 6
#define open_roof 4
#define close_roof 3

#define limit_switch_open_door 23
#define limit_switch_close_door 25
#define limit_switch_open_roof 29
#define limit_switch_close_roof 27
```

Figure 3.8: Define connection pins

- Create a data_transfer string to contain information from control device variables and states. Data will be transmitted through the NRF24L01 radio module to the center.

```
void loop() {
  const char text[100];
  for (byte len = 1; len <= data_transfer.length() + 1; len++) {
    data_transfer.toCharArray(text, len);
  }
  radio.write(&text, sizeof(text));
  delay(500);
}
```

Figure 3.9: Send sensor values and device status

Read and check the temperature, soil moisture, rain, and light sensor values to control the corresponding device on and off based on the installed value.

```
if (temperature > 29) {
  digitalWrite(fan, HIGH);
  data_transfer += '2';
} else {
  digitalWrite(fan, LOW);
  data_transfer += '1';
}
data_transfer += 'd';
```

Figure 3.10: Read and check sensor values and control device

3.2.3 Programming for central controller

- Declare necessary libraries for the installer on ESP8266 and other library links such as SPI, nRF24L01, RF24, and LiquidCrystal_I2C.h to support wireless transmission functions using the RF protocol and screen control LCD screen.

```
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266HTTPClient.h>

char ssid[] = "ThienNhan";
char pass[] = "20102001";

RF24 radio(2, 0);  // CE, CSN
const byte address[6] = "00001";

LiquidCrystal_I2C lcd1(0x26, 20, 4);
LiquidCrystal_I2C lcd2(0x27, 20, 4);
```

Figure 3.11: Declare the central system library

- Parse the data from the data string received from the nRF24L01 module and check whether the characters exist or not, then process the data and convert the extracted values to integers and save them in corresponding variables in the greenhouse system.

```
void processRadioData(String data) {
  int a = data.indexOf("a");
  int b = data.indexOf("b");
  int c = data.indexOf("c");
  int d = data.indexOf("d");
  int e = data.indexOf("e");
  int f = data.indexOf("f");
  int g = data.indexOf("g");
  int h = data.indexOf("h");
  int k = data.indexOf("k");
  int l = data.indexOf("l");
  int m = data.indexOf("m");
  int w = data.indexOf("w");

  if (a >= 0 && b >= 0 && c >= 0 && d >= 0 && e >= 0 && f >= 0 && g >= 0 && h >= 0 && k >= 0 && l >= 0 && m >= 0 && w >= 0) {
    temperature = (data.substring(a + 1, b)).toInt();
    humidity = (data.substring(b + 1, c)).toInt();
    soil_moisture = (data.substring(d + 1, e)).toInt();
    status_rain = (data.substring(f + 1, g)).toInt();
    status_light_sensor = (data.substring(g + 1, h)).toInt();
    status_led = (data.substring(k + 1, l)).toInt();
    status_fan = (data.substring(c + 1, d)).toInt();
    status_pump = (data.substring(e + 1, f)).toInt();
    status_door = (data.substring(l + 1, m)).toInt();
    roof = (data.substring(h + 1, k)).toInt();
    int houseNumber = (data.substring(m + 1, w)).toInt();
```

Figure 3.12: Analyze and check data

- Process new data sent from the nRF24L01 module, read that data and convert it to a data_string, then call the processRadioData() function to process and display the data on the serial monitor with a time gap of 3 seconds between processing times.

```
void loop() {
  String data_ = "";
  if (radio.available()) {
    char text[100] = { 0 };
    radio.read(&text, sizeof(text));
    data_ = (String)text;
    Serial.println(data_);
  }
  if (data_ != "") {
    processRadioData(data_);
    delay(3000);
  }
}
```

Figure 3.13: Handling data sending

3.2.4 Results of hardware programming

- Data is sent from greenhouse node 1.

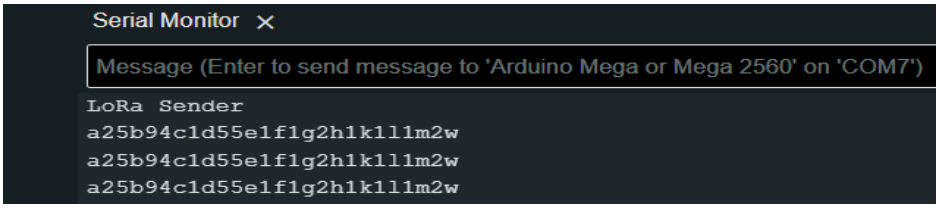


Figure 3.14: Data is sent from greenhouse node 1

- Data is sent from greenhouse node 2.

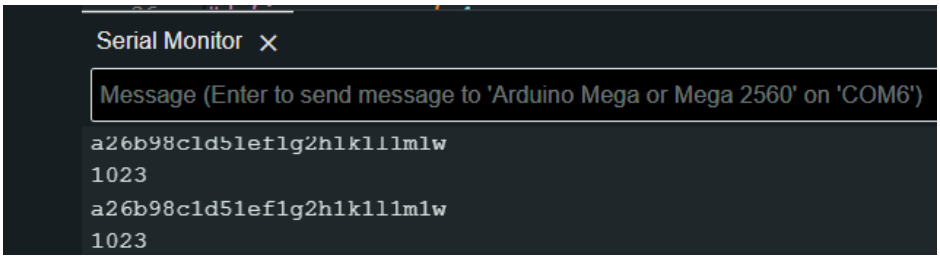


Figure 3.15: Data is sent from greenhouse node 2

- The central controller receives data from 2 greenhouses

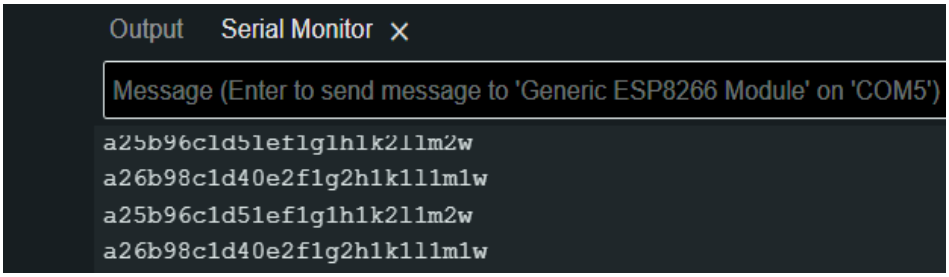


Figure 3.16: The central controller receives data from 2 greenhouses

3.3 DEPLOYING AND BUILDING SERVER

3.3.1 Seting up MongoDB database

- Create a MongoDB account and create a project to store the database.

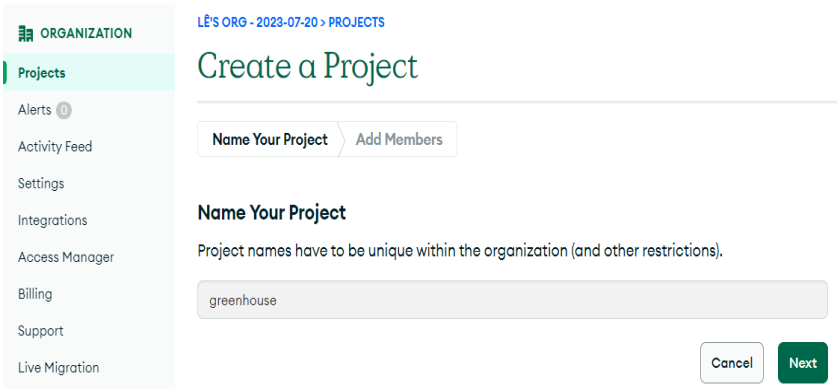


Figure 3.17: Create project for MongoDB database

- Create a database and two collections, greenhouse 1 and greenhouse 2, to store the data.

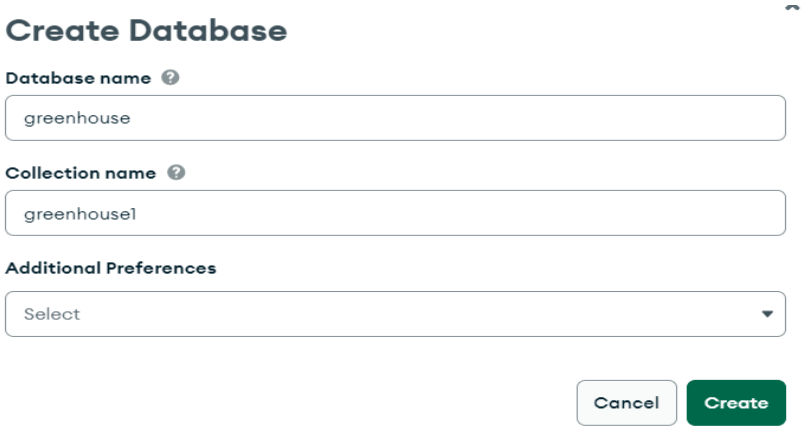


Figure 3.18: Create database and collection

3.3.2 Programming for NodeJS Server

- Initialize the server and handle system-related requests.

```
dotenv.config();
connectDatabase();
const app = express();
app.use(express.json());
app.use(errorMiddleware);
app.use(greenhouseRoutes);
app.get('/', (req, res) => {
  res.send('Welcome to the smart greenhouse server system!');
});
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Figure 3.19: Initialized and processed at the server

- Use the Mongoose library to connect to the MongoDB database. Using the URI connection method uses MongoDB's DNS seedlist connection format, which allows the use of SRV records to identify MongoDB nodes and includes user information, passwords, host addresses, and database names.

```
const connectDatabase = async () => {
  try {
    await mongoose.connect(process.env.MONGODB_URI, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log('Connected to MongoDB');
  } catch (error) {
    console.error('MongoDB connection error:', error);
  }
};
```

```
MONGODB_URI=mongodb+srv://ltnhan19ce:NHANlt2010@cluster0.j9zpsgl.mongodb.net/greenhouse?retryWrites=true&w=majority
```

Figure 3.20: Connect data to MongoDB

- Schema definition of data in MongoDB for greenhouse data houses 1 and 2.

```
const DataGreenhouse1 = mongoose.model('greenHouse1', dataGreenhouseSchema);
const DataGreenhouse2 = mongoose.model('greenHouse2', dataGreenhouseSchema);
```

Figure 3.21: Definition of structural data greenhouse

- Use the mapping between values 1 and 2 in the corresponding text to describe the status of greenhouse components for ease of understanding and use.

```
const rainMap = createStatusMap(1, 2, 'No Rain', 'Rain');
const lightSensorMap = createStatusMap(1, 2, "It's Dark", 'Bright Sky');
const ledMap = createStatusMap(1, 2, 'Light Off', 'Light On');
const fanMap = createStatusMap(1, 2, 'Fan Turns Off', 'Fan Turns On');
const pumpMap = createStatusMap(1, 2, 'Pump Turns Off', 'Pump Turns On');
const doorMap = createStatusMap(1, 2, 'Close Door', 'Open Door');
const roofMap = createStatusMap(1, 2, 'Close Roof', 'Open Roof');
```

Figure 3.22: Mapping values corresponding to states

- Process input data and store it into the MongoDB database corresponding to each greenhouse system.

```
data.temperature = { value: data.temperature, unit: '°C' };
data.humidity = { value: data.humidity, unit: '%' };
data.soil_moisture = { value: data.soil_moisture, unit: '%' };
data.status_rain = rainMap[data.status_rain];
data.status_light_sensor = lightSensorMap[data.status_light_sensor];
data.status_led = ledMap[data.status_led];
data.status_fan = fanMap[data.status_fan];
data.status_pump = pumpMap[data.status_pump];
data.status_door = doorMap[data.status_door];
data.roof = roofMap[data.roof];

const newData = new DataGreenhouse1(data);
await newData.save();
```

Figure 3.23: Process input data and store data

- Define routers in Express to handle routers related to greenhouse data.

```
router.route('/api/data/greenhouse-1').get(getGreenhouse1Data).post(postGreenhouse1Data);
router.route('/api/data/greenhouse-2').get(getGreenhouse2Data).post(postGreenhouse2Data);
```

Figure 3.24: Define routers in Express

- Error handling in the system helps manage and log errors while running the application.

```
const errorMiddleware = (err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Something went wrong!');
};
```

Figure 3.25: Handling system errors

3.3.3 Testing the Server

- Use virtual data created from Postman to GET to check if the value is sent and saved to MongoDB database, if yes the server is working normally, if not then the server is experiencing an error.

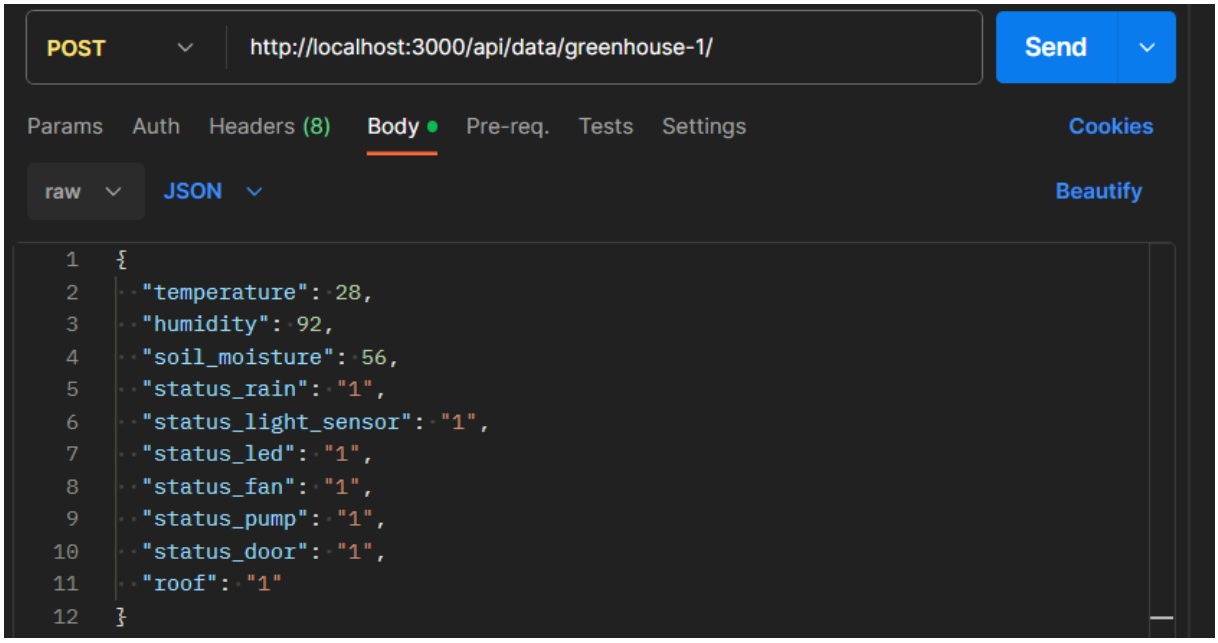


Figure 3.26: Test data for house 1 and house 2 using Postman

- Greenhouse Data 1 runs on http://localhost:3000/api/data/greenhouse-1/

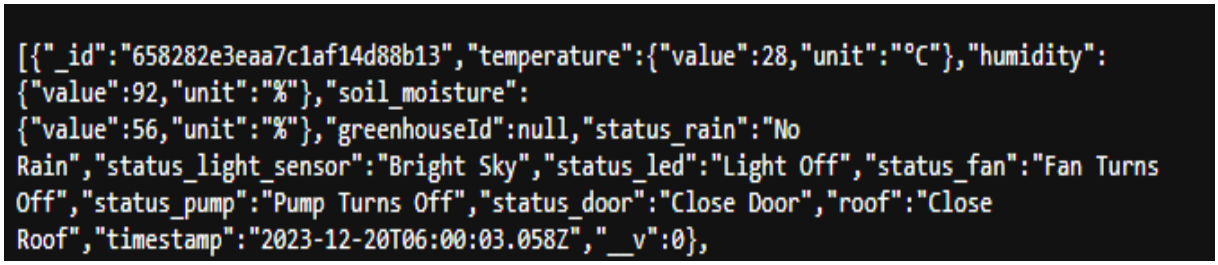


Figure 3.27: Get greenhouse number data 1

- Greenhouse Data 2 runs on http://localhost:3000/api/data/greenhouse-2/

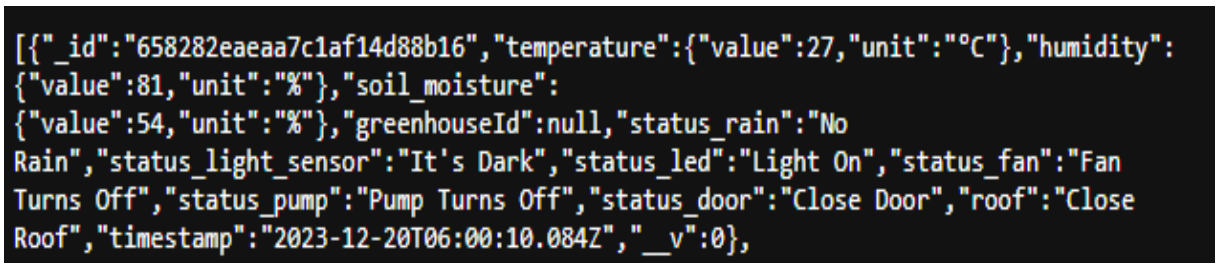


Figure 3.28: Get greenhouse number data 2

3.3.4 Deploying Server to hosting

- Upload the entire NodeJS Server source code to GitHub.
- Proceed to build an account, start creating hosting at the website Render.com, and connect to github to deploy the server NodeJS.

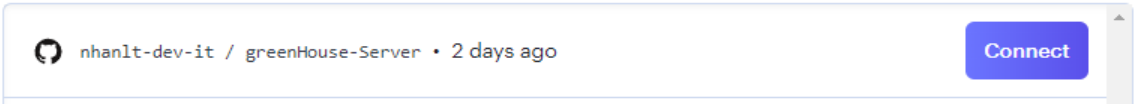


Figure 3.29: Connect hosting to github

- Create a domain name to use for the greenhouse system server and deploy.

Figure 3.30: Create domain name

3.3.5 Results of Server implementation

- Successfully built a server and deployed it with the domain name "https://green-house-orre.onrender.com/" as a server to save data into MongoDB.

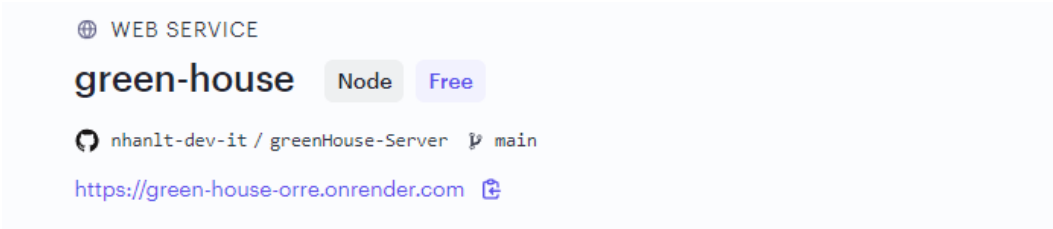


Figure 3.31: Server implementation results

3.4 ESP8266 CONNECTION TO THE SERVER

3.4.1 Server connection

- Declare the variables used to connect to the server and specify the Server address and data transfer port.

```
const char* serverAddress = "green-house-orre.onrender.com";
const int serverPort = 443;
```

Figure 3.32: ***Declare the connection variable to the server***

- Send all data from greenhouse 1 and greenhouse 2 to the Server via the https protocol in JSON format. The data contains input parameters corresponding to the sensor and device status.

```
String path = "/api/data/greenhouse-" + String(greenhouseNumber);
String url = "https://" + String(serverAddress) + path;

if (http.begin(client, url)) {
  http.addHeader("Content-Type", "application/json");

  String postData = "{\"temperature\": \"";
  postData.concat(String(temperature));
  postData.concat("\",\"humidity\":\"");
  postData.concat(String(humidity));
  postData.concat("\",\"soil_moisture\":\"");
  postData.concat(String(soil_moisture));
  postData.concat("\",\"status_light_sensor\":\"");
  postData.concat(String(status_light_sensor));
  postData.concat("\",\"status_rain\":\"");
  postData.concat(String(status_rain));
  postData.concat("\",\"status_led\":\"");
  postData.concat(String(status_led));
  postData.concat("\",\"status_fan\":\"");
  postData.concat(String(status_fan));
  postData.concat("\",\"status_pump\":\"");
  postData.concat(String(status_pump));
  postData.concat("\",\"status_door\":\"");
  postData.concat(String(status_door));
  postData.concat("\",\"roof\":\"");
  postData.concat(String(roof));
  postData.concat("\"}");

  int httpCode = http.POST(postData);
```

Figure 3.33: ***Send data to the Server for each greenhouse***

3.4.2 Results of ESP8266 connection to MongoDB

- The data from each house is processed and sent to the Server in turn.

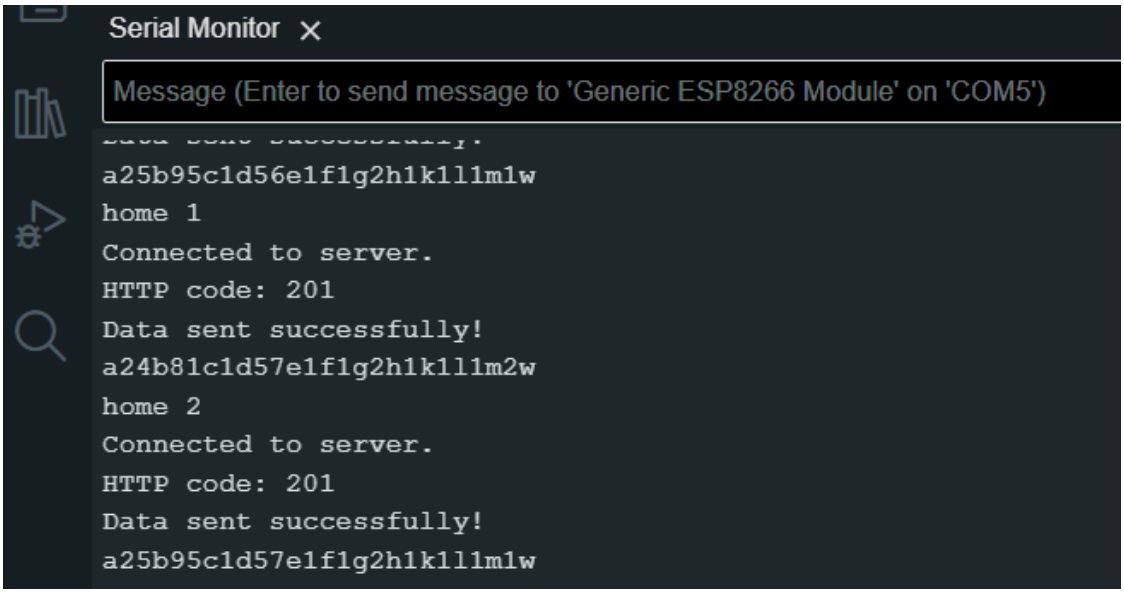


Figure 3.34: Greenhouse data sent to the server successfully

- Store the data of each house corresponding to two collections, greenhouse1 and greenhouse2, in MongoDB in the most intuitive and easy-to-manage way.

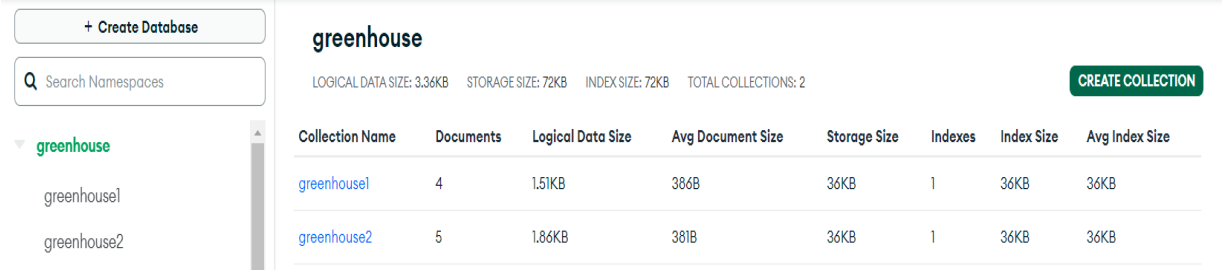


Figure 3.35: Greenhouse data at MongoDB

- Detailed data on sensor information and the status of greenhouse device 1.

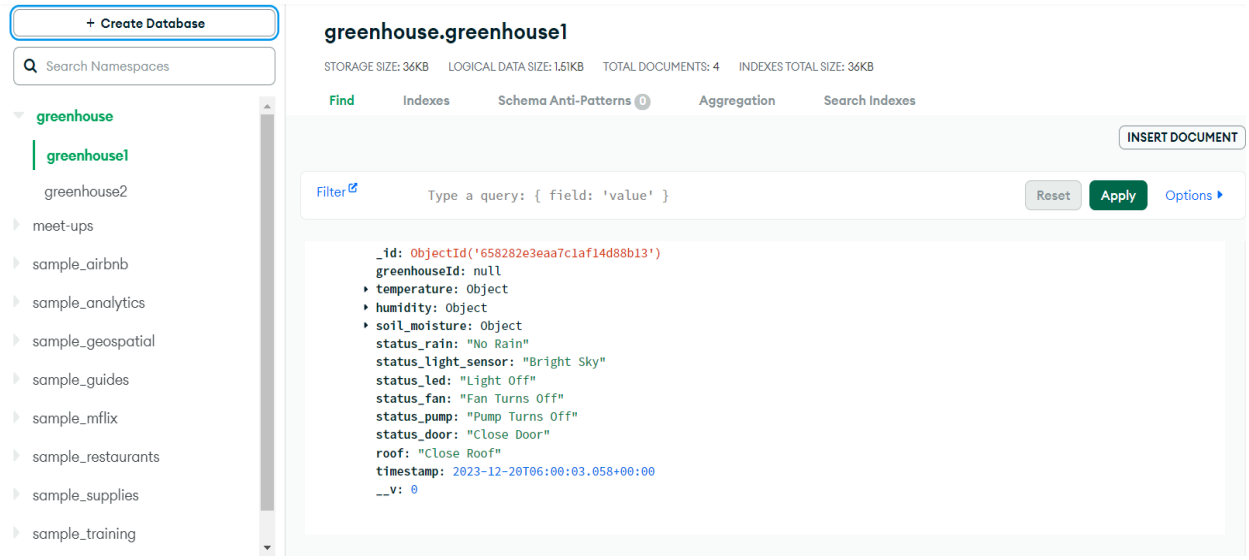


Figure 3.36: Detailed data of greenhouse 1

- Detailed data on sensor information and the status of greenhouse device 2.

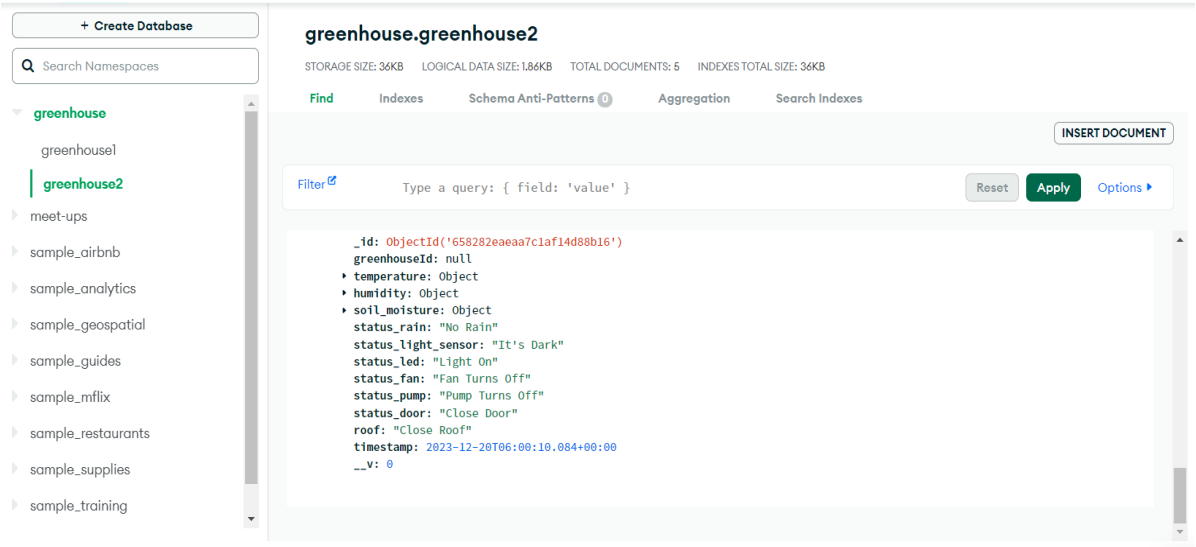


Figure 3.37: Detailed data about greenhouse 2

3.5 OUTSYSTEM CONNECT TO MONGODB DATABASE

3.5.1 Outsystem connection

- Set up an OutSystem to connect to the MongoDB database.

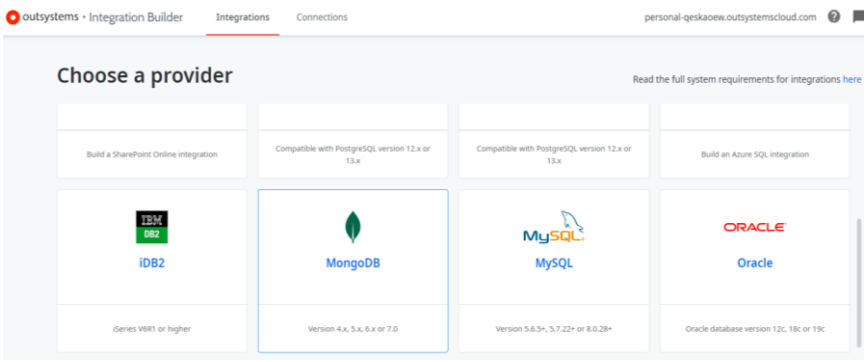


Figure 3.38: Choose to connect MongoDB to the Outsystem system

- Create and integrate data into OutSystem using account information, including name, password, hostname, and MongoDB database, to connect and retrieve data.

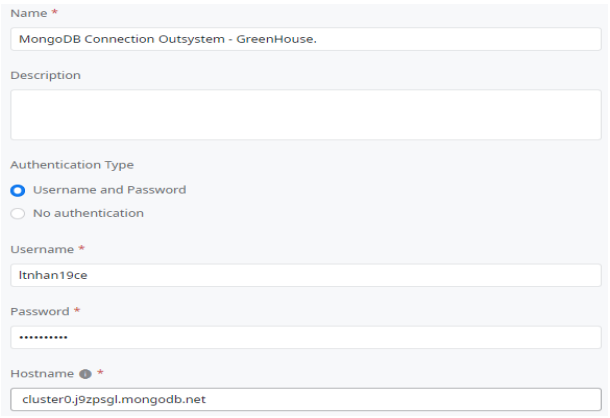


Figure 3.39: Create and log in an account to connect Outsystem with MongoDB

- Connect to the greenhouse database and retrieve data from two collections of greenhouse systems 1 and 2.

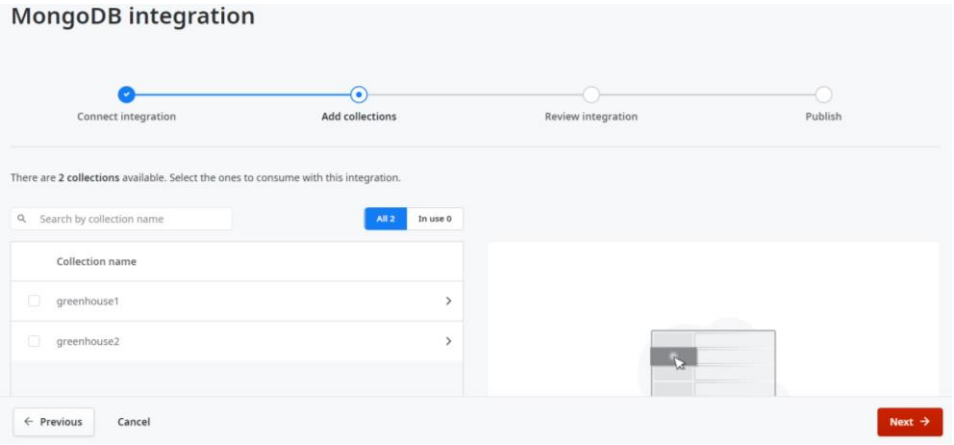


Figure 3.40: Retrieve greenhouse data to Outsystem

- Push greenhouse data from MongoDB to an external system to perform data retrieval for the application.

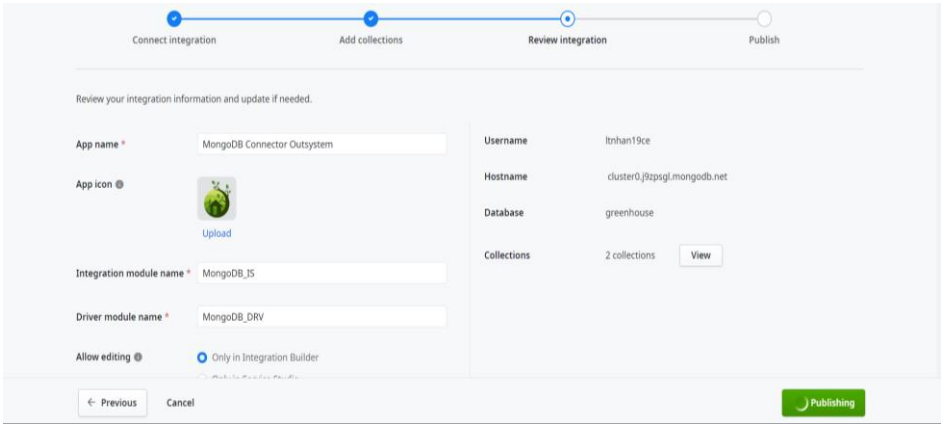


Figure 3.41: Public MongoDB data to the Outsystem system

3.5.2 Results of building Outsystem to MongoDB

- Data has been successfully integrated from MongoDB into the OutSystem.

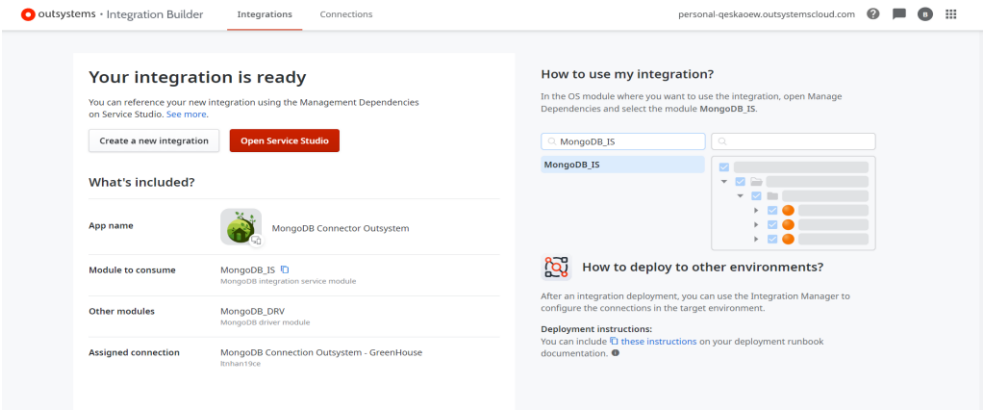


Figure 3.42: Integrate MongoDB data into Outsystem

- In the application, open manage dependencies and find MongoDB_IS and MongoDB_DRV to install into the application to perform the process of retrieving greenhouse data from the MongoDB database.

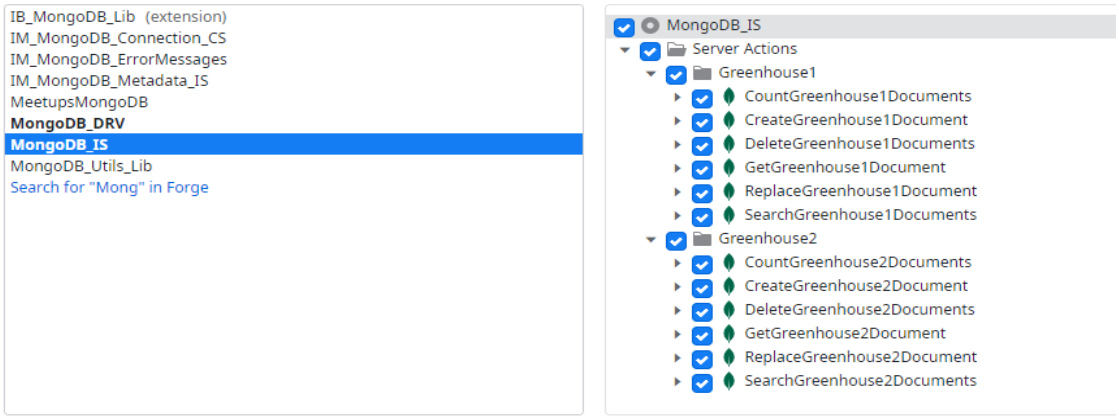


Figure 3.43: Dependencies MongoDB

- The system has been successfully integrated into the Outsystem application for cross-platform deployment.

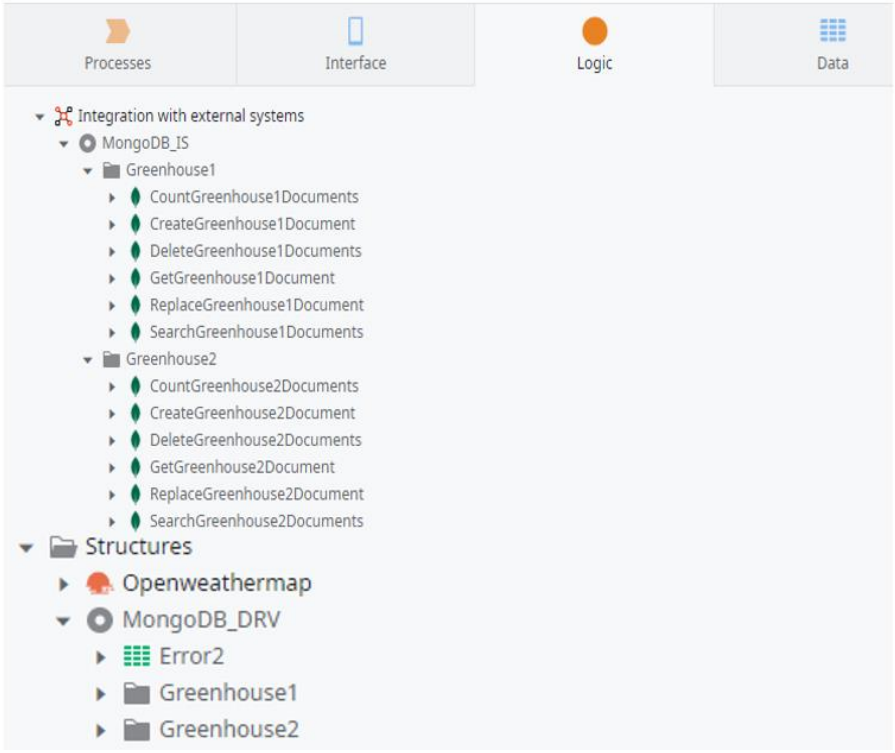


Figure 3.44: Data successfully integrated into the application

3.6 MULTI-PLATFORM OUTSYSTEM APPLICATION PROGRAMMING

3.6.1 Mobile and web applications

- Build a UserExtend database to store user account information from the application corresponding to User data in the Outsystem system.

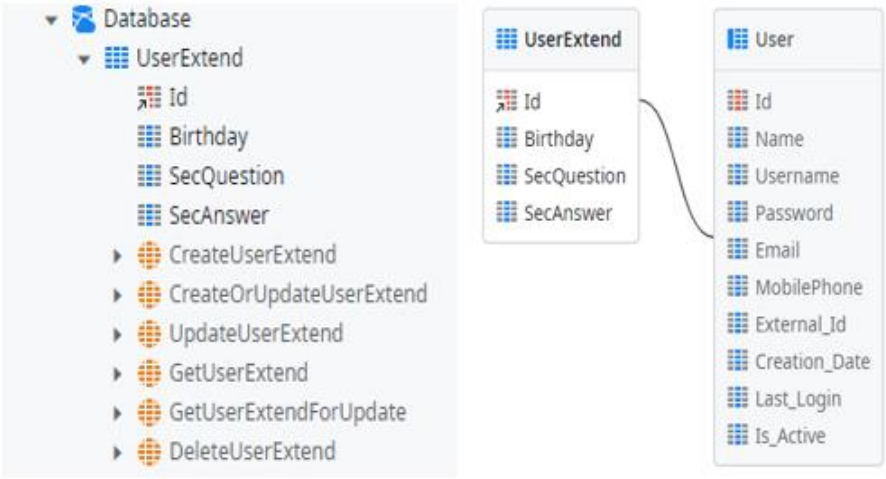


Figure 3.45:: Build database for application

- Build logic for the application to handle the application's activity flow.

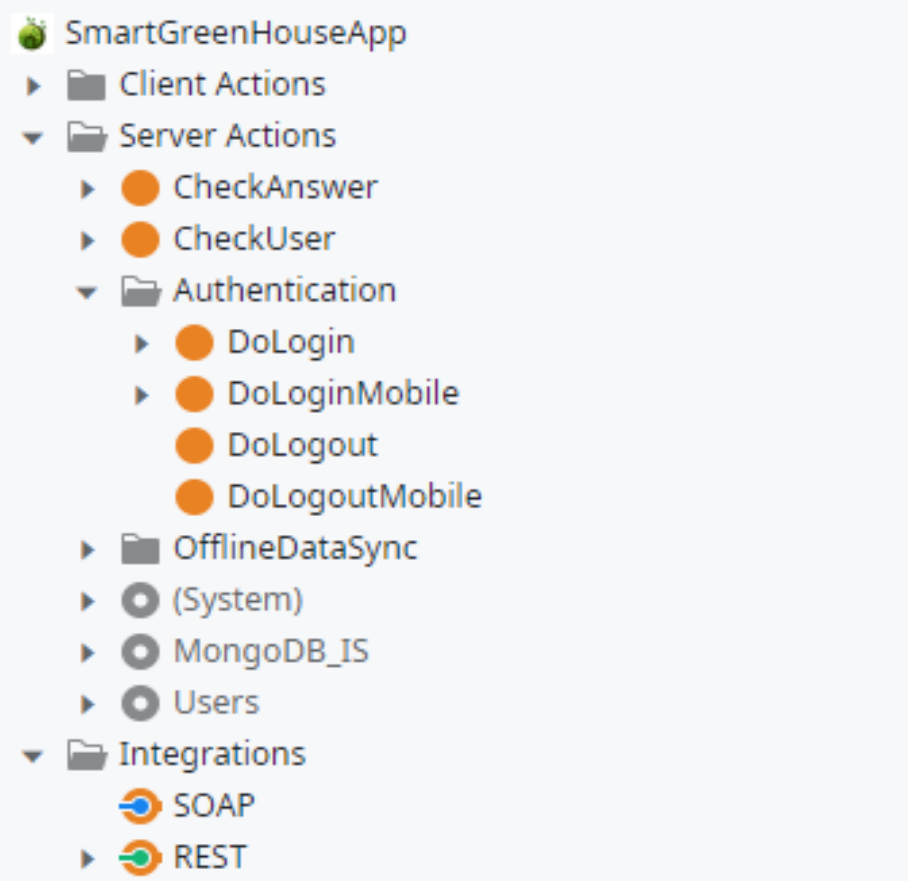


Figure 3.46: Build logic for the application

- Build interfaces and design information pages about greenhouse system management.

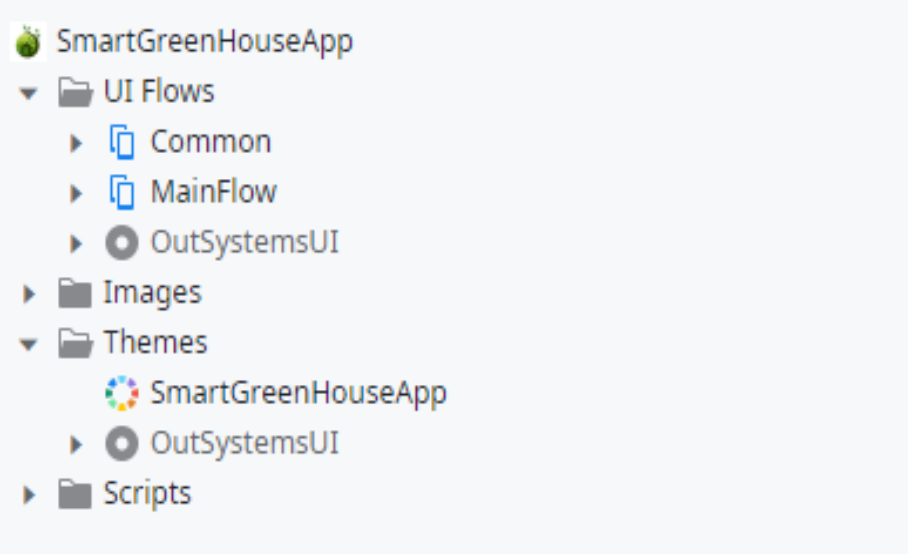


Figure 3.47: Build interface for the application

3.6.2 Results of applications

- **Install apps :**

+ For Android, scan the code below to access the link to download the apk file to install on your device and perform login, monitoring applications, and greenhouse system management.

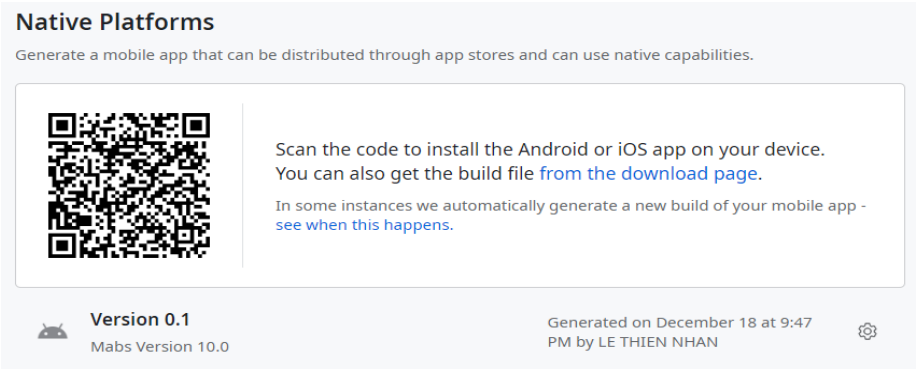


Figure 3.48: QR code native platforms

+ For Android and iOS, you can scan below this code to access the web directly and monitor and manage the greenhouse system quickly and easily.

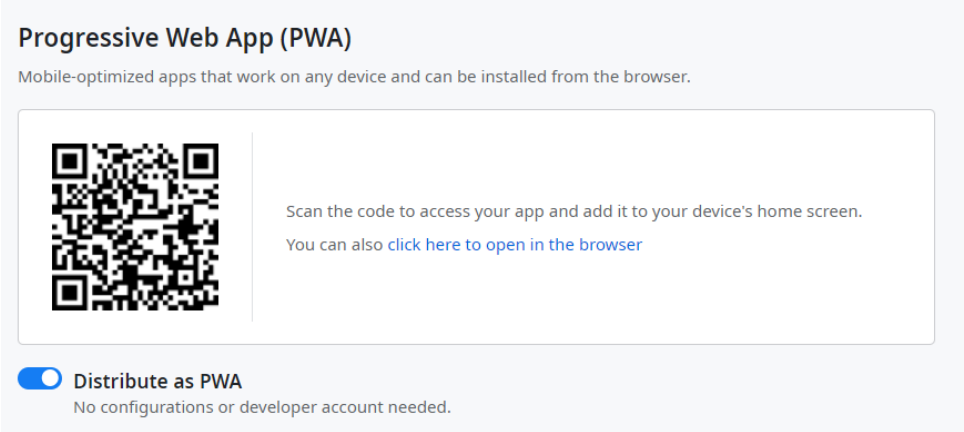


Figure 3.49: QR code web app

- **Mobile application**

+ Login and register interface page

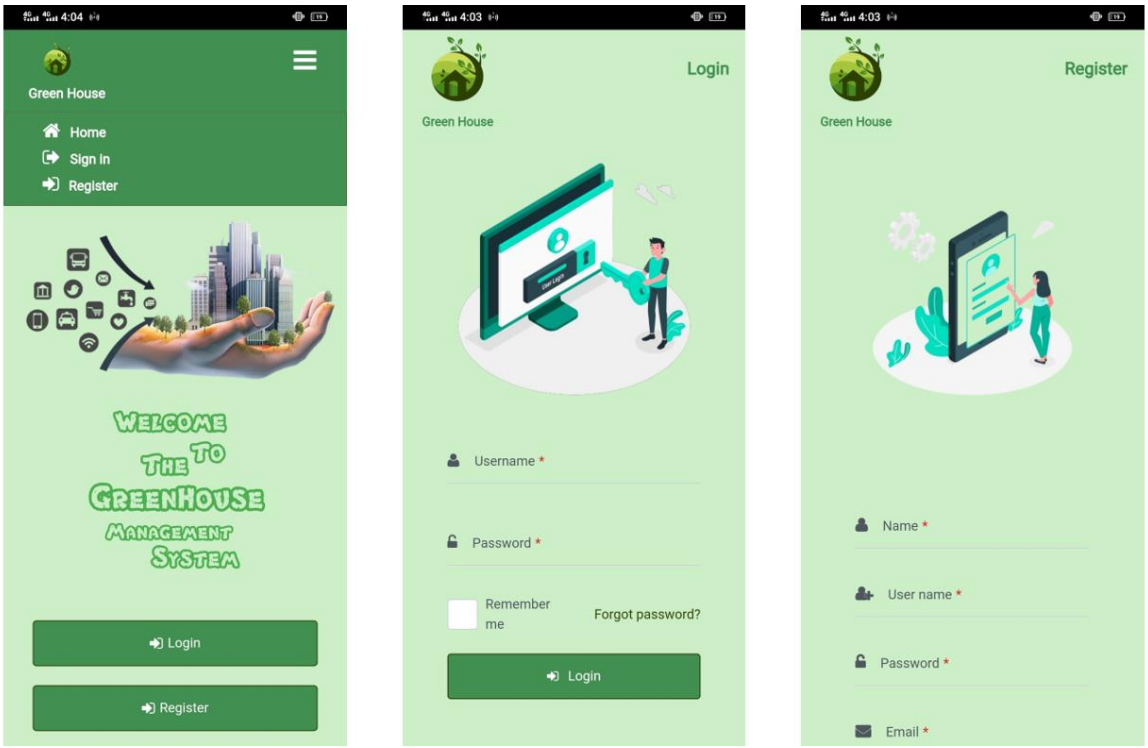


Figure 3.50: Login and register page

+ Fogort password và reset password interface page

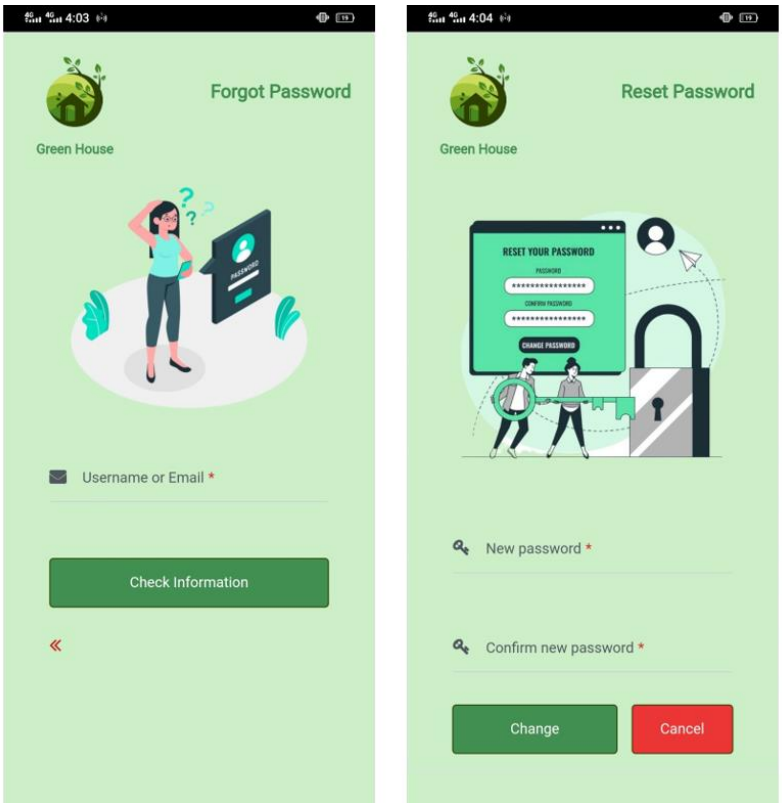


Figure 3.51 Forgot password và reset password page

+ Home, devices, statistical interface page

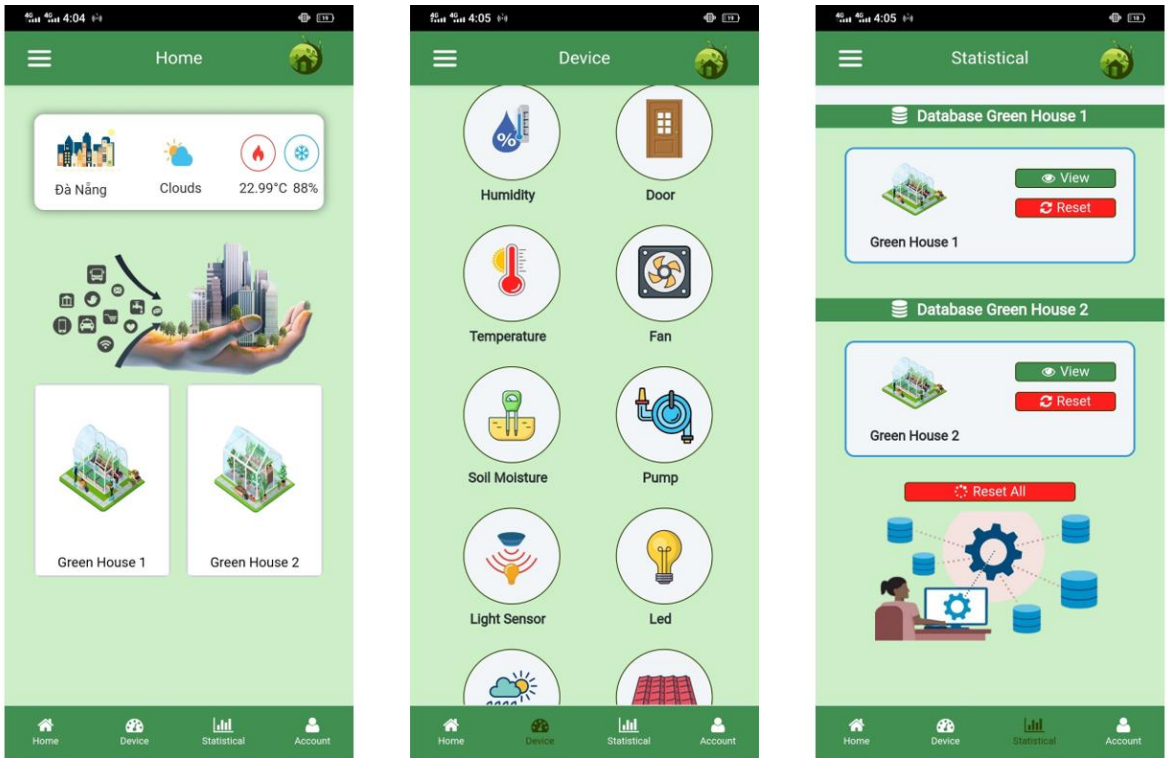


Figure 3.52: Home, devices, statistical page

+ Greenhouse 1 and greenhouse 2 monitoring information interface page

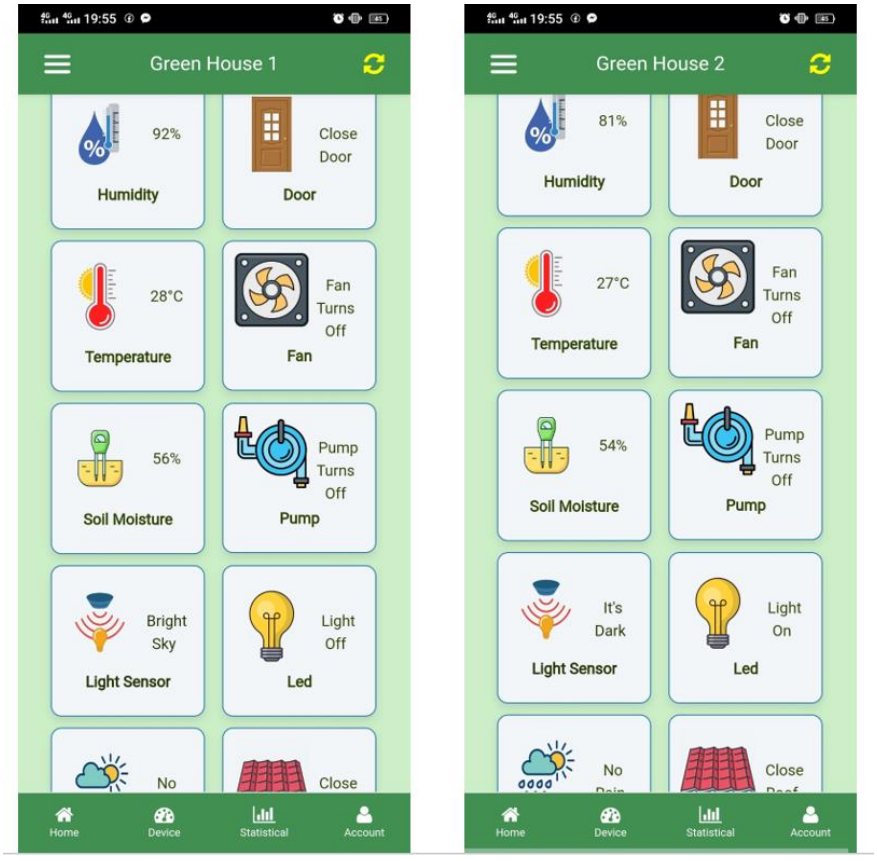


Figure 3.53: Greenhouse monitoring information page

+ Interface page with detailed information of each sensor, device and latest data upload time of greenhouse 1 and greenhouse 2

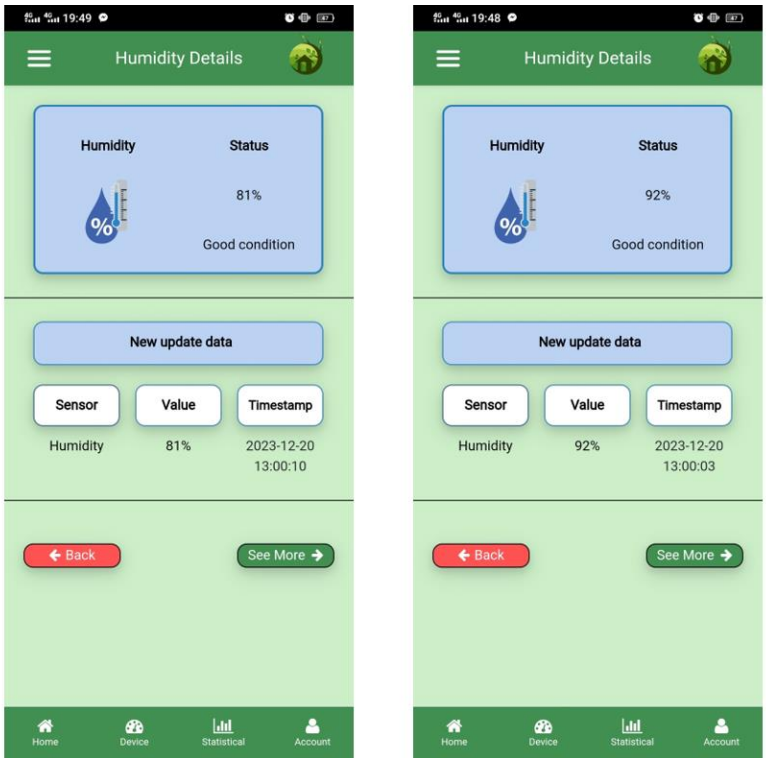


Figure 3.54: Detailed information page about greenhouse

+ Interface page to monitor sensors and device status of 2 greenhouses

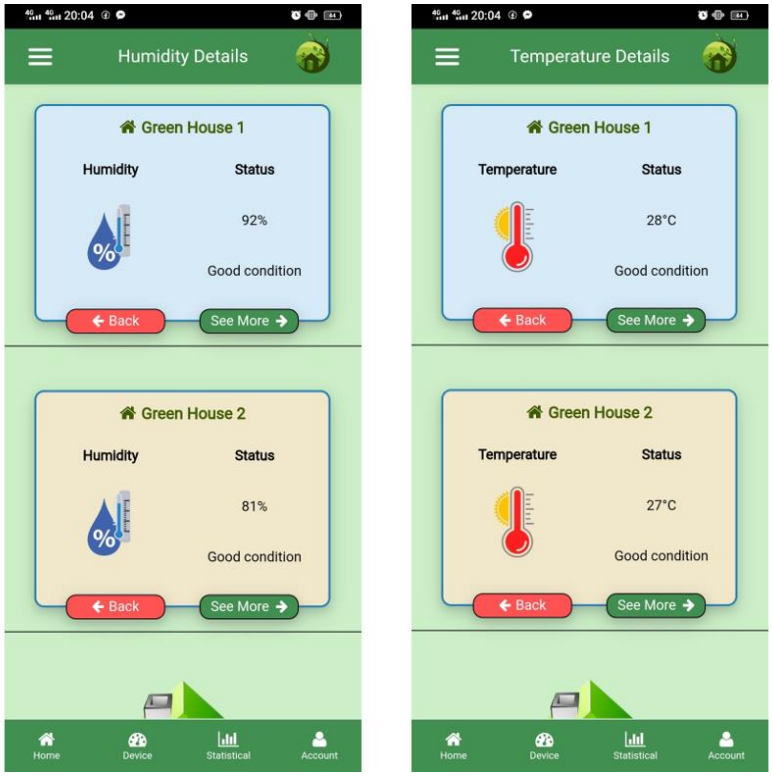


Figure 3.55: Sensor and device data page of 2 greenhouses

+ Database view interface page with all data of each greenhouse system

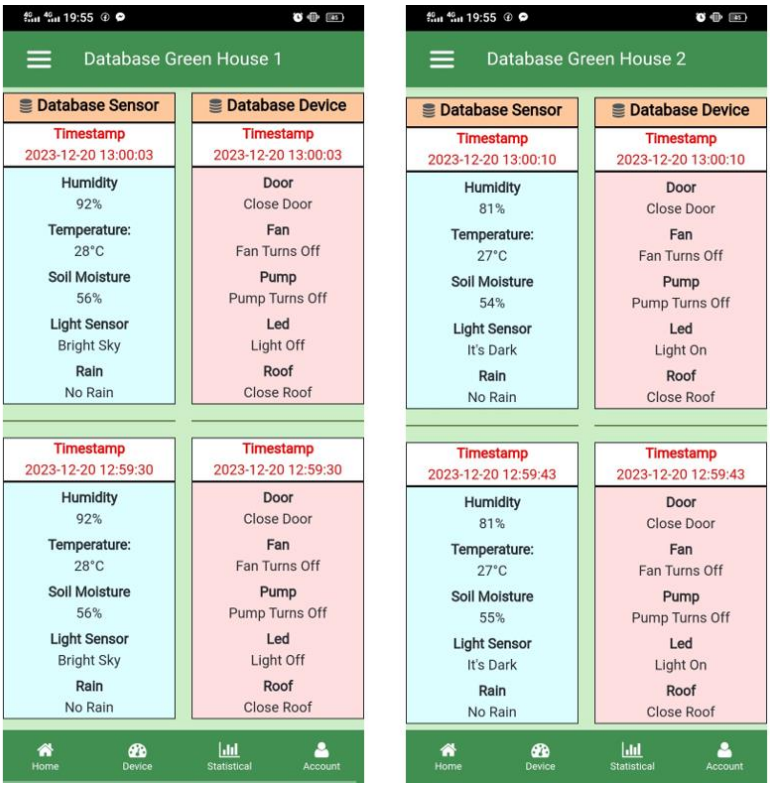


Figure 3.56: Database view page

+ The interface page resets the database for greenhouse 1 and greenhouse 2 and the entire greenhouse

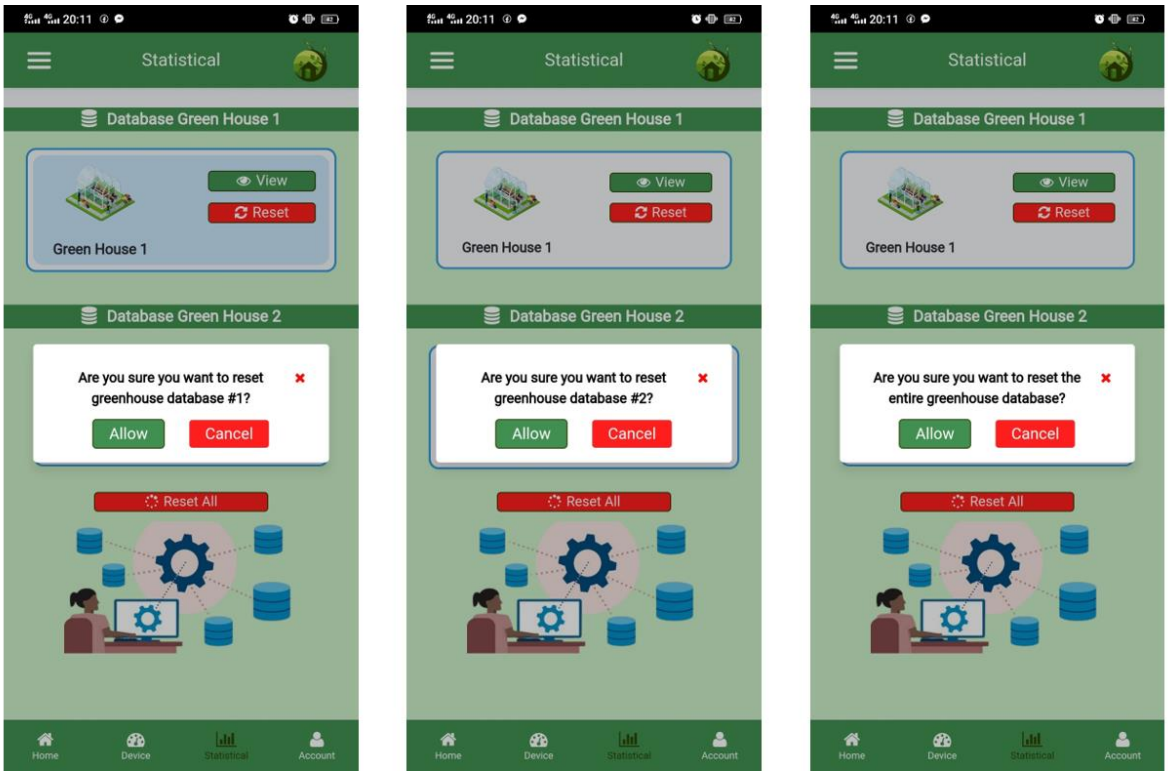


Figure 3.57: Data reset page

+ Account information interface page

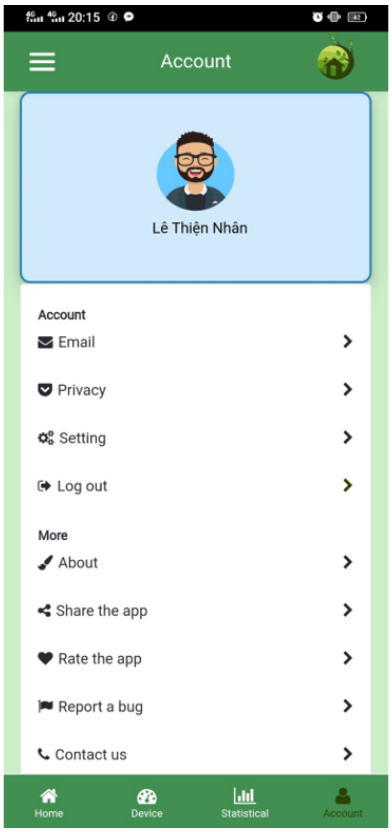


Figure 3.58: Account page

- Web application:

Websites are built with similar functionality to mobile applications.

+ Home interface page

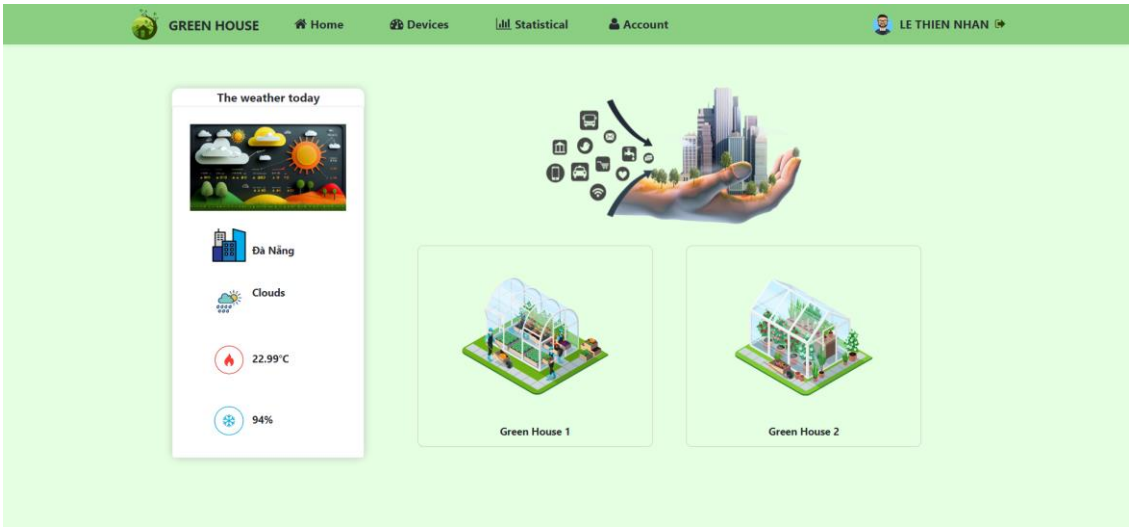


Figure 3.59: Home page

+ Devices interface page

Figure 3.60: Devices page

+ Statistical interface page

Figure 3.61: Statistical page

+ Greenhouse monitoring information interface page

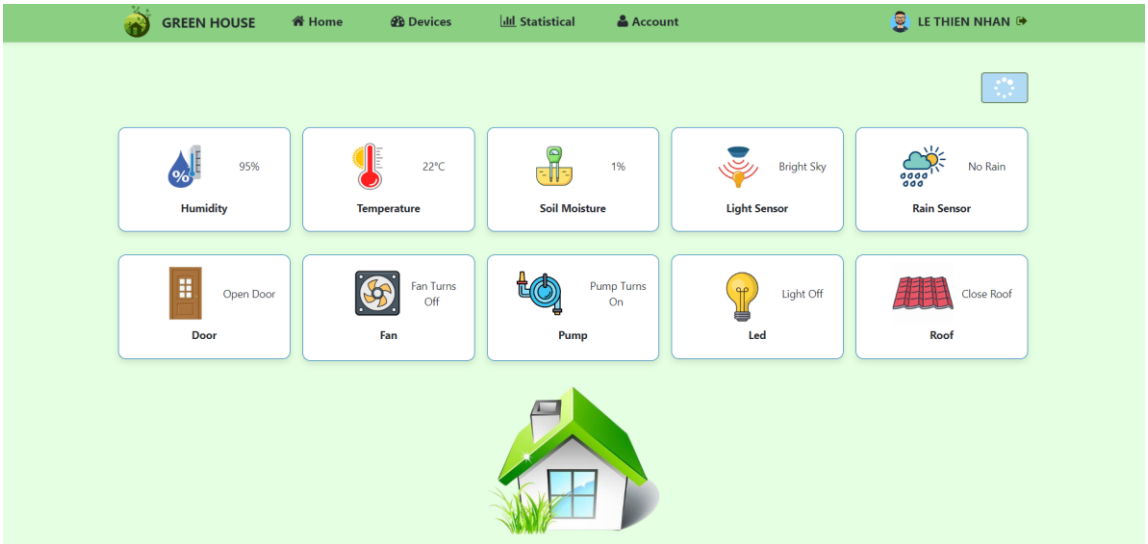


Figure 3.62: Greenhouse monitoring information interface page

+Detailed information page about sensors and greenhouse equipment

Figure 3.63: Detailed information page about sensors and greenhouse equipment

+ The data page greenhouse information

Figure 3.64: The data page greenhouse information

+ Account page

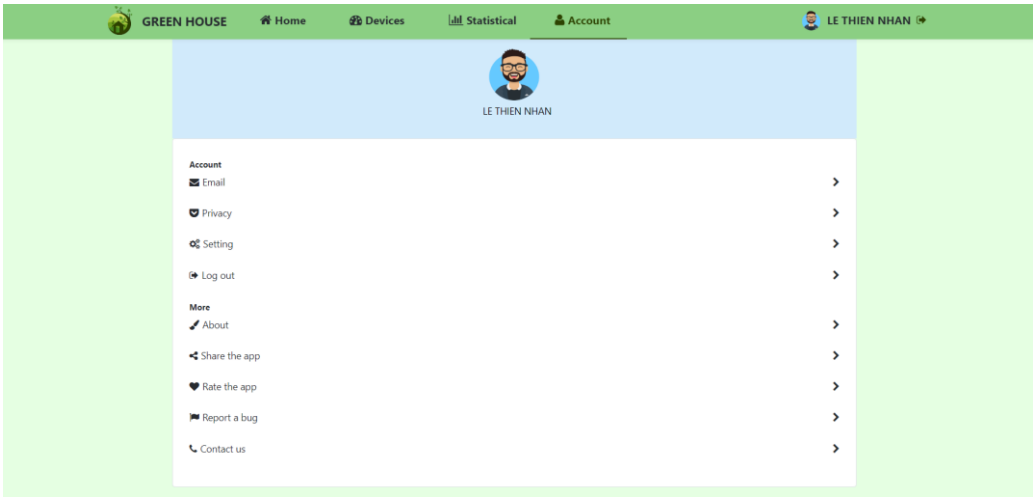


Figure 3.65: Account page

3.7 SYSTEM REVIEW

3.7.1 Smart greenhouse system results

- Greenhouse 1

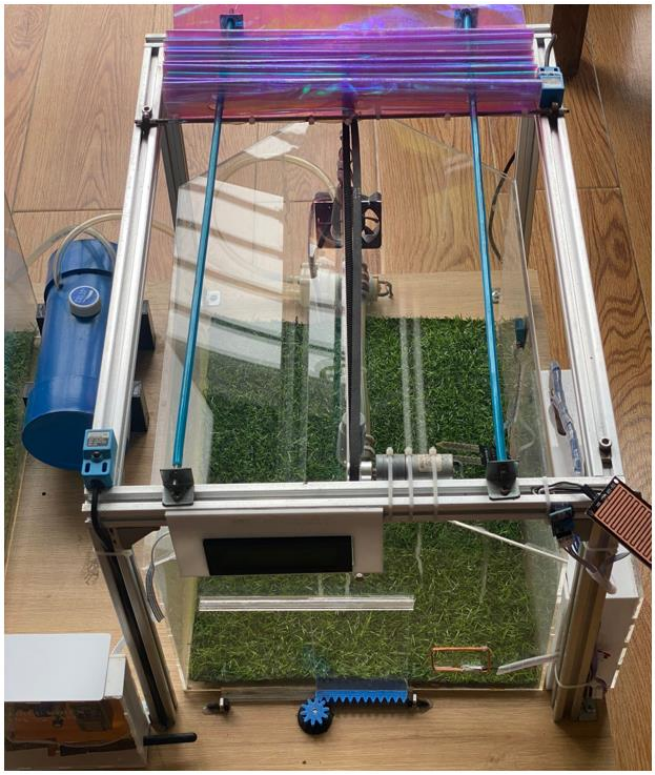


Figure 3.66: Greenhouse model 1

- Greenhouse 2

Figure 3.67: Greenhouse model 2

- Entire greenhouse system

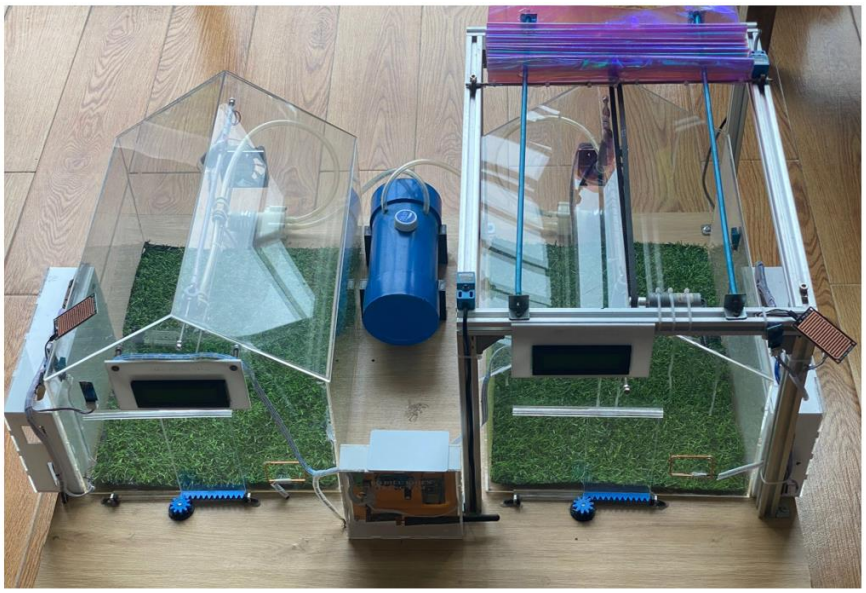


Figure 3.68: Smart greenhouse system

3.7.2 System testing

Perform system testing within 10 hours to measure sensor values as well as the operating status of equipment in the greenhouse system. Below is a statistical table of the greenhouse's operating status as measured hourly.

Time	Humidity	Temperature	Soil moisture	Light sensor	Rain sensor	Door	Fan	Pump	Led	Roof
12:00 PM	80%	32°C	48%	Bright sky	No rain	Close door	Fan turn on	Pump turn on	Light off	Open roof
13:00 PM	81%	32°C	50%	Bright sky	No rain	Close door	Fan turn on	Pump turn off	Light off	Open roof
14:00 PM	85%	31°C	49%	Bright sky	No rain	Close door	Fan turn on	Pump turn on	Light off	Open roof
15:00 PM	85%	32°C	49%	Bright sky	No rain	Close door	Fan turn on	Pump turn on	Light off	Open roof

16:00 PM	87%	29°C	52%	Bright sky	No rain	Close door	Fan turn off	Pump turn off	Light off	Open roof
17:00 PM	87%	28°C	50%	Bright sky	No rain	Close door	Fan turn off	Pump turn off	Light off	Close roof
18:00 PM	88%	27°C	49%	It's dark	No rain	Close door	Fan turn off	Pump turn on	Light on	Close roof
19:00 PM	90%	25°C	53%	It's dark	No rain	Close door	Fan turn off	Pump turn off	Light on	Close roof
20:00 PM	92%	25°C	51%	It's dark	No rain	Close door	Fan turn off	Pump turn off	Light on	Close roof
21:00 PM	95%	23°C	50%	It's dark	No rain	Close door	Fan turn off	Pump turn off	Light on	Close roof

Table 3.1: Statistics table of greenhouse system operating status

Through the statistical table we can see the operating status of the greenhouse system as follows:

- The sensors all work well and measure environmental values.
- Sensor data is accurately displayed on the LCD screen of each greenhouse
- Device status is properly controlled based on sensor values and conditions are set so that each device automatically turns on and off according to environmental changes.
- Sensor information and device status are continuously updated in the app. Additionally, each greenhouse's database can be directly monitored to see details every time new data is updated.

3.7.3 System review

- Meets design requirements.
- Meet safety standards and ensure the operation and use of electrical components and equipment.

- Maintenance is easy through clear, detailed design blocks.
- Meet equipment automation requirements based on design processes.
- Solve data problems by storing data in a MongoDB database.
- The server has been built but does not yet meet real-time performance standards.
- A new programming language app Outsystem handles cross-platform building across all devices.
- Easily monitor and manage sensors and device status through cross-platform applications on any device, such as mobile or web.

3.8 SUMMARY

Chapter 3 plays the most important role as the place to carry out the steps to implement the greenhouse system implementation process, including main steps such as hardware implementation, building servers, and hardware and software connections. Each step will be implemented in detail and clearly to fully grasp and understand the components and functions.

- For the step of hardware implementation, the focus will be on building and completing the printed circuit, then programming each greenhouse device and central controller to ensure efficient packaging and data transmission via LoRa waves. most effective and optimal.
- For the step of building the server connection, establish and create connections and data fields suitable for the greenhouse system, and connect to the MongoDB database system. After a successful build, the server must be tested to see if it is working properly. If available, it will be deployed to hosting to deploy the system.
- For the step of hardware and software connections, it is the core content of the system that helps the system connect to the server to send data to MongoDB. After the data is successfully uploaded, Outsystem will be set up to connect to MongoDB. Next, Outsystem will retrieve all of the greenhouse collection data in the MongoDB database. Then, push to the Outsystem system and go to the Outsystem service studio application to fetch data and proceed to build cross-platform applications for the system.

Through the above deployment step, can understand and grasp each component in the system as well as the deployment process for hardware, software or servers. Thereby, the desired results were achieved compared to the initial design requirements and the system evaluation was clear and meticulous.

CONCLUSION AND DEVELOPMENT DIRECTION

Conclusion

Content has been implemented:

- Successfully designed and implemmentation the greenhouse and central controller.
- Successfully designed a system with high automation and automatic interaction between sensors and devices.
- The greenhouse system model is designed carefully and in detail.
- Successfully deployed and built a greenhouse system with a self-setup Server using NodeJS.
- Successfully deployed and uploaded the Server to render.com for use.
- Successfully established a Server connection to the MongoDB database to store all information, sensors, and device status for greenhouse systems 1 and 2.
- Successfully configured and set up Outsystem connect to the MongoDB database to retrieve greenhouse database data to build cross-platform applications.
- Successfully built Android, iOS, and Web applications based on the new Outsystem programming language.
- The app can monitor all sensor information and device status for each greenhouse.
- The app features detailed tracking of each sensor and device, as well as an updated time of the latest data sent from the greenhouse.
- The application can display statistics about all submitted and stored databases.
- The application can reset the database to avoid data overflow.

Limit:

- The monitoring application is still simple and not beautiful.
- Real-time requests for unloaded data require a 30-second wait to avoid Server crashes.
- The system cannot add control features to the application due to a conflict with the sensor system.

Development direction

- Edit the application interface to make it more beautiful.
- Use more stable hosting so data can be accessed in real-time.
- Add the feature to control the device manually and avoid conflicts with sensors.
- Develop new functions in the system to check sensor status.

REFERENCES

Articles - books:

[1] Paperback (2021), Fast PCB Design with Altium Designer

[2] Michael Margolis (2011), Arduino Cookbook

[3] Major R.Thakur (2018), NodeMCU ESP8266 Communication Methods and Protocols.

[4] Ethan Brown (2019), Web Development with Node & Express: Leveraging the JavaScript Stack.

[5] Amit Phaltankar (2020) , MongoDB Fundamentals.

Website:

[1] https://www.altium.com/documentation/

[2] https://www.arduino.cc/en/Guide/

[3] https://www.codecademy.com/learn/learn-node-js/

[4] https://www.mongodb.com/docs/

[5] https://success.outsystems.com/documentation/