**Hanoi University of Science and Technology**

**School of Information and Communication Technology**

**–––––––––––––\*–––––––––––––**

**HÀ NỘI January 2019**

Contents

[Chapter 1. Introduction 8](#_Toc536316381)

[1.1. Problem statement 8](#_Toc536316382)

[1.2. Techniques for growing hydroponic vegetables 10](#_Toc536316383)

[1.3. Growth characteristics of some hydroponic crops 13](#_Toc536316384)

[1.3.1. Lettuce 13](#_Toc536316385)

[1.3.2. Tomato 14](#_Toc536316386)

[1.4. Survey of the existing IoT application hydroponics system 15](#_Toc536316387)

[1.5. General requirements of the IoT application hydroponics system 17](#_Toc536316388)

[1.6. Solution 18](#_Toc536316389)

[Chapter 2. System analysis and design 20](#_Toc536316390)

[2.1. Analysis of system requirements 20](#_Toc536316391)

[2.2. System architecture design 23](#_Toc536316392)

[2.2.1. Overview diagram of the system 23](#_Toc536316393)

[2.2.2. System architecture 25](#_Toc536316394)

[2.3. Used technologies 29](#_Toc536316395)

[2.3.1. MQTT protocol 29](#_Toc536316396)

[2.3.2. MQTT Eclipse Paho library 33](#_Toc536316397)

[2.3.3. MVC (Model-View-Controller) model 33](#_Toc536316398)

[2.3.4. Spring Framework 35](#_Toc536316399)

[2.3.5. NodeJS in building web services on the server 36](#_Toc536316400)

[Chapter 3. Development of server application 37](#_Toc536316401)

[3.1. General design of the application on the server 37](#_Toc536316402)

[3.2. Build server communication services with devices via MQTT Broker 39](#_Toc536316403)

[3.2.1. Service to authenticate devices to participate in the system 42](#_Toc536316404)

[3.2.2. Data collection service for sensors submitted by devices 43](#_Toc536316405)

[3.2.3. Device control command service 44](#_Toc536316406)

[3.2.4. Online device polling service (KeepAlive) 45](#_Toc536316407)

[3.3. Store data with the database 46](#_Toc536316408)

[3.3.1. Database design 47](#_Toc536316409)

[3.3.2. Data connection and query from server application 56](#_Toc536316410)

[3.4. Web service for management, monitoring and control 58](#_Toc536316411)

[3.4.1. Functional analysis and design 58](#_Toc536316412)

[3.4.2. Interface design 65](#_Toc536316413)

[3.4.3. Packaging and deployment 72](#_Toc536316414)

[3.5. Data analysis service for automatic control 75](#_Toc536316415)

[Chapter 4. Device firmware development 77](#_Toc536316416)

[4.1. Firmware building platform for ESP32 device 77](#_Toc536316417)

[4.1.1. Hardware resources of ESP32 device 77](#_Toc536316418)

[4.1.2. Real-time operating system (RTOS) platform 80](#_Toc536316419)

[4.1.3. Device firmware construction using FreeRTOS platform 92](#_Toc536316420)

[4.2. Design firmware architecture on ESP32 device 95](#_Toc536316421)

[4.3. Sensor data collection task 98](#_Toc536316422)

[4.3.1. Collect temperature and humidity parameters 99](#_Toc536316423)

[4.3.2. Collect pH parameter 102](#_Toc536316424)

[4.3.3. Collect nutrient concentration TDS parameter 106](#_Toc536316425)

[4.3.4. Collect light intensity parameter 112](#_Toc536316426)

[4.4. Communicate with the server via MQTT Broker 113](#_Toc536316427)

[4.4.1. Authentication function for devices participating in the system 113](#_Toc536316428)

[4.4.2. Function of sending data collected from sensors 116](#_Toc536316429)

[4.4.3. Function to receive control commands from the server 117](#_Toc536316430)

[4.4.4. Function of sending packets to report active status (KeepAlive) 119](#_Toc536316431)

[4.5. The task of executing control commands 120](#_Toc536316432)

[Chapter 5. Testbed results 124](#_Toc536316433)

[5.1. Build testbed test 124](#_Toc536316434)

[5.2. Test results of server software with communication services with devices 125](#_Toc536316435)

[5.2.1. Results of deploying device authentication service 126](#_Toc536316436)

[5.2.2. Results of deploying sensor data collection service 127](#_Toc536316437)

[5.2.3. Service deployment results send control commands to the device 128](#_Toc536316438)

LIST of Figures

[Figure 1‑1. An example of a hydroponic plantation 10](#_Toc536316321)

[Figure 2‑1. Overview diagram of the system 23](#_Toc536316322)

[Figure 2‑2. Architecture of system components 25](#_Toc536316323)

[Figure 2‑3. Architecture and operation mechanism of MQTT protocol 29](#_Toc536316324)

[Figure 2‑4. Illustration of 3 types of QoS of MQTT protocol 31](#_Toc536316325)

[Figure 2‑5. Illustrating MVC model applied in building applications on the server 34](#_Toc536316326)

[Figure 2‑6. Processing mechanism of request of spring framework 35](#_Toc536316327)

[Figure 3‑1. General Use-Case diagram of server-side application 37](#_Toc536316328)

[Figure 3‑2. Intelligent diagram of self-communication between a device and the server via MQTT broker 40](#_Toc536316329)

[Figure 3‑3. Sequence diagram of device authentication services 42](#_Toc536316330)

[Figure 3‑4. Sequence diagram for collecting sensor data service from submitted devices 43](#_Toc536316331)

[Figure 3‑5. Service sequence diagram for control commands to devices 45](#_Toc536316332)

[Figure 3‑6. Service device attendance sequence diagram 46](#_Toc536316333)

[Figure 3‑7. Diagram of relational database design 48](#_Toc536316334)

[Figure 3‑8. Class diagram for data access 57](#_Toc536316335)

[Figure 3‑9. Class diagram for data connection and query 58](#_Toc536316336)

[Figure 3‑10. Use case diagram of the web service application 60](#_Toc536316337)

[Figure 3‑11. Homepage screenshot displayed on a desktop PC 66](#_Toc536316338)

[Figure 3‑12. Homepage screenshot displayed on a smartphone 67](#_Toc536316339)

[Figure 3‑13. Login page screenshot 67](#_Toc536316340)

[Figure 3‑14. User group list management page screenshot 68](#_Toc536316341)

[Figure 3‑15. User list management page screenshot 68](#_Toc536316342)

[Figure 3‑16. User management page screenshot 69](#_Toc536316343)

[Figure 3‑17. Frame and crop management page screenshot 70](#_Toc536316344)

[Figure 3‑18. Device, sensor, actuator management page screenshot 71](#_Toc536316345)

[Figure 3‑19. Package diagram of the web service provider using NodeJS 74](#_Toc536316346)

[Figure 3‑20. Deployment diagram of the web service delivery system 75](#_Toc536316347)

[Figure 4‑1. ESP32 Dev-Kit module used for collecting and controlling devices 78](#_Toc536316348)

[Figure 4‑2. Sơ đồ khối của ESP32 78](#_Toc536316349)

[Figure 4‑3. Basic architecture of RTOS operating system 81](#_Toc536316350)

[Figure 4‑4. Multitasking execution mechanism 82](#_Toc536316351)

[Figure 4‑5. Diagram of the status of a task when executed on RTOS 84](#_Toc536316352)

[Figure 4‑6. Scheduling mechanism for tasks to be executed 85](#_Toc536316353)

[Figure 4‑7. Scheduling with authorization for tasks 86](#_Toc536316354)

[Figure 4‑8. Communication mechanism between two signal event tasks 87](#_Toc536316355)

[Figure 4‑9. Communication mechanism between two messages using the message queue 88](#_Toc536316356)

[Figure 4‑10. Communication mechanism using the message queue with moderation mechanism 89](#_Toc536316357)

[Figure 4‑11. Communication mechanism for 2 tasks using the semaphore alarm engine 90](#_Toc536316358)

[Figure 4‑12. Moderation mechanism between two tasks using mutex 91](#_Toc536316359)

[Figure 4‑13. The basic task structure diagram deployed on FreeRTOS 93](#_Toc536316360)

[Figure 4‑14. Firmware construction architecture architecture for ESP32 devices 97](#_Toc536316361)

[Figure 4‑15. Temperature and humidity sensors used in testbed testing 99](#_Toc536316362)

[Figure 4‑16. Principle diagram of connection with DHT22 temperature and humidity sensor module 101](#_Toc536316363)

[Figure 4‑17. Sensors for measuring pH levels in solutions 103](#_Toc536316364)

[Figure 4‑18. Voltage circuit diagram used to calibrate the output voltage of pH sensor compatible with the voltage range used on ADC of ESP32 105](#_Toc536316365)

[Figure 4‑19. Sensors measure nutrient concentrations in TDS solution via EC conductivity 107](#_Toc536316366)

[Figure 4‑20. Calibration diagram of EC measurement value 111](#_Toc536316367)

[Figure 4‑21. Experimental measurement results from DHT22, pH, and EC sensors 112](#_Toc536316368)

[Figure 4‑22. Diagram of steps to authenticate devices participating in the system 113](#_Toc536316369)

[Figure 4‑23. Algorithm flowchart on firmware includes steps for authentication, data sending, receiving control commands and status status messages 114](#_Toc536316370)

[Figure 4‑24. Sensor data sending scenario from device to server 116](#_Toc536316371)

[Figure 4‑25. Algorithm flowchart sends sensor data to the server 116](#_Toc536316372)

[Figure 4‑26. Diagram of steps to receive control commands from server to device 117](#_Toc536316373)

[Figure 4‑27. Flowchart algorithm receives control commands from the server sent to the device 118](#_Toc536316374)

[Figure 4‑28. Diagram of steps to send device activity status message to server 119](#_Toc536316375)

[Figure 4‑29. The algorithm flow sends the device's status message to the server 119](#_Toc536316376)

[Figure 4‑30. Storage structure and handling control commands in a command\_queue queue 121](#_Toc536316377)

[Figure 4‑31. Flowchart algorithm executes control commands 122](#_Toc536316378)

[Figure 5‑1. Images of lettuce cultivars used as testbed 125](#_Toc536316379)

[Figure 5‑2. The image of the test device receives the control command from the server and executes the control command to turn on the nutrient return pump 129](#_Toc536316380)

List of Tables

[Table 1‑1. EC index of some common plants 12](#_Toc536316439)

[Table 3‑1. Specification of **user\_group** relationship 49](#_Toc536316440)

[Table 3‑2. Specification of **user** relationship 50](#_Toc536316441)

[Table 3‑3. Specification of **frame** relationship 51](#_Toc536316442)

[Table 3‑4. Specification of **plant\_type** relationship 51](#_Toc536316443)

[Table 3‑5. Specification of **crop** relationship 52](#_Toc536316444)

[Table 3‑6. Specification of **device\_type** relationship 52](#_Toc536316445)

[Table 3‑7. Specification of **device** relationship 53](#_Toc536316446)

[Table 3‑8. Specification of **sensor\_type** relationship 54](#_Toc536316447)

[Table 3‑9. Specification of **sensor** relationship 54](#_Toc536316448)

[Table 3‑10. Specification of **sensor\_data** relationship 54](#_Toc536316449)

[Table 3‑11. Specification of **actuator\_type** relationship 55](#_Toc536316450)

[Table 3‑12. Specification of **actuator** relationship 55](#_Toc536316451)

[Table 3‑13. Specification of **command** relationship 56](#_Toc536316452)

[Table 3‑14. Specification of the **Register** use case 61](#_Toc536316453)

[Table 3‑15. Specification of the **Login** use case 61](#_Toc536316454)

[Table 3‑16. Specification of the **Manage Properties** use case 62](#_Toc536316455)

[Table 3‑17. Specification of the **Manage Customer Accounts** use case 62](#_Toc536316456)

[Table 3‑18. Specification of the **Perform Maintenance Operations** use case 63](#_Toc536316457)

[Table 3‑19. Specification of the **Manage Technical Moderator Accounts** use case 64](#_Toc536316458)

# Introduction

## Problem statement

In recent years, smart agriculture is one of the key development directions to address the challenges of the agricultural sector (improving quality, increasing output,..). And "hydroponics" is an important branch of smart agriculture with the ability to increase production 10 times more than traditional agriculture, without using land, avoiding the use of pesticides. Những năm gần đây, nông nghiệp thông minh đang là một trong những hướng phát triển trọng điểm nhằm giải quyết các thách thức của ngành nông nghiệp (nâng cao chất lượng, tăng sản lượng,…). Và “thủy canh” là một nhánh quan trọng trong nông nghiệp thông minh với khả năng nâng cao sản lượng gấp 10 lần nông nghiệp truyền thống, lại không cần sử dụng đất, tránh được việc sử dụng thuốc bảo vệ thực vật.

Those advantages are multiplied through the application of Internet of Things (IoT) technology to hydroponic systems. Many developed countries have developed hydroponic systems that use IoT solutions to collect crop data, analyze those data and make decisions to control plant habitats. Những lợi thế đó được nhân lên nhiều lần thông qua việc ứng dụng công nghệ Internet of Things (IoT) vào những hệ thống thủy canh. Rất nhiều những nước phát triển đã xây dựng những hệ thống thủy canh sử dụng các giải pháp IoT để thu thập dữ liệu cây trồng, phân tích các dữ liệu đó và đưa ra những quyết định kiểm soát môi trường sống của thực vật.

In Vietnam, however, despite having been available since 2012, hydroponic production is still very limited compared to traditional agriculture. Although hydroponic farming has started to grow rapidly since 2015, in the form of small vegetable platforms on the roofs of buildings or condominiums, with limited IoT impletemented IoT, there are sttill several drawbacks: Không giống như các nước phát triển, sản xuất thủy canh ở Việt Nam vẫn còn rất hạn chế so với hình thức nông nghiệp truyền thống dù rằng “thủy canh” đã có mặt tại nước ta từ năm 2012. Gần đây, kể từ năm 2015, thủy canh đang bắt đầu phát triển nhanh chóng dưới hình thức những giàn trồng rau nhỏ trên sân thượng các tòa nhà hay khu chung cư. Đã có một số những giải pháp ứng dụng IoT với các vườn rau mini đó song vẫn có những nhược điểm như:

Lack of automated analysis and control procedures. Thiếu các quy trình phân tích và điều khiển tự động.

Low productivity. Chưa đạt được hiệu suất cao.

Can only be used in small gardens. Chỉ có thể sử dụng cho những vườn nhỏ.

Therefore, to increase hydroponic production in Vietnam, as well as other developing countries, it is necessary to apply IoT solutions with: Do đó, để tăng sản lượng thủy canh ở Việt Nam cũng như các nước đang phát triển khác thì điều quan trọng nhất là có thể áp dụng được các giải pháp IoT với những yêu cầu:

Low cost: includes both initial and operating costs. Chi phí thấp: bao gồm cả chi phí ban đầu và chi phí vận hành.

Expandable: suitable for small to large models. Có khả năng mở rộng: thích hợp với các mô hình từ cỡ nhở đến lớn.

High-tech applications that allow farmers with limited hydroponics experience to be able to operate their hydroponic systems easily. Ứng dụng công nghệ cao, cho phép người nông dân không có nhiều kinh nghiệm thủy canh vẫn có thể vận hành hệ thống thủy canh của họ một cách dễ dàng.

High yields and quality while minimizing production costs. Sản lượng cao đi kèm với chất lượng cao trong khi giảm thiểu chi phí sản xuất.

To meet these requirements, we will present a hydroponic system model that includes the following modules: Để đáp ứng các yêu cầu này, đề tài này sẽ đưa ra một mô hình thống thủy canh bao gồm những phân hệ sau:

IoT infrastructure module: Phân hệ hạ tầng IoT:

* + Include IoT devices, sensors, devices that can affect the plant's habitat. Bao gồm các thiết bị IoT, các cảm biến, các thiết bị có khả năng gây ảnh hưởng đến môi trường sống của cây.
  + Tasked with collecting data from the sensor, receiving control commands and executing control mechanisms. Phân hệ này sẽ có các nhiệm vụ gồm: thu thập dữ liệu từ cảm biến, nhận các lệnh điều khiển và thực hiện cơ chế điều khiển.

Connection managing module: Phân hệ quản lý kết nối:

* + The medium between the device and the data processing layer: Includes a mechanism for managing the delivery of messages from the device side. Bao gồm cơ chế quản lý việc gửi nhận các bản tin từ phía thiết bị, là trung gian giữa thiết bị và tầng xử lý dữ liệu.

Data processing module: Phân hệ xử lý dữ liệu:

* + Process and manage data. Phân hệ này gồm hai nhiệm vụ chính là phân tích dữ liệu và quản lý dữ liệu.
  + Received data will be analyzed and evaluated to make control decisions for the equipment. Dữ liệu được gửi về sẽ được phân tích và đánh giá nhằm đưa ra các quyết định điều khiển cho các thiết bị.
  + Management applications help users easily manage objects, equipment. Ứng dụng quản lý giúp người dùng dễ dàng quản lý các đối tượng, thiết bị.

## Techniques for growing hydroponic vegetables

Hydroponics is a technique of cultivating non-land plants that are planted directly into a nutrient environment or substrate that is not soil. The substrates can be sand, rice husk, coconut shell, peat... hydroponic technology is one of the modern gardening. Thủy canh là kỹ thuật trồng cây không dùng đất mà trồng trực tiếp vào môi trường dinh dưỡng hoặc giá thể mà không phải là đất. Các giá thể có thể là cát, trấu, vỏ xơ dừa, than bùn... kỹ thuật thủy canh là một trong những nghề làm vườn hiện đại.

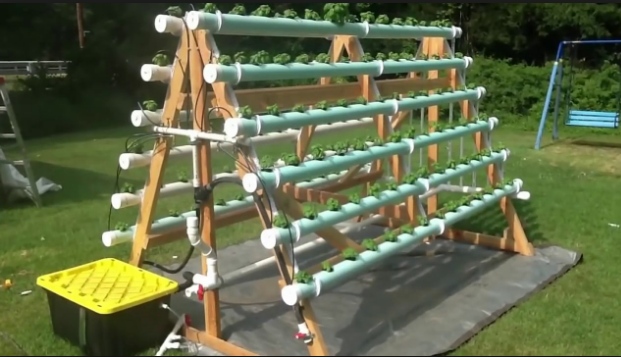


Figure 1‑1. An example of a hydroponic plantation

The secret of this technique is to provide enough and timely plants for essential mineral elements. Provide adequate food, ensure sufficient light, CO2 for photosynthesis, O2 for respiration, plants can grow healthy according to the growers' wishes. Bí quyết của kỹ thuật này là cung cấp đủ và đúng lúc cho cây trồng các nguyên tố khoáng cần thiết. Cung cấp đầy đủ cái ăn, đảm bảo đủ ánh sáng, CO2 cho quá trình quang hợp, O2 cho quá trình hô hấp, cây trồng có thể phát triển khỏe mạnh theo ý muốn người trồng.

Hydroponic technology has been studied since the 17th century, until now this technology has been completed, towards clean, green, unpolluted agricultural products. Hydroponic vegetable cultivation can grow both in small scale and commercial scale. Công nghệ thủy canh đã được nghiên cứu từ thế kỷ 17, đến nay công nghệ này đã hoàn thiện, hướng đến nông sản sạch, xanh, không ô nhiễm. Trồng rau thủy canh có thể phát triển cả quy mô nhỏ lẻ và quy mô thương mại.

Some basic hydroponic farming models: Một số mô hình canh tác thủy canh cơ bản:

Wick type system. Hệ thống dạng bấc.

Static hydroponic system. Hệ thống thủy canh tĩnh.

The system is flooded and periodically withdraws. Hệ thống ngập và rút định kỳ.

Drip system. Hệ thống nhỏ giọt.

NFT nutrient membrane system. Hệ thống màng dinh dưỡng NFT.

Aeroponics. Khí canh.

Advantages of hydroponic vegetable cultivation: Ưu điểm của trồng rau thủy canh:

No need to do soil. Không phải làm đất.

No weeds. Không có cỏ dại.

Many crops can be grown, possibly off-season. Trồng được nhiều vụ, có thể trái vụ.

No need to water. Không cần tưới.

No need to use pesticides. Không cần sử dụng thuốc trừ sâu bệnh.

Productivity higher than 25% to 50%. Năng suất cao hơn 25% đến 50%.

Disadvantages of hydroponic vegetable cultivation: Nhược điểm của trồng rau thủy canh:

The cost of installing systems and materials for growing hydroponic vegetables is quite high. Chi phí lắp đặt hệ thống và nguyên liệu trồng rau thủy canh khá cao.

Hydroponic production uses only nutrient solution, it is mixed with appropriate proportions with necessary element for plant development, structure of organic materials such as N, K, P, Na, Ca. The solutions are of type A and B; a pH up solution, which plays a role in increasing the pH when the pH in the nutrient solution is low; a pH down solution, which plays a role in reducing the pH when the pH in the nutrient solution is high. Sản xuất thủy canh chỉ sử dụng dung dịch dinh dưỡng, nó được pha với tỉ lệ thích hợp với thành phần nguyên tố cần thiết cho sự phát triển cây trồng, cấu tạo lên các vật chất hữu cơ như N, K, P, Na, Ca. Các dung dịch có dạng A, B; một dung dịch pH up, có vai trò tăng độ pH lên khi độ pH trong dung dịch dinh dưỡng thấp; một dung dịch pH down, có vai trò giảm độ pH xuống khi độ pH trong dung dịch dinh dưỡng cao.

Solutions A and B will be mixed into water at different rates according to the EC (electro-conductivity) of each crop. Các dung dịch A, B sẽ được pha trộn vào nước với tỷ lệ khác nhau theo EC (electro-conductivity) của mỗi loại cây trồng.

The EC (electro-conductivity) index is an index that describes the total concentration of ions dissolved in the solution. EC can be expressed in a number of different units, but typical units are used to measure EC as millisiemens per centimeter (mS / cm). The EC index does not describe the concentration of each substance in the solution nor does it indicate the level of balance of nutrients in the solution. Chỉ số EC (electro-conductivity) là chỉ số diễn tả tổng nồng độ ion hòa tan trong dung dịch. EC có thể được thể hiện bằng một số đơn vị khác nhau nhưng đơn vị tiêu biểu được dùng để đo lường EC là millisiemens trên centimet (mS / cm). Chỉ số EC không diễn tả nồng độ của từng chất trong dung dịch đồng thời cũng không thể hiện mức độ cân bằng của các chất dinh dưỡng trong dung dịch.

Table 1‑1. EC index of some common plants

|  |  |
| --- | --- |
| **Plant Cây trồng** | **EC (ms/cm)** |
| Carnation Cẩm chướng | 2.4 – 5.0 |
| Cymbidium Địa lan | 0.6 – 1.5 |
| Rose Hoa hồng | 1.5 – 2.4 |
| Tomato Cà chua | 2.4 – 5.0 |
| Salad Xà-lách | 0.6 – 1.5 |
| Strawberry Dâu tây | 1.5 – 2.4 |
| Chili Ớt | 1.5 – 2.4 |

During the growth process, plants absorb the minerals they need. Therefore, maintaining EC at a stable level is very important. If the solution has a high EC index, the plant's water uptake takes place faster than mineral absorption. This increases the concentration of the solution and causes poisoning to the plant. Then we have to add water to the environment. Conversely, if the EC is low, the plant will absorb minerals faster than water absorption. At that time, the concentration of solution decreases sharply, the plant will not be provided with enough minerals, slow growth and poor growth. Trong suốt quá trình tăng trưởng, cây hấp thu khoáng chất mà chúng cần. Do vậy duy trì EC ở một mức ổn định là rất quan trọng. Nếu dung dịch có chỉ số EC cao thì sự hấp thu nước của cây diễn ra nhanh hơn sự hấp thu khoáng chất. Điều này làm nồng độ dung dịch tăng cao và gây ngộ độc cho cây. Khi đó ta phải bổ sung thêm nước vào môi trường. Ngược lại, nếu EC thấp, cây sẽ hấp thu khoáng chất nhanh hơn hấp thu nước. Khi đó, nồng độ dung dịch giảm mạnh, cây sẽ không được cung cấp đầy đủ khoáng chất, chậm lớn và phát triển kém.

## Growth characteristics of some hydroponic crops

### Lettuce

Lettuce belongs to the root system of eating piles, eating shallow on the soil surface, eating 20-30cm wide, so the tree does not flood. Usually there are 10-24 yellow or green flowers on a tree. Flowers bloom from the time of sunlight to noon, pollination is best at 9-10am. Rau xà lách thuộc hệ rễ ăn cọc, ăn nông trên bề mặt đất, ăn rộng 20-30cm, bởi vậy cây không chịu ngập úng. Thường có 10-24 hoa màu vàng hay vàng lục trên một cây. Hoa nở từ lúc có ánh sáng mặt trời đến trưa, thụ phấn tốt nhất lúc 9-10h sáng.

Temperature: Nhiệt độ:

Appropriate temperature for growth and vegetation of lettuce is 15-18oC. Nhiệt độ thích hợp cho sự sinh trưởng, sinh dưỡng của xà lách là 15-18oC.

The appropriate temperature for plants to grow during the day is 20oC. Nhiệt độ thích hợp cho cây phát triển vào ban ngày là 20oC.

The appropriate temperature for plants to grow at night is 18oC. Nhiệt độ thích hợp cho cây phát triển vào ban đêm là 18oC.

High temperature above 22oC makes seeds germ long and reduces the quality of leaves and corn. Nhiệt độ cao trên 22oC làm mầm hạt kéo dài và làm giảm chất lượng của lá và bắp.

pH: Độ pH:

The most appropriate pH for lettuce is 6-6.5. Độ pH thích hợp nhất cho xà lách là 6-6.5.

pH < 5.5 plants only absorb medium and microelements: Fe, Mn, Bo, Cu, Zn, leading to stunted plants because they cannot absorb the plurality of elements N, P, K, Ca, Mg, Mo. Độ pH < 5.5 cây chỉ hấp thụ được trung lượng và vi lượng: Fe, Mn, Bo, Cu, Zn, dẫn đến cây còi cọc do không hấp thụ được nguyên tố đa lượng N, P, K, Ca, Mg, Mo.

pH > 7 plants absorb the plurality of elements but less absorb Fe, Mn, Bo, Cu, Zn, leading to yellow leaves. Độ pH > 7 cây hấp thụ nguyên tố đa lượng nhưng lại kém hấp thụ Fe, Mn, Bo, Cu, Zn, dẫn đến lá cây vàng vọt.

CO2:

Particularly with lettuce, CO2 is very important for growing plants. Riêng với xà lách khí CO2 rất quan trọng cho cây sinh trưởng.

In greenhouses people spray CO2 at a concentration of 1000–1500 (ppm). Trong nhà kính người ta phun CO2 với nồng độ 1000–1500 (ppm).

Humidity and lighting: Độ ẩm và ánh sáng:

Lettuce is a moist-loving tree, air humidity is 65% -75%. Xà lách là cây ưa ẩm, độ ẩm không khí là 65%-75%.

Often the first stage of the plant needs more light than the next stage. Thường giai đoạn đầu của cây cần ánh sáng nhiều hơn giai đoạn sau.

Lighting 16h in the first 10 days significantly increases productivity. Chiếu sáng 16h trong 10 ngày đầu làm tăng năng suất đáng kể.

### Tomato

Tomato has two types of limited and infinite growth. Flowers are yellow, grow in clusters of 3 to 30 flowers. Flowering after planting 50-70 days. Flowering time 10-55 days. Cà chua có hai loại sinh trưởng có hạn và vô hạn. Hoa màu vàng, mọc thành chùm từ 3 đến 30 hoa. Ra hoa sau khi trồng 50-70 ngày. Thời gian ra hoa 10-55 ngày.

Temperature: Nhiệt độ:

Suitable temperature for high yielding tomatoes, good quality is about 21-24oC. Nhiệt độ thích hợp cho cà chua có năng suất cao, chất lượng tốt là khoảng 21-24oC.

If the night temperature is lower than 4-5oC, the plants give more flowers. Nếu nhiệt độ đêm thấp hơn ngày 4-5oC thì cây cho nhiều hoa hơn.

Temperatures below 12oC will cause serious damage. Nhiệt độ dưới 12oC kéo dài sẽ gây thiệt hại nghiêm trọng.

Prolonged temperature above 27oC also limits flowering and fruiting. Nhiệt độ trên 27oC kéo dài cũng hạn chế ra hoa, đậu quả.

Tomato plants will die if the temperature exceeds 38oC. Cây cà chua sẽ chết nếu nhiệt độ vượt ngưỡng 38oC.

Before and after the pollination period if the night temperature is > 21oC, the ability to flower is reduced. Trước và sau thời gian thụ phấn nếu nhiệt độ ban đêm > 21oC thì giảm khả năng ra hoa.

At the seeding stage of tomato seeds, the best temperature to germinate tomato seeds is from 20-28oC. Ở giai đoạn gieo mầm hạt cà chua, nhiệt độ tốt nhất để hạt cà chua nảy mầm là từ 20-28oC.

After germination, the temperature is suitable for the seedlings to grow from 15-18oC. Sau khi nảy mầm thì nhiệt độ thích hợp để cây con sinh trưởng là từ 15-18oC.

pH: Độ pH:

Tomato grows best in an environment with a pH of about 6-6.5. Cà chua phát triển tốt nhất trong môi trường có độ pH vào khoảng 6-6.5.

High humidity and prolonged flooding will reduce growth potential. Độ ẩm cao và ngập nước kéo dài sẽ làm giảm khả năng sinh trưởng.

Soil pH less than 5 or higher than 9 can be toxic to plants. Độ pH của đất thấp hơn 5 hoặc cao hơn 9 có thể gây độc cho cây.

CO2

Appropriate CO2 concentration for tomatoes is about 800-1,000 (ppm). Nồng độ CO2 phù hợp cho cà chua vào khoảng 800-1.000 (ppm).

Humidity and lighting: Độ ẩm và ánh sáng:

The best air humidity for tomatoes is about 45-60%. Độ ẩm không khí tốt nhất cho cà chua vào khoảng 45-60%.

Tomato plants need to be exposed to light 6-8 hours/day. Cây cà chua cần được tiếp xúc với ánh sáng 6-8 tiếng/ngày.

Thus, each type of crop has different growth conditions, in each different stage. This leads to the need to have appropriate care control mechanisms for each type of crop in each time of their growth. Như vậy, mỗi loại cây trồng đều có những điều kiện sinh trưởng khác nhau, trong từng giai đoạn khác nhau. Điều này dẫn tới nhu cầu cần phải có cơ chế điều khiển chăm sóc phù hợp cho từng loại cây trồng trong từng thời điểm sinh trưởng của chúng.

## Survey of the existing IoT application hydroponics system

There are basic parameters that need to be monitored and adjusted for any type of plants. Hence, even high-tech hydroponic systems will have some similar functions. Therefore, to capture the basic functions of a hydroponic system, we will examine the systems already in the market. Dù là loại cây trồng nào cũng sẽ có những thông số cơ bản cần được theo dõi và điều chỉnh. Do đó, về cơ bản, các hệ thống thủy canh công nghệ cao sẽ có những chức năng gần tương tự nhau. Vì vậy, để nắm bắt được các chức năng cơ bản của một hệ thống thủy canh, ta sẽ tiến hành khảo sát các hệ thống đã có mặt trên thị trường.

On the market today, there are some products that combine IoT and hydroponic plant production such as Hachi, Greenbot or Lisado, and Hachi is most prominent one. Trên thị trường hiện nay có một số sản phẩm trồng rau thủy canh kết hợp IoT như Hachi, Greenbot, Lisado. Trong đó Hachi nổi bật nhất với độ hoàn thiện của sản phẩm:

* 1. Hachi's product suit targets two main customers - urban residents and farm owners. Bộ sản phẩm của Hachi nhắm đến hai đối tượng khách hàng chính là cư dân đô thị và chủ trang trại.
  2. Urban-house version includes features like: Phiên bản dành cho nhà phố, với các chức năng:
  + Monitor humidity, soil moisture. Giám sát độ ẩm không khí, độ ẩm đất.
  + Monitor light intensity. Giám sát cường độ ánh sáng.
  + Issue warnings and directions when environmental conditions exceed the warning level. Đưa ra cảnh báo và hướng xử lý khi các điều kiện môi trường vượt qua mức cảnh báo.
  + Have control mode option so that users can easily set the appropriate factor and alert conditions most suitable to the plants. Thiết lập chế độ điều khiển tùy chọn để người dùng có thể dễ dàng thiết lập các yếu tố và điều kiện cảnh báo phù hợp với loại cây trồng nhất.
  + Control devices such as pumps, LEDs or cameras through smartphones and Internet. Điều khiển các thiết bị như máy bơm, đèn LED, máy sục, Camera thông qua smartphone tại bất kỳ đâu có mạng.
  1. Large-farm version include features like: Phiên bản dành cho trang trại lớn, với các chức năng:
  + Monitor humidity, soil moisture. Giám sát độ ẩm không khí, độ ẩm.
  + Monitor light intensity. Giám sát cường độ ánh sáng.
  + Monitor pH in water. Giám sát nồng độ pH trong nước.
  + Issue warnings and directions when environmental conditions exceed the warning level. Đưa ra cảnh báo và hướng xử lý khi các điều kiện môi trường vượt qua mức cảnh báo.
  + Have control mode option so that users can easily set the appropriate factor and alert conditions most suitable to the plants. Thiết lập chế độ điều khiển tùy chọn để người dùng có thể dễ dàng thiết lập các yếu tố và điều kiện cảnh báo phù hợp với loại cây trồng của mình nhất.
  + Control devices such as pumps, LEDs or cameras through smartphones and Internet. Điều khiển các thiết bị như máy bơm, đèn LED, máy sục, Camera thông qua smartphone tại bất kỳ đâu có kết nối mạng.
  + Support Barcode for easy traceability of agricultural products. Hỗ trợ Qrcode để dễ dàng truy xuất nguồn gốc nông sản.
  + Expected to use the Zigbee wireless connection protocol to connect slave nodes together and use Wi-Fi to connect to the master node. Dự tính sẽ sử dụng giao thức kết nối không dây Zigbee để kết nối các node với nhau và sử dụng wifi để kết nối với node chủ.

Hachi's products can monitor the factors that affect the growth of plants such as temperature, humidity, light intensity, pH level, give notice to the user, predict the harvest date. The user can also control plant-monitoring devices semi-automatically. However, Hachi's products are not capable of analyzing data to provide control strategies to save system resources. Sản phẩm của Hachi đã giám sát được các yếu tố ảnh hưởng đến sự sinh trưởng của cây như nhiệt độ, độ ẩm, cường độ sáng, nồng độ pH, đưa ra được thông báo cho người dùng, dự đoán ngày thu hoạch. Bên cạnh đó người dùng có thể điều khiển ngược lại các thiết bị tại giàn đề chăm sóc cây nhưng mới chỉ là bán tự động. Ngoài ra sản phẩm của Hachi chưa có khả năng phân tích dữ liệu để đưa ra chiến lược điều khiển nhằm tiết kiệm các tài nguyên hệ thống.

**Therefore, a fully automatic vegetable cultivation system, capable of self-analyzing and supporting users who has little knowledge of cultivation is not available. Vì vậy, một hệ thống canh tác rau hoàn toàn tự động, có khả năng tự phân tích và hỗ trợ người dùng có ít kiến thức về canh tác là không có sẵn.**

## General requirements of the IoT application hydroponics system

After examining the existing hydroponic systems on the market, it can be seen that the general requirements of a hydroponic system will include: Sau khi khảo sát các hệ thống thủy canh hiện có trên thị trường, có thể nhận thấy những yêu cầu chung của một hệ thống thủy canh sẽ bao gồm:

Device authentication to ensure that information sent to the server is correctly corresponded to the ones on the monitored frames. Xác thực thiết bị để đảm bảo thông tin gửi về server là chính xác của thiết bị trên giàn đang giám sát.

The system consists of control circuits capable of controlling the device via the control pins and using Wi-fi to communicate with the server. Hệ thống bao gồm các thiết bị có khả năng thu thập dữ liệu từ các cảm biến và nhận lệnh điều khiển thông qua khả năng kết nối mạng không dây (wifi, 3G) giữa bản thân và server.

The system includes sensors that measure the environmental parameters that may affect the development of the plant and send data to the processor server. Hệ thống bao gồm các cảm biến đo những thông số môi trường có khả năng ảnh hưởng đến sự phát triển của cây trồng và gửi dữ liệu về máy chủ xử lý.

The system is scalable with larger models and is suitable for a wide variety of control circuits. Hệ thống đảm báo tính dễ mở rộng với mô hình lớn hơn và phù hợp cho nhiều loại thiết bị thu thập điều khiển khác nhau.

The system ensures real-time accuracy. Hệ thống đảm bảo tính thời gian thực.

The system is fully automatic, but users can manually control it if desired. Hệ thống hoàn toàn tự động, đồng thời người dùng cũng có thể điều khiển bằng tay nếu muốn.

The system can analyze sent data to make resource-saving decisions. Hệ thống có khả năng phân tích dữ liệu gửi về để đưa ra những quyết định giúp tiết kiệm tài nguyên.

The system includes an object management module that helps users capture information of objects in the system. Hệ thống bao gồm các module quản lý đối tượng giúp người dùng nắm bắt thông tin của các đối tượng trong hệ thống.

The system retains a control log. Hệ thống có lưu trữ lại các thao tác điều khiển từ cả phía người dùng và tự động.

The system can display diagrams of real time measured data. Hệ thống có khả năng hiển thị biểu đồ giá trị các thông số đo được theo thời gian thực.

These are the most common requirements of an IoT application system that fully meets the criteria outlined in the question set. No current system has fully met these requirements. Đây là những yêu cầu chung nhất của một hệ thống thủy canh ứng dụng IoT đáp ứng đầy đủ những tiêu chí đã nêu trong phần đặt vấn đề. Có thể thấy, các hệ thống hiện nay chưa đáp ứng được hoàn toàn những yêu cầu này.

## Solution

To meet the above requirements, we will need to address the following issues in the scope of this project: Để đáp ứng được những yêu cầu đã nêu trên, đề tài này sẽ hướng tới xây dựng một hệ thống giải quyết các vấn đề sau:

The system should be able to allow users with minimal expertise to grow and harvest vegetables of good quality. Hệ thống phải hỗ trợ được người dùng không có kiến thức chuyên môn để vẫn có thể trồng và thu hoạch rau quả với chất lượng tốt.

Raising plant in the form of hydroponic. Trồng cây dưới dạng thủy canh.

The sensor system monitors important parameters for hydroponic plants. Hệ thống các cảm biến giám sát các chỉ số quan trọng đối với cây trồng thủy canh.

The server-side application receives data to controls the corresponding devices so that the plant gets the best state for growth. Ứng dụng phía server tiếp nhận dữ liệu và dựa trên dữ liệu đó điều khiển các thiết bị tương ứng để cây trồng có được trạng thái tốt nhất cho sự tăng trưởng.

Data to be collected includes: temperature, humidity, CO2, pH. Các dữ liệu cần thu thập bao gồm: nhiệt độ, độ ẩm, nồng độ dinh dưỡng trong dung dịch TDS, nồng độ pH.

A client-side application that includes mobile and web-based platforms to make it easy for users to track crop activity and intervene if they want. Ứng dụng phía client bao gồm nền tảng di động và nền tảng web giúp người dùng dễ dàng theo dõi được diễn biến của cây trồng và can thiệp nếu người đó có chuyên môn.

Use communication protocols to send data between control circuits to the data processing server. Sử dụng các giao thức truyền thông gửi nhận dữ liệu giữa các mạch điều khiển với máy chủ xử lý dữ liệu.

Use optimization algorithms to save system resources. Áp dụng một số thuật toán tối ưu nhằm phân tích dữ liệu gửi về và đưa ra các quyết định giúp tiết kiệm tài nguyên hệ thống.

Have a database to store information of each plant in each stage of development. Thiết kế CSDL có khả năng lưu trữ thông tin của từng loại cây trồng trong từng giai đoạn phát triển.

These are the basic issues that need to be solved when building an IoT hydroponic system, and we will need to come up with solutions to address these issues. Đây là các vấn đề cơ bản cần giải quyết được khi xây dựng một hệ thống thủy canh ứng dụng IoT, từ đó hệ thống được xây dựng cần tiến hành giải quyết các vấn đề này.

# System analysis and design

## Analysis of system requirements

Through practical surveys and proposed solutions, the built system will meet the following requirements: Thông qua khảo sát thực tiễn và giải pháp đề xuất, hệ thống được xây dựng sẽ đáp ứng các yêu cầu sau:

The overall system consists of 2 components: Hệ thống tổng thể gồm 2 thành phần:

* + The devices have two main functions: (1) data collection is the environmental parameters monitored during plant growth and development to be sent to the server; (2) Receiving control commands from the sending server and executing commands to control the actuating facilities that are paired with the device. Các thiết bị có 2 chức năng chính: (1) thu thập dữ liệu là các thông số môi trường được theo dõi trong quá trình sinh trưởng và phát triển của cây trồng để gửi lên server; (2) nhận các lệnh điều khiển từ server gửi xuống và thực thi các lệnh điều khiển các cơ cấp chấp hành được ghép nối với thiết bị.
  + Server collects monitoring data from sent devices, performs storage, data management, data analysis, control commands to devices. Web services allow data management and monitoring and control of the entire process of plant growth and development. Server thu thập dữ liệu giám sát từ các thiết bị gửi về, thực hiện lưu trữ, quản lý dữ liệu, phân tích dữ liệu, ra lệnh điều khiển đến các thiết bị. Dịch vụ web cho phép quản lý dữ liệu và giám sát, điều khiển toàn bộ quá trình sinh trưởng phát triển của cây trồng.

Server application: Ứng dụng phía server:

* + Ability to authenticate devices to ensure that only registered devices are allowed to participate in the system. Có khả năng xác thực thiết bị để đảm bảo chỉ cho phép các thiết bị đã đăng ký mới được phép tham gia vào hệ thống.
  + Receiving sensor data from all devices sent in real time to monitor and monitor plant parameters at the same time of various plant and plant platforms. Tiếp nhận dữ liệu cảm biến từ tất cả các thiết bị gửi lên theo thời gian thực để theo dõi giám sát các thông số cây trồng đồng thời của nhiều giàn cây, vườn cây trồng khác nhau.
  + Organize data storage of many different plant rigs. Tổ chức lưu trữ dữ liệu của nhiều giàn cây trồng khác nhau.
  + Providing web services for data management, data presentation (chart of monitoring parameters). Cung cấp dịch vụ web để quản lý dữ liệu, trình bày dữ liệu (biểu đồ các thông số giám sát).
  + Provides a control feature interface so that users can dictate device control, or integrate with automatic control features of tree care. Cung cấp giao diện tính năng điều khiển để người dùng có thể ra lệnh điều khiển thiết bị, hoặc tích hợp với các tính năng điều khiển tự động quá trình chăm sóc cây.
  + Administration, user authorization, device administration register with the system. Quản trị, phân quyền người dùng, quản trị thiết bị đăng ký với hệ thống.

The side of the equipment collected and controlled: Phía các thiết bị thu thập và điều khiển:

* + Devices to collect and control are capable of controlling actuators such as nutrient-phase pump, nutrient pump, light, heating, cooling fan, etc. The control pins and the ability to connect to wifi, send and receive data between themselves and the server. Các thiết bị thu thập, điều khiển có khả năng điều khiển các cơ cấu chấp hành (actuators) như bơm pha dung dịch dinh dưỡng, bơm dung dịch dinh dưỡng, đèn chiếu sáng, đèn sưởi ấm, quạt làm mát, … thông qua các chân điều khiển và có khả năng kết nối wifi, gửi nhận dữ liệu giữa bản thân và server.
  + Collecting devices can be paired, using flexible sensors to measure environmental parameters affecting plant growth and development such as temperature, humidity, pH, nutrient concentration. in TDS solution,... Các thiết bị thu thập có thể ghép nối, sử dụng được linh hoạt các cảm biến đo những thông số môi trường ảnh hưởng đến sự sinh trưởng phát triển của cây trong như nhiệt độ, độ ẩm, nồng độ pH, nồng độ dinh dưỡng trong dung dịch TDS,…
  + Devices that communicate with the server via the internet, are able to connect to wireless networks via WiFi or 3G telecommunications networks. Các thiết bị giao tiếp với server thông qua mạng internet, có khả năng kết nối mạng không dây qua WiFi hoặc mạng viễn thông 3G.
  + The device collects sensor data and sends collected data to the server periodically, ensuring real-time availability. Các thiết bị thu thập dữ liệu cảm biến theo và gửi dữ liệu thu thập lên server theo định kỳ, đảm bảo tính thời gian thực.

Other requirements: Các yêu cầu khác:

* + The system is fully automated, and users can also control it manually if desired. Hệ thống hoàn toàn tự động, đồng thời người dùng cũng có thể điều khiển bằng tay nếu muốn.
  + The system has the ability to analyze data sent to make decisions to save resources. Hệ thống có khả năng phân tích dữ liệu gửi về để đưa ra những quyết định giúp tiết kiệm tài nguyên.
  + The system includes object management modules that help users capture information of objects in the system. Hệ thống bao gồm các module quản lý đối tượng giúp người dùng nắm bắt thông tin của các đối tượng trong hệ thống.
  + The system retains control operations from both the user and the automatic. Hệ thống có lưu trữ lại các thao tác điều khiển từ cả phía người dùng và tự động.

The main tasks need to be done: Các công việc chính cần thực hiện:

1. Build applications on the server including the main functions: Xây dựng ứng dụng trên server gồm các chức năng chính:
   * Provide communication services between the server and the collection and control devices included: Cung cấp các dịch vụ giao tiếp giữa server với các thiết bị thu thập và điều khiển bao gồm:
     + The device authentication service has been registered before joining the system. Dịch vụ xác thực thiết bị đã được đăng ký trước khi tham gia vào hệ thống.
     + Service of collecting periodic sensor data from posted devices. Dịch vụ thu thập dữ liệu cảm biến định kỳ từ các thiết bị gửi lên.
     + The service commands the control to the specified device. Dịch vụ ra lệnh điều khiển tới các thiết bị được chỉ định.
     + Service attendance status of devices (KeepAlive). Dịch vụ điểm danh trạng thái hoạt động của các thiết bị (KeepAlive).
   * Organize data, store data collected in the database. Tổ chức dữ liệu, lưu trữ dữ liệu được thu thập trong cơ sở dữ liệu.
   * Build web services that allow data management, display collected information and be able to send commands to control devices for the server. Xây dựng dịch vụ web cho phép quản lý dữ liệu, hiển thị thông tin thu thập và có khả năng gửi lệnh điều khiển các thiết bị cho server.
   * Data analysis functions are integrated as additional modules into the system. This function allows automatic control of the plant care process. Chức năng phân tích dữ liệu (data analysis) được tích hợp dưới dạng mô-đun bổ sung vào hệ thống. Chức năng này cho phép điều khiển tự động quá trình chăm sóc cây trồng.
2. Build the firmware on the device to collect sensor data, communicate with the server (send sensor data, receive control commands), and execute control commands to actuators. Xây dựng firmware trên thiết bị để thu thập dữ liệu sensors, giao tiếp với server (gửi dữ liệu sensors, nhận lệnh điều khiển), và thực hiện các lệnh điều khiển đến các cơ cấu chấp hành (actuators).

## System architecture design

In order to be able to develop the system components to ensure the requirements, the first important job is to design the overall architecture of the system. The construction of the overall architecture for the system helps to control errors and unexpected cases of the system, predicting changes that affect the construction and development of the system from which there are Specific measures for each case. The construction of the overall architecture of the system includes the design of the overall diagram for the system, the construction of the overall architecture and the design of the functional components that need to be met by the system. Để có thể triển khai xây dựng được các thành phần hệ thống đảm bảo yêu cầu đặt ra, công việc quan trọng đầu tiên là thiết kế kiến trúc tổng thể của hệ thống. Việc xây dựng kiến trúc tổng thể cho hệ thống giúp cho việc kiểm soát sai sót và các trường hợp không mong muốn của hệ thống, dự đoán được những thay đổi làm ảnh hưởng đến việc xây dựng và phát triển hệ thống từ đó có những biện pháp cụ thể cho từng trường hợp. Việc xây dựng kiến trúc tổng thể của hệ thống bao gồm việc thiết kế sơ đồ tổng quan cho hệ thống, xây dựng kiến trúc tổng thể và thiết kế các thành phần chức năng cần đáp ứng của hệ thống.

### Overview diagram of the system

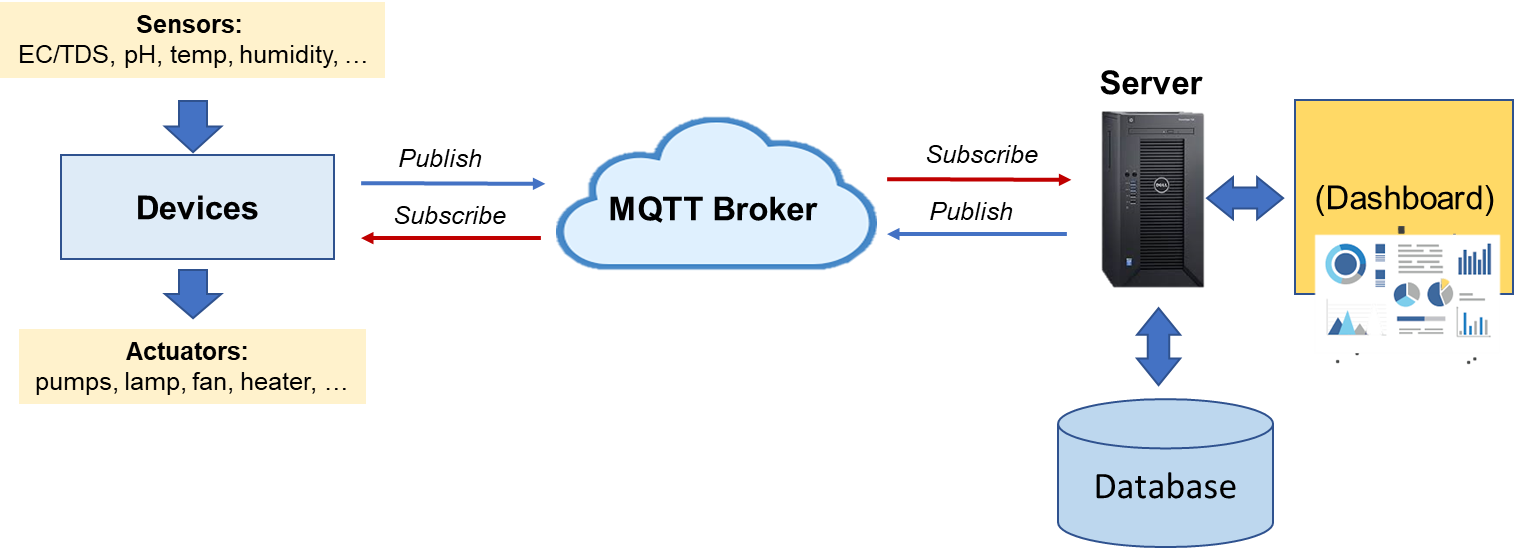


Figure 2‑1. Overview diagram of the system

The overall model of the system consists of 2 components: Mô hình tổng quan của hệ thống gồm có 2 thành phần:

Devices are low-cost embedded computers designed with sensors to collect parameters to monitor the planting process, and to pair with actuators ( actuators) to carry out control of the tree care process. Các thiết bị là các máy tính nhúng giá thành rẻ được thiết kế với chức năng ghép nối với các cảm biến (sensors) để thu thập các thông số giám sát quá trình trồng cây, đồng thời ghép nối với các cơ cấu chấp hành (actuators) để thực hiện điều khiển quá trình chăm sóc cây.

A server with application collects data from the device, stores data with the database and displays, displays data monitoring the growth of the plant for each plant, each set selected; It is possible to order automatic control of the tree care process by analyzing collected data. Một server với ứng dụng thu thập dữ liệu gửi lên từ các thiết bị, lưu trữ dữ liệu với cơ sở dữ liệu và trình bày, hiển thị dữ liệu giám sát quá trình sinh trưởng của cây trồng ứng với mỗi giàn cây, mỗi thiết bị được lựa chọn; có khả năng ra lệnh điều khiển quá trình chăm sóc cây tự động nhờ vào phân tích dữ liệu thu thập.

Devices and servers are designed to communicate with each other through an MQTT broker intermediary based on the MQTT protocol to transmit data between devices with the server. MQTT protocol is a very suitable choice for data transfer problem in IoT technology application. MQTT is suitable for transferring small size packets with high frequency from many different receiving devices. Compared to using the http protocol to communicate between devices with the server, using the MQTT protocol allows to reduce the server design response to a large number of different clients, while also separating the operation between the server and client devices (no need to know each other's existence), the server and client devices only communicate with each other via the publish (send data) mechanism and subscribe (receive data) to / from MQTT broker. Through the use of MQTT broker as an intermediary, it is easy to design communication scenarios between the server and client devices to meet the required functions of the system. Các thiết bị và server được thiết kế giao tiếp với nhau thông qua một MQTT broker trung gian dựa trên giao thức MQTT để truyền nhận dữ liệu giữa các thiết bị với server. Giao thức MQTT là một lựa chọn rất phù hợp cho bài toán truyền nhận dữ liệu trong ứng dụng công nghệ IoT. MQTT phù hợp cho việc truyền các gói tin kích thước nhỏ với tần suất cao từ nhiều thiết bị gửi nhận khác nhau. So với việc sử dụng giao thức http để giao tiếp giữa các thiết bị với server, sử dụng giao thức MQTT cho phép giảm tải việc thiết kế server đáp ứng một số lượng lớn client khác nhau, đồng thời tách biệt hoạt động giữa server và các thiết bị client (không cần nhận biết đến sự tồn tại của nhau), server và các thiết bị client chỉ giao tiếp với nhau thông qua cơ chế publish (gửi dữ liệu) và subscribe (nhận dữ liệu) lên/từ MQTT broker. Thông qua việc sử dụng MQTT broker làm trung gian có thể dễ dàng thiết kế các kịch bản giao tiếp giữa server và các thiết bị client để đáp ứng được các chức năng yêu cầu của hệ thống.

### System architecture

From the overall design of the system identified above, the architecture of the detailed components of the system continues to be designed using top-down methods. The following figure depicts the detailed architecture of the system components, and the communication scenario between the server and the control devices. Từ thiết kế tổng quan của hệ thống được xác định ở trên, kiến trúc các thành phần chi tiết của hệ thống tiếp tục được thiết kế xây dựng theo phương pháp top-down. Hình sau mô tả kiến trúc chi tiết các thành phần của hệ thống, cùng kịch bản giao tiếp giữa server và các thiết bị thu thập điều khiển.

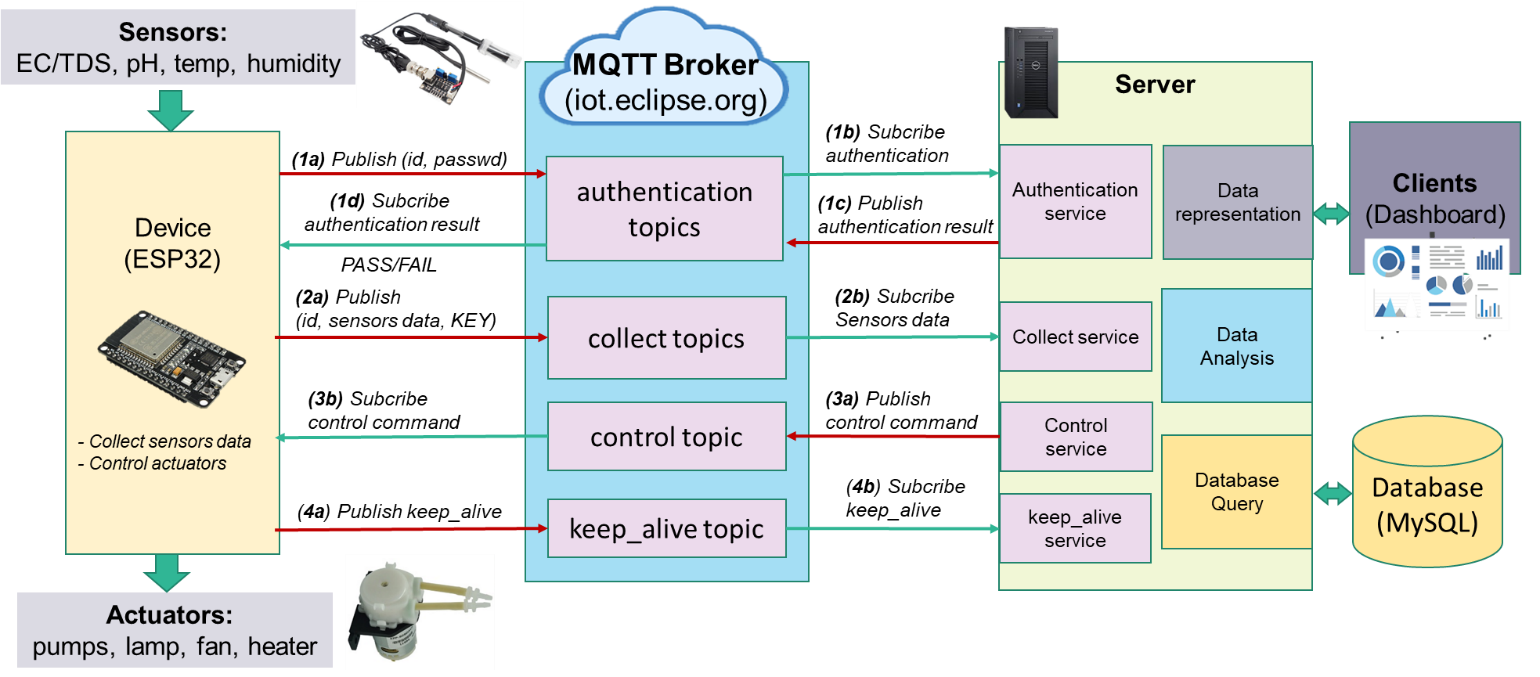


Figure 2‑2. Architecture of system components

Detailed architecture system divided into 3 components: Kiến trúc chi tiết hệ thống chia làm 3 thành phần:

**Component 1. On the side of the gathering device controls. Phần 1. Phía các thiết bị thu thập điều khiển.**

The objective of these devices is to collect sensor data and perform control of the crop care process. Each device is designed to be attached to a hydroponic crop rig. Mục tiêu của các thiết bị này là thu thập các dữ liệu cảm biến và thực hiện điều khiển quá trình chăm sóc cây trồng. Mỗi thiết bị được thiết kế để gắn với một giàn cây trồng thủy canh.

The topic will be to design and build a hardware device paired with sensors such as nutrient concentrations in TDS solution, pH, temperature, humidity, light,... and conditioning structures. Control is the switching relay of control devices to take care of hydroponic plants such as nutrient pump, nutrient pump, fan, heating lamp,... The device is an embedded computer with low cost, low power consumption, proper configuration, capable of wifi connection. Pairing with sensors and actuators is designed to be plug-and-play, flexible and scalable, customizing with additional sensors and motors. Different control profiles. Đề tài sẽ hướng tới thiết kế, xây dựng một thiết bị phần cứng ghép nối với các cảm biến như nồng độ dinh dưỡng trong dung dịch TDS, nồng độ pH, nhiệt độ, độ ẩm, ánh sáng,… và các cơ cấu điều khiển là các rơ le đóng cắt các thiết bị điều khiển chăm sóc cây trồng thủy canh như bơm dung dịch dinh dưỡng, bơm pha dinh dưỡng, quạt, đèn sưởi,… Thiết bị được chế tạo là một máy tính nhúng có giá thành rẻ, công suất tiêu thụ thấp, cấu hình phù hợp, có khả năng kết nối wifi. Việc ghép nối với các cảm biến và các cơ cấu chấp hành được thiết kế có khả năng dễ dàng kết nối (plug-and-play), linh hoạt và có thể mở rộng tùy chỉnh thêm bớt với các loại cảm biến cũng như cơ cấu điều khiển khác nhau.

To meet these requirements, the topic aims to build an embedded computer based on the IoT ESP32-WROOM device. This is a device for IoT in accordance with the requirements to build the system. Để đáp ứng những yêu cầu trên, đề tài hướng tới xây dựng một máy tính nhúng dựa trên thiết bị IoT ESP32-WROOM. Đây là một thiết bị cho IoT phù hợp với các yêu cầu để xây dựng hệ thống.

**Component 2. The MQTT broker is the connection intermediary that serves to transmit data between client devices and the server.Phần 2. MQTT broker là trung gian kết nối phục vụ truyền nhận dữ liệu giữa các thiết bị client với server.**

MQTT broker acts as a secondary server, acting as a connection intermediary for communication between devices and servers. Packets from the device are sent to the server via the MQTT broker, and the control packets from the server transfer to the devices via MQTT broker. The mechanism of communication through publishing (sending data) and subscriptions (receiving data) from topics associated with the content of the message (message). The script is designed with 4 main topic groups: authentication, sending / receiving sensor data, sending / receiving control commands, sending / receiving device status (KeepAlive). MQTT broker đóng vai trò là một server thứ cấp, làm nhiệm vụ trung gian kết nối phục vụ truyền nhận giữa các thiết bị và server. Các gói tin từ thiết bị được gửi lên server thông qua MQTT broker, đồng thời các gói tin điều khiển từ server truyền tới các thiết bị thông qua MQTT broker. Cơ chế giao tiếp qua việc publish (gửi dữ liệu) và subscribe (nhận dữ liệu) từ các topic gắn với nội dung gói tin (message). Kịch bản được thiết kế gồm có 4 nhóm topic chính: xác thực, gửi/nhận dữ liệu cảm biến, gửi/nhận lệnh điều khiển, gửi/nhận điểm danh trạng thái thiết bị (KeepAlive).

**Component 3. Server application.Phần 3. Ứng dụng phía server.**

This section performs the main tasks including: Phần này thực hiện các nhiệm vụ chính gồm:

1. Providing communication services with devices including: authenticating devices, receiving data sent from devices, issuing control commands to the desired device, taking attendance of devices. Cung cấp các dịch vụ giao tiếp với các thiết bị gồm: xác thực thiết bị, nhận dữ liệu gửi từ các thiết bị, ra lệnh điều khiển tới thiết bị mong muốn, điểm danh trạng thái hoạt động của các thiết bị.
2. Store data into the database, providing query operations with the database. Lưu trữ dữ liệu vào cơ sở dữ liệu, cung cấp các thao tác truy vấn với cơ sở dữ liệu.
3. Provides web services for managing, storing, presenting data and controlling commands. The presentation of the data is built in the form of a dashboard with representation charts, monitoring collected data for a selected collection device. Provides console to perform care control during plant growth. The control can be manual from user interaction, or done automatically thanks to automatic control settings, data analysis processes. Cung cấp dịch vụ web cho phép quản lý, lưu trữ, trình bày dữ liệu và ra lệnh điều khiển. Việc trình bày dữ liệu được xây dựng dưới dạng dashboard (bảng giám sát điều khiển) với các biểu đồ biểu diễn, giám sát dữ liệu thu thập ứng với một thiết bị thu thập được lựa chọn. Cung cấp giao diện điều khiển để thực hiện điều khiển chăm sóc trong quá trình sinh trưởng của cây trồng. Việc điều khiển có thể là thủ công từ tương tác của người dùng, hoặc được thực hiện tự động hoàn toàn nhờ vào các cài đặt điều khiển tự động, các xử lý từ phân tích dữ liệu.
4. The function of data analysis for automatic control is built as a module integrated into the server system. Chức năng phân tích dữ liệu (data analysis) phục vụ cho điều khiển tự động được xây dựng dưới dạng mô đun tích hợp vào hệ thống server.

**The communication between devices with server consists of 4 functions:Kịch bản giao tiếp giữa các thiết bị với server gồm 4 chức năng:**

1. Authentication process
2. The device should first be registered with the system through the management function. The registered device will be granted an ID identifier, authentication code (authentication\_code) and a security code (security\_code) that will be attached to the data packet sent to the server after the server is successfully verified.
3. Authentication is performed first when the device is started and a successful internet connection via a wifi access point and a successful connection to the MQTT broker is used in the system.
4. The device will send a packet without authentication request information (including ID identifier and authentication code authentication\_code) provided when registering the device on the system administration page.
5. The device authentication service on the server will continuously listen for authentication requests from the device. When receiving an MQTT broker authentication packet from a device, the server checks in the database and responds to the device to authenticate success or failure. Only participating devices can enter the system after successfully passing the authentication step.
6. Data sending and receival
7. Nếu kết quả trả về là pass thì thiết bị sẽ publish lên MQTT broker những message là thông số nhận được từ sensor bao gồm id, sensor data, key value theo topic collect \_data.
8. Những dữ liệu liên tục đực gửi về MQTT broker, phía server sẽ sử dụng các dịch vụ collect để subcribe dữ liệu sensor được gửi về
9. Control process
10. Data sent will be performed with data analysis with standard data sets which are suitable parameters for each plant variety. If the data is found to be inappropriate, the server side will execute control commands by publishing control commands to the control topic at MQTT broker.
11. Devices that perform subcribe commands that the device sends and then executes commands as required
12. The process of taking attendance status of devices (KeepAlive)
13. Devices will periodically send keep\_alive packets to notify their active status to the server via MQTT broker.
14. The server listens to the packet informing the operating status of all devices to capture which devices remain connected after each time interval.

## Used technologies

### MQTT protocol

* **Introduction to MQTT: Tổng quan về MQTT:**
  + MQTT (Message Queing Telemetry Transport) was designed by Andy Stanford-Clark and Arlen Nipper engineers in 1999. It is a publish / subscribe message delivery protocol, designed simply, using bandwidth Low, high reliability, and able to operate in unstable transmission conditions. Therefore, it is suitable for M2M and IoT applications. MQTT (Message Queing Telemetry Transport) được thiết kế bởi hai kĩ sư Andy Stanford-Clark và Arlen Nipper vào năm 1999. Nó là một giao thức truyền thông điệp theo mô hình publish/subscribe, được thiết kế đơn giản, sử dụng băng thông thấp, độ tin cậy cao, và có khả năng hoạt động trong điều kiện đường truyền thiếu ổn định. Vì vậy nó phù hợp cho các ứng dụng M2M và IoT.
  + MQTT operates according to client / server model, each sensor is a client, server is a broker intermediary component, via TCP / IP protocol. MQTT hoạt động theo mô hình client/server, mỗi một cảm biến là một client, server là thành phần trung gian broker, thông qua giao thức TCP/IP.
* **Architecture and operation mechanism of MQTT protocol: Kiến trúc và cơ chế hoạt động của giao thức MQTT:**

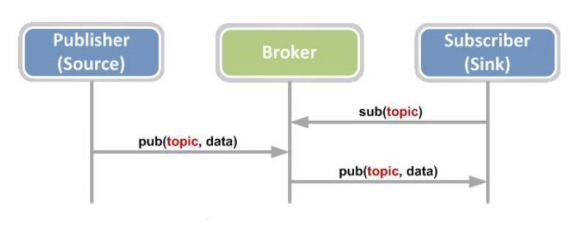
**

Figure 2‑3. Architecture and operation mechanism of MQTT protocol

* In the system using MQTT, clients connect to a MQTT server (broker). Each client will subsribe a few topics to listen to the message. Trong hệ thống sử dụng MQTT, các client kết nối đến một MQTT server (broker). Mỗi client sẽ subsribe một vài topic để lắng nghe message.
* Message: The data is transported within the network MQTT protocol for applications. Each message will have the topic name and QoS. Message: Các dữ liệu được vận chuyển bên trong giao thức MQTT trên mạng cho các ứng dụng. Mỗi một message sẽ có tên topic và QoS.
* Client: A program or device that uses the MQTT protocol, it connects to the Server and takes action. Client: Một chương trình hoặc thiết bị sử dụng giao thức MQTT, nó kết nối đến Server và thực hiện hành động.
  + Publish: Sends messages to other clients. Gửi các message đến các client khác.
  + Subscribe: Registers a topic where the client wants to receive messages. Đăng ký một topic mà client muốn nhận message.
  + Unsubscribe: Cancels registration. Hủy bỏ đăng ký.
  + Disconnect: Disconnects from the server. Ngắt kết nối đến server.
* Server: An intermediate component between clients to receive subscriptions from clients and simultaneously publish messages to clients. Broker actions: Server: Là một thành phần trung gian đứng giữa các client để nhận các subscribe từ client và đồng thời publish các message đến các client. Những hành động của Broker:
  + Accept the connection from Client. Chấp nhận kết nối đến từ Client.
  + Accept the messages that the Client publishes. Chấp nhận các message mà Client publish.
  + Publish the messages to the subscribed Client. Publish các message đến Client đã subscribe.
  + Make subscription requests and unsubscribe from the Client. Thực hiện các yêu cầu subscribe và unsubscribe từ Client.
* Topic: Are labels attached to a message to match the subscription to the server. The server will send messages to the client via the registered subsciption. Topic: Là các nhãn được gắn cho một message để khớp với các subscription đến server. Server sẽ gửi các message đến Client thông qua subsciption đã đăng ký.
* **Quality of service – QoS**

Quality of Service is an agreement between the sender and the receiver in securing message distribution. There are 3 levels of QoS in MQTT: Quality of Service là một thỏa thuận giữa bên gửi và bên nhận trong việc đảm bảo phân phối message. Có 3 cấp độ QoS trong MQTT:

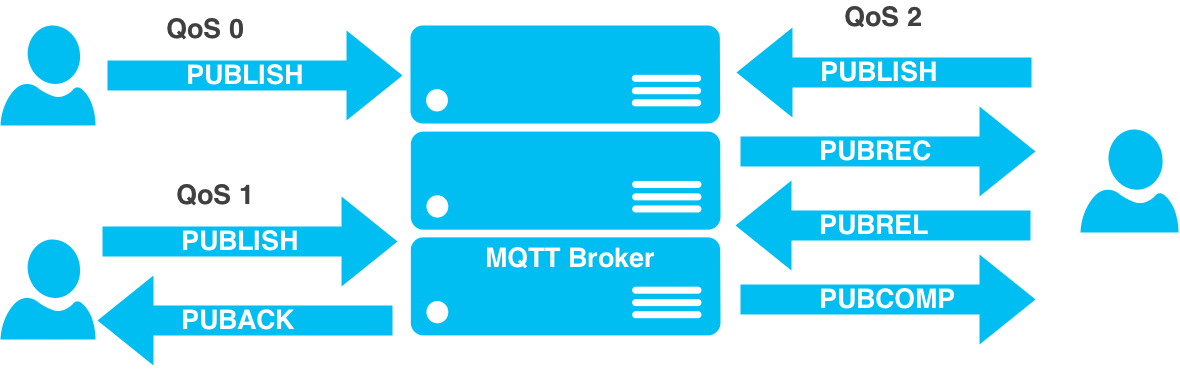


Figure 2‑4. Illustration of 3 types of QoS of MQTT protocol

* + At most once (QoS 0):
    - The messages arrive at the server up to once. The transmission is completely based on the reliability of TCP/IP. Các message đến server tối đa một lần. Việc truyền nhận hoàn toàn dựa vào tính tin cậy của TC/IP.
  + At least once (QoS 1):
    - The server side receives the message identified by a PUBACK message. Phía server nhận message được xác định bởi một message PUBACK.
    - If there is an error, or a confirmation message is not received after a certain period of time, it will return the message and set the DUP bit in the header of the message. So messages are sent to the server at least once. Nếu có lỗi, hay message xác nhận không nhận được sau một khoảng thời gian nhất định thì sẽ gửi lại message và thiết lập bit DUP tron header của message. Vì vậy message được gửi tới server ít nhất một lần.
    - If the client does not receive a PUBACK message or for a defined period or an error occurs, the client will return the PUBLISH message with the established DUP. Nếu client không nhận được message PUBACK hoặc trong một khoảng thời gian được định nghĩa hoặc có lỗi xảy ra thì client sẽ gửi lại message PUBLISH với DUP đã được thiết lập.
    - When receiving a repeat message from the client, the server publishes messages to subscriber and sends another PUBACK message. Khi nhận một message lặp lại từ phía client, server sẽ publish các message đến các subscriber và gửi một message PUBACK khác.
  + Exactly once (QoS 2):
    - Repeat messages do not transfer to the application. This is the highest level when distributing messages, repeated messages are not accepted. When QoS = 2, a message has a message ID in its header. Message bị lặp lại không chuyển đến ứng dụng. Đây là mức độ cao nhất khi phân phối message, message lặp lại không được chấp nhận. Khi QoS = 2, một message có một message ID trong phần header của nó.
    - If there is an error or after a period of time, the protocol flow will redo the result of the last confirmation message or PUBLISH or PUBREL. So make sure the message reaches the subscriber exactly once. Nếu có lỗi hoặc sau một khoảng thời gian, luồng protocol sẽ thực hiện lại kết quả của message xác nhận cuối cùng hoặc PUBLISH hoặc PUBREL. Vì vậy đảm bảo message đến subscriber đúng một lần.
* **Security in MQTT: Bảo mật trong MQTT:**
  + Security in MQTT is divided into multiple layers, each layer has a different attack type. MQTT is a lightweight protocol so there are not many security mechanisms, in fact often use SSL / TLS encryption. Bảo mật trong MQTT được chia ra làm nhiều layer, mỗi một layer có một kiểu tấn công khác nhau. MQTT là giao thức gọn nhẹ nên không có nhiều cơ chế bảo mật, trong thực tế thường sử dụng mã hóa SSL/TLS.
  + At the network layer using a physical security network or VPN as the platform for the connection between Client and Broker is a way to provide reliable security and connectivity. Ở tầng mạng sử dụng một mạng bảo mật vật lý hay VPN như nền tảng cho kết nối giữa Client và Broker là một cách để cung cấp bảo mật và kết nối đáng tin cậy.
  + At the application layer, MQTT provides a user / password authentication when a client connects to the broker. Another way is to use download encryption on the application layer so that information is transmitted securely when there is no full transport encryption. Ở tầng ứng dụng, MQTT cung cấp một xác thực user/password khi một client kết nối đến broker. Một cách khác là sử dụng mã hóa tải trên tầng ứng dụng để thông tin được truyền đi an toàn khi không có mã hóa giao vận đầy đủ.

### MQTT Eclipse Paho library

To deploy the system using the MQTT protocol, we selected a popular MQTT broker provider, iot.eclipse.org, or broker.hivemq.com. These are MQTT brokers that are widely used, providing advantages for developing systems that use this protocol. Để triển khai hệ thống sử dụng giao thức MQTT, chúng tôi lựa chọn một nhà cung cấp MQTT broker phổ biến là iot.eclipse.org, hoặc broker.hivemq.com. Đây là các MQTT broker được sử dụng rộng rãi, cung cấp thuận lợi cho phát triển các hệ thống sử dụng giao thức này.

In addition to building communication modules between devices and servers, we also use the client's MQTT library sets. Typically, the server side will use the Java Paho Client library set as a MQTT library written in java to develop applications running on the JVM or compatible platforms like Android. Java Paho Client provides two APIs: Ngoài ra để xây dựng mô đun giao tiếp giữa các thiết bị và server, chúng tôi cũng sử dụng các bộ thư viện MQTT client của các cung cấp. Tiêu biểu, phía server sẽ sử dụng bộ thư viện Java Paho Client là một thư viện MQTT được viết bằng java để phát triển các ứng dụng chạy trên JVM hoặc các nền tảng tương thích như Android. Java Paho Client cung cấp hai API:

MqttAsyncClient is a completely asynchronous API, the incoming data will be processed in the pre-registered Callback Interface. MqttAsyncClient là một API hoàn toàn bất đồng bộ, các dữ liệu gửi đến sẽ được xử lý trong những Callback Interface đăng kí trước.

MqttClient is a synchronous wrapper around the MqttAsyncClient, where functions appear synchronized with the application. MqttClient là một wrapper đồng bộ xung quanh MqttAsyncClient, nơi các chức năng xuất hiện đồng bộ với ứng dụng.

The side of the embedded computer device will use the library of Embedded C/C++ MQTT client compatible for building communication module via MQTT on software for developed computer devices. Phía thiết bị máy tính nhúng sẽ sử dụng bộ thư viện Embedded C/C++ MQTT client tương thích cho xây dựng mô đun giao tiếp qua MQTT trên phần mềm cho thiết bị máy tính nhúng được phát triển.

### MVC (Model-View-Controller) model

Data processing is always a problem not simply because a data sent to the server or sent from the server must always go through many processing processes that must be stored in the most basic form to be able to proceed to transform into different forms to meet the changing needs of the user interface. Xử lý dữ liệu luôn là một vấn đề không đơn giản do một dữ liệu được gửi về server hay gửi đi từ server luôn phải đỏi hỏi đi qua rất nhiều quá trình xử lý là phải được lưu trữ dưới dạng cơ bản nhất để có thể tiến hành biến đổi ra thành mọi dạng khác nhau đáp ứng như cầu thay đổi nhanh chóng của giao diện người dùng.

Therefore, the processing and saving of data must be broken down into several parts, each of which has its own role and generally all systems will consist of 3 main components: Do đó công việc xử lý và lưu dữ liệu phải được chia nhỏ ra thành nhiều phần, mỗi phần đều có vai trò riêng của nó và nhìn chung mọi hệ thống đều sẽ bao gồm 3 thành phần chính là:

Model: Object components and data. Thành phần đối tượng và dữ liệu.

View: User interface. Giao diện người dùng.

Controller: Data processing and control components. Thành phần xử lý dữ liệu và điều khiển.

This model fully satisfies the requirement of clearly separating the role of each component in the system, so we will apply the very popular MVC model to the current system. Mô hình này đáp ứng đầy đủ yêu cầu về khả năng phân tách rõ ràng vai trò của từng thành phần trong hệ thống, do đó ta sẽ áp dụng mô hình MVC đang rất thịnh hành cho hệ thống hiện tại.

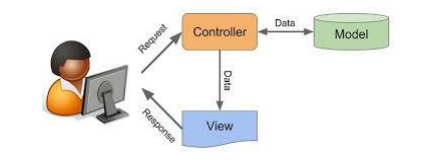
**

Figure 2‑5. Illustrating MVC model applied in building applications on the server

Accordingly, the system will be made into 3 parts: Theo đó hệ thống sẽ được ra làm 3 phần:

Model: Includes Entities and data and classes that interact with data. Bao gồm các Entity và dữ liệu và các lớp tương tác với dữ liệu.

View: We will consider as the client side is the view, inside each view that needs to be divided into a smaller size MVC model. Ta sẽ coi như phía client side chính là các view, bên trong mỗi view đó lại cần phải chia thành một mô hình MVC cỡ nhỏ hơn.

Controller: Includes logical processing and mapping of requests from users. Bao gồm các xử lý logic và thành phần mapping các request từ người dùng.

### Spring Framework

Spring framework is the most popular open source platform for building applications. Spring's core features can be used to develop simple Java Desktop applications, mobile applications, and Java Web applications. Spring framework là một nền tảng mã nguồn mở phổ biến nhất để xây dựng các ứng dụng. Những tính năng cốt lõi của Spring có thể được sử dụng để phát triển các ứng dụng Java Desktop, ứng dụng mobile, Java Web một cách đơn giản.

This project will use Spring framework to support the server-side application development process, providing a convenient mechanism for deploying the system's functional services. Dự án này sẽ sử dụng Spring framework để hỗ trợ quá trình xây dựng ứng dụng phía server, cung cấp cơ chế thuận tiện để triển khai các dịch vụ chức năng của hệ thống.

The mechanism of processing request and response of Spring Framework is shown in the diagram: Cơ chế xử lí request và response của Spring Framework được thể hiện qua sơ đồ:

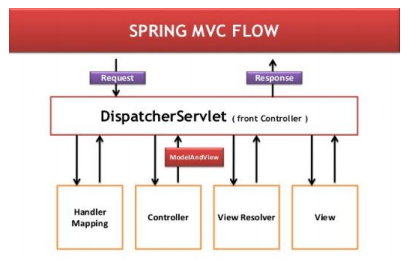
**

Figure 2‑6. Processing mechanism of request of spring framework

When there is a URL request sent to the execution steps will be as follows: Khi có một yêu cầu URL gửi đến các bước thực hiện sẽ như sau:

* + - 1. The Spring Front Controller (DispatcherServlet) receives a request that it will find the appropriate Handler Mapping. Spring Front Controller (DispatcherServlet) nhận yêu cầu nó sẽ tìm đến Handler Mapping thích hợp.
      2. Handler Mapping will map the client's request to the appropriate controller. Handler Mapping sẽ ánh xạ yêu cầu của client đến controller thích hợp.
      3. Then DispatcherServlet will send the request to the appropriate Controller. Sau đó DispatcherServlet sẽ gửi yêu cầu đến Controller thích hợp.
      4. After executing the process from the client's request, it executes the logic specified in the controller and finally returns the ModelAndView object. Sau khi thực hiện tiến trình từ yêu cầu của client, nó thực thi các logic được xác định trong controller và cuối cùng trả về đối tượng ModelAndView.
      5. Based on the values in ModelAndView, the Font Controller will send a request to View Resolver, View Resolver will search based on the config file view that Controller returns. Dựa trên các giá trị trong ModelAndView, Font Controller sẽ gửi yêu cầu đến View Resolver, View Resolver sẽ tìm kiếm dựa trên các config file view mà Controller trả về.
      6. The final Font Controller will find the view file previously defined by View Resolver and send the result to the user. Cuối cùng Font Controller sẽ tìm đến file view đã được View Resolver xác định trước đó và gửi kết quả cho người dùng.

### NodeJS in building web services on the server

# Development of server application

## General design of the application on the server

The server-side system will have 2 main users who are the system administrator (admin) with full authority to use the system and ordinary users are customers, the owners of the participating hydrological platforms into the system, granted a number of rights to use certain systems related to monitoring and controlling the hydroponic trusses they own during the planting process. Based on the analysis of user needs, through the overall design of the system, we have designed the general Use-Case diagram for the server-side application part as follows: Hệ thống phía server sẽ có 2 đối tượng người dùng chính là người quản trị hệ thống (admin) có toàn quyền với sử dụng hệ thống và người sử dụng thông thường là các khách hàng, người chủ sở hữu các giàn thủy canh tham gia vào hệ thống, được cấp một số quyền hạn sử dụng hệ thống nhất định liên quan đến giám sát và điều khiển các giàn thủy canh mà họ sở hữu trong quá trình trồng cây. Dựa trên phân tích những nhu cầu của người dùng, thông qua thiết kế tổng quan của hệ thống, ta có thiết kế biểu đồ Use-Case tổng quát cho phần ứng dụng phía server như sau:



Figure 3‑1. General Use-Case diagram of server-side application

Use-case specification: Đặc tả use-case:

* Description: The management system allows users to log into the system to monitor the data obtained from the device sent and perform device control. Miêu tả: Hệ thống quản lý cho phép người dùng đăng nhập vào hệ thống theo dõi dữ liệu thu được từ thiết bị gửi về và thực hiện điều khiển thiết bị.
* Input conditions: Users can log in to the system and be given the function to manage, monitor and control. The device is allocated to users who are registered with pre-defined identifiers and codes. Điều kiện đầu vào: Người dùng có thể đăng nhập vào hệ thống và được phân quyền chức năng để quản lý, giám sát và điều khiển. Thiết bị được cấp phát cho người dùng được đăng ký với định danh và mã xác thực qui định trước.
* Activation condition: User login successfully and device is started. Điều kiện kích hoạt: Người dùng đăng nhập thành công và thiết bị được khởi động.
* Event flow: Dòng sự kiện:
  + Device side: The device performs two main tasks: data transmission and other device control when receiving control commands. Before reading and sending data to the upper layer, the device must be authenticated by sending an authentication packet to MQTT broker, if the device is authentic, the device will be published and subscribe to the data obtained by a topic. up MQTT broker as a newsletter. When receiving a control command from the server, it will be allowed to control other devices if authenticated. Phía thiết bị: Thiết bị thực hiện hai nhiệm vụ chính là truyền gửi dữ liệu và thực hiện điều khiển thiết bị khác khi nhận lệnh điều khiển. Trước khi đọc và gửi dữ liệu lên tầng trên nó thì thiết bị phải được xác thực bằng việc gửi gói tin xác thực lên MQTT broker, nếu đúng thiết bị xác thực thì thiết bị sẽ được publish và subcribe dữ liệu thu được bằng một topic lên MQTT broker dưới dạng bản tin. Khi nhận được lệnh điều khiển từ phía server thì nó sẽ được phép điều khiển các thiết bị khác nếu đã được xác thực.
  + Server side: Users will be logged into the system and assigned to perform two management and control functions. For the management module, users are allowed to manage the information of the system and information of objects to be managed. At the same time when logging into the user system will be authenticated by sending an authentication packet including id, username, password to MQTT broker, when the account is authenticated, the system will simultaneously receive the sent data. from the device and the user is allowed to search and manipulate the device to analyze the parameters affecting the development of the cultivated plant variety and can control the device to obtain information Favorable numbers for plants. User device control must also be through authentication with an intermediate application, MQTT. Admin users will perform all functions of normal users and perform management and authorization for ordinary users. Phía server: Người dùng sẽ được đăng nhập vào hệ thống và được phân quyền để thực hiện 2 chức năng quản lý và điều khiển. Đối với phân hệ quản lý, người dùng được phép quản lý các thông tin của hệ thống và thông tin các đối tượng cần quản lý. Đồng thời khi đăng nhập vào hệ thống người dùng sẽ được xác thực thông qua việc gửi gói tin xác thực bao gồm id, username, password lên MQTT broker, khi tài khoản đươc xác thực thì hệ thống sẽ đồng thời nhận được dữ liệu gửi về từ phía thiết bị và người dùng được phép tìm kiếm và thao tác trên bộ thiết bị đó để phân tích những thông số ảnh hưởng đến việc phát triển của giống cây trồng đang theo dõi và có thể điều khiển các thiết bị để có thông số thuận lợi cho cây trồng. Việc điều khiển thiết bị của người dùng cũng phải thông qua việc xác thực với ứng dụng trung gian là MQTT. Người dùng là Admin sẽ thực hiện được tất cả các chức năng của người dùng thông thường và thực hiện quản lý và phân quyền cho người dùng thông thường.
* End condition: Account not authenticated and login failed. The device is broken, the power supply is disconnected or there is no network connection. Điều kiện kết thúc: Tài khoản không được xác thực và đăng nhập không thành công. Thiết bị hỏng, nguồn cấp ngắt hoặc không có kết nôi mạng.
* Status after completion: The system returns to the state before the event stream starts, the device returns to its original state as it starts up. Trạng thái sau khi kết thúc: Hệ thống quay về trạng thái trước khi dòng sự kiện bắt đầu, thiết bị quay về trạng thái ban đầu như bắt đầu khởi động.
* Special requirements: The system continuously receives data sent from the device and must be in realtime. Các yêu cầu đặc biệt: Hệ thống liên tục nhận dữ liệu gửi về từ phía thiết bị và phải ở trạng thái realtime.

## Build server communication services with devices via MQTT Broker

This section will carry out the first important functional group of server-side applications and services for communication between the server and control collection devices as designed. This function group consists of 4 parts: Phần này sẽ tiến hành nhóm chức năng quan trọng đầu tiên của ứng dụng phía server và các dịch vụ phục vụ giao tiếp giữa server với các thiết bị thu thập điều khiển như đã được thiết kế. Nhóm chức năng này gồm có 4 phần:

1. Device authentication service participating in the system. Dịch vụ xác thực thiết bị tham gia vào hệ thống.
2. Service of collecting sensor data from devices sent through MQTT broker Dịch vụ thu thập dữ liệu cảm biến từ các thiết bị gửi tới thông qua MQTT broker.
3. The service sends the control command to the specified device. Dịch vụ gửi lệnh điều khiển tới các thiết bị được chỉ định.
4. Service attendance status of devices (KeepAlive). Dịch vụ điểm danh trạng thái hoạt động của các thiết bị (KeepAlive).

The communication scenario of these 4 services needs to be designed to allow simultaneous service of multiple connected devices and must ensure all unexpected situations can occur when operating the system such as: Network connection is interrupted, or packets sent to the wrong format. Kịch bản giao tiếp của 4 dịch vụ này cần được thiết kế để cho phép phục vụ đồng thời nhiều thiết bị kết nối tới và phải đảm bảo tất cả các trường hợp không mong muốn có thể xảy ra khi vận hành hệ thống như: kết nối mạng bị gián đoạn, hay các gói tin gửi tới không đúng khuôn dạng.

A screenshot of a video game

Description generated with high confidence

Figure 3‑2. Intelligent diagram of self-communication between a device and the server via MQTT broker

Describe the scenario of communicating with the device's MQTT Broker: Mô tả kịch bản giao tiếp với MQTT Broker phía thiết bị:

1. When starting the system, the system must be connected to the network and the intermediate component must be started as MQTT Broker. Khi bắt đầu khởi động hệ thống thì hệ thống bắt buộc phải có kết nối mạng và phải được khởi động thành phần trung gian là MQTT Broker.
2. The system and equipment must be connected to the MQTT Broker, and if the connection is successful, it will go to the device authentication step. If the connection fails, it will return to Step 1. Hệ thống và thiết bị phải được kết nối với MQTT Broker, nếu kết nối thành công thì sẽ sang bước xác thực thiết bị. Nếu kết nối không thành công thì sẽ quay trở lại Bước 1.
3. When the connection is successful, the authentication will be carried out, on the server side will authenticate the user by sending the authentication message to the MQTT, the device will send the active device authentication message to the MQTT broker. Khi kết nối thành công thì việc xác thực sẽ được tiến hành, ở phía Server sẽ xác thực người dùng bằng việc gửi bản tin xác thực đến MQTT, ở thiết bị sẽ gửi bản tin xác thực thiết bị hoạt động đến MQTT broker.
4. If the system is already authenticated, it will proceed to send and receive data from MQTT broker, on the server side, it will perform subcribe data and store it in the database, on the device side, it will publish data to MQTT Broker so that the server can subscribe. In case the system has not been authenticated, it is necessary to resubmit the authentication to conduct a subscription to answer the authentication, if the authentication is successful, the data will be sent. Otherwise, if authentication fails, it will return to Step 1. Nếu hệ thống đã xác thực rồi thì sẽ tiến hành việc gửi và nhận dữ liệu từ MQTT broker, ở phía server sẽ thực hiện subcribe dữ liệu gửi lên và lưu trữ vào database, ở phía thiết bị sẽ thực hiện publish dữ liệu lên MQTT Broker để server có thể subcribe. Trong trường hợp hệ thống chưa được xác thực thì phải tiến hành gửi lại xác thực để tiến hành subcribe để trả lời xác thực, nếu xác thực thành công mới được gửi dữ liệu. Ngược lại nếu xác thực không thành công sẽ quay trở lại Bước 1.

Communication scenario between MQTT broker and server: Kịch bản giao tiếp giữa MQTT broker với server:

1. The server will be connected to the MQTT broker. Server sẽ được kết nối với MQTT broker.
2. If there is a connection, it will perform user authentication via username and password. If there is no connection, it will reconnect. Nếu có kết nối thì sẽ thực hiện quá trình xác thực người dùng thông qua username, password. Nếu không có kết nối thì sẽ kết nối lại.
3. Proceed to authenticate users, if the authentication is successful, then proceed to subscribe to authenticate replies. If not, then reconnect. Tiến hành xác thực người dùng, nếu việc xác thực thành công thì mới tiến hành subcribe để trả lời xác thực. Nếu chưa thì tiến hành kết nối lại.
4. After conducting the authentication, if successful, the system will listen to the message that the device sends to MQTT and receive data on the MQTT broker. The bulletins will be sent continuously, the system must also ensure listening to changes that occur continuously to conduct the tracking of changes of the database. Sau khi tiến hành trả lời xác thực, nếu thành công thì hệ thống sẽ lắng nghe bản tin mà thiết bị gửi lên MQTT và nhận dữ liệu trên MQTT broker. Các bản tin sẽ được gửi lên liên tục, hệ thống cũng phải đảm bảo việc lắng nghe những thay đổi xảy ra liên tục để tiến hành thao tác lưu vết sự thay đổi của cơ sở dữ liệu.

### Service to authenticate devices to participate in the system

The mechanism for deploying communication services with devices through MQTT broker is implemented using Java library Mqtt Paho. Authentication service is built using a MqttAuthenticationServices class to reinstall MqttCallback in the Mqtt Paho library provided. This service allows to serve authentication requirements of many incoming devices. Cơ chế để triển khai các dịch vụ giao tiếp với thiết bị thông qua MQTT broker được thực hiện sử dụng thư viện Java Mqtt Paho. Dịch vụ xác thực được xây dựng bằng một lớp MqttAuthenticationServices cài đặt lại MqttCallback trong thư viện Mqtt Paho cung cấp. Dịch vụ này cho phép phục vụ yêu cầu xác thực của nhiều thiết bị gửi tới.



Figure 3‑3. Sequence diagram of device authentication services

Authentication packets sent from the device are specified in JSON format as follows: Gói tin xác thực gửi từ thiết bị được qui định định dạng JSON như sau:

{

"id" : %s,

"password" : "nct\_laboratory",

"device\_type" : "ESP32"

}

### Data collection service for sensors submitted by devices

This service continuously listens collect\_data\_topic packets from the MQTT broker due to the successfully authenticated devices sent. All sensor data packets from the devices are sent into a "nct\_collect" topic, the server will distinguish which device's packets through the device identifier field sent in the packet. In addition, the security\_key field is a secret string that has been granted to the device by the server when the authentication will be used to verify that the packet is sent by the authenticated device, improving the security of the system. This field avoids fake packets from fake devices sent to the system. Dịch vụ này liên tục lắng nghe các gói tin collect\_data\_topic từ MQTT broker do các thiết bị đã được xác thực thành công gửi tới. Tất các các gói tin dữ liệu cảm biến từ các thiết bị đều được gửi chung vào một topic “nct\_collect”, server sẽ phân biệt gói tin của thiết bị nào thông qua trường định danh thiết bị được gửi trong gói tin. Ngoài ra trường security\_key là một chuỗi bí mật đã được server cấp cho thiết bị khi xác thực sẽ dùng để kiểm tra đúng là gói tin do thiết bị được xác thực gửi tới, nâng cao bảo mật cho hệ thống. Trường này sẽ tránh được các gói tin giả từ các thiết bị giả mạo gửi tới hệ thống.



Figure 3‑4. Sequence diagram for collecting sensor data service from submitted devices

The format of the packet to collect sensor data from the devices sent is JSON format as follows: Khuôn dạng gói tin thu thập dữ liệu cảm biến từ các thiết bị gửi tới được định dạng JSON như sau:

{

"id" : %s, // id của ESP32

"secret\_key" : "%s", // từ mã để xác thực gói tin

"timestamp" : "%s", // thời gian gửi

"packet\_no" : %lu, // số thứ tự gói tin

"temperature" : %.1f, // dữ liệu nhiệt độ

"humidity" : %.1f, // dữ liệu độ ẩm

"tds" : %.1f, // dữ liệu tds

"pH" : %.1f, // dữ liệu pH

"water\_level" : 100 // mực nước còn lại

}

### Device control command service

This service allows sending control commands from the server to the specified device. The sending of control commands is done manually by the user for the hydroponic rig they own or is done completely automatically by the system. The server sends packets containing the control statement to the desired device via the packet's topic. Each device receives packets containing control commands through a topic associated with the device identifier nct\_control\_<deviceid>. Dịch vụ này cho phép gửi các lệnh điều khiển từ server tới các thiết bị được chỉ định. Việc gửi lệnh điều khiển được thực hiện thủ công bởi người dùng đối với giàn thủy canh họ sở hữu hoặc được thực hiện hoàn toàn tự động bởi hệ thống. Server sẽ gửi các gói tin chứa câu lệnh điều khiển tới thiết bị mong muốn thông qua topic của gói tin. Mỗi thiết bị nhận gói tin chứa lệnh điều khiển qua một topic gắn với định danh của thiết bị nct\_control\_<deviceid>



Figure 3‑5. Service sequence diagram for control commands to devices

The packet contains the control command formatted with json. In which dev\_id is the identifier of the control command receiver, actuator\_name indicates the name of the control device such as PUMP, LAMP, HEATER, ... and the ON/OFF action specified by the "action" field along with the parameter of the action determined by the "param" field. The device that receives the control command will execute that control command. Gói tin chứa lệnh điều khiển được định dạng bằng json. Trong đó dev\_id là định danh của thiết bị nhận lệnh điều khiển, actuator\_name cho biết tên của cơ cấu sẽ điều khiển như PUMP, LAMP, HEATER, … và hành động ON/OFF xác định bởi trường “action” cùng với tham số của hành động được xác định bởi trường “param”. Thiết bị nhận được lệnh điều khiển sẽ tiến hành thực hiện lệnh điều khiển đó

{

"dev\_id" : number,

"command\_no" : number,

"actuator\_id" : number,

"actuator\_name" : string,

"action" : string,

"param" : number

}

### Online device polling service (KeepAlive)

Devices in the process of operation may for some reason be disconnected in the middle, not participating in the system, then there is a mechanism to monitor devices that are in active state. (active) or lost connection with the system (inactive). This function is performed through the service built by MqttKeepAliveServices class. This service allows to receive notifications from active devices and updates to a device management list. This process is done periodically after a certain amount of time. Các thiết bị trong quá trình đang hoạt động có thể vì một lý do nào đó mất kết nối giữa chừng, không tham gia vào hệ thống, khi đó cần có cơ chế để theo dõi các thiết bị đang ở trạng thái duy trì hoạt động (active) hay bị mất kết nối với hệ thống (inactive). Chức năng này được thực hiện thông qua dịch vụ được xây dựng bởi lớp MqttKeepAliveServices. Dịch vụ này cho phép nhận các thông báo từ các thiết bị đang hoạt động và cập nhật vào một danh sách quản lý các thiết bị. Quá trình này được thực hiện định kỳ sau một khoản thời gian nhất định.



Figure 3‑6. Service device attendance sequence diagram

The package contains the operation status message sent from the JSON format device as follows: Gói tin chứa thông báo trạng thái hoạt động gửi từ thiết bị có định dạng json như sau:

{

"id" : %s,

"secret\_key" : "%s",

"timestamp" : "%s",

"active" : "true"

}

## Store data with the database

This section describes database design, storage issues and data queries. Within the framework of this project, data is stored in a relational database using MySQL administration system[[1]](#footnote-1). Phần này mô tả thiết kế cơ sở dữ liệu, vấn đề lưu trữ và truy vấn dữ liệu. Trong khuôn khổ đề tài này, dữ liệu được lưu trữ trong một CSDL quan hệ sử dụng hệ quản trị MySQL[[2]](#footnote-2).

### Database design

For the purpose of storing information of the hydroponic growing system in households applying IoT techniques, the information that needs to be stored includes: customer profile information, information about the equipment being deployed for each customer, information on crops, data collected from sensors, system control information, and system administration information. With this consideration, the relational database system is designed as shown in the following figure. The specification of relationships is described in detail in the next tables. Với mục đích lưu trữ thông tin của hệ thống trồng giàn thuỷ canh trong các hộ gia đình áp dụng các kỹ thuật IoT, các thông tin cần được lưu trữ bao gồm: dữ liệu thông tin hồ sơ khách hàng, thông tin về các thiết bị được triển khai cho từng khách hàng, thông tin các vụ trồng, dữ liệu thu thập từ cảm các biến, thông tin điều khiển của hệ thống, và thông tin quản trị hệ thống. Với nhận định như vậy, hệ cơ sở dữ liệu quan hệ được thiết kế như hình dưới. Đặc tả các quan hệ được mô tả chi tiết trong các bảng tiếp theo.



Figure 3‑7. Diagram of relational database design

The **user\_group** relationship to store information about user groups is specified in the following table. Currently, the system is designed with 3 user groups including: Customers, Technical Moderators, Administrators. However, the relationship is designed to be open, so that users can be added or modified easily in the future. Quan hệ **user\_group** để lưu trữ thông tin về các nhóm người dùng được đặc tả trong bảng dưới. Hiện tại, hệ thống được thiết kế với 3 nhóm người dùng gồm: Khách hàng, Điều hành viên kỹ thuật, Quản trị viên. Tuy nhiên, quan hệ được thiết kế mở, các người dùng có thể được bổ sung hay điều chỉnh về sau một cách dễ dàng.

Table 3‑1. Specification of **user\_group** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of user group  ID của nhóm người dùng |
| name | VARCHAR | NOT NULL | Short name of user group  Tên ngắn của nhóm người dùng |
| full\_name | VARCHAR | NOT NULL | Full name of user group  Tên đầy đủ của nhóm người dùng |
| level | INT | NOT NULL | User permission level, with values including: 1 = Customer, 5 = Operator, 10 = Administrator  Mức phân quyền của người dùng, với các giá trị gồm: 1 = Khách hàng, 5 = Điều hành viên kỹ thuật, 10 = Quản trị viên |

The **user** relationship to store user profile information is specified in the following table. Quan hệ **user** để lưu trữ thông tin về hồ sơ người dùng được đặc tả trong bảng dưới.

Table 3‑2. Specification of **user** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of user  ID của người dùng |
| login | VARCHAR | NOT NULL | User login  Tên đăng nhập của người dùng |
| full\_name | VARCHAR | NOT NULL | User full name  Họ tên người dùng |
| password | VARCHAR | NOT NULL | Encrypted user password  Mật khẩu đã được mã hoá |
| user\_group | INT | FOREIGN KEY | ID of user group  ID nhóm người dùng |
| create\_time | DATETIME | NOT NULL | Time of account creation  Thời gian tạo tài khoản |
| last\_login | DATETIME |  | Time of last login if available, or NULL if never logged in  Thời gian lần đăng nhập mới nhất nếu có, hoặc NULL nếu chưa đăng nhập |
| deactivate\_time | DATETIME |  | Account closure time if available, or NULL if the account is still active  Thời gian đóng tài khoản nếu có, hoặc NULL nếu tài khoản còn đang hoạt động |

The **frame** relationship to store information about each client’s platforms is specified in the following table. Quan hệ **frame** để lưu trữ thông tin về các giàn trồng của từng khách hàng được đặc tả trong bảng dưới.

Table 3‑3. Specification of **frame** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of frame  ID của giàn trồng |
| name | VARCHAR | NOT NULL | Name of frame  Tên của giàn trồng |
| user | VARCHAR | FOREIGN KEY | User ID  ID của người dùng |
| create\_time | DATETIME | NOT NULL | Initialization time of frame  Thời gian khởi tạo giàn trồng |
| retire\_time | DATETIME |  | Removal time of the frame from the system, or NULL if the frame is still in use  Thời gian gỡ bỏ giàn trồng khỏi hệ thống nếu có, hoặc NULL nếu giàn còn đang sử dụng |

The **plant\_type** relationship to store information about crops is specified in the following table. Quan hệ **plant\_type** để lưu trữ thông tin về các loại cây trồng được đặc tả trong bảng dưới.

Table 3‑4. Specification of **plant\_type** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of plant type  ID của loại cây trồng |
| name | VARCHAR | NOT NULL | Name of plant type  Tên của loại cây trồng |

The **crop** relationship to store information about the crop is described in the following table. Quan hệ **crop** để lưu trữ thông tin về các vụ trồng được đặc tả trong bảng dưới.

Table 3‑5. Specification of **crop** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of crop  ID của vụ trồng |
| name | VARCHAR | NOT NULL | Name of crop  Tên của vụ trồng |
| frame | INT | FOREIGN KEY | Frame ID  ID của giàn trồng |
| plant\_type | INT | FOREIGN KEY | Plant type ID  ID của loại cây trồng |
| start\_time | DATETIME | NOT NULL | Start time of crop  Thời gian bắt đầu vụ trồng |
| end\_time | DATETIME |  | End time of crop if it is finished, or NULL if not  Thời gian kết thúc vụ trồng nếu đã kết thúc, hoặc NULL nếu vụ trồng chưa kết thúc |

The **device\_type** relationship to store information about device types is specified in the following table. Quan hệ **device\_type** để lưu trữ thông tin về các loại thiết bị được đặc tả trong bảng dưới.

Table 3‑6. Specification of **device\_type** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of device type  ID của loại thiết bị |
| name | VARCHAR | NOT NULL | Name of device type  Tên của loại thiết bị |

The **device** relationship to store information about devices is specified in the following table. Quan hệ **device** để lưu trữ thông tin về các thiết bị được đặc tả trong bảng dưới.

Table 3‑7. Specification of **device** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of device  ID của thiết bị |
| frame | INT | FOREIGN KEY | ID of frame  ID của giàn trồng |
| device\_type | INT | FOREIGN KEY | ID of device type  ID của loại thiết bị |
| password | VARCHAR | NOT NULL | Encrypted password for device login  Mật khẩu đăng nhập của thiết bị đã được mã hoá |
| security\_code | VARCHAR |  | Security code for device  Mã bảo vệ của thiết bị |
| excluded | BOOLEAN | NOT NULL | Whether the device is excluded from the system  Thông tin thiết bị còn được sử dụng hay không |

The **sensor\_type** relationship to store information about the sensor types is specified in the following table. Quan hệ **sensor\_type** để lưu trữ thông tin về các loại cảm biến được đặc tả trong bảng dưới.

Table 3‑8. Specification of **sensor\_type** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of sensor type  ID của loại cảm biến |
| name | VARCHAR | NOT NULL | Name of sensor type  Tên của loại cảm biến |

The **sensor** relationship to store information about sensors specified in the following table. Quan hệ **sensor** để lưu trữ thông tin về các cảm biến được đặc tả trong bảng dưới.

Table 3‑9. Specification of **sensor** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of sensor  ID của cảm biến |
| name | VARCHAR | NOT NULL | Name of sensor  Tên của cảm biến |
| device | INT | FOREIGN KEY | ID of device  ID của thiết bị |
| sensor\_type | INT | FOREIGN KEY | ID of sensor type  ID của loại cảm biến |

The **sensor\_data** relationship to store information about sensor information collected is specified in the following table. Quan hệ **sensor\_data** để lưu trữ thông tin về các thông tin thu thập từ cảm biến được đặc tả trong bảng dưới.

Table 3‑10. Specification of **sensor\_data** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of measurement  ID của giá trị đo |
| sensor | INT | FOREIGN KEY | ID of sensor  ID của cảm biến |
| value | FLOAT | NOT NULL | Measurement value  Giá trị đo |
| time | DATETIME | NOT NULL | Time of measurement  Thời gian đo |

The **actuator\_type** relationship to store information about the types of actuators is specified in the following table. Quan hệ **actuator\_type** để lưu trữ thông tin về các loại thiết bị chấp hành được đặc tả trong bảng dưới.

Table 3‑11. Specification of **actuator\_type** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of actuator type  ID của loại thiết bị chấp hành |
| name | VARCHAR | NOT NULL | Name of actuator type  Tên của loại thiết bị chấp hành |

The **actuator** relationship to store information about the actuators is specified in the following table. Quan hệ **actuator** để lưu trữ thông tin về các thiết bị chấp hành được đặc tả trong bảng dưới.

Table 3‑12. Specification of **actuator** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of actuator  ID của thiết bị chấp hành |
| name | VARCHAR | NOT NULL | Name of actuator  Tên của thiết bị chấp hành |
| device | INT | FOREIGN KEY | ID of device  ID của thiết bị |
| actuator\_type | INT | FOREIGN KEY | ID of actuator type  ID của loại thiết bị chấp hành |

The **command** relation to store information about control statements specified in the following table. Quan hệ **command** để lưu trữ thông tin về các câu lệnh điều khiển được đặc tả trong bảng dưới.

Table 3‑13. Specification of **command** relationship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | ID of command  ID của câu lệnh điều khiển |
| device | INT | FOREIGN KEY | ID of device  ID của thiết bị |
| params | VARCHAR |  | Control parameters  Các tham số điều khiển |
| time | DATETIME | NOT NULL | Time of command  Thời gian gửi lệnh điều khiển |
| is\_done | BOOLEAN | NOT NULL | Whether the command has been processed  Thông tin xác định câu lệnh điều khiển đã được xử lý hay chưa |

### Data connection and query from server application

In order for the application to run on the server built with Java using Spring Framework that can retrieve data, classes as shown in the following tables are built. The data, basically corresponds to the main relationships in the database, is packaged in the **database.data** package. Connection, query and data conversion classes are packaged in the **database.connection** package. The connection and queries to the database are done via a JDBC object. Để ứng dụng chạy trên máy chủ được xây dựng bằng Java sử dụng Spring Framework[[3]](#footnote-3) có thể truy xuất được dữ liệu, các lớp như các hình sau được xây dựng. Các lớp chứa thuộc tính từ các thành phần dữ liệu về cơ bản tương ứng với các quan hệ chính trong CSDL, được đóng gói trong gói **database.data**. Các lớp kết nối và truy vấn và chuyển đổi dữ liệu, được đóng gói trong gói **database.connection**. Kết nối và các truy vấn tới CSDL được thực hiện thông qua một đối tượng JDBC.



Figure 3‑8. Class diagram for data access



Figure 3‑9. Class diagram for data connection and query

## Web service for management, monitoring and control

In the framework of this project, the web service is built to serve as interactions between customers and systems, and also allows technical moderator and administrators to conveniently perform tracking, monitoring, maintenance and administration tasks of the system. Trong khuôn khổ của đề tài này, dịch vụ web được xây dựng nhằm phục vụ tương tác giữa là các khách hàng và hệ thống, đồng thời cũng cho phép kỹ thuật viên và quản trị viên có thể thực hiện các tác vụ theo dõi, giám sát, bảo trì và quản trị hệ thống một cách thuận tiện.

### Functional analysis and design

The main functions of the web service include: Các chức năng chính của dịch vụ web bao gồm:

* Functions to support customers to register and control their resources including crops, frames, equipment,... Các chức năng hỗ trợ khách hàng đăng ký, kiểm soát các giàn trồng, vụ trồng, thiết bị,… của mình.
* Allows administrators and technical moderators to operate, monitor and maintain the system. Cho phép các quản trị viên và quản lý kỹ thuật vận hành, giám sát và bảo trì hệ thống.

Four actors of the system include: Bốn tác nhân của hệ thống bao gồm:

* Guest: Visitors having not logged in or registered an account. Khách vãng lai chưa đăng nhập hay đăng ký tài khoản.
* Customer: Customer having registered an account. Khách hàng đã đăng ký tài khoản.
* Operator. Điều hành viên kỹ thuật.
* Administrator. Quản trị viên.

With this consideration, functions are designed as shown in the following figure and specified in the following tables. Với nhận định này, các chức năng được thiết kế như hình sau và được đặc tả trong các bảng tiếp theo.



Figure 3‑10. Use case diagram of the web service application

Table 3‑14. Specification of the **Register** use case

|  |  |
| --- | --- |
| **Name of use case**  **Tên ca sử dụng** | **Register** |
| Main actor  Tác nhân chính | Guest |
| Purpose  Mục đích | Customer registers to create user accounts. Khách hàng đăng ký tạo tài khoản người dung. |
| Activation  Kích hoạt | Click the account registration button. Ấn nút đăng ký tài khoản. |
| Description  Mô tả | 1. User fills in the profile information, then presses the create account button. Người dùng điền các thông tin khai báo hồ sơ cá nhân, rồi ấn nút tạo tài khoản. 2. System sends confirmation email. Hệ thống gửi email yêu cầu xác nhận. 3. User opens email to confirm account creation. Người dùng mở email để xác nhận tạo tài khoản. |

Table 3‑15. Specification of the **Login** use case

|  |  |
| --- | --- |
| **Name of use case**  **Tên ca sử dụng** | **Login** |
| Main actor  Tác nhân chính | Customer, Technical Moderator |
| Purpose  Mục đích | Customer already having an account or technical moderator having logged into the system. Khách hàng đã có tài khoản hoặc điều hành viên kỹ thuật đăng nhập vào hệ thống. |
| Activation  Kích hoạt | Press the login button. Ấn nút đăng nhập. |
| Description  Mô tả | 1. User enters login information including username and password, then presses the login button. Người dùng điền thông tin đăng nhập gồm tên đăng nhập và mật khẩu, rồi ấn nút đăng nhập. 2. System checks the login information the allows login. Hệ thống kiểm tra khớp thông tin đăng nhập để cho phép đăng nhập hay không. |

Table 3‑16. Specification of the **Manage Properties** use case

|  |  |
| --- | --- |
| **Name of use case**  **Tên ca sử dụng** | **Manage Properties** |
| Main actor  Tác nhân chính | Customer, Technical Moderator |
| Purpose  Mục đích | Customer or technical moderator already having an account to manage their resources, including frames, crops, devices, sensors, and actuators. Khách hàng đã có tài khoản hoặc điều hành viên kỹ thuật quản lý các tài nguyên của mình. Các tài nguyên bao gồm giàn trồng, vụ trồng, thiết bị, cảm biến, thiết bị chấp hành. |
| Activation  Kích hoạt | Select the corresponding items from the menu after logging into the corresponding account. Chọn các mục tương ứng từ menu sau khi đã đăng nhập vào tài khoản tương ứng. |
| Description  Mô tả | 1. User selects resources and view information or perform operations such as add, remove, update information. Người dùng lựa chọn các tài nguyên rồi xem thông tin hoặc thực hiện các thao tác thêm, bớt, cập nhật thông tin. 2. If the user performs add, remove, update operations, the system needs to ensure the information is valid and saves to the database. Nếu người dùng thực hiện các thao tác thêm, bớt, cập nhật, hệ thống cần đảm bảo các thông tin hợp lệ và lưu vào CSDL. |

Table 3‑17. Specification of the **Manage Customer Accounts** use case

|  |  |
| --- | --- |
| **Name of use case**  **Tên ca sử dụng** | **Manage Customer Accounts** |
| Main actor  Tác nhân chính | Technical Moderator |
| Purpose  Mục đích | Technical moderator manages customer accounts. Điều hành viên kỹ thuật quản lý các tài khoản khách hàng. |
| Activation  Kích hoạt | Select the corresponding items from the menu in the section for administrators and moderators after logging in to the account. Chọn các mục tương ứng từ menu trong mục dành cho quản trị viên và điều hành viên sau khi đã đăng nhập vào tài khoản. |
| Description  Mô tả | 1. Technical moderator views the resources or performs operations such as add, remove, update information. Điều hành viên kỹ thuật lựa chọn các tài nguyên rồi xem thông tin hoặc thực hiện các thao tác thêm, bớt, cập nhật thông tin. 2. If the user performs add, remove, update operations, the system needs to ensure that the information is valid and saves in the database. Nếu người dùng thực hiện các thao tác thêm, bớt, cập nhật, hệ thống cần đảm bảo các thông tin hợp lệ và lưu vào CSDL. |

Table 3‑18. Specification of the **Perform Maintenance Operations** use case

|  |  |
| --- | --- |
| **Name of use case**  **Tên ca sử dụng** | **Perform Maintenance Operations** |
| Main actor  Tác nhân chính | Administrator |
| Purpose  Mục đích | Administrator supervises and operates the system. Quản trị viên giám sát và vận hành, bảo trì hệ thống. |
| Activation  Kích hoạt | Select the corresponding items from the menu in the administrator section after logging in to the account. Chọn các mục tương ứng từ menu trong mục dành cho quản trị viên sau khi đã đăng nhập vào tài khoản. |
| Description  Mô tả | 1. Administrator views information or performs operations and maintenance tasks if necessary. Quản trị viên xem thông tin hoặc thực hiện các tác vụ vận hành và bảo trì nếu cần thiết. 2. System checks the necessary conditions and executes the request if the conditions are guaranteed. Hệ thống kiểm tra các điều kiện cần thiết và thực hiện yêu cầu nếu các điều kiện được đảm bảo. |

Table 3‑19. Specification of the **Manage Technical Moderator Accounts** use case

|  |  |
| --- | --- |
| **Name of use case**  **Tên ca sử dụng** | **Manage Technical Moderator Accounts** |
| Main actor  Tác nhân chính | Administrator |
| Purpose  Mục đích | Administrator manages the accounts of the technical moderators. Quản trị viên quản trị các tài khoản của các điều hành viên kỹ thuật. |
| Activation  Kích hoạt | Select the corresponding items from the menu in the administrator section after logging in to the account. Chọn các mục tương ứng từ menu trong mục dành cho quản trị viên sau khi đã đăng nhập vào tài khoản. |
| Description  Mô tả | 1. Administrator views information or performs tasks such as creating, deleting, and changing technical moderator accounts if necessary. Quản trị viên xem thông tin hoặc thực hiện các tác vụ như tạo, xoá, thay đổi tài khoản các điều hành viên kỹ thuật nếu cần thiết. 2. System checks the necessary conditions and executes the request if the conditions are guaranteed. Hệ thống kiểm tra các điều kiện cần thiết và thực hiện yêu cầu nếu các điều kiện đảm bảo. |

### Interface design

To develop the web interface, the following client-side programming libraries are used in this project: Để xây dựng giao diện web, các thư viện cho lập trình phía client chính sau đây được sử dụng trong đề tài này:

* JQuery[[4]](#footnote-4): Library helps work with DOM objects, handle CSS and perform AJAX asynchronous queries. Thư viện giúp làm việc với các đối tượng DOM, xử lý CSS và thực hiện các truy vấn không đồng bộ AJAX.
* Bootstrap[[5]](#footnote-5): Responsive web design library allows layout design and display of customized content on devices with different screen sizes such as desktop PCs, mobile devices, etc. This library is developed and published by Twitter. Thư viện thiết kế giao diện web đáp ứng (responsive) cho phép phân bố và hiển thị nội dung tuỳ biến trên các thiết bị với màn hình kích thước khác nhau như máy tính để bàn, thiết bị di động, v.v… Thư viện này được phát triển bởi Twitter.

The interface is designed with a menu and login information at the top bar. Figure 3‑11 shows the interface of the home page shown on a desktop PC when the user is not logged in. When displayed on a mobile device with a smaller screen, the menu will automatically be collapsed to the top corner as shown in Figure 3‑12. Giao diện được thiết kế với một menu và thông tin đăng nhập ở trên cùng. Hình 3-11 minh hoạ giao diện của trang chủ thể hiện trên máy tính để bàn khi người dùng chưa đăng nhập. Khi hiển thị trên thiết bị di động có màn hình nhỏ hơn, menu sẽ tự động được thu gọn sang góc màn hình như trên Hình 3-12.

The interface of some of the main pages is shown on the following images, including the login page (Figure 3‑13), user group list management page (Figure 3‑14), user list management page (Figure 3‑15), user management page (Figure 3‑16), frame and crop management page (Figure 3‑17), device, sensor, actuator management page (Figure 3‑18). Giao diện một số trang làm việc chính được thể hiện trên các hình tiếp theo, bao gồm giao diện trang đăng nhập hệ thống, giao diện quản trị danh sách nhóm người dùng, giao diện quản trị danh sách người dùng, giao diện quản trị hồ sơ người dùng, giao diện quản trị thông tin giàn trồng và các vụ trồng, giao diện quản trị thiết bị trung tâm, cảm biến, thiết bị chấp hành.

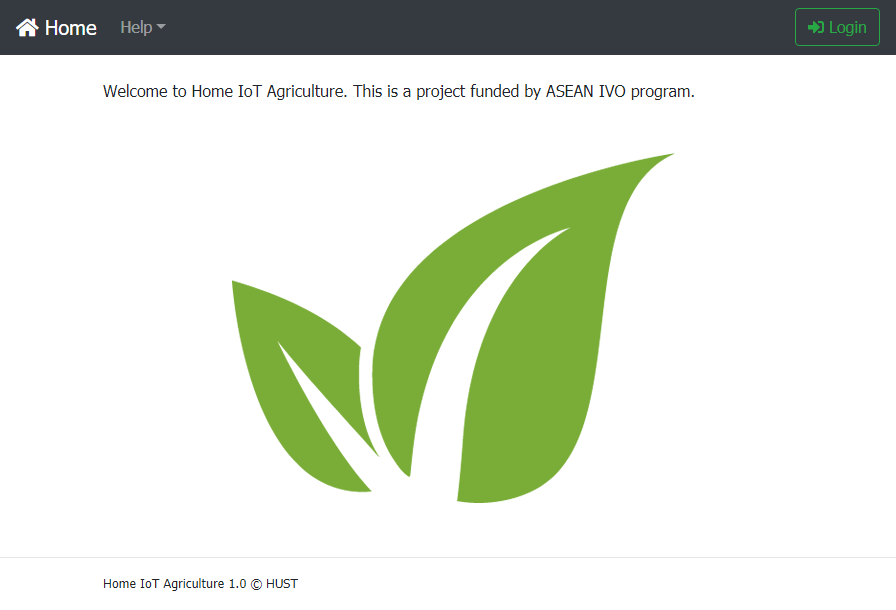


Figure 3‑11. Homepage screenshot displayed on a desktop PC

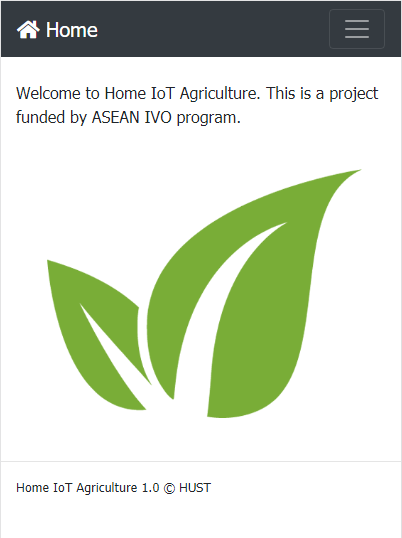


Figure 3‑12. Homepage screenshot displayed on a smartphone

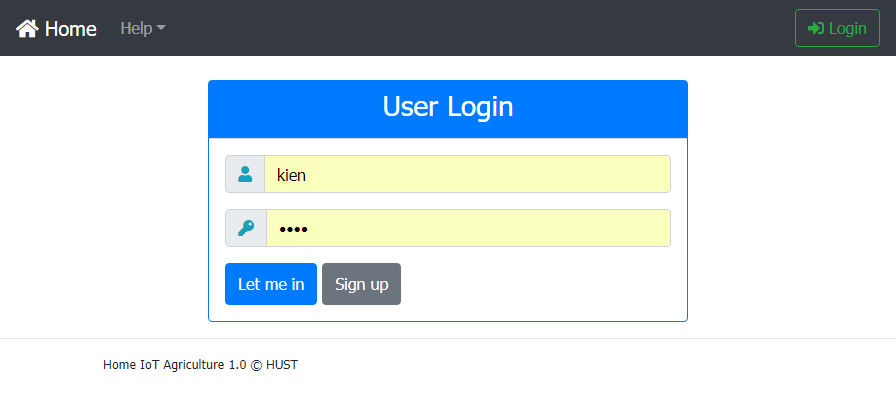


Figure 3‑13. Login page screenshot

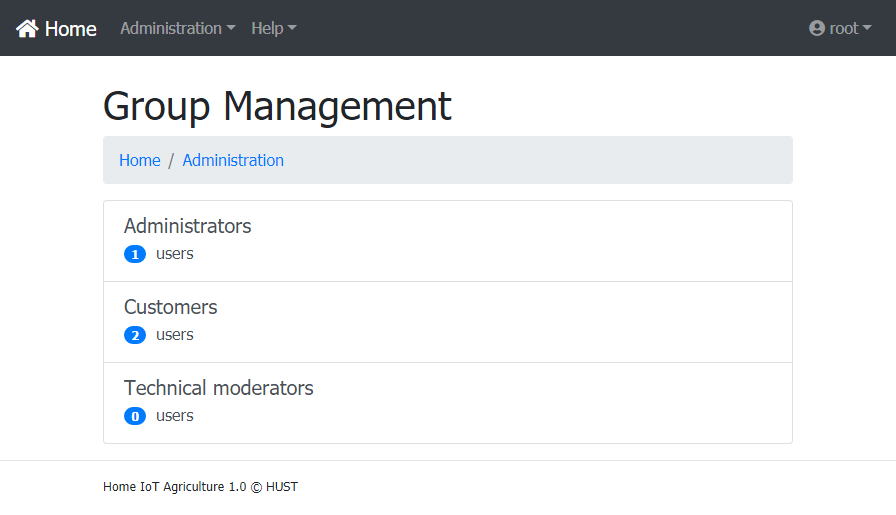


Figure 3‑14. User group list management page screenshot

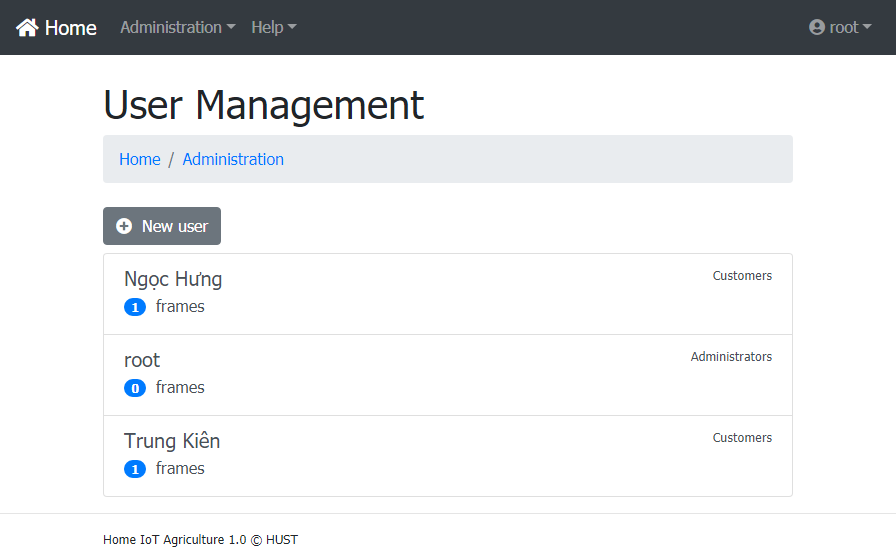


Figure 3‑15. User list management page screenshot

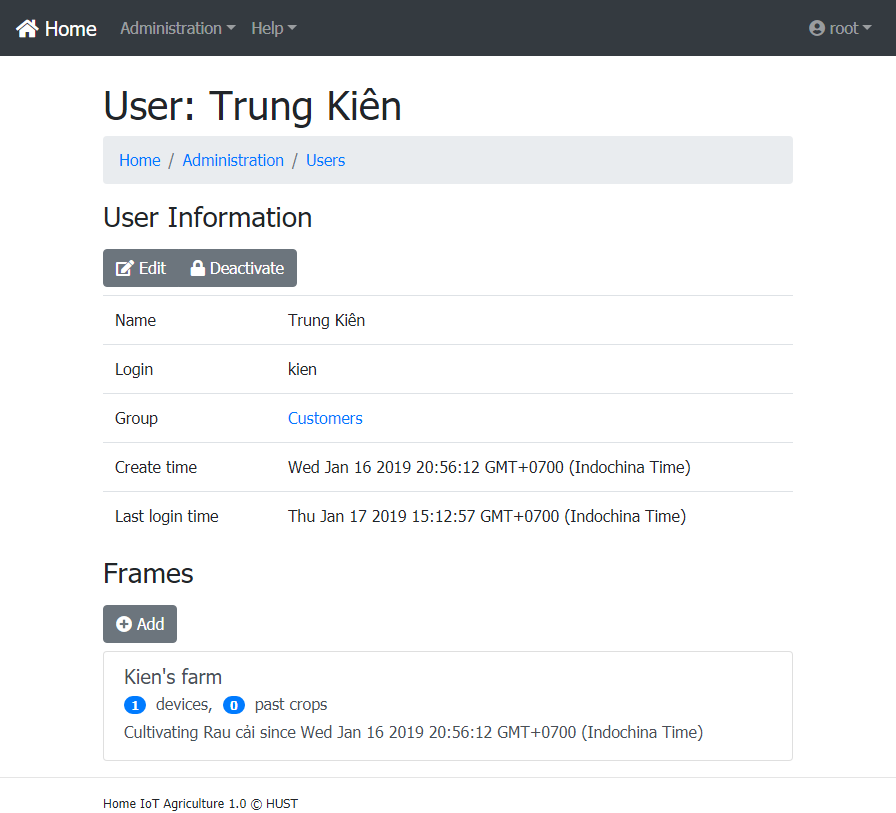


Figure 3‑16. User management page screenshot

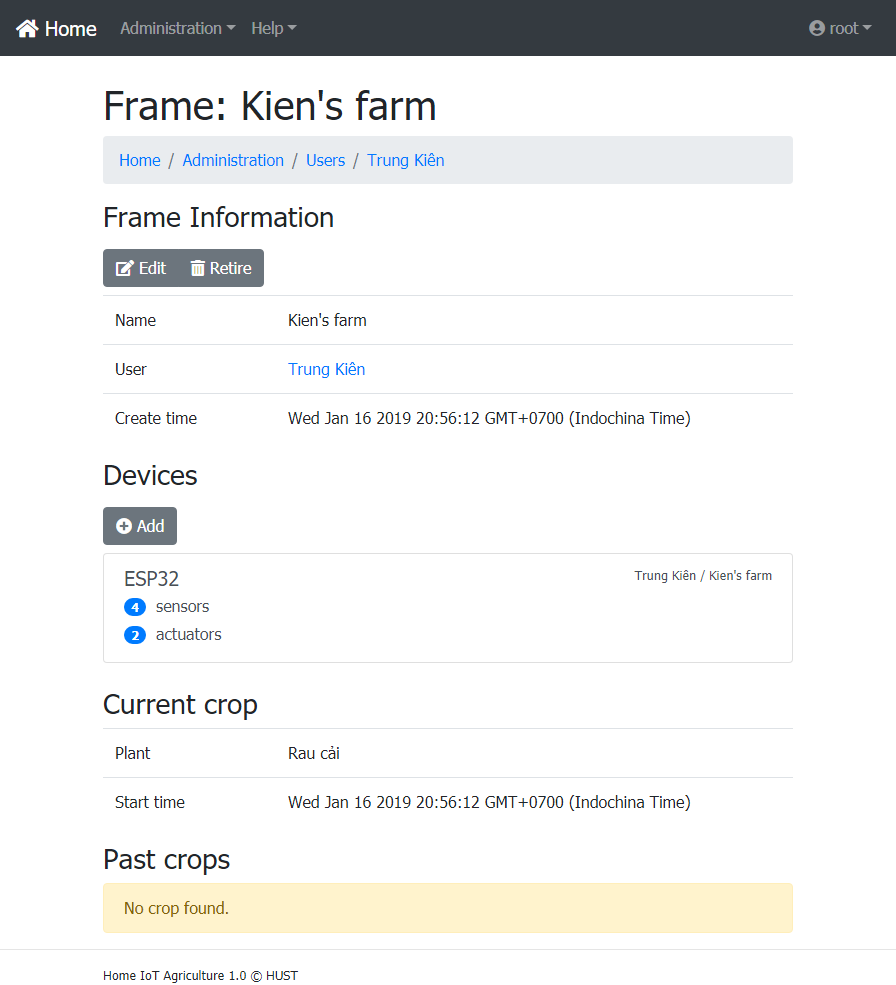


Figure 3‑17. Frame and crop management page screenshot

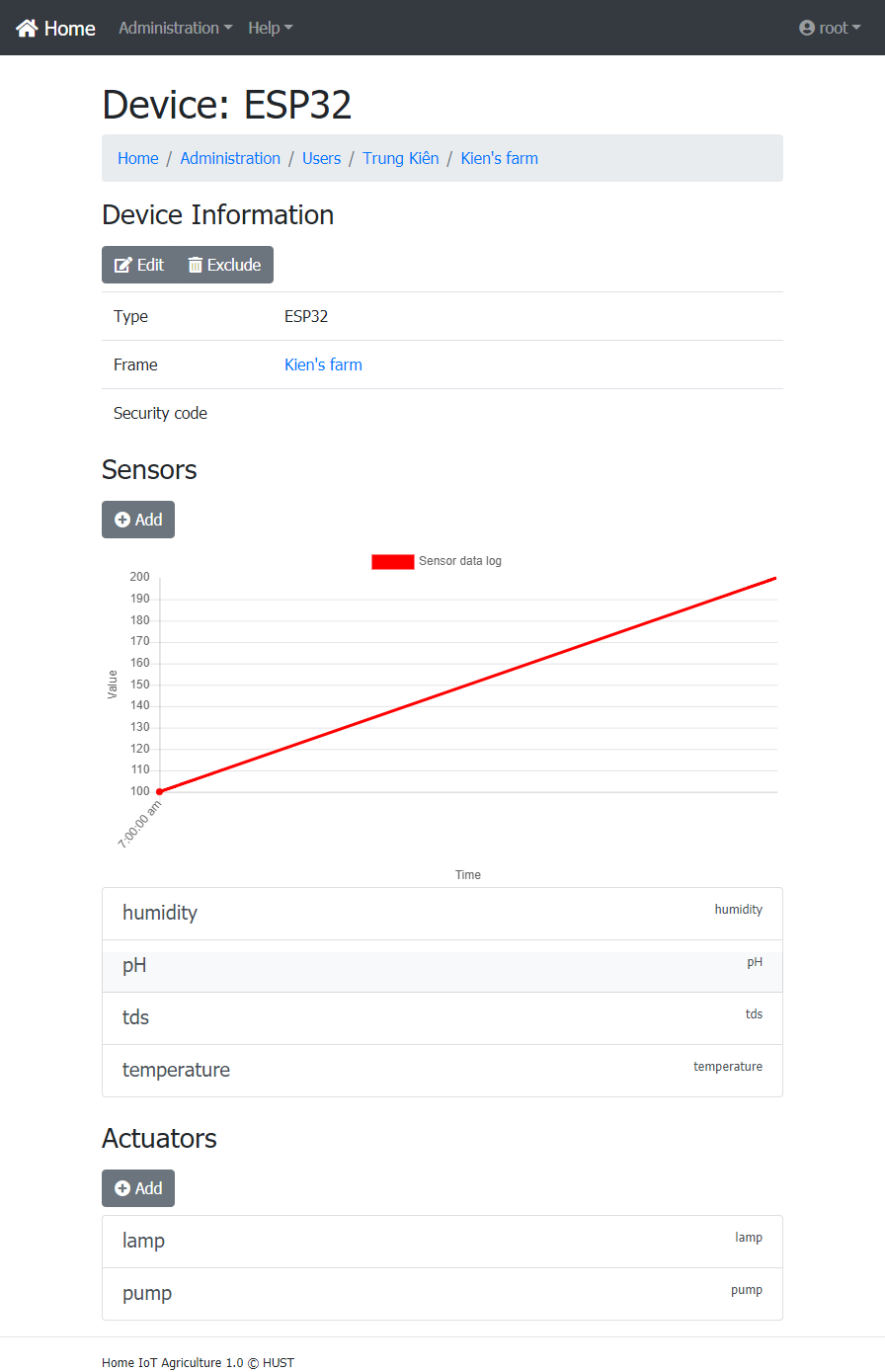


Figure 3‑18. Device, sensor, actuator management page screenshot

### Packaging and deployment

The web service application is built using NodeJS[[6]](#footnote-6) platform. This is an application platform with some major technical advantages, of which the biggest is the single threading execution and the non-blocking (asynchronous) input/output mechanisms, making programming simpler in many situations, while increasing processing speed and increasing system scalability, minimizing errors and problems caused by parallel programming in traditional programming languages. Dịch vụ web được xây dựng sử dụng nền tảng NodeJS[[7]](#footnote-7). Đây là một nền tảng ứng dụng với một số ưu điểm chính về mặt kỹ thuật, trong đó ưu điểm lớn nhất là tính đơn luồng và cơ chế xuất nhập không đồng bộ, giúp cho việc lập trình được đơn giản hoá trong nhiều vấn đề, đồng thời tăng tốc độ xử lý và tăng khả năng mở rộng hệ thống, giảm thiểu lỗi và các vấn đề do lập trình song song gây ra trong các ngôn ngữ lập trình truyền thống.

The organization of the web service application in this project is shown in the following figure. The application includes 3 modules: Việc xây dựng ứng dụng web của đề tài này được tổ chức như hình dưới. Ứng dụng gồm 3 module con:

* Database layer: makes connection and queries to the database, is developed using mysql[[8]](#footnote-8) package. Tầng CSDL: thực hiện kết nối và các truy vấn tới CSDL, được xây dựng dựa trên gói mysql.
* CLI interface: helping administrators to perform some operations directly at the server machine running the application, such as viewing logs, restarting the system, etc. Giao diện dòng lệnh: giúp quản trị viên có thể thực hiện một số thao tác trực tiếp tại máy chủ chạy app, như xem log, tạm dừng hệ thống, khởi động lại hệ thống, v.v…
* Web interface: is the main function of the application, providing features that can be accessed via the web interface using the browser. This module is developed using packages for server-side programming: Giao diện web: là chức năng chính của phần mềm, cung cấp các tính năng có thể được truy cập qua giao diện web sử dụng trình duyệt. Module này được xây dựng dựa trên một số gói chính cho lập trình phía server:
  + express[[9]](#footnote-9): Web service development platform for NodeJS. Nền tảng phát triển dịch vụ web cho NodeJS.
  + pug[[10]](#footnote-10): Template processing library for web design. Thư viện xử lý khuôn mẫu (template) cho thiết kế web.
  + morgan[[11]](#footnote-11): Logging library for web requests. Thư viện ghi log cho các truy vấn qua web.

The system after being built can be deployed according to the diagram in the next figure. Web service application connects to database server to query data, IoT server to send control commands directly from users when needed, and provides services for web users to access. Hệ thống sau khi được xây dựng sẽ được triển khai theo sơ đồ trong hình tiếp theo. Phần mềm dịch vụ web kết nối với máy chủ CSDL để truy vấn CSDL, máy chủ IoT để gửi lệnh điều khiển trực tiếp từ người dùng khi cần, và cung cấp dịch vụ cho các trình duyệt web của người dùng truy cập tới.



Figure 3‑19. Package diagram of the web service provider using NodeJS



Figure 3‑20. Deployment diagram of the web service delivery system

## Data analysis service for automatic control

Data from sensors such as nutrient concentrations in solution, pH, temperature, and moisture content were collected from monitoring devices throughout the growth and development of plants for each plant Hydroponic. This data is stored in the database. Since then, the system has been built more functions of data analysis to serve the problem of automatic control of crop care process. Data analysis functions will be built in the form of modules integrated into the system. This is the next job that the topic will do. Dữ liệu từ các cảm biến như nồng độ dinh dưỡng trong dung dịch, nồng độ pH, nhiệt độ, độ ẩm được thu thập từ các thiết bị giám trong suốt quá trình sinh trưởng và phát triển của cây trồng ứng với mỗi giàn trồng cây thủy canh. Dữ liệu này được lưu trữ trong cơ sở dữ liệu. Từ đó hệ thống được xây dựng thêm chức năng phân tích dữ liệu (data analysis) để phục vụ cho bài toán điều khiển tự động quá trình chăm sóc cây trồng. Chức năng phân tích dữ liệu sẽ được xây dựng dưới dạng các mô đun tích hợp vào hệ thống. Đây là một công việc tiếp theo mà đề tài sẽ thực hiện.

# Device firmware development

## Firmware building platform for ESP32 device

As defined in the overall design analysis of the entire system, in this section, we proceed to design for the software (firmware) on the collection and control devices. The system's solution is to build control equipment based on ESP32 embedded computers with the necessary sensors and control mechanisms. The tasks on the device are relatively complicated including network connection via wifi and communication functions to send and receive data via MQTT protocol, so software on ESP32 will be built on the operating system platform Embed FreeRTOS, a compact embedded operating system that supports mechanisms for managing tasks and resources on the device. In order to be able to deploy the necessary functions of this device-side software, it is necessary to approach the survey of ESP32 hardware resources provided as well as FreeRTOS operating system capabilities as the foundation for software development. Như đã xác định trong phần phân tích thiết kế tổng quan của cả hệ thống, trong phần này, chúng tôi tiến hành thiết kế cho phần mềm (firmware) trên các thiết bị thu thập và điều khiển. Giải pháp của hệ thống là sẽ xây dựng thiết bị thu thập điều khiển dựa trên máy tính nhúng ESP32 ghép nối với các cảm biến và các cơ cấu điều khiển cần thiết. Các nhiệm vụ trên thiết bị là tương đối phức tạp bao gồm kết nối mạng qua wifi và các chức năng giao tiếp gửi nhận dữ liệu thông qua giao thức MQTT, vì vậy phần mềm trên ESP32 sẽ được xây dựng trên nền tảng hệ điều hành nhúng FreeRTOS, một hệ điều hành nhúng nhỏ gọn hỗ trợ các cơ chế quản lý các tác vụ và tài nguyên trên thiết bị. Để có thể triển khai các chức năng cần thiết của phần mềm phía thiết bị này, cần tiếp cận khảo sát các tài nguyên phần cứng của ESP32 cung cấp cũng như các khả năng của hệ điều hành FreeRTOS làm nền tảng để xây dựng phần mềm.

### Hardware resources of ESP32 device

ESP32 is an embedded computer with lots of features, integrated on one IC. The following figure is an image of ESP32 Dev-Kit hardware device used for collecting and controlling equipment. ESP32 là một máy tính nhúng với rất nhiều tính năng, được tích hợp trên một vi mạch. Hình dưới là hình ảnh của thiết bị phần cứng ESP32 Dev-Kit được sử dụng cho thiết bị thu thập và điều khiển.



Figure 4‑1. ESP32 Dev-Kit module used for collecting and controlling devices

The diagram of functional blocks of ESP32 is shown in the following figure. Sơ đồ các khối chức năng của ESP32 được thể hiện qua hình sau.

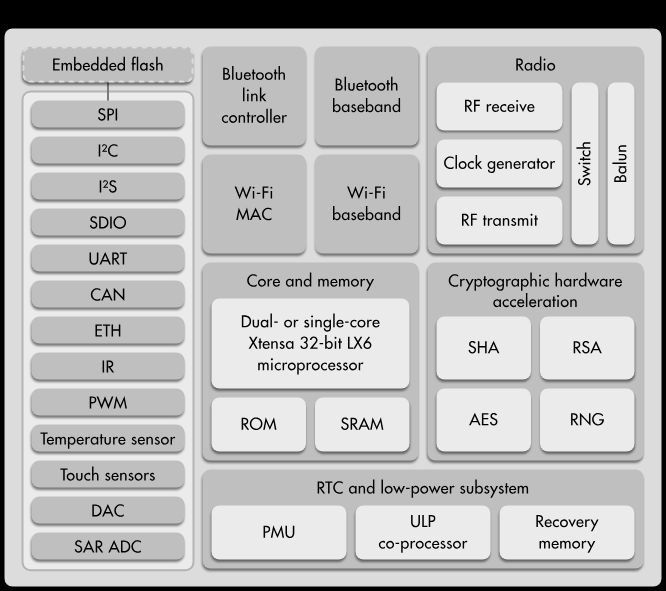


Figure 4‑2. Sơ đồ khối của ESP32

ESP32 hardware specification: Đặc điểm phần cứng của ESP32:

Processors:

* + Main processor: Tensilica Xtensa 32-bit LX6 microprocessor
  + Cores: 2 or 1 (depending on variation)
  + All chips in the ESP32 series are dual-core except for ESP32-S0WD, which is single-core.
    - Clock frequency: up to 240 MHz
    - Performance: up to 600 DMIPS
  + Ultra low power co-processor:  allows you to do ADC conversions, computation, and level thresholds while in deep sleep.

Wireless connectivity:

* + Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
  + Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE)

Internal memory:

* + ROM: 448 KiB
  + SRAM: 520 KiB (For booting and core functions)
  + RTC fast SRAM: 8 KiB (For data and instruction)
  + RTC slow SRAM: 8 KiB (For data storage and main CPU during RTC Boot from the deep-sleep mode)
  + eFuse: 1 Kibit (For co-processor accessing during deep-sleep mode)
  + Embedded flash: Flash connected internally via IO16, IO17, SD\_CMD, SD\_CLK, SD\_DATA\_0 and SD\_DATA\_1 on ESP32-D2WD and ESP32-PICO-D4.
    - 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips)
    - 2 MiB (ESP32-D2WD chip)
    - 4 MiB (ESP32-PICO-D4 SiP module)

**External flash & SRAM:** ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches.

* + Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported.
  + Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.
  + ESP32 chips with embedded flash do not support the address mapping between external flash and peripherals.

**Peripheral input/output:** Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I²C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I²S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

**Security:**

* + IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
  + Secure boot
  + Flash encryption
  + 1024-bit OTP, up to 768-bit for customers
  + Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

### Real-time operating system (RTOS) platform

The RTOS operating system is an essential platform in the field of building applications on embedded systems with a larger and more complex program size. RTOS will split complex applications into smaller and more manageable components. Benefits that RTOS provides: Hệ điều hành RTOS một nền tảng cần thiết trong trường xây dựng các ứng dụng trên hệ nhúng có kích thước chương trình lớn dần, độ phức tạp cao. RTOS sẽ chia nhỏ ứng dụng phức tạp thành các thành phần nhỏ hơn và dễ quản lý hơn. Các lợi ích mà RTOS cung cấp:

Allow multiple tasks to run at the same time (multi-tasking). Cho phép nhiều tác vụ chạy cùng 1 lúc (multi-tasking).

Has hardware resource management and provides services for different tasks. Có quản lý tài nguyên về phần cứng và cung cấp các dịch vụ cho các tác vụ khác nhau.

Share resources simply: provide a mechanism to divide the memory and peripheral requirements of MCU. Chia sẻ tài nguyên một cách đơn giản: cung cấp cơ chế để phân chia các yêu cầu về bộ nhớ và ngoại vi của MCU.

Easy to debug and develop: Everyone in the group can work independently, programmers can avoid interactions with interrupts, timers, and hardware (this is not recommended because I understand hardware will still be much better). Dễ debug và phát triển: Mọi người trong nhóm có thể làm việc một cách độc lập, các lập trình viên thì có thể tránh được các tương tác với ngắt, timer, với phần cứng (cái này mình không khuyến khích lắm vì hiểu được phần cứng vẫn sẽ tốt hơn nhiều).

Increase flexibility and ease of maintenance: through API of RTOS,... Tăng tính linh động và dễ dàng bảo trì: thông qua API của RTOS,…

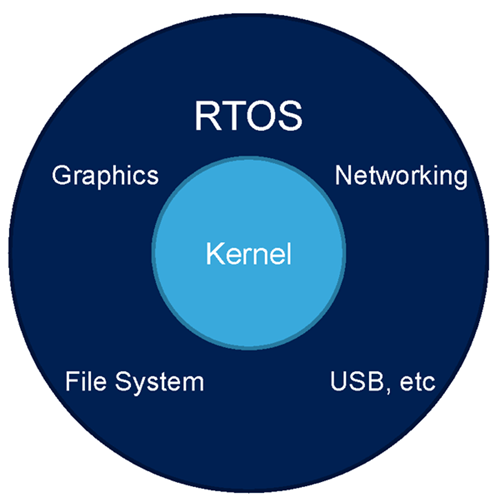


Figure 4‑3. Basic architecture of RTOS operating system

To exploit the features of RTOS provided for building applications on RTOS platforms, it is necessary to study and understand the basic features that the operating system provides such as the kernel, task, and scheduler architecture. The following section will present detailed research on these contents. Để khai thác các tính năng của RTOS cung cấp cho việc xây dựng ứng dụng trên nền tảng RTOS, cần nghiên cứu, tìm hiểu những đặc điểm cơ bản mà hệ điều hành cung cấp như kiến trúc nhân (kernel), các tác vụ (tasks), lập lịch (scheduler). Phần sau sẽ trình bày những nghiên cứu chi tiết về những nội dung này.

#### OS kernel (Nhân hệ điều hành)

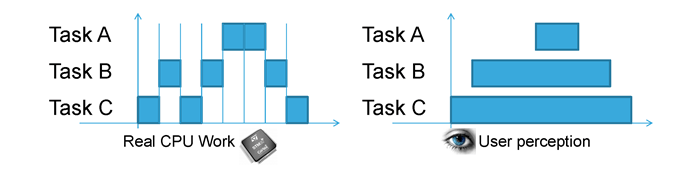


Figure 4‑4. Multitasking execution mechanism

The kernel will have the task of managing multiple tasks at the same time, each task will run for several ms. At the end of the normal task: Kernel sẽ có nhiệm vụ quản lý nhiều task cùng chạy 1 lúc, mỗi task thường chạy mất vài ms. Tại lúc kết thúc task thường:

Save task state. Lưu trạng thái task.

The CPU register will load the status of the next task. Thanh ghi CPU sẽ load trạng thái của task tiếp theo.

The next task needs about a few ms to perform. Task tiếp theo cần khoảng vài ms để thực hiện.

Since the CPU performs the task very quickly, under the user perspective, tasks are almost done continuously. Vì CPU thực hiện tác vụ rất nhanh nên dưới góc nhìn người dùng thì hầu như các task là được thực hiện một cách liên tục.

#### Task (Tác vụ)

A task is a process, this process can run continuously in an endless loop and never stop. For each task, the only belief is that only it is running and can use up the available resources of the processor although it is a fact, it still has to share this resource with other tasks. Một tác vụ (task) là một tiến trình, tiến trình này có thể chạy liên tục trong vòng lặp vô tận và không bao giờ dừng lại. Với mỗi task thì có niềm tin duy nhất là chỉ mình nó đang chạy và có thể sử dụng hết nguồn tài nguyên sẵn có của bộ xử lý mặc dù là thực tế thì nó vẫn phải chia sẻ nguồn tài nguyên này với các task khác.

An application may have many different tasks. For example, an automatic vending machine can be built with the following tasks: Một ứng dụng có thể có nhiều tác vụ khác nhau. Ví dụ một máy bán nước tự động có thể được xây dựng với các tác vụ như sau:

The task of managing user selections. Tác vụ quản lý việc lựa chọn của người dùng.

Action to properly verify the amount the user has paid. Tác vụ để kiểm tra đúng số tiền người dùng đã trả.

Action for motor control/drinking water supply structure. Tác vụ để điều khiển động cơ/cơ cấu cung cấp nước uống.

Kernel sẽ quản lý việc chuyển đổi giữa các task, nó sẽ lưu lại ngữ cảnh của task sắp bị hủy và khôi phục lại ngữ cảnh của task tiếp theo bằng cách: The kernel will manage the transition between tasks, it will save the context of the task to be canceled and restore the context of the next task by:

Check the predefined execution time (the time slice is generated by the systick interrupt). Kiểm tra thời gian thực thi đã được định nghĩa trước (time slice được tạo ra bởi ngắt systick).

When there are unblocking events a higher authority task occurs (signal, queue, semaphore, ...). Khi có các sự kiện unblocking một task có quyền cao hơn xảy ra (signal, queue, semaphore,…).

When the task calls Yield() to force the kernel to switch to other tasks without waiting for the end of the time slice. Khi task gọi hàm Yield() để ép Kernel chuyển sang các task khác mà không phải chờ cho hết time slice.

When booting, the kernel will create a default task called Task Idle. To create a task, you need to declare the task definition function, then create task and memory allocation, provided by the operating system API. Khi khởi động thì kernel sẽ tạo ra một task mặc định gọi là Idle Task. Để tạo một task thì cần phải khai báo hàm định nghĩa task, sau đó tạo task và cấp phát bộ nhớ, được cung cấp bởi API của hệ điều hành.

A task in RTOS usually has the following states: Một task trong RTOS thường có các trạng thái như sau:

RUNNING: đang thực thi

READY: sẵn sàng để thực hiện

WAITING: chờ sự kiện

INACTIVE: không được kích hoạt

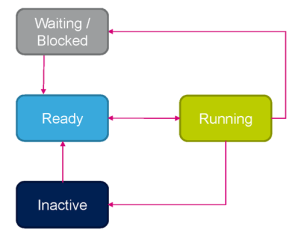


Figure 4‑5. Diagram of the status of a task when executed on RTOS

#### Scheduler (Cơ chế lập lịch của hệ điều hành)

To manage the execution of tasks on the system, a scheduler will take care of scheduling tasks to be executed. The scheduler is an important component of the kernel, which determines which tasks will be executed. There are several rules for scheduling the following: Để quản lý sự thực thi của các tác vụ trên hệ điều, một bộ lập lịch (scheduler) sẽ đảm nhiệm việc lập lịch cho các tác vụ được thực thi. Bộ lập lịch là một thành phần quan trọng của nhân hệ điều hành (kernel), quyết định tác vụ nào sẽ được thực thi. Có một số luật qui định cho việc lập lịch (scheduling) như sau:

Cooperative: Giống với lập trình thông thường, mỗi task chỉ có thể thực thi khi task đang chạy dừng lại, nhược điểm của nó là task này có thể dùng hết tất cả tài nguyên của CPU. Cooperative: Just like normal programming, each task can only be executed when the task is running, its drawback is that this task can use all CPU resources.

Round-robin: each task is executed in a predetermined time slice and has no priority. Round-robin: mỗi task được thực hiện trong thời gian định trước (time slice) và không có ưu tiên.

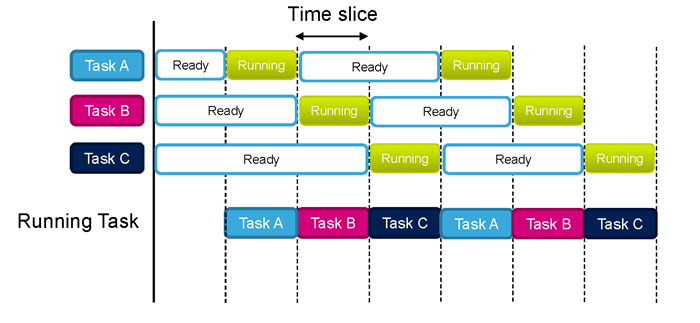


Figure 4‑6. Scheduling mechanism for tasks to be executed

Priority base: The highest delegated task will be performed first, if the tasks with the same permissions are the same as round-robin, lower priority tasks will be performed until the end of the time slice. Priority base: Task được phân quyền cao nhất sẽ được thực hiện trước, nếu các task có cùng quyền như nhau thì sẽ giống với round-robin, các task có mức ưu tiên thấp hơn sẽ được thực hiện cho đến cuối time slice.

* + Task A waits for events. Task A chờ event.
  + Task B waits for events. Task B chờ event.
  + Task B event ready. Task B event sẵn sàng.
  + Task A event ready. Task A event sẵn sàng.

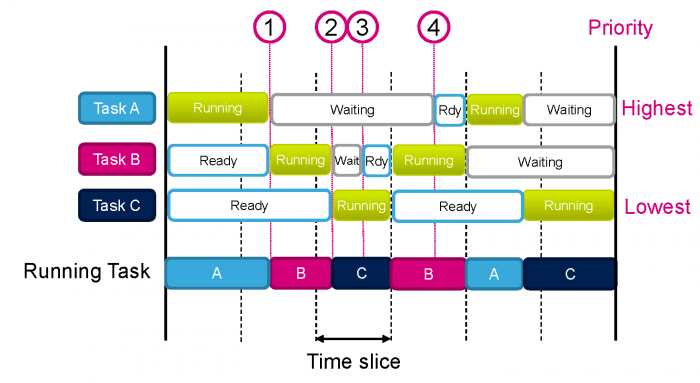


Figure 4‑7. Scheduling with authorization for tasks

Priority-based pre-emptive: The highest priority tasks always yield lower priority tasks than before. Priority-based pre-emptive: Các task có mức ưu tiên cao nhất luôn nhường các task có mức ưu tiên thấp hơn thực thi trước.

#### Inter-task communication and resource sharing (Giao tiếp giữa các tác vụ và chia sẻ tài nguyên)

The tasks need to connect and exchange data together to be able to share resources, some mechanisms are provided including: Các task cần phải kết nối và trao đổi dữ liệu với nhau để có thể chia sẻ tài nguyên, một số cơ chế được cung cấp gồm:

Inter-task communication: Với giao tiếp giữa các tác vụ:

* + Signal events. Đồng bộ các task.
  + Message queue. Trao đổi tin nhắn giữa các task trong hoạt động giống như FIFO.
  + Mail queue. Trao đổi dữ liệu giữa các task sử dụng hằng đợi của khối bộ nhớ.

Resource sharing: Với chia sẻ tài nguyên:

* + Semaphores. Truy xuất tài nguyên liên tục từ các task khác nhau.
  + Mutex. Đồng bộ hóa truy cập tài nguyên sử dụng Mutual Exclusion.

**Signal event: Truyền tín hiệu giữa các tác vụ:**

Signal event is used to synchronize tasks, such as catching tasks that must be performed at a certain event. Signal event được dùng để đồng bộ các task, ví dụ như bắt task phải thực thi tại một sự kiện nào đó được định sẵn.

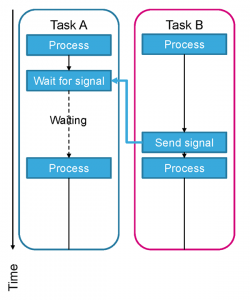


Figure 4‑8. Communication mechanism between two signal event tasks

Example: A washing machine has 2 tasks, Task A controls the motor, Task B reads the water level from the input water sensor. Ví dụ: Một cái máy giặt có 2 task là Task A điều khiển động cơ, Task B đọc mức nước từ cảm biến nước đầu vào.

Task A needs to wait for the water to fill before starting the engine. This can be done using the signal event. Task A cần phải chờ nước đầy trước khi khởi động động cơ. Việc này có thể thực hiện được bằng cách sử dụng signal event.

Task A must wait for the signal event from Task B before starting the engine. Task A phải chờ signal event từ Task B trước khi khởi động động cơ.

When it detects that the water has reached the required level, Task B will send a signal to Task A. Khi phát hiện nước đã đạt tới mức yêu cầu thì Task B sẽ gửi tín hiệu tới Task A.

In this case, the task will wait for the signal before executing, it will be in the state of WAITING until the signal is set. In addition, we can set one or more signals in any other task. Với trường hợp này thì task sẽ đợi tín hiệu trước khi thực thi, nó sẽ nằm trong trạng thái là WAITING cho đến khi signal được set. Ngoài ra ta có thể set một hoặc nhiều signal trong bất kỳ các task nào khác.

Each task can be assigned a maximum of 32 signal events. Mỗi task có thể được gán tối đa là 32 signal event.

The advantage is that it performs quickly, uses less RAM than the semaphore and message queue, but has the disadvantage that is only used when a task receives a signal. Ưu điểm của nó là thực hiện nhanh, sử dụng ít RAM hơn so với semaphore và message queue nhưng có nhược điểm lại chỉ được dùng khi một task nhận được signal.

**Message queue: Cơ chế hàng đợi thông điệp:**

Message queues are mechanisms that allow tasks to be connected, which is a FIFO buffer defined by length (the number of elements that the buffer can store) and the data size (the size of the components). in the buffer). A typical application is the buffer for Serial I / O, the buffer for the command is sent to the task. Message queue là cơ chế cho phép các task có thể kết nối với nhau, nó là một FIFO buffer được định nghĩa bởi độ dài (số phần tử mà buffer có thể lưu trữ) và kích thước dữ liệu (kích thước của các thành phần trong buffer). Một ứng dụng tiêu biểu là buffer cho Serial I/O, buffer cho lệnh được gửi tới task.

https://i2.wp.com/hocarm.org/wp-content/uploads/2017/06/Message-queue.png?resize=400%2C61&ssl=1

Figure 4‑9. Communication mechanism between two messages using the message queue

Task that can write to the queue: Task có thể ghi vào hàng đợi:

The task will be blocked when sending data to a full message queue. Task sẽ bị khóa (block) khi gửi dữ liệu tới một message queue đầy đủ.

Tasks will be completely unblocked when the memory in the message queue is empty. Task sẽ hết bị khóa (unblock) khi bộ nhớ trong message queue trống.

If multiple tasks are blocked, the task with the highest priority will be unblocked first. Trường hợp nhiều task mà bị block thì task với mức ưu tiên cao nhất sẽ được unblock trước.

Task that can read from the queue: Task có thể đọc từ hàng đợi:

The task will be blocked if the message queue is empty. Task sẽ bị block nếu message queue trống.

Task will be unblocked if there is data in the message queue. Task sẽ được unblock nếu có dữ liệu trong message queue.

Similarly write the task to be unblocked based on priority. Tương tự ghi thì task được unblock dựa trên mức độ ưu tiên.

Example: Ví dụ:

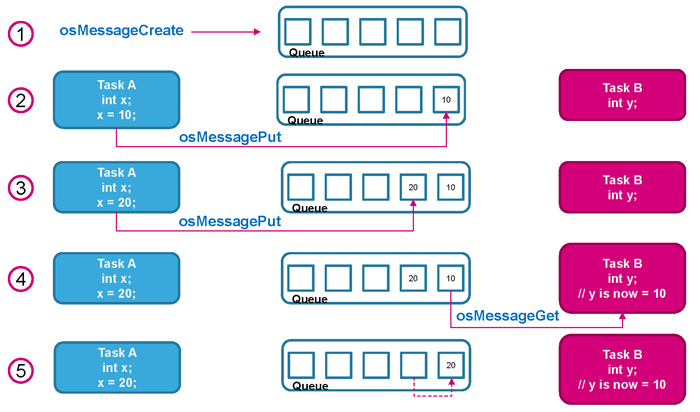


Figure 4‑10. Communication mechanism using the message queue with moderation mechanism

***Semaphore:* Cơ chế cơ báo:**

Used to synchronize tasks with other events in the system. There are 2 types: Được sử dụng để đồng bộ task với các sự kiện khác trong hệ thống. Có 2 loại:

Binary semaphore:

Special case of counting semaphore. Trường hợp đặc biệt của counting semaphore.

There is only 1 token. Có duy nhất 1 token.

Only 1 synchronous activity. Chỉ có 1 hoạt động đồng bộ.

Counting semaphore:

There are many synchronous activities. Có nhiều hoạt động đồng bộ.

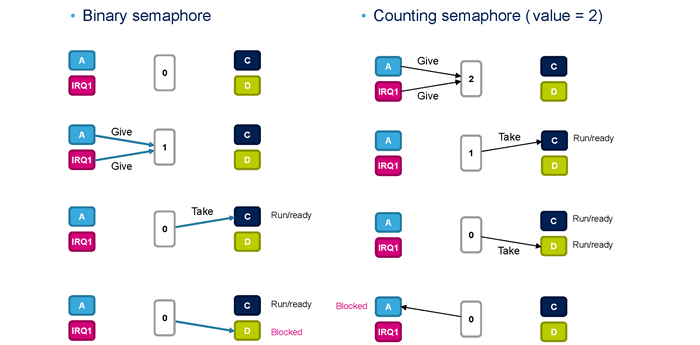


Figure 4‑11. Communication mechanism for 2 tasks using the semaphore alarm engine

Counting semaphore is used to: Couting semaphore được dùng để:

Counting event

An event handler will ‘give’ the semaphore when an event occurs (increase the semaphore count value). Một event handler sẽ ‘give’ semaphore khi có event xảy ra (tăng giá trị đếm semaphore).

A task handler will ‘take’ semaphore when it executes the event (reduce the semaphore count). Một task handler sẽ ‘take’ semaphore khi nó thực thi sự kiện (giảm giá trị đếm semaphore).

Count value is the difference between the number of events occurring and the number of events executed. Count value là khác nhau giữa số sự kiện xảy ra và số sự kiện được thực thi.

In the case of a counting event, the semaphore is initialized to a zero value. Trong trường hợp counting event thì semaphore được khởi tạo giá trị đếm bằng 0.

Resource management

Count value will indicate the number of available resources. Count value sẽ chỉ ra số resource sẵn có.

To control and control task resources based on the count value of the semaphore, if the count value drops to 0, there is no free resource. Để điều khiển và kiểm soát được resource của task dựa trên count value của semaphore (giá trị giảm), nếu count value giảm xuống bằng 0 nghĩa là không có resource nào free.

When a task finishes with the resource, it will give the semaphore back to increase the count value of the semaphore. Khi một task finish với resource thì nó sẽ give semaphore trở lại để tăng count value của semaphore.

In the case of resouce management, count value will be equal to the max value of count value when the semaphore is created. Trong trường hợp resouce management thì count value sẽ bằng với giá trị max của count value khi semaphore được tạo.

***Mutex:* Cơ chế loại trừ:**

Use for mutial exclution, acting as a token to protect shared resources. A task if you want to access shared resources. Sử dụng cho việc loại trừ (mutial exclution), hoạt động như là một token để bảo vệ tài nguyên được chia sẻ. Một task nếu muốn truy cập vào tài nguyên chia sẻ.

Need to request (wait) the mutex before accessing shared resources. Cần yêu cầu (đợi) mutex trước khi truy cập vào tài nguyên chia sẻ.

Give the token when ending with the resource. Đưa ra token khi kết thúc với tài nguyên.

At each time, only 1 task has a mutex. Other tasks that want the same mutex must be blocked until the old task releases the mutex. Tại mỗi một thời điểm thì chỉ có 1 task có được mutex. Những task khác muốn cùng mutex thì phải block cho đến khi task cũ thả mutex ra.

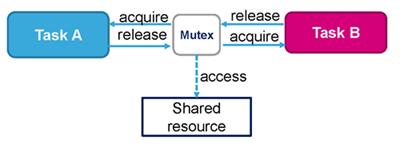


Figure 4‑12. Moderation mechanism between two tasks using mutex

Basically, Mutex is like a binary semaphore but is used for exclusion, not synchronization. In addition, it includes the priority inheritance mechanism to minimize the problem of priority reversal, this mechanism can be understood simply by the following example: Về cơ bản thì Mutex giống như binary semaphore nhưng được sử dụng cho việc loại trừ chứ không phải đồng bộ. Ngoài ra thì nó bao gồm cơ chế thừa kế mức độ ưu tiên (Priority inheritance mechanism) để giảm thiểu vấn đề đảo ngược ưu tiên, cơ chế này có thể hiểu đơn giản qua ví dụ sau:

Task A (low priority) requires a mutex. Task A (low priority) yêu cầu mutex.

Task B (high priority) wants to request the same mutex. Task B (high priority) muốn yêu cầu cùng mutex trên.

Task A's priority will be temporarily taken to Task B to allow Task A to be executed. Mức độ ưu tiên của Task A sẽ được đưa tạm về Task B để cho phép Task A được thực thi.

Task A will drop the mutex, the priority will be restored and allow Task B to continue executing. Task A sẽ thả mutex ra, mức độ ưu tiên sẽ được khôi phục lại và cho phép Task B tiếp tục thực thi.

### Device firmware construction using FreeRTOS platform

FreeRTOS is an open source real-time operating system (Real Time Operating System) developed by Real Time Engineers Ltd, founded and owned by Richard Barry. FreeRTOS is designed to be suitable for many compact embedded systems because it only performs very few functions such as basic memory and task management mechanism, important API functions for synchronization mechanism. It does not provide pre-loaded network interfaces, drivers, or file systems like other high-end embedded operating systems. However, FreeRTOS has many advantages, supports many microcontroller architectures, compact size (4.3 Kbytes after compiling on ARM7), written in C language and can be used and developed with many different C compilers (GCC, OpenWatcom, Keil, IAR, Eclipse, ...), allowing unlimited running tasks simultaneously, without limiting execution priority, hardware exploitation capabilities. In addition, it also allows deploying moderation mechanisms between processes such as: queues, counting semaphore, mutexes. FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry. FreeRTOS được thiết kế phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như: cơ chế quản lý bộ nhớ và tác vụ cơ bản, các hàm API quan trọng cho cơ chế đồng bộ. Nó không cung cấp sẵn các giao tiếp mạng, drivers, hay hệ thống quản lý tệp (file system) như những hệ điều hành nhúng cao cấp khác. Tuy vậy, FreeRTOS có nhiều ưu điểm, hỗ trợ nhiều kiến trúc vi điều khiển khác nhau, kích thước nhỏ gọn (4.3 Kbytes sau khi biên dịch trên ARM7), được viết bằng ngôn ngữ C và có thể sử dụng, phát triển với nhiều trình biên dịch C khác nhau (GCC, OpenWatcom, Keil, IAR, Eclipse, …), cho phép không giới hạn các tác vụ chạy đồng thời, không hạn chế quyền ưu tiên thực thi, khả năng khai thác phần cứng. Ngoài ra, nó cũng cho phép triển khai các cơ chế điều độ giữa các tiến trình như: queues, counting semaphore, mutexes.

The ESP32 software is built based on exploiting the capabilities provided by the FreeRTOS operating system, including: system scheduling, system resource management, and task management mechanisms. Phần mềm ESP32 được xây dựng sẽ dựa trên việc khai thác những khả năng được cung cấp của hệ điều hành FreeRTOS, bao gồm: cơ chế lập lịch, quản lý tài nguyên của hệ thống, cơ chế quản lý các tác vụ.

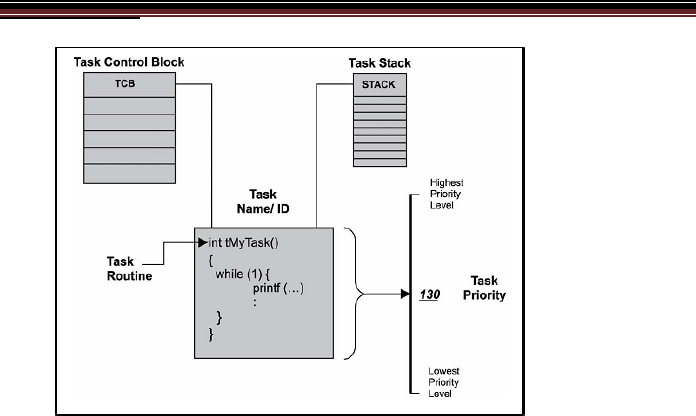


Figure 4‑13. The basic task structure diagram deployed on FreeRTOS

FreeRTOS provides api functions to work with tasks. Functions to be used include: FreeRTOS cung cấp các hàm api để làm việc với các tác vụ (task). Các hàm sẽ được sử dụng bao gồm:

**Create task: Tạo task:**

portBASE\_TYPE xTaskCreate(

pdTASK\_CODE pvTaskCode,

const portCHAR \* const pcName,

unsigned portSHORT usStackDepth,

void \*pvParameters,

unsigned portBASE\_TYPE uxPriority,

xTaskHandle \*pvCreatedTask);

Description: Create a Task and add the list of tasks ready to execute. Mô tả: Tạo một Task và thêm vào danh sách các task sẵn sàng để thực thi.

Parameters: Tham số:

* + pvTaskCode: Pointer to the function to perform for the task. Pointer đến hàm thực hiện cho task.
  + pcName: Descriptive name for that task (only for system debugging purposes). The maximum length of the name is defined by configMAX\_TASK\_NAME\_LEN. Tên mô tả cho task đó (chỉ dùng cho mục đích debug hệ thống). Chiều dài tối đa của tên được định nghĩa bằng configMAX\_TASK\_NAME\_LEN.
  + usStackDepth: The size of the stack represents the number of variables that the task can manage. Kích thước của stack đặc trưng cho số lượng biến mà task có thể quản lý được.
  + pvParameters: Pointer is used as a parameter of task creation. Pointer được sử dụng như là tham số của việc tạo task.
  + uxPriority: Task priority. Độ ưu tiên của task.
  + pvCreatedTask: Pointer points to the task created. Pointer trỏ đến task được tạo.

Return value: pdPASS if the task is successfully created. Trị trả về: pdPASS nếu như task được tạo thành công.

**Delete task: Xóa task:**

void vTaskDelete(

xTaskHandle pxTask);

Description: Delete a task from RTOS. Mô tả: Xóa một task từ RTOS.

Parameters: Tham số:

* + pxTask: Pointer to the task to be deleted. Pointer đến task cần xóa.

Return value: None. Trị trả về: Không có.

**Delay task: Tạm dừng task:**

void vTaskDelay(

portTickType xTicksToDelay);

Description: Delay (block) task for a period since the last function was called (the number of ticks that the task has blocked). Delay (block) task một khoảng thời gian kể từ lần cuối hàm được gọi (số ticks mà task đó bị block).

Parameters: Tham số:

* + xTicksToDelay: Number of ticks that task is blocked. Số tick mà task bị block.

Return value: None. Trị trả về: Không có.

## Design firmware architecture on ESP32 device

The firmware architecture on the ESP32 device is designed into separate tasks, each of which allows to run independently of each other. The tasks that are designed include: Kiến trúc firmware trên thiết bị ESP32 được thiết kế thành các tác vụ riêng biệt, mỗi tác vụ cho phép chạy độc lập với nhau. Các task vụ được thiết kế bao gồm:

**Task 1.** Perform event handling with MQTT. After starting, the ESP32 device will connect to the internet via wifi via an access point (wifi access point) provided. If the wifi connection is successful, the device will start the MQTT service to connect to the specified MQTT Broker for communication with the server. Task 1 is a main task that will handle the communication events with MQTT broker. Thực hiện xử lý sự kiện với MQTT. Sau khi khởi động, thiết bị ESP32 sẽ thực hiện kết nối với internet qua wifi thông qua một điểm truy cập (wifi access point) được cung cấp. Nếu kết nối wifi thành công, thiết bị sẽ khởi động dịch vụ MQTT để kết nối tới MQTT Broker được chỉ định phục vụ cho việc giao tiếp với server. Task 1 là một tác vụ chính sẽ thực hiện xử lý các sự kiện giao tiếp với MQTT broker.

MQTT broker communication events include: Các sự kiện giao tiếp với MQTT broker gồm:

MQTT\_EVENT\_CONNECTED: The event occurs when the device is successfully connected to the MQTT broker. If the device is authenticated, it will proceed to the next step of creating the task of sending data to the server via broker and receiving the control command from the server via the broker. Sự kiện xảy ra khi thiết bị được kết nối thành công tới MQTT broker. Nếu thiết bị đã được xác thực, nó sẽ được tiến hành bước tiếp theo là tạo tác vụ gửi dữ liệu lên server qua broker và nhận lệnh điều khiển từ server qua broker.

MQTT\_EVENT\_DISCONNECTED: The event occurs when a connection is lost with MQTT broker, the device needs to re-connect. Sự kiện xảy ra khi bị mất kết nối với MQTT broker, thiết bị cần thực hiện lại việc kết nối.

MQTT\_EVENT\_SUBSCRIBED

MQTT\_EVENT\_UNSUBSCRIBED

MQTT\_EVENT\_PUBLISHED

MQTT\_EVENT\_DATA: The event occurs when receiving data from the server sent to the device via the broker. ESP32 device data listened to receive an authentication request and control commands sent from the server. Sự kiện xảy ra khi nhận được dữ liệu từ server gửi tới thiết bị thông qua broker. Dữ liệu thiết bị ESP32 lắng nghe để nhận là trả lời yêu cầu xác thực và các lệnh điều khiển gửi từ server.

MQTT\_EVENT\_ERROR

**Task 2:** The task of periodically reading data from sensors and saving to a buffer data sent by task 4. This task is independent of MQTT communication. The frequency of reading data is set via the parameter sent from the server in the packet to answer the device authentication successfully. Tác vụ thực hiện việc định kỳ đọc dữ liệu từ các cảm biến và lưu vào một bộ đệm dữ liệu chời gửi đi bởi task 4. Tác vụ này độc lập với việc giao tiếp MQTT. Tần suất đọc dữ liệu được thiết lập thông qua tham số gửi từ server trong gói tin trả lời xác thực thiết bị thành công.

**Task 3**: The task to perform the actuators for the tree care process. This task is also performed independently, periodically checking the control command in the command\_queue storage structure, if any, then executing the required command. Tác vụ để thực hiện điều khiển các cơ cấu (actuators) cho quá trình chăm sóc cây. Tác vụ này cũng được thực hiện độc lập, định kỳ kiểm tra lệnh điều khiển trong cấu trúc lưu trữ command\_queue, nếu có thì thực hiện lệnh được yêu cầu.

**Task 4**: The task of sending collected data from the sensors stored in the buffer to the server via MQTT broker. This task is initialized when the device is successfully authenticated. The frequency of sending data packets is determined through parameters sent from the server via successful authentication response packets. Tác vụ gửi dữ liệu thu thập từ các cảm biến đã được lưu trong bộ đệm lên server qua MQTT broker. Tác vụ này được khởi tạo khi thiết bị được xác thực thành công. Tần suất gửi gói tin dữ liệu được xác định thông qua tham số gửi từ server qua gói tin trả lời xác thực thành công.

**Task 5**: The task of sending the packet informs its active status on the server via MQTT broker. This task is also initialized after the device is successfully authenticated and periodically sends the packet at a frequency specified by the setting parameter. Tác vụ gửi gói tin thông báo trạng thái đang hoạt động của nó lên server thông qua MQTT broker. Tác vụ này cũng được khởi tạo sau khi thiết bị được xác thực thành công và định kỳ gửi gói tin với tần suất xác định bởi tham số thiết lập.

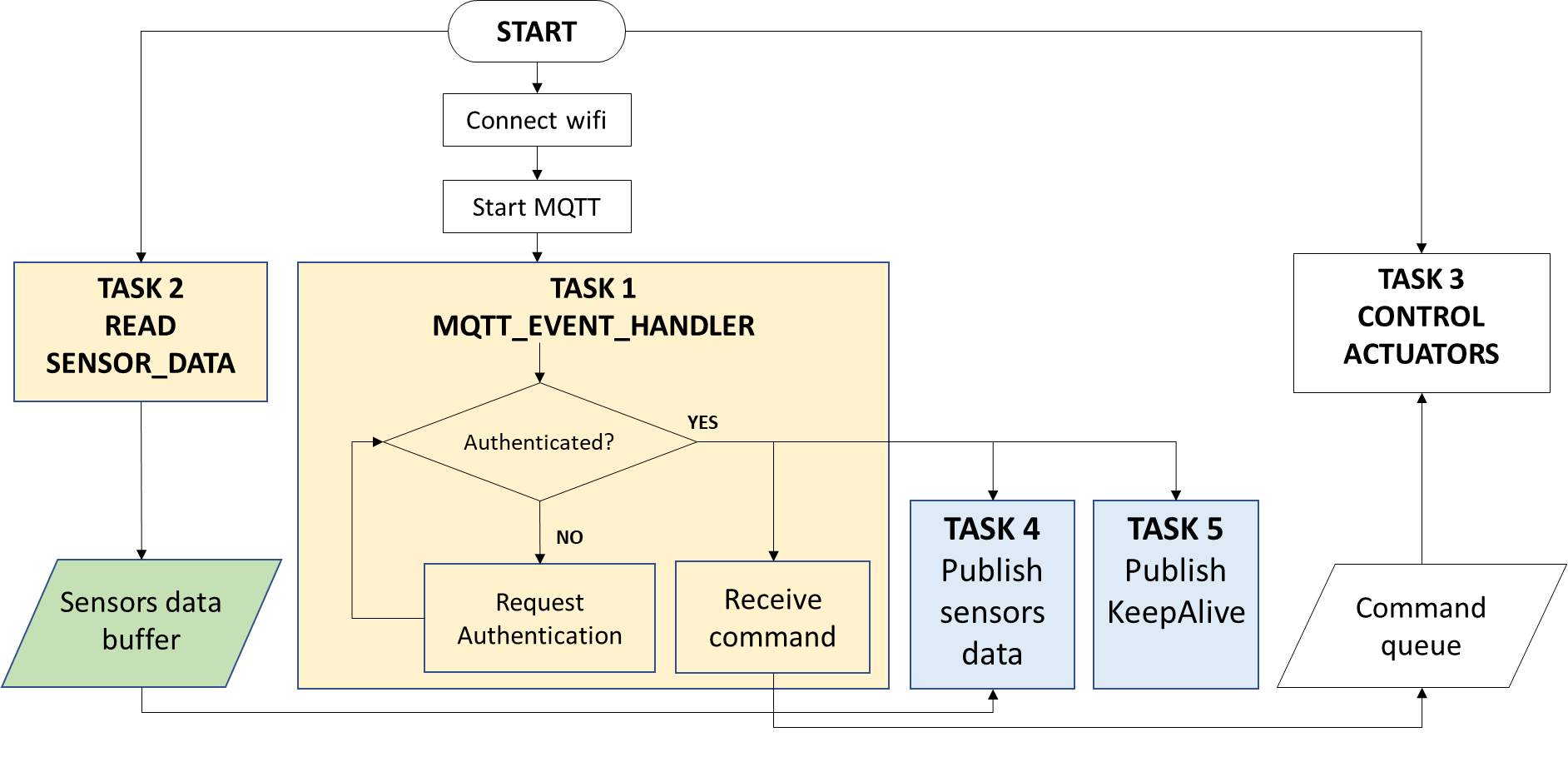


Figure 4‑14. Firmware construction architecture architecture for ESP32 devices

Operation of ESP32: Quá trình hoạt động của ESP32:

After booting, ESP will: Sau khi khởi động, ESP sẽ lần lượt:

* + Connect to WiFi. Kết nối WiFi.
  + Initialize task for reading data from sensors. Khởi tạo task đọc dữ liệu từ các cảm biến.
  + Initialize the command manager to control peripheral devices. Khởi tạo task quản lý command điều khiển thiết bị ngoại vi.
  + Connection to MQTT Broker. Kết nối tới MQTT Broker.

After the broker connection succeeds, ESP turns publishing and subscribes to the topic "nct\_authentication" and "nct\_authentication\_result\_% id" to send and receive packets for authentication. Sau khi kết nối broker thành công, ESP lần lượt publish và subscribe các topic “nct\_authentication” và “nct\_authentication\_result\_%id” để gửi nhận gói tin phục vụ cho quá trình xác thực.

After the authentication process is successful, ESP will perform the following tasks: Sau khi quá trình xác thực thành công, ESP sẽ thực hiện các công việc sau:

* + Initialize keep\_alive packet sending tasks periodically to notify the server that the device is still active. Khởi tạo các tác vụ gửi gói tin keep\_alive định kỳ để thông báo ho server rằng thiết bị vẫn hoạt động.
  + Subscribe topic “nct\_control\_%id” để nhận các gói tin điều khiển từ server. Subscribe topic "nct\_control\_% id" to receive control packets from the server.
  + Initialization task periodically sends sensor data to the server. Khởi tạo tác vụ định kỳ gửi dữ liệu sensors cho server.

When receiving control packets from the sending server, ESP will process and save control commands to "command\_queue", the command management task will periodically take command from command\_queue to control the peripheral devices: pump, fanlight,... Khi nhận được các gói tin điều khiển từ server gửi, ESP sẽ xử lý và lưu lệnh điều khiển vào “command\_queue”, task quản lý command sẽ đinh kỳ lấy command từ command\_queue để thực hiện điều khiển các thiết bị ngoại vi: bơm, đèn, quạt,...

## Sensor data collection task

The sensor's data acquisition device is openly designed (both in hardware and software) to allow flexibility in connection with different sensors, easily changing the type and number of sensors. To do this, the hardware needs to be designed with storage ports that are compatible with different types of sensors of different sensors. The software on the device is designed with a task dedicated to reading data from sensors. Provide different functions to work with each type of sensor. This section describes reading data from several types of sensors used in the test step during testbed construction in the laboratory. The selected sensors include: DHT22 to collect temperature, humidity, pH DFRobot, EC/TDS DFRobot 0300 sensor. Thiết bị thu thập dữ liệu của cảm biến được thiết kế mở (về cả phần cứng và phần mềm) để cho phép linh hoạt trong kết nối với các loại cảm biến khác nhau, dễ dàng thay đổi loại cũng như số lượng các cảm biến. Muốn vậy, phần cứng cần được thiết kế dữ trữ với các cổng kết nối tương thích với các loại chuẩn đầu đo của nhiều loại cảm biến khác nhau. Phần mềm trên thiết bị được thiết kế với một tác vụ dành riêng cho việc đọc dữ liệu từ các cảm biến. Cung cấp các hàm khác nhau làm việc với mỗi loại cảm biến. Phần này mô tả việc đọc dữ liệu từ một số loại cảm biến được sử dụng ở bước thử nghiệm trong quá trình xây dựng testbed tại phòng thí nghiệm. Các cảm biến được lựa chọn gồm: DHT22 để thu thập nhiệt độ, độ ẩm, pH DFRobot, cảm biến EC/TDS DFRobot 0300.

The sensor data collection task is initialized as soon as ESP has finished booting, periodically collecting data from the sensors above. Tác vụ thu thập dữ liệu từ cảm biến được khởi tạo ngay khi ESP khởi động xong, có nhiệm vụ định kỳ thu thập dữ liệu từ các cảm biến nói trên.

The sensor data reading cycle is a parameter that can be configured to change as desired, currently select sensor readings every 5 seconds. All sensors are read at the same time and write results to the cache, data variables - the following data will overwrite the previous data. When ESP sends data to the server, it reads data from these cache. Chu kỳ đọc dữ liệu cảm biến là một tham số có thể cấu hình thay đổi theo mong muốn, hiện tại lựa chọn đọc giá trị cảm biến sau mỗi chu kỳ 5 giây. Tất cả các cảm biến đều được đọc cùng lúc và ghi kết quả vào bộ nhớ đệm, các biến dữ liệu – dữ liệu sau sẽ ghi đè lên dữ liệu trước. Khi ESP gửi dữ liệu lên server sẽ đọc dữ liệu từ các bộ nhớ đệm này.

### Collect temperature and humidity parameters

Sensor: DHT22.

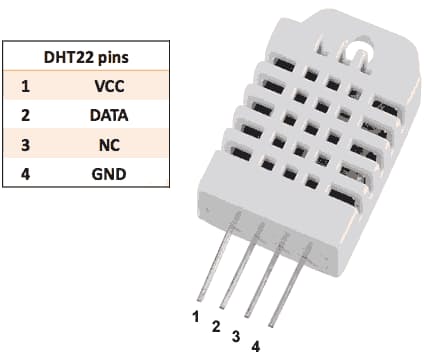


Figure 4‑15. Temperature and humidity sensors used in testbed testing

DHT22 temperature and humidity sensor with high stability, capable of continuous operation for a long time, while the temperature range, humidity can be measured quite wide. Cảm biến số nhiệt độ và độ ẩm DHT22 với độ ổn định cao, có khả năng hoạt động liên tục trong thời gian dài, đồng thời dải nhiệt độ, độ ẩm có thể đo được khá rộng.

**Specification: Thông số kỹ thuật:**

Operating voltage: 5VDC. Điện áp hoạt động: 5VDC.

Operating humidity range: 0% - 100%RH, tolerance ±2%RH. Dải độ ẩm hoạt động: 0% - 100% RH, sai số ±2%RH.

Operating temperature range: -40°C ~ 80°C, error of ±0.5°C. Dải nhiệt độ hoạt động: -40°C ~ 80°C, sai số ±0.5°C.

No need for extra components. Không cần thêm linh kiện ngoài.

Works with long-term stability. Hoạt động với độ ổn định lâu dài.

**Communication: Cách giao tiếp:**

Connection with ESP32: Kết nối với ESP32:

|  |  |
| --- | --- |
| * **ESP32** | * **Sensor DHT22** |
| * 5V (sử dụng nguồn của Arduino) | * Vcc |
| * GND | * GND |
| * GPIO4 | * OUT |

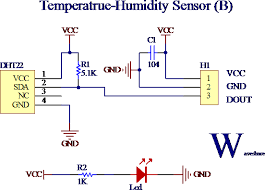


Figure 4‑16. Principle diagram of connection with DHT22 temperature and humidity sensor module

Data reading: Reading DHT22 uses two library functions, DHT22.c (in esp-idf/components/esp32) and DHT22.h (in esp-idf/components/esp32/include). The function to read the humidity is getHumidity() and the function to read the air temperature is getTemperature(). Cách đọc dữ liệu: Cách đọc DHT22 sử dụng hai hàm thư viện là DHT22.c (trong esp- idf/components/esp32) và DHT22.h (trong esp-idf/components/esp32/include). Hàm để đọc độ ẩm là getHumidity() và hàm để đọc nhiệt độ không khí là getTemperature().

Data: RH data (humidity) = 16 Bits; T data (temperature) = 16 Bits; checksum = 8 Bits (checksum is used to check the data after it has finished reading). Về dữ liệu: RH data (Độ ẩm) = 16 Bits; T data (Nhiệt độ) = 16 Bits; check-sum = 8 Bits (checksum được sử dụng để kiểm tra dữ liệu sau khi đã hoàn thành việc đọc).

Suppose MCU receives 40 Bits of data from DHT22 (for example: 0000 0010 1000 1100 0000 0001 0101 1111 1110 1110). Giả sử MCU nhận được 40 Bits dữ liệu từ DHT22 (ví dụ là: 0000 0010 1000 1100 0000 0001 0101 1111 1110 1110).

* + Step 1: Convert 16 Bits of data to read the moisture from binary to decimal, 0000 0010 1000 1100 → 652. We calculate the parameter RH = 652/10 = 65.2%RH. Bước 1: Chuyển đổi 16 Bits data đọc độ ẩm từ hệ nhị phân sang hệ thập phân, 0000 0010 1000 1100 → 652. Ta tính được thông số RH=652/10=65.2%RH.
  + Step 2: Convert 16 Bits of data to read the temperature from binary to decimal, 0000 0001 0101 1111 → 351. We calculate the parameter T = 351/10 = 35.1°C. When the highest bit is 1, the temperature will be below 0 degrees Celsius. For example: 1000 0000 0110 0101, T = negative 10.1°C: 16 bits T data. Bước 2: Chuyển đổi 16 Bits data đọc nhiệt độ từ hệ nhị phân sang hệ thập phân, 0000 0001 0101 1111 → 351. Ta tính được thông số T=351/10=35.1°C. Khi bit cao nhất là 1, nhiệt độ sẽ dưới 0 độ Celsius. Ví dụ: 1000 0000 0110 0101, T= âm 10.1°C: 16 bits T data.
  + Step 3: Check sum is calculated as the sum of 8 bits of each RH data and T data part (eg 0000 0010 + 1000 1100 + 0000 0001 + 0101 1111 = 1110 1110). Bước 3: Check sum được tính bằng tổng của 8 bits của mỗi phần RH data và T data (Ví dụ: 0000 0010+1000 1100+0000 0001+0101 1111=1110 1110).
* About signal and clock pulse Timings: The cycle of each data processing must be greater than 2 seconds. Về tín hiệu và xung đồng hồ Timings: Chu kì mỗi lần xử lí dữ liệu phải lớn hơn 2 giây.
  + Step 1: Send low pulse with a time greater than 1~10 ms. Bước 1: Gửi xung thấp với khoảng thời gian lớn hơn 1~10 ms.
  + Step 2: Send high pulses with a time period greater than 20~40 us. Bước 2: Gửi xung cao với khoảng thời gian lớn hơn 20~40 us.
  + Step 3: When DHT22 detects a signal starting from ESP32, it will pull the bus down to about 80us to send the notification signal to return, then pull the bus up to about 80us to prepare data sent about. Bước 3: Khi DHT22 phát hiện được tín hiệu bắt đầu từ ESP32, nó sẽ kéo thấp bus xuống một khoảng 80us để gửi tín hiệu thông báo sẽ trả về, sau đó sẽ kéo cao bus lên một khoảng 80us để chuẩn bị gửi dữ liệu về.
  + Step 4: When DHT22 sends data to ESP32, each sent bit starts with a 50us low signal, the length of the high level signal then determines the "1" or "0" bit. Bước 4: Khi DHT22 gửi dữ liệu về ESP32, mỗi bit được gửi đi bắt đầu với tín hiệu mức thấp kéo dài 50us, độ dài của tín hiệu mức cao sau đó sẽ quyết định là bit “1” hay “0”.
    - 0: 26~28 us
    - 1: 70 us

### Collect pH parameter

Sensor: Analog pH Sensor DFRobot.

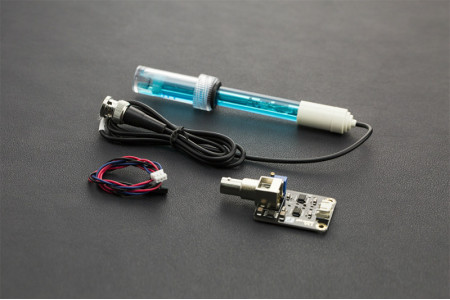


Figure 4‑17. Sensors for measuring pH levels in solutions

RFRobot's pH meter is designed to ensure simplicity, convenience and ease of connection. On the signal processing circuit of the built-in sensor LED acts as a power indicator, a BNC connector to connect to the pH sensor and a PH2.0 port to connect to the MCU via the signal pins Analog. Sensors are used in water quality testing, control and detection of foreign components, used in aquaculture and hydroponics. Cảm biến đo độ pH của RFRobot được thiết kế để đảm bảo sự đơn giản, tiện dụng và dễ dàng kết nối. Trên mạch xử lý tín hiệu của cảm biến được tích hợp đèn LED hoạt động như một đèn báo nguồn, một cổng kết nối BNC để kết nối với cảm biến pH và một cổng PH2.0 để kết nối với MCU thông qua chân tín hiệu Analog. Cảm biến được ứng dụng trong việc kiểm tra chất lượng nước, kiểm soát, phát hiện thành phần lạ trong nước, dùng trong nuôi trồng thuỷ sản, thủy canh.

**Specification: Thông số kỹ thuật:**

Power supply: 5.00V (The closer the power level is 5.00V, the higher the accuracy). Nguồn cung cấp: 5.00V (Nguồn càng gần mức 5.00V, độ chính xác càng tăng).

Size: 43mm×32mm. Kích thước: 43mm×32mm.

Measuring range: pH range from 0-14. Phạm vi đo: độ pH trong khoảng từ 0 – 14.

Measuring temperature: 0-60℃. Nhiệt độ đo: 0-60℃.

Accuracy: ± 0.1pH (25 ℃ - ideal temperature). Độ chính xác: ±0.1pH (25℃ - nhiệt độ lý tưởng).

Response time: min 1min. Thời gian phản hồi: ≤ 1min.

PH sensor with BNC port connection. Cảm biến pH với kết nối cổng BNC.

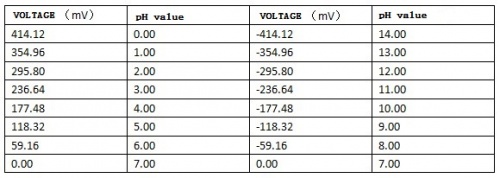
Connection port PH2.0 (3 pins). Cổng kết nối PH2.0 (3 chân).

There are adjustable resistors. Có biến trở điều chỉnh.

Power LED indicator. Đèn LED báo nguồn.

Wire length from pH sensor to BNC connector: 660mm. Độ dài dây từ cảm biến pH đến cổng kết nối BNC: 660mm.

PH electrode characteristics: the pH value corresponding to the voltage is described as the following table: Đặc điểm điện cực pH: thông số giá trị pH tương ứng với điện áp được mô tả như bảng sau:



**Communication: Cách giao tiếp:**

Connection with ESP32: Kết nối với ESP32:

|  |  |
| --- | --- |
| * ESP32 | * Sensor pH |
| * 5V | * Vcc |
| * GND | * GND |
| * GPIO32 (ADC1\_CHANNEL\_4) | * OUT |
|  |  |

Data reading: With the ADC1 port of ESP32, we have two functions to adjust the desired accuracy and the signal amplitude degradation is adc1\_config\_width() and adc1\_config\_channel\_atten(). Specific to the PH sensor we use ADC\_WIDTH\_10Bit (data read width is 10 bits) and ADC\_ATTEN\_DB\_0 (attenuation equal to 0). Cách đọc dữ liệu: Với cổng ADC1 của ESP32, ta có hai hàm để điều chỉnh độ chính xác theo mong muốn và độ suy giảm biên độ tín hiệu là adc1\_config\_width() và adc1\_config\_channel\_atten(). Cụ thể với sensor PH ta sử dụng ADC\_WIDTH\_10Bit (độ rộng đọc dữ liệu là 10 bit) và ADC\_ATTEN\_DB\_0 (độ suy giảm bằng 0).

* + Instead, to read the 5V signal amplitude of the PH sensor, we use a voltage divider circuit: Thay vào đó để đọc biên độ tín hiệu là 5V của sensor PH, ta sử dụng mạch phân áp:

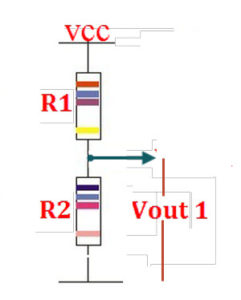


Figure 4‑18. Voltage circuit diagram used to calibrate the output voltage of pH sensor compatible with the voltage range used on ADC of ESP32

* + Specifically, with R1 = 39KOhm and R2 is 11KOhm. So the value of pH after converting ADC will have to apply the formula of the differential bridge circuit: Vout1 = (Vcc.R2)/(R1+R2) with Vout1 is the voltage of the input into ESP32; Vcc is the voltage of the readout data from sensor, R2 is 11KOhm and R1+R2 is 50KOhm (see below for details). Cụ thể với R1 = 39KOhm và R2 là 11KOhm. Vì vậy giá trị của pH sau khi chuyển đổi ADC sẽ phải áp dụng công thức của mạch cầu phân áp: Vout1= (Vcc.R2)/(R1+R2) với Vout1 là điện áp của dữ liệu vào ESP32; Vcc là điện áp của dữ liệu đọc ra từ sensor, R2 là 11KOhm và R1+R2 là 50KOhm (cụ thể xem ở dưới công thức).
  + Then the current measured from the PH sensor will go into the voltage bridge and come out with the signal amplitude will be reduced to 0 to 1.1V in accordance with ESP32. Khi đó dòng điện đo được từ sensor PH sẽ đi vào mạch cầu phân áp và đi ra với biên độ tín hiệu sẽ được giảm xuống là 0 tới 1.1 V phù hợp với ESP32.
  + The ADC signal can then be read by esp-idf's adc1\_get\_raw() function. Sau đó có thể đọc được tín hiệu ADC bằng hàm adc1\_get\_raw() của esp-idf.

How to calculate data: Cách tính toán dữ liệu:

For each measurement: The signal is read 10 times with a delay of 20ms and then averaged (avoiding noise). The measured value is converted to the average Volt value with the formula: Với mỗi lần đo: Tín hiệu được đọc 10 lần với delay là 20ms và sau đó được lấy trung bình (tránh nhiễu). Giá trị đo được chuyển sang giá trị Vôn trung bình với công thức:

float voltageAvgValue = (float)avgValue \* 1.1 / 1023 / 11 \* 50;

* + voltageAvgValue: pH value in V. Giá trị pH theo V.
  + avgValue: Average value from analog pin (0 to 1023 bits). Giá trị trung bình đo được từ chân analog (từ 0 đến 1023 bit).
  + 1.1: Signal amplitude of ESP32. Biên độ tín hiệu của ESP32.
  + 1023: ADC\_WIDTH\_10Bit.

The formula changes the readable pH value from V to the true pH value: Công thức đổi giá trị pH đọc được từ V ra giá trị pH thực sự:

float phValue = 3.5 \* voltageAvgValue + Offset

* + Offset is calibrated when shortening the BNC port before using the sensor. Offset được hiệu chuẩn khi làm ngắn cổng BNC trước khi sử dụng sensor.

### Collect nutrient concentration TDS parameter

Sensors: Analog Electrical Conductivity Sensor DFRobot (DFR0300). This DFRobot sensor is integrated with a conductivity probe with a DS18B20 (water proof) temperature sensor. Cảm biến sử dụng: Analog Electrical Conductivity Sensor DFRobot (DFR0300). Cảm biến Bộ cảm biến DFRobot này được tích hợp một đầu đo độ dẫn điện cùng với một cảm biến nhiệt độ DS18B20 (chống nước).



Figure 4‑19. Sensors measure nutrient concentrations in TDS solution via EC conductivity

Conductivity is the ability to carry electric current. It is the reciprocal of the resistivity. Water conductivity is an important indicator of water quality measurement. It may reflect the level of electrolytes in the water. Depending on the concentration of the electrolyte, the conductivity of the aqueous solution is different. Độ dẫn điện là khả năng chất mang dòng điện. Nó là đối ứng của điện trở suất. Độ dẫn của nước là một chỉ số quan trọng trong việc đo lường chất lượng nước. Nó có thể phản ánh mức độ chất điện giải có trong nước. Tùy thuộc vào nồng độ của chất điện phân, độ dẫn của dung dịch nước là khác nhau.

**Specification: Thông số kỹ thuật:**

Operating voltage: 5V. Điện áp hoạt động: 5V.

PCB size: 45mm x 32 mm. Kích thước PCB: 45mm x 32 mm.

Measuring range: 1 ms/cm - 20 ms/cm. Phạm vi đo: 1 ms/cm – 20 ms/cm.

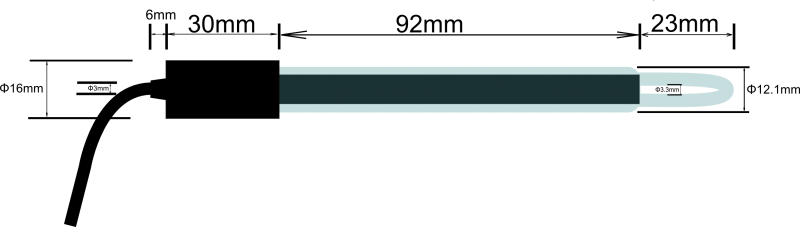
Operating temperature: 5-40oC. Nhiệt độ hoạt động: 5-40oC.

Accuracy: <10% FS (specific accuracy depends on the accuracy of the calibration solution). Độ chính xác: < 10% FS (độ chính xác cụ thể phụ thuộc vào độ chính xác của giải pháp hiệu chuẩn).

Temperature sensor: DS18B20. Cảm biến nhiệt độ: DS18B20.

Conductive electrode (K = 1 electrode constant, BNC connector). Điện cực dẫn điện (hằng số điện cực K=1, đầu nối BNC).

Electrode size: Kích thước điện cực:



**Communication: Cách giao tiếp:**

Connection with ESP32: Kết nối với ESP32:

|  |  |  |
| --- | --- | --- |
| **ESP32** | **Sensor EC Electrode** | **Sensor DS18B20** |
| 5V | Vcc | Vcc |
| GND | GND | GND |
| GPIO33 (ADC1\_CHANNEL\_5) | OUT |  |
| GPIO25 (DAC1) |  | OUT |

How to read data: DS18b20 temperature measuring sensor uses 1 WIRE communication standard (signal path and feed voltage path can be shared on one conductor). In addition, we use the esp-idf RMT module to send and receive infrared control signals (support for more accurate reading/writing timeslots in One Wire communication). We use the available library function esp32-ds18b20 for communication between DS18B20 and ESP32. Cách đọc dữ liệu: Cảm biến đo nhiệt độ DS18b20 sử dụng chuẩn giao tiếp 1 WIRE (đường dẫn tín hiệu và đường dẫn điện áp nguồn nuôi có thể dùng chung trên một dây dẫn). Ngoài ra ta sử dụng module RMT của esp-idf để gửi và nhận các tín hiệu điều khiển hồng ngoại (hỗ trợ cho việc đọc/ viết các timeslots trong giao tiếp One Wire được chính xác hơn). Ta sử dụng hàm thư viện esp32-ds18b20 có sẵn cho giao tiếp giữa DS18B20 và ESP32.

* + The DS18B20 connected to the same bus is distinguished from each other by a unique 64-bit address (64-bit serial number). This 8 bytes (64 bits) is divided into three main parts: Starting with LSB, the first byte is the family of 8-bit family codes (8-bit family codes) that determine the device type. The next 6 bytes store the device's private address. The last byte (MSB) is the cyclic redundancy check data integrity check (CRC) whose value corresponds to the value of the first 7 bytes. Thanks to the CRC byte, it helps the master determine whether or not the read address is corrupted. DS18B20 kết nối với cùng một bus được phân biệt với nhau nhờ 64 bit địa chỉ duy nhất (64-bit serial number). 8 byte (64 bit) này và được chia làm ba phần chính: Bắt đầu với LSB, là byte đầu tiên là mã họ thiết bị có độ lớn 8 bit (8-bit family codes) xác định kiểu thiết bị. 6 byte tiếp theo lưu trữ địa chỉ riêng của thiết bị. Byte cuối cùng (MSB) là byte kiểm tra tính toàn vẹn dữ liệu cyclic redundancy check (CRC) có giá trị tương ứng với giá trị của 7 byte đầu tiên. Nhờ byte CRC giúp cho master xác định có địa chỉ được đọc có bị lỗi hay không.
  + Four basic operations of the bus 1 wire is Reset / Presence, send bit 1, send bit 0, and read bit. Operate bytes such as sending bytes and reading bytes based on bit manipulation. Bốn thao tác hoạt động cơ bản của bus 1 wire là Reset/Presence, gửi bit 1, gửi bit 0, và đọc bit. Thao tác byte như gửi byte và đọc byte dựa trên thao tác từng bit.
  + EC Meter V1.0 conductivity measurement sensor: Similar to the PH measuring sensor, we use the ADC1 port to read the Analog signal of the EC sensor and use the differential bridge circuit to reduce the signal amplitude to 0 to 1.1 V. Cảm biến đo độ dẫn điện EC Meter V1.0: Tương tự với cảm biến đo PH, ta sử dụng cổng ADC1 để đọc tín hiệu Analog của sensor EC và sử dụng mạch cầu phân áp để giảm biên độ tín hiệu xuống 0 tới 1.1 V.

Data calculation: For each measurement: Tính toán dữ liệu: Với mỗi lần đo:

* + Step 1: The signal is read 20 times with the delay of 20ms and then averaged (avoiding noise). The formula used is similar to the formula for measuring pH. Bước 1: Tín hiệu được đọc 20 lần với delay là 20ms và sau đó được lấy trung bình (tránh nhiễu). Công thức được sử dụng tương tự với công thức đo pH.
  + Step 2: We calculate the temperature coefficient from the measured value from DS18B20 compared to the calibration temperature of 25oC: Bước 2: Ta tính hệ số nhiệt độ từ giá trị đo được từ DS18B20 so sánh với nhiệt độ hiệu chuẩn 25oC:

float TempCoefficient=1.0+0.0185\*(water\_temperature-25.0);

* + Step 3: Calculate the voltage coefficient by dividing the average voltage value by the temperature coefficient. Bước 3: Tính hệ số hiệu điện thế bằng phép chia giá trị hiệu điện thế trung bình với hệ số nhiệt độ.

float CoefficientVolatge = (float)averageVoltage/TempCoefficient;

* + Step 4: Compare voltage coefficients with levels and have EC values based on the corresponding formula of levels (ECcurrent is the measured EC value). Bước 4: So sánh hệ số hiệu điện thế với các mức và có giá trị EC dựa trên công thức tương ứng của các mức (ECcurrent là giá trị EC đo được).
    - EC < 1ms/cm: No solution. Không có dung dịch.
    - EC > 20ms/cm: Out of the range. Ngoài phạm vi đo.
    - 1ms/cm<EC<=3ms/cm: ECcurrent=6.84\*CoefficientVolatge-64.32;
    - 3ms/cm<EC<=10ms/cm: ECcurrent=6.98\*CoefficientVolatge-127;
    - 10ms/cm<EC<20ms/cm: ECcurrent=5.3\*CoefficientVolatge+2278;

Calibration of measurement value: Because the circuit constant of each electrode is different, the accuracy is not very high, so it needs calibration. The calibration steps are as follows: Hiệu chuẩn giá trị đo: Vì hằng số mạch của mỗi điện cực là khác nhau nên độ chính xác không cao lắm nên cần hiệu chuẩn. Các bước hiệu chuẩn như sau:

* + Make the solution temperature at 25oC. Làm cho nhiệt độ dung dịch ở mức 25oC.
  + Make wire according to connection diagram. Làm dây theo sơ đồ kết nối.
  + Load sample code. Tải mã mẫu.
  + Insert the probes (conductive electrode and temperature sensor) into the head solution. With this solution, we get 1413 us/cm, the average voltage is V1. We have point A (V1, 1.1413) in the picture. Chèn các đầu đò (điện cực dẫn điện và cảm biến nhiệt độ) vào dung dịch đầu. Với dung dịch này, ta có kết quả 1413 us/cm, điện áp trung bình là V1. Ta có điểm A(V1, 1.1413) trong hình.
  + Remove the probe and clean it with pure water. Lấy đầu dò ra và làm sạch bằng nước tinh khiết.
  + Submerge again in another EC sample solution resulting in point B (V2.12.88). Nhấn chìm một lần nữa vào dung dịch EC mẫu khác thu được kết quả là điểm B(V2,12.88).
  + With two points A and B, we will draw a straight line to describe the relationship between analog reading and EC. Với hai điểm A và B ta sẽ vẽ đường thẳng để mô tả mối quan hệ giữa đọc tương tự và EC.

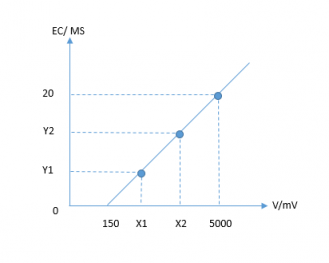


Figure 4‑20. Calibration diagram of EC measurement value

Experimental results to collect data from DHT22, pH Sensor and EC Sensor, where: Kết quả thử nghiệm để thu thập dữ liệu từ DHT22, Cảm biến pH và Cảm biến EC, trong đó

Water Temperature: Water temperature from DS18B20. Nhiệt độ nước từ DS18B20.

pH: pH value from pH Sensor. Giá trị pH từ pH Sensor.

EC: EC value from EC Sensor. Giá trị EC từ EC Sensor.

Air Temp: Air temperature from DHT22. Nhiệt độ không khí từ DHT22.

Humidity: Humidity from DHT22. Độ ẩm từ DHT22.

Evaluation: pH, Water Temperature, Air Temp and Humidity parameters are quite stable and accurate. The collected value from the EC/TDS sensor needs to be calibrated according to the solution temperature. Đánh giá: Các thông số pH, Water Temperature, Air Temp và Humidity khá ổn định và chính xác. Giá trị thu thập từ cảm biến EC/TDS cần được hiệu chỉnh (calibration) theo nhiệt độ dung dịch.

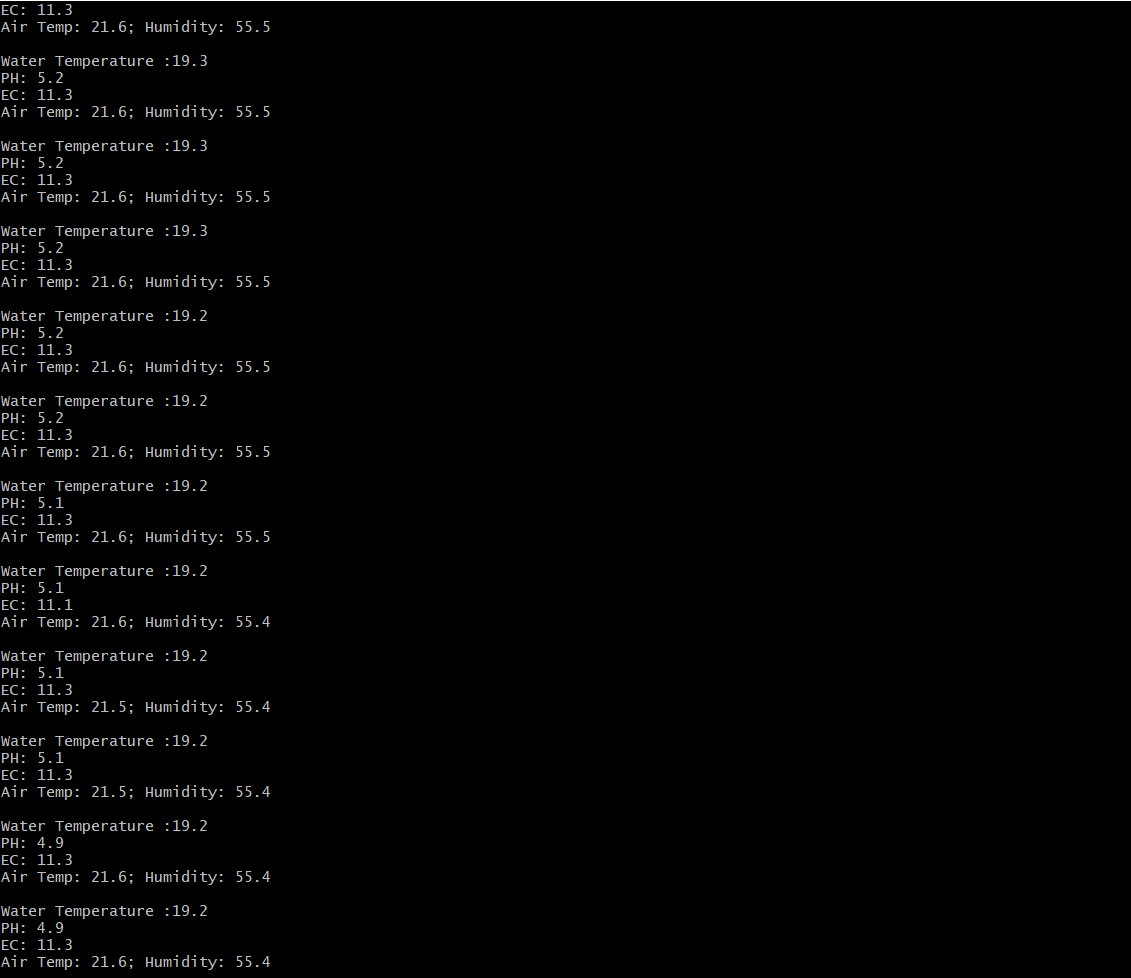
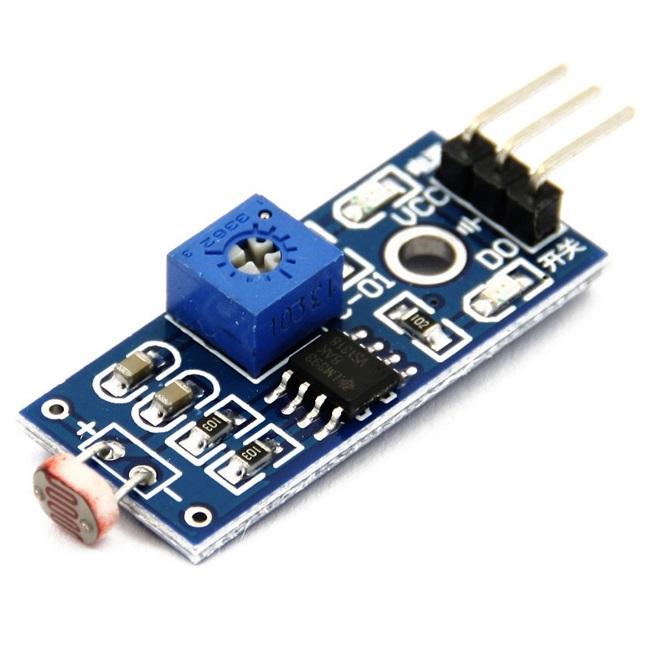


Figure 4‑21. Experimental measurement results from DHT22, pH, and EC sensors

### Collect light intensity parameter

Sensor: Light sensor.



## Communicate with the server via MQTT Broker

### Authentication function for devices participating in the system



Figure 4‑22. Diagram of steps to authenticate devices participating in the system

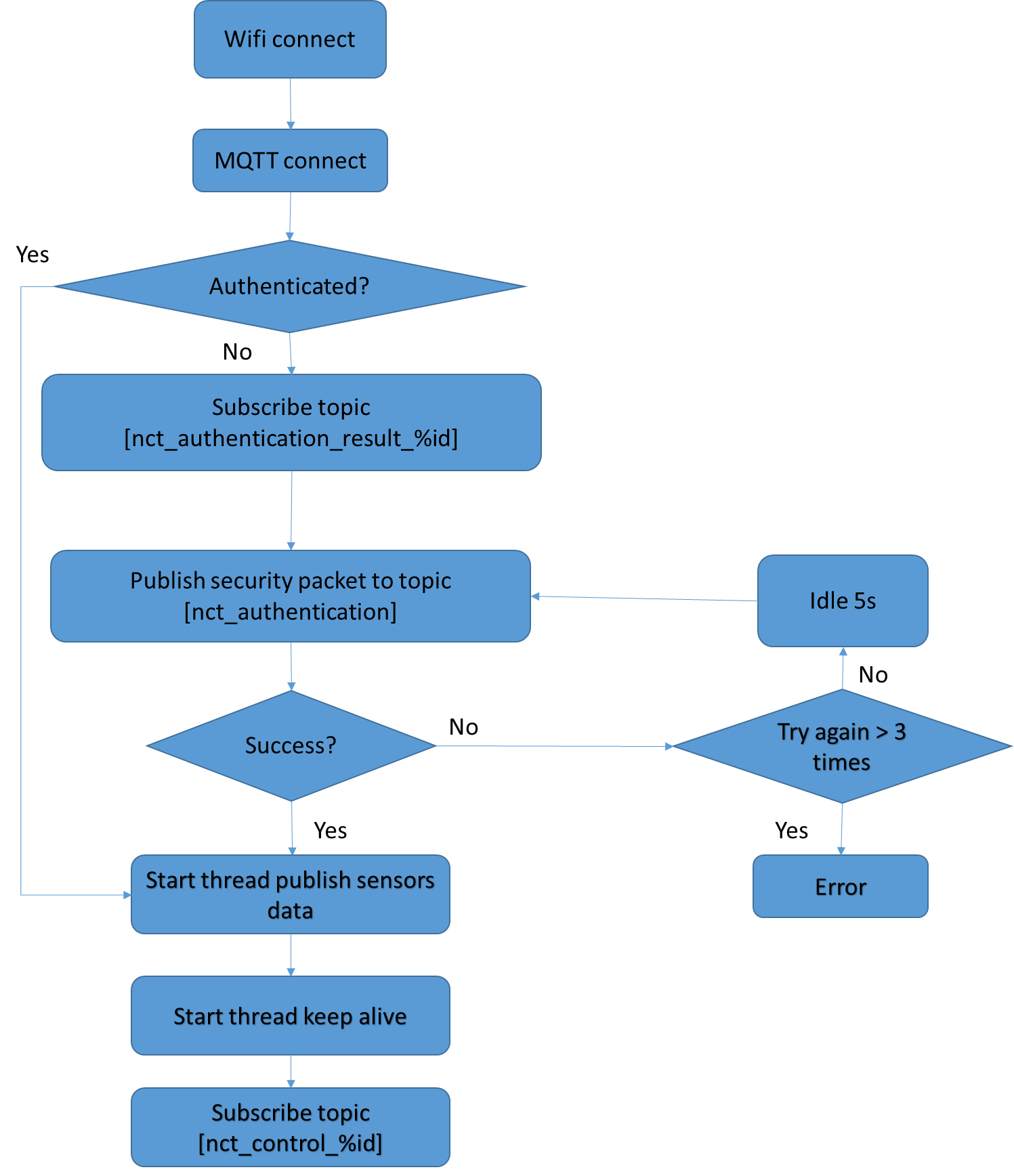


Figure 4‑23. Algorithm flowchart on firmware includes steps for authentication, data sending, receiving control commands and status status messages

After ESP successfully connects to the MQTT Broker, it will subscribe to the topic "nct\_authentication\_result\_%id" and publish a packet to the topic "nct\_authentication", the format is JSON as follows: Sau khi ESP kết nối thành công với MQTT Broker, nó sẽ lần lượt subscribe topic “nct\_authentication\_result\_%id” và publish một gói tin nhận dạng tới topic “nct\_authentication”, gói tin có định dạng JSON như sau:

{

"id" : %s,

"password" : "nct\_laboratory",

"device\_type" : "ESP32"

}

where: trong đó:

The "id" field is the id of the ESP device, which is responsible for identifying the device in the system, with the string type. Trường “id” là id của thiết bị ESP, có nhiệm vụ định danh thiết bị trong hệ thống, có kiểu chuỗi ký tự.

The "password" field is from the code to confirm, there is a string of characters. Trường “password” là từ mã để xác nhận, có kiểu chuỗi ký tự.

The "device\_type" field is a classification for the device, serving future expansion when there are many different types of devices, of type string. Trường “device\_type” là phân loại cho thiết bị, phục vụ việc mở rộng trong tương lại khi có nhiều loại thiết bị khác nhau, có kiểu chuỗi ký tự.

On the server side, also subscribe to the topic "nct\_authentication" and publish a confirmation packet to the topic "nct\_authentication\_result\_%id", the packet is in JSON format as follows: Về phía server, cũng lần lượt subscribe topic “nct\_authentication” và publish một gói tin xác nhận tới topic “nct\_authentication\_result\_%id”, gói tin có định dạng JSON như sau:

{

"status" : "pass",

"time\_period" : number,

"secret\_key" : string[16]

}

where: trong đó:

Whether the "status" field shows the successful authentication status (pass / not pass). Trường “status” thể hiện trạng thái xác thực thành công hay không (pass/not pass).

The "time\_period" field is the server cycle that requires ESP to send sensor data to the server. Trường “time\_period” là chu kỳ server yêu cầu ESP gửi dữ liệu cảm biến cho server.

The "secret\_key" field provided by the server to ESP aims to authenticate ESP's future packets. Trường “secret\_key” do server cung cấp cho ESP nhằm mục đích xác thực các gói tin sau này của ESP gửi đi.

### Function of sending data collected from sensors



Figure 4‑24. Sensor data sending scenario from device to server

ESP will periodically (after the period specified by "time\_period") send the collected data to the server. ESP sẽ đinh kỳ (sau khoảng thời gian được quy định bởi “time\_period”) gửi dữ liệu đã thu thập được cho server.

This sending is intended to be done by a thread running parallel to the other collection and control tasks in ESP. Việc gửi này định thực hiện bởi một luồng chạy song song với các tác vụ điều khiển và thu thập khác trong ESP.

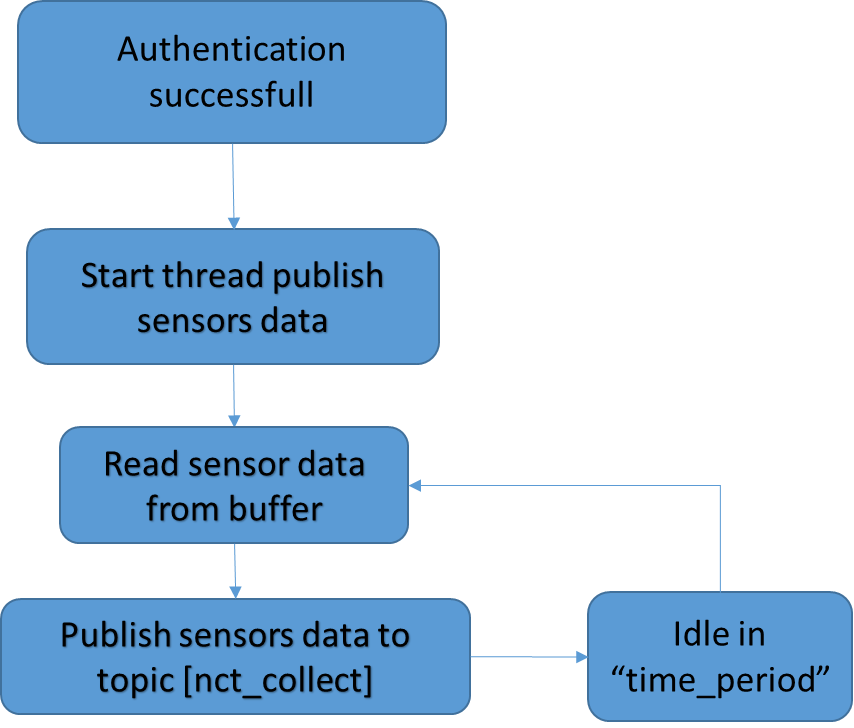


Figure 4‑25. Algorithm flowchart sends sensor data to the server

The data will be packaged in JSON format and published to the topic "nct\_collect", the format is as follows: Các dữ liệu sẽ được đóng gói theo định dạng JSON và publish lên topic “nct\_collect”, gói tin có định dạng như sau:

{

"id" :%s, // id của ESP32

"secret\_key" :"%s", // từ mã để xác thực gói tin

"timestamp" :"%s", // thời gian gửi

"packet\_no" :%lu, // số thứ tự gói tin

"temperature" :%.1f, // dữ liệu nhiệt độ

"humidity" :%.1f, // dữ liệu độ ẩm

"tds" :%.1f, // dữ liệu tds

"pH" :%.1f, // dữ liệu pH

"water\_level" :100 // mực nước còn lại

}

### Function to receive control commands from the server



Figure 4‑26. Diagram of steps to receive control commands from server to device

ESP after authentication, will subscribe to the topic "nct\_control\_%id", ESP will receive commands to control the server sent in this topic. ESP sau khi xác thực, sẽ subscribe topic “nct\_control\_%id”, ESP sẽ nhận các lệnh điều khiển được server gửi lên tại topic này.

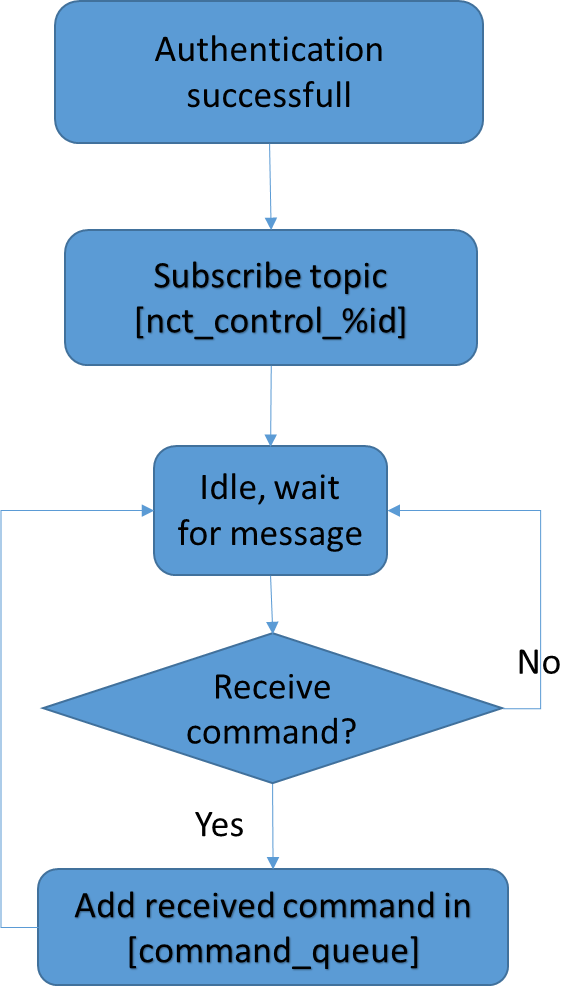


Figure 4‑27. Flowchart algorithm receives control commands from the server sent to the device

JSON-formatted packets will be interpreted and stored at "command\_queue" and then taken in turn to execute. Package format is as follows: Các gói tin có định dạng JSON sẽ được thông dịch và lưu trữ tại “command\_queue” rồi sau đó sẽ được lần lượt lấy ra để thực thi. Định dạng gói tin như sau:

{

"dev\_id" :number,

"command\_no" :number,

"actuator\_id" :number,

"actuator\_name" :string,

"action" :string,

"param" :number

}

How to execute the commands will be more clearly stated in Section 4.5. Cách thức thực thi các lệnh sẽ được nói rõ hơn ở Phần 4.5.

### Function of sending packets to report active status (KeepAlive)



Figure 4‑28. Diagram of steps to send device activity status message to server

ESP after authentication, will periodically send a keep\_alive packet, which tells the server that ESP is still running smoothly. ESP sau khi xác thực, sẽ định kỳ gửi gửi một gói tin keep\_alive, có nhiệm vụ thông báo cho server biết rằng ESP vẫn đang chạy ổn định.

This task is performed by a thread that runs in parallel with other tasks in ESP. Tác vụ này được thực hiện bởi một luồng chạy song song với các tác vụ khác trong ESP.

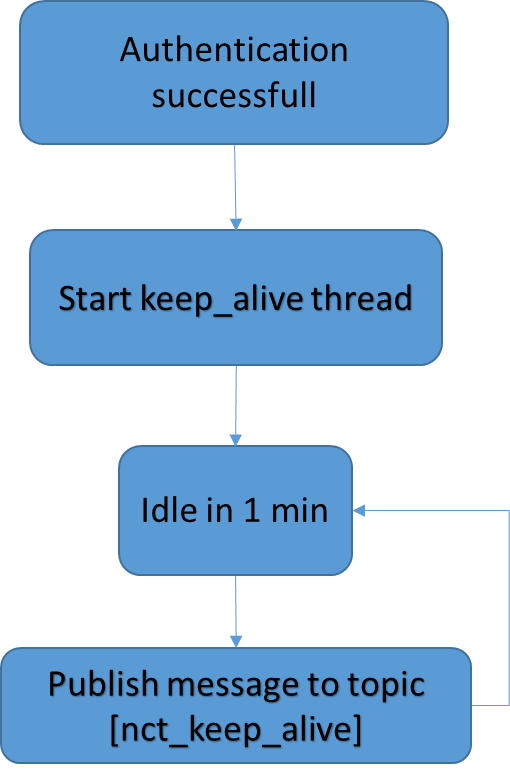


Figure 4‑29. The algorithm flow sends the device's status message to the server

The package is in JSON format as follows: Gói tin có định dạng JSON như sau:

{

"id" :%s,

"secret\_key" :"%s",

"timestamp" :"%s",

"active" :"true"

}

The server will rely on these packets to identify how many devices are still running in the system. Server sẽ dựa vào những gói tin này để nhận biết có bao nhiêu thiết bị vẫn đang chạy trong hệ thống.

## The task of executing control commands

Control commands received from the server sent to the device will be stored in a command queue (command\_queue). The control command execution task will continuously get control commands (if any) from this queue to execute. Implementation is handled in a separate, parallel operation and does not affect other device work. Tác vụ thực thi lệnh điều khiển sẽ liên tục lấy các lệnh điều khiển (nếu có) từ hàng đợi này ra để thực hiện. Việc thực hiện được xử lý trong một tác vụ độc lập, song song và không ảnh hưởng tới các công việc khác của thiết bị.

The structure of "command\_queue" is as follows: Cấu trúc của “command\_queue” như sau:

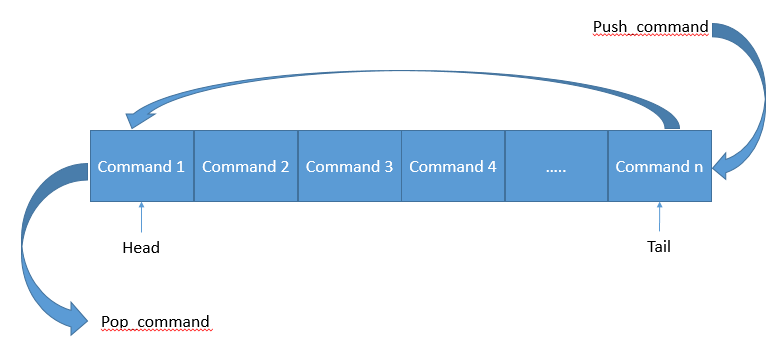


Figure 4‑30. Storage structure and handling control commands in a command\_queue queue

Each time pop\_command - take 1 command in the queue to execute, Head indicator will increase by 1 unit, if Head> = Max\_size\_queue, Head = 0. Mỗi lần pop\_command – lấy 1 command trong hàng đợi ra để thực hiện, chỉ số Head sẽ tăng thêm 1 đơn vị, nếu Head >= Max\_size\_queue, Head = 0.

Each time push\_command - adding a command to the queue, the Tail index will increase by 1 unit, if Tail> = MAX\_COMMAND\_QUEUE, Tail = 0. Mỗi lần push\_command – thêm 1 command vào hàng đợi, chỉ số Tail sẽ tăng thêm 1 đơn vị, nếu Tail >= MAX\_COMMAND\_QUEUE, Tail = 0.

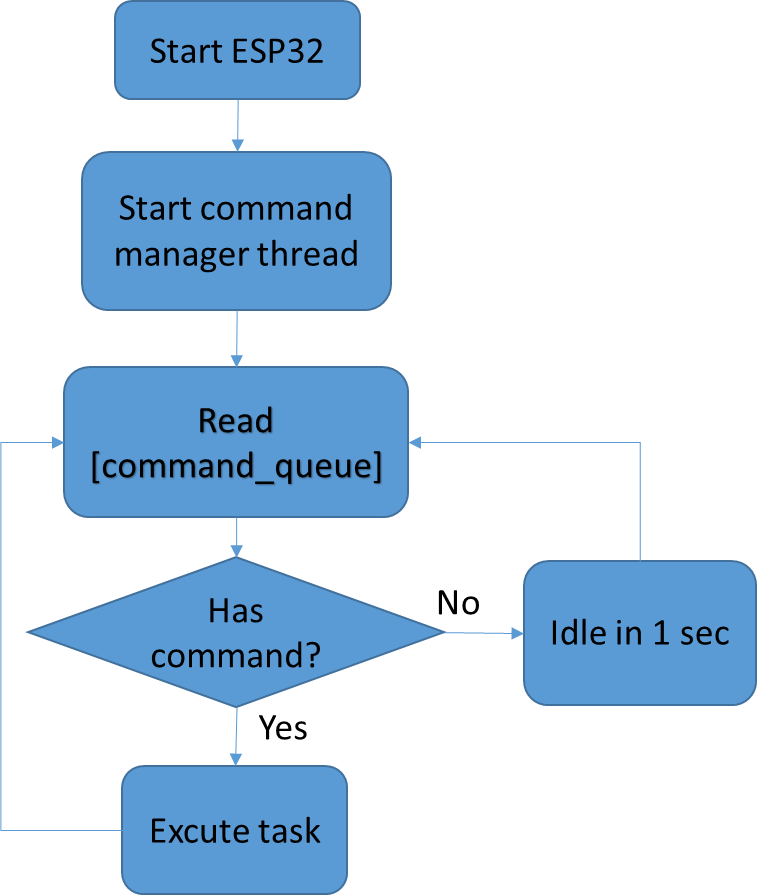


Figure 4‑31. Flowchart algorithm executes control commands

After booting, the ESP will create a parallel stream to perform the management and execution of the command. This stream continuously checks "command\_queue" - the queue used to store control commands received from the server, and executes it if any. Sau khi khởi động, ESP sẽ tạo một luồng song song để thực hiện việc quản lý và thực thi lệnh điều khiển. Luồng này liên tục kiểm tra “command\_queue” – hàng đợi dùng để lưu các lệnh điều khiển nhận được từ server, và thực thi nếu có.

When executing a command, ESP creates a separate thread to run the peripheral through the corresponding GPIOs. This thread will be assigned a handler for flow management purposes. Khi thực thi một lệnh, ESP sẽ tạo ra một luồng riêng khác để chạy thiết bị ngoại vi thông qua các GPIO tương ứng. Luồng này sẽ được gán một handler nhằm mục đích quản lý luồng.

The peripheral device flow will run until the specified condition is reached (on time or until the environmental parameters change to the appropriate level: for example, the fan will run until the humidity drops). will automatically interrupt the peripheral and delete the stream to save memory. Luồng chạy thiết bị ngoại vi sẽ chạy cho đến khi đạt điều kiện quy đinh (về thời gian hoặc cho đến khi các thông số môi trường thay đổi đến mức thích hợp: ví dụ quạt sẽ chạy cho đến khi độ ẩm giảm xuống) thì sẽ tự động ngắt thiết bị ngoại vi và xóa luồng để tiết kiệm bộ nhớ.

If the following command commands the same peripheral device of the preceding command, the following command clears the execution flow of the previous command and creates a new execution flow. Nếu lệnh đến sau điều khiển cùng 1 thiết bị ngoại vi của lệnh đến trước thì lệnh đến sau xóa luồng thực hiện của lệnh đến trước và tạo một luồng thực thi mới.

# Testbed results

The system is implemented experimentally in the laboratory to evaluate the components developed and developed by the whole system. Hệ thống được triển khai thực nghiệm tại phòng thí nghiệm nhằm đánh giá các thành phần được xây dựng phát triển của cả hệ thống.

In this section we report the progress results at the present time of the experimental running system. The results are logged on the server software. Trong phần này chúng tôi báo cáo lại kết quả tiến độ ở thời điểm hiện tại của hệ thống đang chạy thực nghiệm. Các kết quả được ghi log trên phần mềm server.

## Build testbed test

Testbed was built to test including a aqua-culture plant with lettuce as shown. Equipment for collecting and controlling fabricated test circuits coupled with temperature, humidity, pH, nutrient concentrations in tds and coupled with relay control devices It allows to control and turn off the pumping device refluxing nutrient solution for growing plants from containers to trusses, injecting and supplementing nutrition A, B, turning on lights and fans. Testbed được xây dựng thử nghiệm gồm một giàn thủy canh trồng rau xà lách như hình minh họa. Thiết bị thu thập và điều khiển được chế tạo mạch thử nghiệm ghép nối với các cảm biến nhiệt độ, độ ẩm, nồng độ pH, nồng độ dinh dưỡng trong dung dịch tds và ghép nối với các cơ cấu điều khiển đóng cắt bằng rơ le cho phép điều khiển tắt bật các thiết bị bơm hồi lưu dung dịch dinh dưỡng nuôi trồng cây từ thùng chứa lên giàn, bơm pha bổ sung dinh dưỡng A, B, tất bật đèn chiếu sáng, quạt.



Figure 5‑1. Images of lettuce cultivars used as testbed

## Test results of server software with communication services with devices

When starting the server software, the device communicating with the device via the MQTT broker will be started, including: Khi khởi động phần mềm server, các dịch vụ giao tiếp với thiết bị thông qua MQTT broker sẽ được khởi động, gồm có:

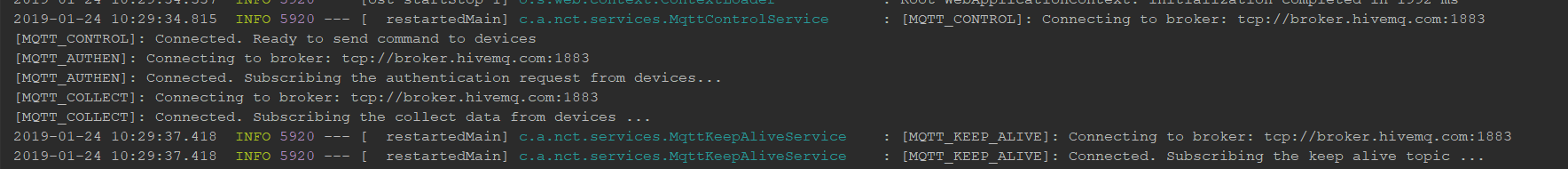
Service for authentication of equipment (MQTT\_AUTHEN). Dịch vụ phục vụ xác thực các thiết bị (MQTT\_AUTHEN).

Service to receive sensor data from sender device (MQTT\_COLLECT). Dịch vụ nhận dữ liệu cảm biến từ thiết bị gửi lên (MQTT\_COLLECT).

Service sending down control device (MQTT\_CONTROL). Dịch vụ gửi điều khiển xuống thiết bị (MQTT\_CONTROL).

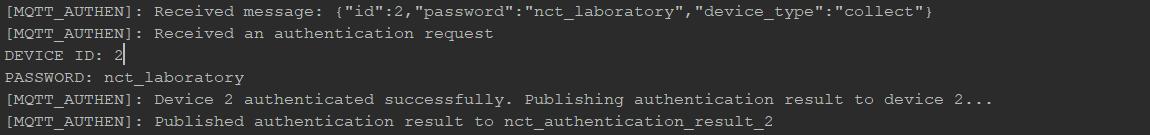
Service to receive notification of operating status from devices (MQTT\_KEEPALIVE). Dịch vụ nhận thông báo trạng thái hoạt động từ các thiết bị (MQTT\_KEEPALIVE).

Each of these services runs on an independent flow that allows the server to simultaneously serve multiple devices with different services. The startup results of the services are illustrated in the server software log as follows: Mỗi dịch vụ này chạy trên một luồng độc lập cho phép server có thể phục vụ đồng thời nhiều thiết bị với các dịch vụ khác nhau. Kết quả khởi động các dịch vụ được minh họa trong ghi log của phần mềm server như sau:



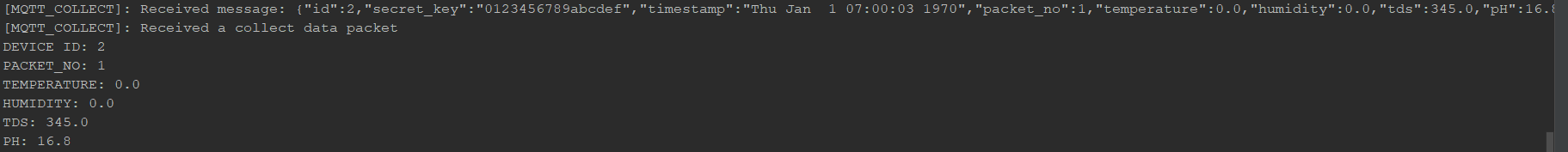
### Results of deploying device authentication service

When a device is started to join the system, it first sends an authentication request to the server. On the server that receives the request to authenticate from the device sent along with information about device identification and authentication code (ID, Password), the server will check if this information is already in the database (can register before using the device). If authentication is successful, the server will return the authentication result to the corresponding device. This process is logged as follows: Khi có một thiết bị được khởi động để tham gia vào hệ thống, đầu tiên nó sẽ gửi yêu cầu xác thực đến server. Trên server nhận được yêu cầu xác thực từ thiết bị gửi lên kèm theo thông tin về định danh thiết bị và mã xác thực (ID, Password), server sẽ kiểm tra thông tin này đã có trong cơ sở dữ liệu hay chưa (được đăng ký trước khi sử dụng thiết bị). Nếu xác thực thành công, server sẽ trả về kết quả xác thực tới thiết bị yêu cầu tương ứng. Quá trình này được ghi log như sau:



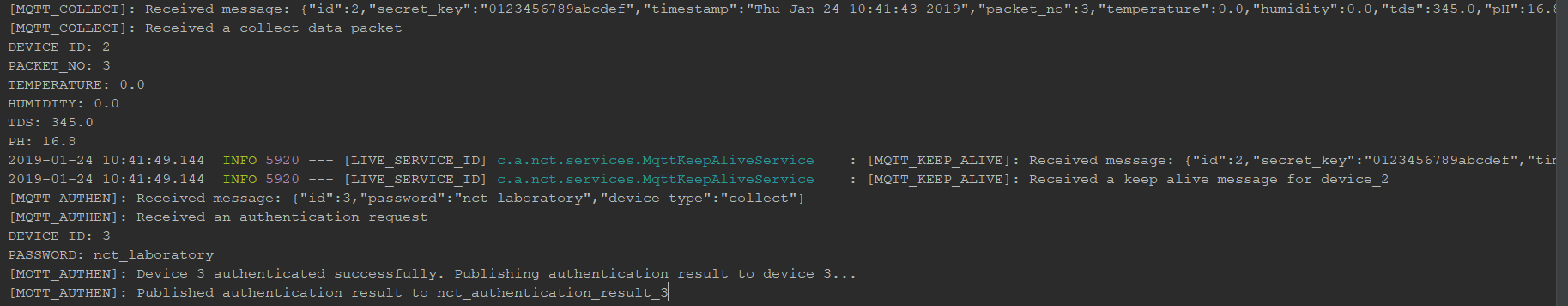
### Results of deploying sensor data collection service

Server software allows to receive data from multiple devices to be sent. Devices that have been successfully authenticated will be allowed to send data to the server. Packaged data packets contain device identifiers and security codes that have been issued during authentication, helping the server separate the device's packets. The server software is also capable of extracting packets packaged in json format and saving the data as collected sensor values into the corresponding database. The result of this process is illustrated in the log on the server as follows: Phần mềm server cho phép nhận dữ liệu từ nhiều thiết bị thu thập gửi tới. Các thiết bị đã được xác thực thành công sẽ được tham gia gửi dữ liệu đến server. Các gói tin dữ liệu được đóng gói chứa định danh thiết bị và mã bảo mật đã được cấp trong quá trình xác thực, giúp server phân tách được các gói tin của thiết bị nào. Phần mềm server cũng có khả năng bóc tách gói tin được đóng gói theo định dạng json và lưu các dữ liệu là các giá trị cảm biến thu thập vào cơ sở dữ liệu tương ứng. Kết quả quá trình này được minh họa trong ghi log trên server như sau:



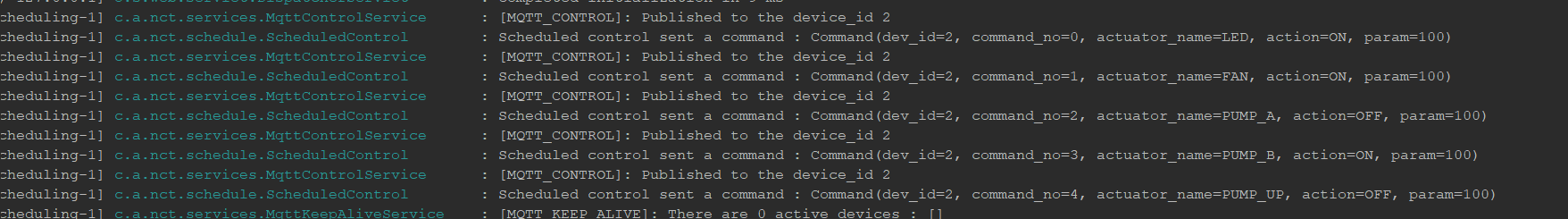
The collected data will be written to the database. Dữ liệu được thu thập sẽ được ghi vào cơ sở dữ liệu.

Authentication and data collection services run independently on two parallel streams, so the server can both collect data of incoming devices while continuing to process authentication requests from the device. other if any. The illustrated results of these two types of processing log as follows: Các dịch vụ xác thực và thu thập dữ liệu chạy độc lập trên 2 luồng song song, vì vậy server có thể vừa thu thập dữ liệu của các thiết bị gửi đến trong khi vẫn tiếp tục xử lý các yêu cầu xác thực từ thiết bị khác nếu có. Kết quả minh họa của 2 loại xử lý này ghi log như sau:



### Service deployment results send control commands to the device

To assess the ability to send control commands to devices, the test server software will periodically send packets containing control commands to the specified device. Illustrative results are recorded as follows: Để đánh giá khả năng gửi lệnh điều khiển tới các thiết bị, phần mềm server thử nghiệm sẽ tiến hành định kỳ gửi các gói tin chứa các lệnh điều khiển tới thiết bị được chỉ định. Kết quả minh họa được ghi log như sau:



The command is sent to the specified device, the device successfully receives and executes the control instructions to the corresponding device. The figure below illustrates that the test circuit device receives a command to turn on the nutrient solution pump and execute this command. Lệnh điều được gửi tới thiết bị chỉ định, thiết bị nhận thành công và thực thi các lệnh điều khiển tới các cơ cấu tương ứng. Hình dưới minh họa thiết bị mạch thử nghiệm nhận được lệnh điều khiển bật bơm dung dịch dinh dưỡng và thực thi lệnh này.

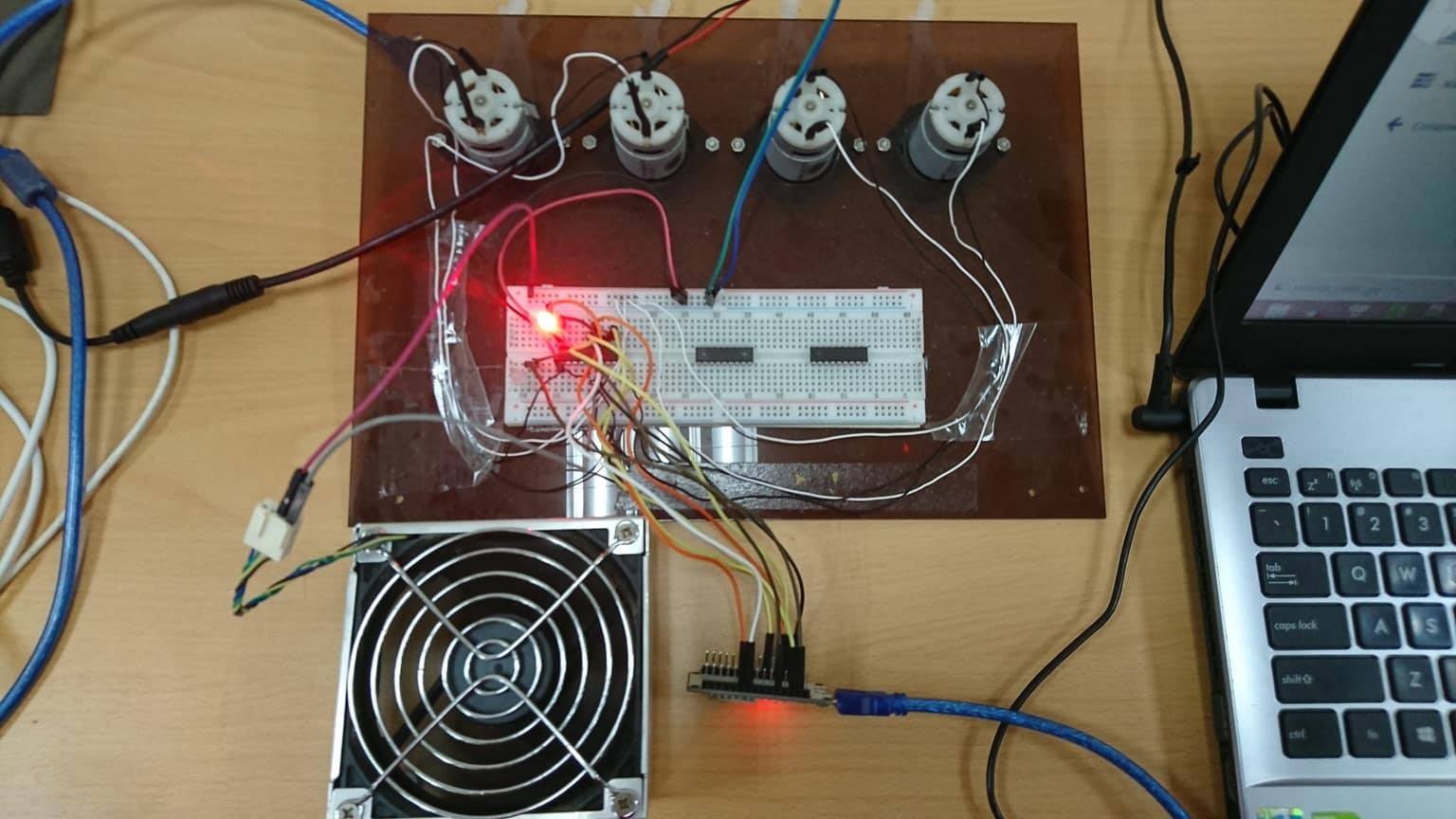


Figure 5‑2. The image of the test device receives the control command from the server and executes the control command to turn on the nutrient return pump

1. <https://www.mysql.com/> [↑](#footnote-ref-1)
2. <https://www.mysql.com/> [↑](#footnote-ref-2)
3. <https://spring.io/> [↑](#footnote-ref-3)
4. <https://jquery.com/> [↑](#footnote-ref-4)
5. <https://getbootstrap.com/> [↑](#footnote-ref-5)
6. <https://nodejs.org/en/about/> [↑](#footnote-ref-6)
7. <https://nodejs.org/en/about/> [↑](#footnote-ref-7)
8. <https://www.npmjs.com/package/mysql> [↑](#footnote-ref-8)
9. <https://expressjs.com/> [↑](#footnote-ref-9)
10. <https://pugjs.org/api/getting-started.html> [↑](#footnote-ref-10)
11. <https://www.npmjs.com/package/morgan> [↑](#footnote-ref-11)