ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ VIỄN THÔNG
BỘ MÔN MÁY TÍNH – HỆ THỐNG NHÚNG

NGUYỄN NGỌC NGUYÊN

Đề tài đồ án tốt nghiệp:

MÔ HÌNH IOT GIÁM SÁT TRỒNG RAU TRONG HỘ GIA ĐÌNH

Chuyên ngành Máy Tính - Hệ Thống Nhúng

TP. Hồ Chí Minh, tháng 7 năm 2025

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG
BỘ MÔN MÁY TÍNH - HỆ THỐNG NHÚNG

NGUYỄN NGỌC NGUYÊN
20200058

Đề tài:

MÔ HÌNH IOT GIÁM SÁT TRỒNG RAU TRONG HỘ GIA ĐÌNH

IOT-BASED MODEL FOR VEGETABLE CULTIVATION MONITORING IN HOUSEHOLD GARDENS

ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN
NGÀNH KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG
CHUYÊN NGÀNH MÁY TÍNH - HỆ THỐNG NHÚNG

NGƯỜI HƯỚNG DẪN KHOA HỌC
TS. NGUYỄN XUÂN VINH

TP. Hồ Chí Minh, tháng 7 năm 2025

LỜI CẢM ƠN

Em xin trân trọng gửi lời cảm ơn đến quý thầy cô khoa Điện tử – Viễn thông, Trường Đại học Khoa học tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh đã tận tình giảng dạy và tạo điều kiện thuận lợi cho em trong suốt quá trình học tập.

Đặc biệt, em xin chân thành cảm ơn thầy Nguyễn Xuân Vinh đã dành thời gian, tâm huyết hướng dẫn và hỗ trợ em trong quá trình thực hiện Đồ án tốt nghiệp này.

Mặc dù đã rất cố gắng, nhưng do hạn chế về mặt thời gian và kinh nghiệm, bài báo cáo không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và góp ý từ quý thầy cô để em có thể hoàn thiện hơn trong tương lai.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Tôi xin cam đoan những số liệu và kết quả nghiên cứu được trình bày trong đồ án tốt nghiệp là trung thực. Tôi cam kết không sao chép nội dung hay kết quả của các nghiên cứu khác đã được công bố. Những tài liệu tham khảo được tôi sử dụng trong báo cáo đã được trích dẫn một cách đầy đủ, rõ ràng về nguồn gốc theo quy định.

Tôi xin chịu trách nhiệm nếu vi phạm các cam kết trên.

Họ tên sinh viên

Nguyễn Ngọc Nguyên

MỤC LỤC

Danh sách chữ viết tắt

Chữ viết tắt	Tiếng Anh	Ý nghĩa tiếng Việt
IoT	Internet of Things	Internet vạn vật
LED	Light Emitting Diode	Điốt phát quang
USB	Universal Serial Bus	Cổng kết nối nối tiếp đa năng
MCU	Microcontroller Unit	Bộ vi điều khiển
ADC	Analog to Digital Converter	Bộ chuyển đổi tín hiệu tương tự sang số
DAC	Digital to Analog Converter	Bộ chuyển đổi tín hiệu số sang tương tự
UART	Universal Asynchronous Receiver Transmitter	Giao tiếp nối tiếp bất đồng bộ
SPI	Serial Peripheral Interface	Giao tiếp ngoại vi nối tiếp
I2C	Inter-Integrated Circuit	Giao tiếp mạch tích hợp nội
BLE	Bluetooth Low Energy	Bluetooth năng lượng thấp
Wi-Fi	Wireless Fidelity	Kết nối mạng không dây
CPU	Central Processing Unit	Bộ xử lý trung tâm
RAM	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
SDRAM	Synchronous Dynamic RAM	RAM động đồng bộ
DC	Direct Current	Dòng điện một chiều
AC	Alternating Current	Dòng điện xoay chiều
GPIO	General Purpose Input/Output	Chân vào/ra đa năng
CSI	Camera Serial Interface	Giao tiếp nối tiếp dành cho camera

DSI	Display Serial Interface	Giao tiếp nối tiếp dành cho hiển thị
DO	Digital Output	Ngõ ra kỹ thuật số
AO	Analog Output	Ngõ ra tương tự
REST API	Representational State Transfer - Application Programming Interface	Giao diện lập trình ứng dụng theo mô hình REST
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
NoSQL	Not Only SQL	Cơ sở dữ liệu không quan hệ

Danh sách hình ảnh trong báo cáo

CHƯƠNG 1: TỔNG QUAN

1.1 Tổng quan về IoT trong quản lý vườn rau hộ gia đình

Công nghệ Internet vạn vật (IoT – Internet of Things) đang ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực đời sống, trong đó có nông nghiệp đô thị và mô hình vườn rau hộ gia đình. Việc áp dụng IoT vào quản lý vườn rau giúp tự động hóa các hoạt động chăm sóc cây trồng như tưới nước, giám sát độ ẩm, ánh sáng, nhiệt độ và dinh dưỡng đất, từ đó nâng cao hiệu quả trồng trọt và tiết kiệm thời gian, công sức cho người sử dụng.

Các hệ thống IoT trong vườn rau hộ gia đình thường bao gồm các cảm biến (độ ẩm đất, nhiệt độ không khí, ánh sáng...), các thiết bị điều khiển (bơm nước, đèn LED trồng cây...), bộ vi xử lý (như Arduino, Raspberry Pi) và nền tảng điều khiển qua smartphone hoặc máy tính. Nhờ khả năng kết nối Internet, người dùng có thể theo dõi tình trạng cây trồng theo thời gian thực và điều khiển hệ thống từ xa thông qua ứng dụng di động.

Ứng dụng IoT không chỉ giúp cải thiện năng suất và chất lượng rau sạch tại nhà mà còn góp phần thúc đẩy lối sống xanh, tiết kiệm tài nguyên và hỗ trợ việc trồng trọt bền vững trong bối cảnh đô thị hóa ngày càng gia tăng.

Hình 1.1: Giám sát vườn qua ứng dụng di động

1.2 Các thành phần chính của hệ thống IoT trong vườn rau hộ gia đình

- Cảm biến (Sensors): Đo độ ẩm đất, nhiệt độ môi trường, ánh sáng, độ pH, độ ẩm không khí... giúp thu thập dữ liệu môi trường theo thời gian thực.

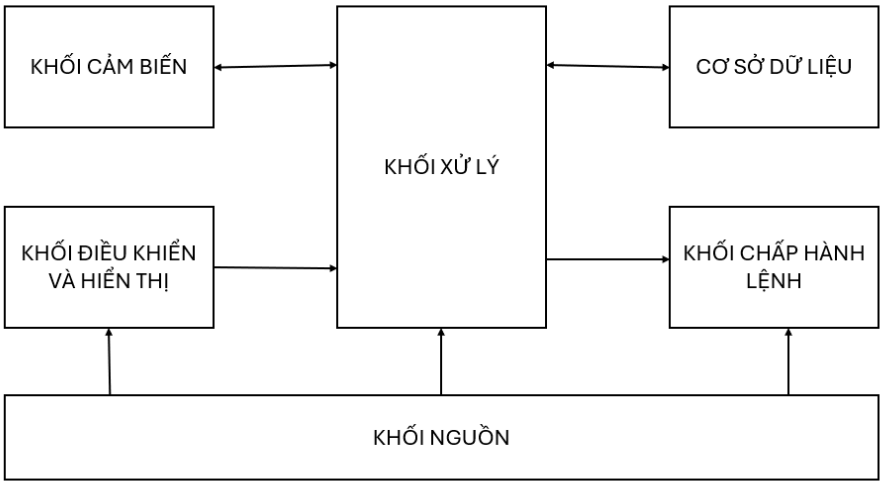
- Bộ điều khiển trung tâm: Thường là vi điều khiển như Arduino, ESP8266, ESP32 hoặc Raspberry Pi có nhiệm vụ xử lý dữ liệu cảm biến và điều khiển các thiết bị đầu ra.
- Thiết bị chấp hành (Actuators): Bao gồm hệ thống tưới nước tự động, đèn LED bổ sung ánh sáng, quạt thông gió... được điều khiển tự động hoặc bán tự động.
- Kết nối mạng: Hệ thống có thể sử dụng Wi-Fi, Zigbee hoặc Bluetooth để truyền dữ liệu đến điện thoại hoặc máy chủ đám mây.
- Giao diện người dùng: Ứng dụng trên điện thoại hoặc trang web giúp người dùng giám sát và điều khiển hệ thống từ xa.

1.3 Lợi ích khi ứng dụng IoT vào quản lý vườn rau hộ gia đình

- Tự động hóa: Giảm thiểu công việc thủ công như tưới nước, theo dõi nhiệt độ, nhắc nhở thời gian bón phân.
- Tiết kiệm tài nguyên: Tưới nước đúng lúc, đúng lượng giúp tiết kiệm nước và tránh lãng phí phân bón.
- Nâng cao năng suất và chất lượng: Duy trì điều kiện lý tưởng cho cây phát triển, hạn chế sâu bệnh.
- Giám sát từ xa: Người dùng có thể quản lý vườn dù không có mặt tại nhà, phù hợp với người bận rộn hoặc thường xuyên đi xa.
- Thân thiện với môi trường: Hạn chế sử dụng hóa chất, thúc đẩy mô hình trồng rau sạch, bền vững trong đô thị.

CHƯƠNG 2: LỰA CHỌN CÁC THIẾT BỊ, LINH KIỆN, THIẾT KẾ PHẦN MỀM VÀ PHẦN CỨNG CỦA HỆ THỐNG

2.1 Sơ đồ khối tổng quan mô hình.



Hình 2.1: Sơ đồ khối tổng quan mô hình

Hệ thống gồm các khối chức năng chính như sau:

- **Khối nguồn:**
  Cung cấp điện năng cho toàn bộ hệ thống. Đây là khối đảm bảo sự hoạt động ổn định và liên tục của các thiết bị trong mô hình như cảm biến, vi điều khiển, các thiết bị chấp hành, và màn hình hiển thị. Nguồn điện có thể được cấp từ adapter hoặc pin dự phòng tùy theo thiết kế.
- **Khối cảm biến:**
  Gồm các cảm biến đo đạc các thông số môi trường như độ ẩm đất, nhiệt độ không khí, ánh sáng và mưa. Dữ liệu từ các cảm biến được thu thập theo thời gian thực và chuyển đến khối xử lý để phân tích. Khối này đóng vai trò then chốt trong việc cung cấp thông tin chính xác để hệ thống đưa ra quyết định điều khiển phù hợp.
- **Khối xử lý:**
  Sử dụng vi điều khiển (như ESP32) để tiếp nhận, phân tích dữ liệu từ khối cảm biến. Dữ liệu được so sánh với ngưỡng tham chiếu hoặc cơ sở dữ liệu định sẵn. Từ đó, hệ thống sẽ đưa ra quyết định điều khiển thích hợp (bật/tắt bơm nước, quạt, đèn...).

- **Cơ sở dữ liệu:**

  Lưu trữ thông tin về các thông số môi trường, lịch sử hoạt động và trạng thái của các thiết bị. Dữ liệu này có thể được đồng bộ với nền tảng điện toán đám mây như Firebase để hỗ trợ giám sát và điều khiển từ xa thông qua ứng dụng.

- **Khối chấp hành lệnh:**

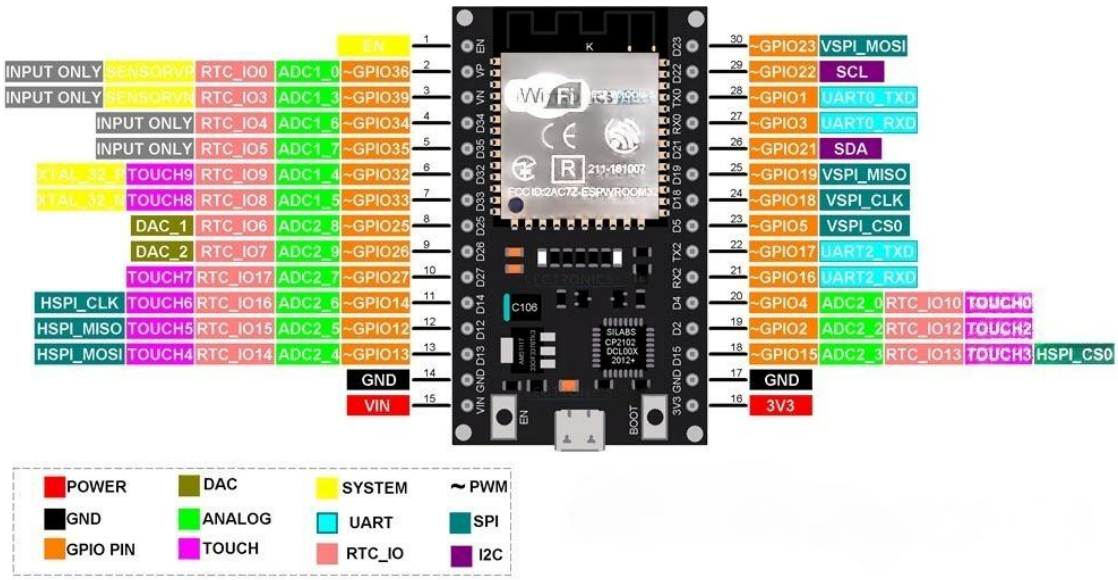
  Gồm các thiết bị thực thi như máy bơm, quạt, đèn chiếu sáng,... Thực hiện các lệnh điều khiển được gửi từ khối xử lý. Việc thực hiện chính xác các lệnh này giúp duy trì điều kiện môi trường phù hợp cho cây trồng.

- **Khối điều khiển và hiển thị:**

  Gồm giao diện điều khiển người dùng (UI), thường là ứng dụng di động hoặc màn hình LCD. Khối này hiển thị trạng thái hệ thống, các thông số đo được và cho phép người dùng theo dõi, can thiệp hoặc điều khiển thủ công khi cần thiết.

2.2 Lựa chọn thiết bị

2.2.1 Module ESP32 WROOM 32 30PIN CP2102



Hình 2.2: Sơ đồ chân ESP32 WROOM 32 30PIN CP2102

ESP32 WROOM-32 là module phát triển nhỏ gọn, tích hợp Wi-Fi và Bluetooth BLE, sử dụng vi điều khiển ESP32 mạnh mẽ. Giao tiếp máy tính qua cổng micro USB với chip nạp CP2102, phù hợp cho các ứng dụng IoT, tự động hóa và điều khiển từ xa.

Thông số kỹ thuật cơ bản:

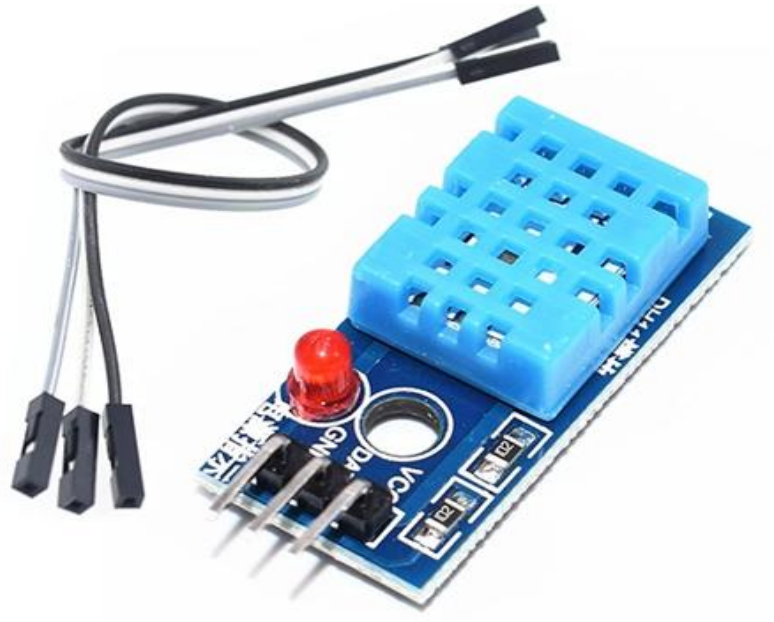
- Vi điều khiển: ESP32 (dual-core, 240MHz, 32-bit)
- RAM: 520KB, Flash: 4MB
- Kết nối: Wi-Fi 802.11 b/g/n, Bluetooth 4.2 (BLE)
- Giao tiếp: 30 chân GPIO (UART, SPI, I2C, PWM, ADC, DAC…)
- Micro USB, chip nạp CP2102
- Nút nhấn: BOOT và EN (Reset)
- Nguồn cấp: 5V qua USB, I/O hoạt động ở mức 3.3V

2.2.2 Cảm biến nhiệt độ, độ ẩm không khí DHT11

DHT11 là cảm biến kỹ thuật số dùng để đo nhiệt độ và độ ẩm môi trường, có kích thước nhỏ, giá thành thấp, dễ sử dụng với vi điều khiển như Arduino, ESP32, Raspberry Pi.

Thông số kỹ thuật cơ bản:

- Loại cảm biến: Nhiệt độ & độ ẩm kỹ thuật số
- Dải đo nhiệt độ: 0 – 50°C (sai số ±2°C)
- Dải đo độ ẩm: 20 – 90% RH (sai số ±5% RH)
- Tần suất đo: 1 lần/giây (1Hz)
- Điện áp hoạt động: 3.3V – 5.5V
- Giao tiếp: Tín hiệu số (1 dây – single wire)
- Kích thước: khoảng 15.5mm x 12mm x 5.5mm



Hình 2.3: Cảm biến nhiệt độ, độ ẩm không khí DHT11

2.2.3 Cảm biến mưa

Nguyên lý hoạt động:

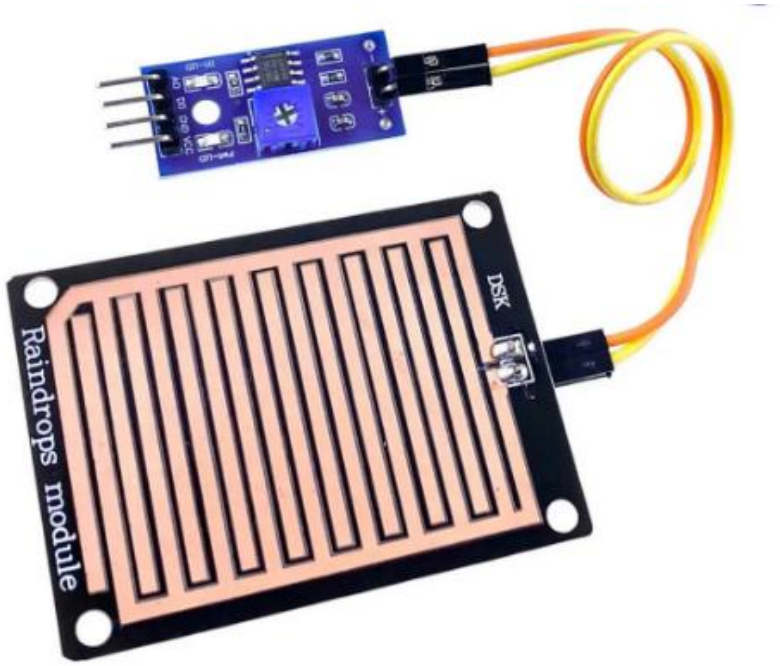
- Cảm biến gồm một tấm mạch có các đường dẫn điện kim loại được in trên bề mặt như một lưới điện.
- Khi không có mưa, các đường dẫn này không dẫn điện (mạch hở).
- Khi có mưa rơi vào bề mặt cảm biến, nước (có tính dẫn điện nhẹ) sẽ kết nối các đường dẫn, tạo ra dòng điện → mạch trở nên dẫn điện, giúp phát hiện có mưa.

Thông số kỹ thuật:

Điện áp hoạt động: 3.3V – 5V DC

Tín hiệu ngõ ra:

- Digital (D0): mức LOW khi có nước mưa
- Analog (A0): tỷ lệ tín hiệu theo mức độ ẩm ướt



Hình 2.4: Cảm biến mưa

2.2.4 Cảm biến quang học LM393

Giới Thiệu:

LM393 là mô-đun cảm biến ánh sáng sử dụng điốt quang, có khả năng phát hiện cường độ ánh sáng môi trường và cho ra tín hiệu đầu ra dạng số (digital) hoặc tương tự (analog). Thích hợp dùng với Arduino, ESP32,...

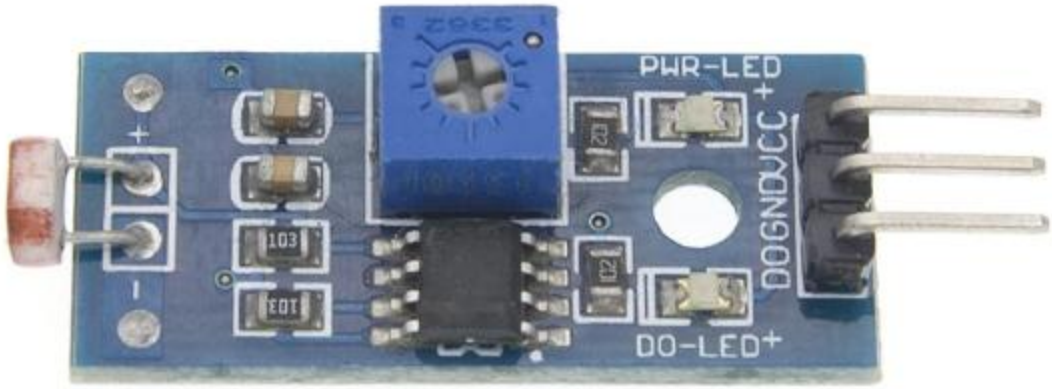
Thông số kỹ thuật:

- Điện áp hoạt động: 3.3V – 5V
- Tín hiệu đầu ra:
    - DO (Digital): mức 0 hoặc 1 tùy theo độ sáng
    - AO (Analog): điện áp tỷ lệ với cường độ ánh sáng
- Chiết áp: dùng để điều chỉnh ngưỡng ánh sáng
- Kích thước: 3.2cm x 1.4cm
- Có lỗ bắt vít: dễ lắp đặt

Nguyên lý hoạt động:

- Khi cường độ ánh sáng thấp hơn ngưỡng, chân DO xuất mức cao (1)
- Khi ánh sáng vượt ngưỡng, chân DO xuất mức thấp (0)

- Hướng phát hiện ánh sáng tốt nhờ tính định hướng của điốt quang



Hình 2.5: Cảm biến quang học LM393

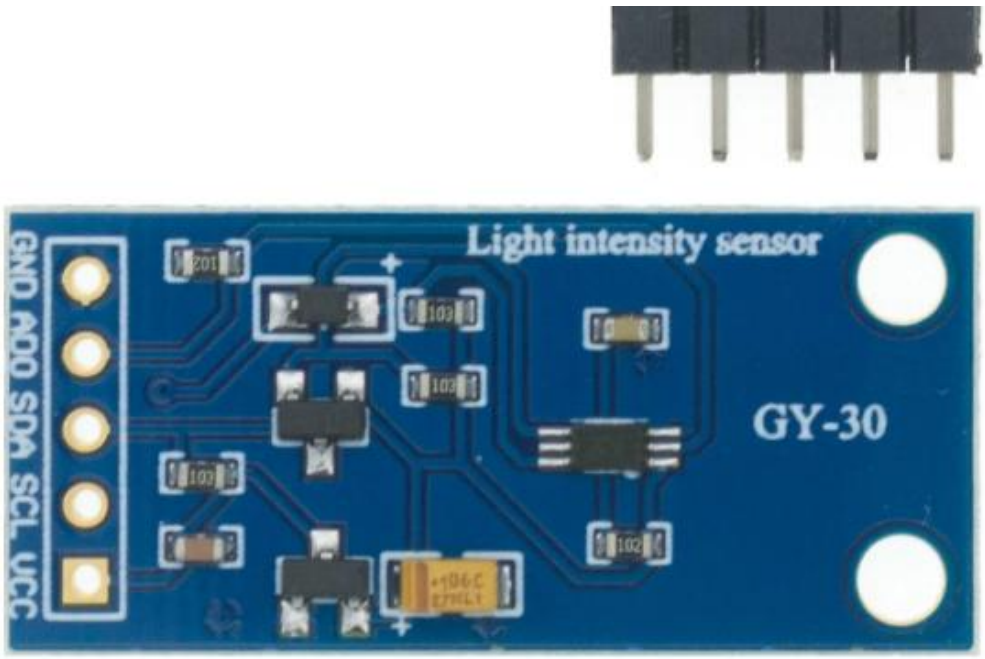
2.2.5 Cảm biến ánh sáng kỹ thuật số GY-302

Thông tin chung:

GY-302 sử dụng chip BH1750FVI của hãng ROHM, là cảm biến ánh sáng kỹ thuật số độ chính xác cao, có thể đo cường độ ánh sáng trong đơn vị lux.

Thông số kỹ thuật:

- Chip xử lý: BH1750FVI (ROHM - Nhật Bản)
- Nguồn hoạt động: 3V – 5V DC
- Dải đo: 0 – 65.535 lux
- Độ phân giải ADC: 16 bit
- Kích thước: 13.9mm x 18.5mm
- Giao tiếp: I2C
- Đặc điểm nổi bật:
    - Đầu ra tín hiệu kỹ thuật số, không cần hiệu chuẩn hoặc tính toán phức tạp
    - Đáp ứng ánh sáng theo phổ gần giống mắt người
    - Đo lường ánh sáng môi trường với độ chính xác cao (tới 1 lux)
    - Ổn định, ít bị nhiễu bởi ánh sáng nền



Hình 2.6: Cảm biến ánh sáng kỹ thuật số GY-302

2.2.6 Cảm biến độ ẩm đất (có ngõ ra số và tương tự)

Chức năng:

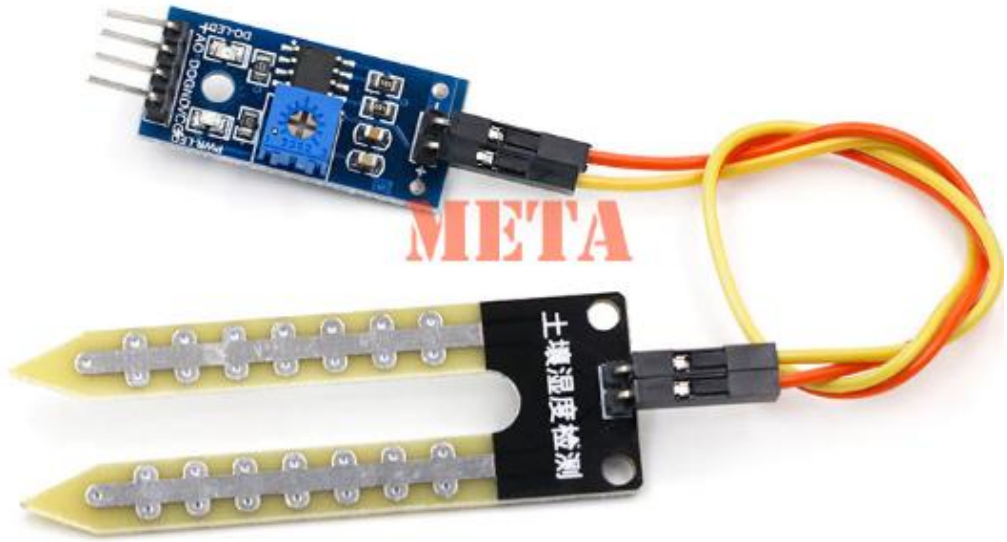
Cảm biến dùng để phát hiện độ ẩm của đất, thường ứng dụng trong hệ thống tưới cây tự động. Khi đất khô, tín hiệu đầu ra DO ở mức cao; khi đủ ẩm, DO ở mức thấp.

Thông số kỹ thuật:

- Điện áp hoạt động: 3.3V ~ 5V
- Tín hiệu đầu ra:
    - DO (Digital): mức cao/thấp tùy theo độ ẩm
    - AO (Analog): tín hiệu tỷ lệ độ ẩm
- Điều chỉnh ngưỡng: thông qua biến trở (màu xanh)
- Chip so sánh: LM393 – hoạt động ổn định
- Kích thước PCB: 3.2 x 1.4 cm
- Dây kết nối:
    - VCC: 3.3V – 5V
    - GND: nối đất
    - DO: đầu ra kỹ thuật số
    - AO: đầu ra tương tự

Ứng dụng:

Hệ thống tưới cây tự động, cảnh báo đất khô, đo độ ẩm đất trong nhà kính, nông nghiệp thông minh,…

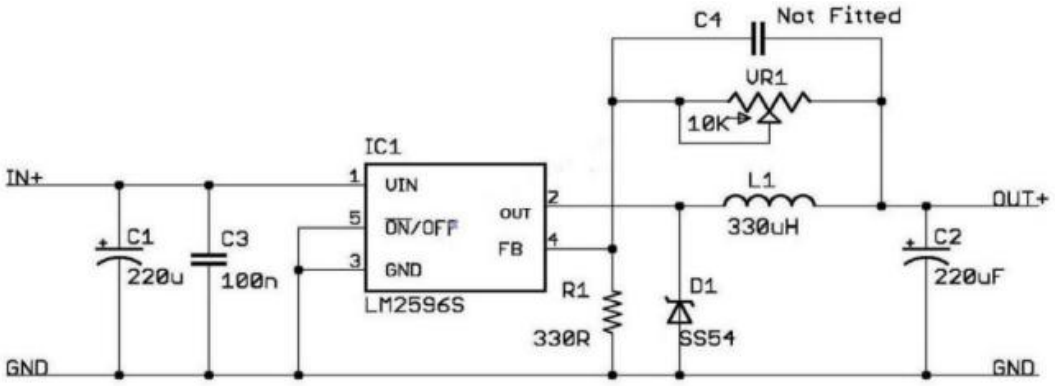


Hình 2.7: Cảm biến độ ẩm đất

2.2.7 Module hạ áp DC-DC LM2596 5V

- Điện áp đầu vào: Từ 3V đến 30V.
- Điện áp đầu ra: Điều chỉnh được trong khoảng 1.5V đến 30V.
- Dòng đáp ứng tối đa là 3A.
- Hiệu suất: 92%
- Công suất: 15W
- Kích thước: 45 (dài) * 20 (rộng) * 14 (cao) mm

Hình 2.8: Module hạ áp DC-DC LM2596 5V

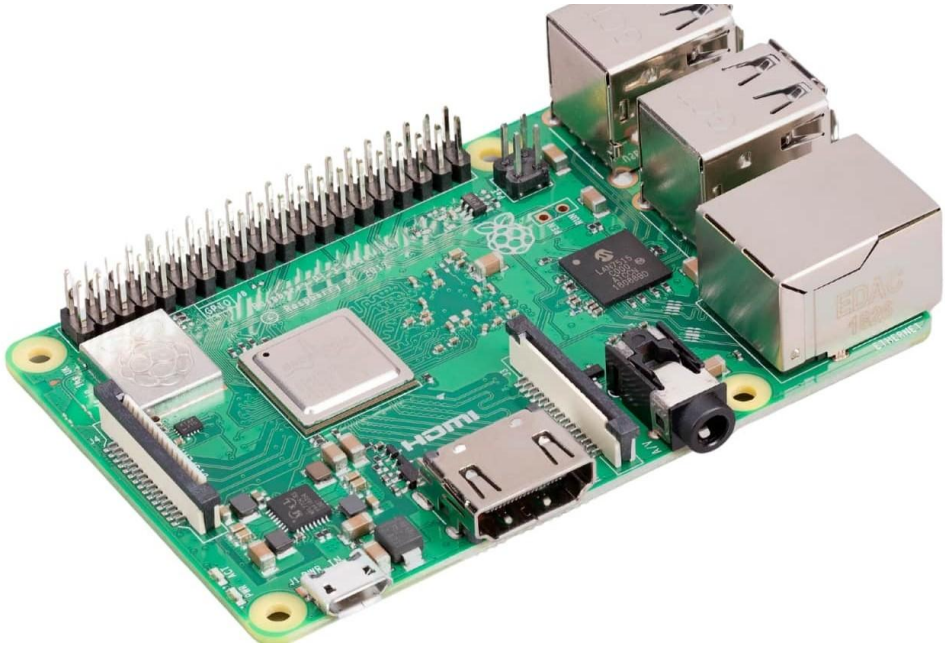


Hình 2.9: Sơ đồ IC LM2596

2.2.8 Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+ là một máy tính mini nhỏ gọn, đa năng, phù hợp cho nhiều ứng dụng từ học tập, phát triển phần mềm đến các dự án IoT và tự động hóa. Thiết bị được trang bị phần cứng mạnh mẽ cùng khả năng kết nối linh hoạt:

- Bộ vi xử lý (CPU): Broadcom BCM2837B0, kiến trúc Cortex-A53 (ARMv8) 64-bit, tốc độ 1.4GHz (4 nhân).
- Bộ nhớ RAM: 1GB LPDDR2 SDRAM.
- Kết nối không dây: Wi-Fi dual-band 2.4GHz và 5GHz (chuẩn IEEE 802.11 b/g/n/ac), Bluetooth 4.2 và Bluetooth Low Energy (BLE).
- Mạng có dây: Ethernet Gigabit qua USB 2.0, hỗ trợ tốc độ truyền tối đa lên tới 300 Mbps.
- Cổng kết nối:
    - 4 cổng USB 2.0
    - 1 cổng HDMI chuẩn
    - Cổng camera CSI
    - Jack âm thanh/video 3.5mm
- GPIO: 40 chân mở rộng cho kết nối thiết bị ngoại vi và phát triển mạch điều khiển.
- Lưu trữ: Hỗ trợ thẻ nhớ MicroSD.
- Kích thước: 85mm x 56mm x 17mm.
- Nguồn điện: 5V/2.5A thông qua cổng Micro USB.



Hình 2.10: Raspberry Pi 3 Model B+

2.2.9 Máy bơm 12V RS385

Máy bơm 12V RS385 là loại máy bơm nước mini sử dụng phổ biến trong thiết bị bể cá, tuần hoàn nước, đồ DIY..

Thông số kỹ thuật:

Điện áp đầu vào 6-12V DC

(Khuyến nghị sử dụng: 9V – 12V, dòng 1A)

Dòng điện tiêu thụ: 0.5A – 0.7A

Công suất tại 6V: khoảng 6W

Nhiệt độ nước tối đa cho phép: 80°C

Tuổi thọ trung bình: khoảng 2.500 giờ hoạt động

2.2.10 Quạt tản nhiệt không chổi than DC 12V

Mục đích điều hòa không khí giúp cây dễ hấp thụ và trao đổi chất

Thông số kỹ thuật:

Điện áp hoạt động: 12VDC

Dòng điện định mức: 0.10A

Kích thước: 5 x 5 x 1 cm

Hình 2.11: Quạt tản nhiệt không chổi than DC 12V

2.2.11 Đèn chiếu sáng 12V

Thông số kỹ thuật:

Thông số	Giá trị
Điện áp định mức	12VDC
Công suất tiêu thụ	22 W/mét
Nhiệt độ màu	3500K / 5700K
Chiều rộng thanh LED	12 mm
Độ dày thanh LED	1.8 mm
Tuổi thọ	50.000 giờ
Số lượng LED	144 bóng / 1 mét

Hình 2.12: Đèn chiếu sáng 12V

2.2.12 Nguồn tổ ong 12V – 10A

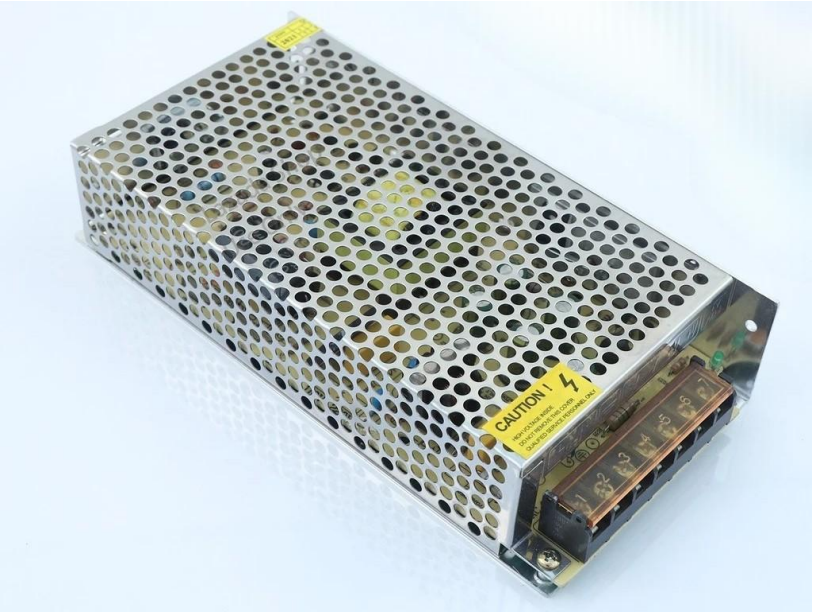
Chức năng:

Dùng chuyển đổi từ điện xoay chiều thành điện 1 chiều cung cấp năng lượng cho hệ thống

Thông số kỹ thuật:

- Điện áp đầu vào: 110V / 220V AC (chuyển đổi được)
- Điện áp đầu ra: 12V DC
- Dòng điện đầu ra: 10A

- Công suất tối đa: 120W – 360W



Hình 2.13: Nguồn tổ ong 12V – 10A

2.2.13 Module 4 Relay Với Opto Cách Ly (5VDC)

Thông số kỹ thuật:

- Điện áp điều khiển: 5V hoặc 12V DC (tùy phiên bản)
- Dòng tiêu thụ mỗi relay: ~80mA
- Công suất tải tối đa:
    - AC: 250V – 10A
    - DC: 30V – 10A
- Loại kích hoạt: Mức thấp (LOW)
- Có opto cách ly: giúp bảo vệ mạch điều khiển
- Đèn báo: LED hiển thị trạng thái hoạt động của từng relay

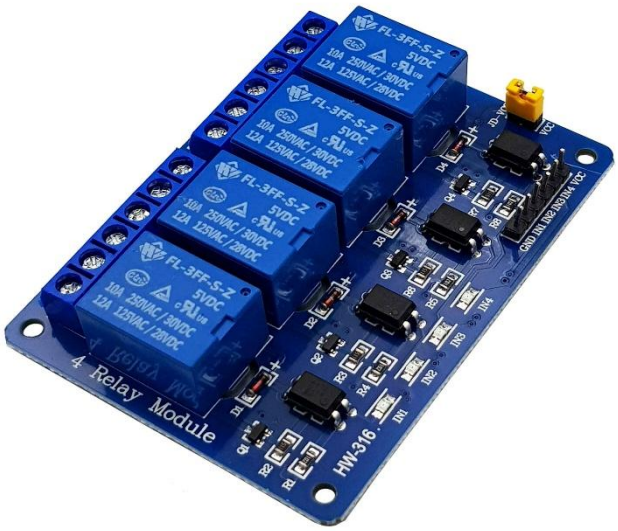
Cấu trúc tiếp điểm:

- COM (Common): chân chung
- NO (Normally Open): thường hở – chỉ nối khi relay kích
- NC (Normally Closed): thường đóng – nối khi relay chưa kích

Ứng dụng:

- Điều khiển thiết bị điện AC/DC công suất lớn

- Tự động hóa, nhà thông minh, robot, máy in 3D, Arduino



Hình 2.14: Module 4 Relay Với Opto Cách Ly (5VDC)

2.2.14 Mạch Điều Khiển Động Cơ Bước ULN2003, Động Cơ Bước 5V

Thông số kỹ thuật:

Mạch đệm ULN2003:

- Điện áp cung cấp: 5 ~ 12VDC.
- Tín hiệu ngõ vào: 4 chân in1, in2, in3, in4.
- Tín hiệu ngõ ra: Jack cắm động cơ bước 28BYJ-48.

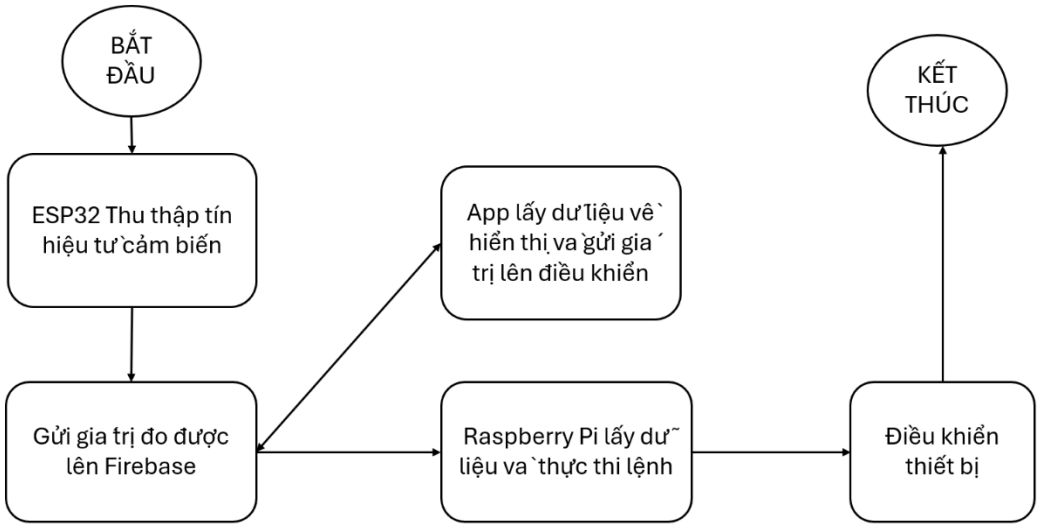
Động cơ 28BYJ-48:

- Điện áp cung cấp: 5VDC
- Số phase: 4.

Hình 2.15: Mạch Điều Khiển Động Cơ Bước ULN2003, Động Cơ Bước 5V

2.3 Lưu đồ thuật toán

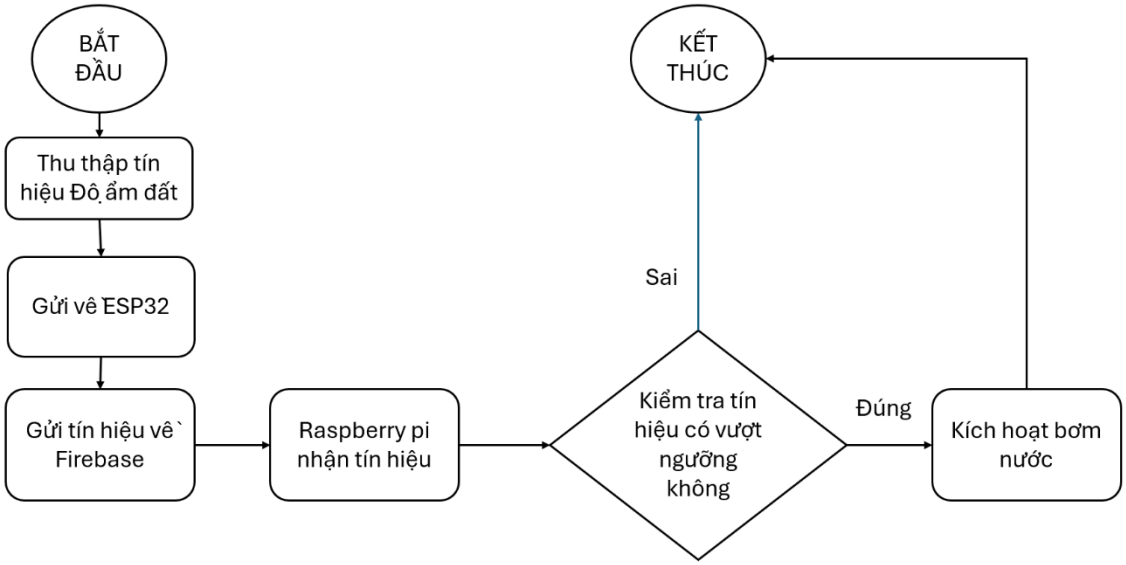
2.3.1 Lưu đồ thuật toán chung



Hình 2.16: Lưu đồ thuật toán chung

ESP32 thu thập dữ liệu từ cảm biến và gửi lên Firebase. Ứng dụng sẽ hiển thị giá trị và gửi lệnh điều khiển. Raspberry Pi lấy dữ liệu, thực thi lệnh, và điều khiển thiết bị tương ứng, hoàn tất chu trình xử lý.

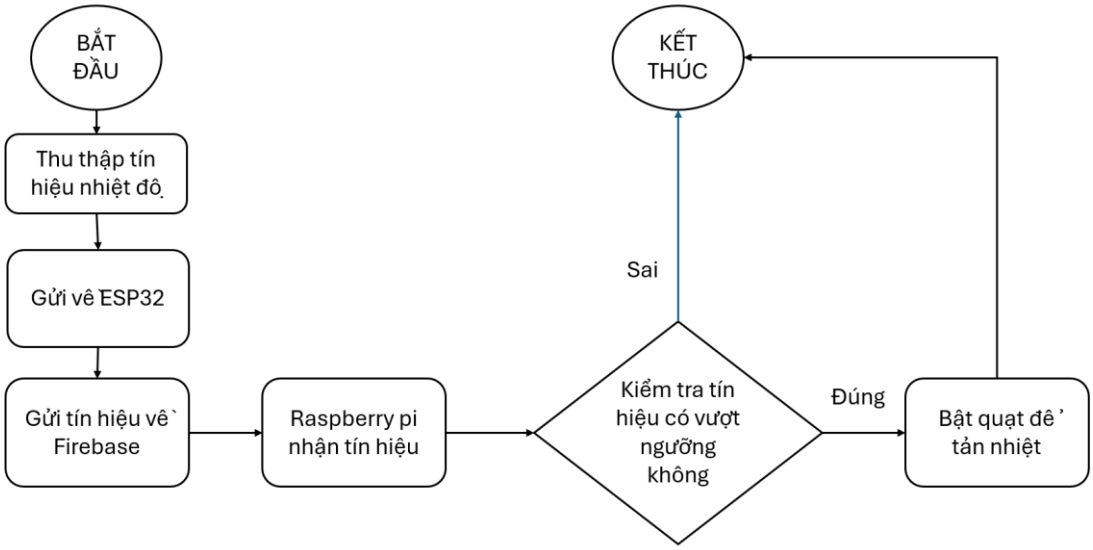
2.3.2 Lưu đồ thuật toán cảm biến độ ẩm đất



Hình 2.17: Lưu đồ thuật toán cảm biến độ ẩm đất

Hình 2.17 mô tả thuật toán cảm biến độ ẩm đất: hệ thống bắt đầu bằng việc thu thập dữ liệu độ ẩm từ cảm biến, gửi về ESP32 và truyền lên Firebase. Raspberry Pi nhận dữ liệu, kiểm tra ngưỡng và nếu vượt ngưỡng thì kích hoạt bơm nước.

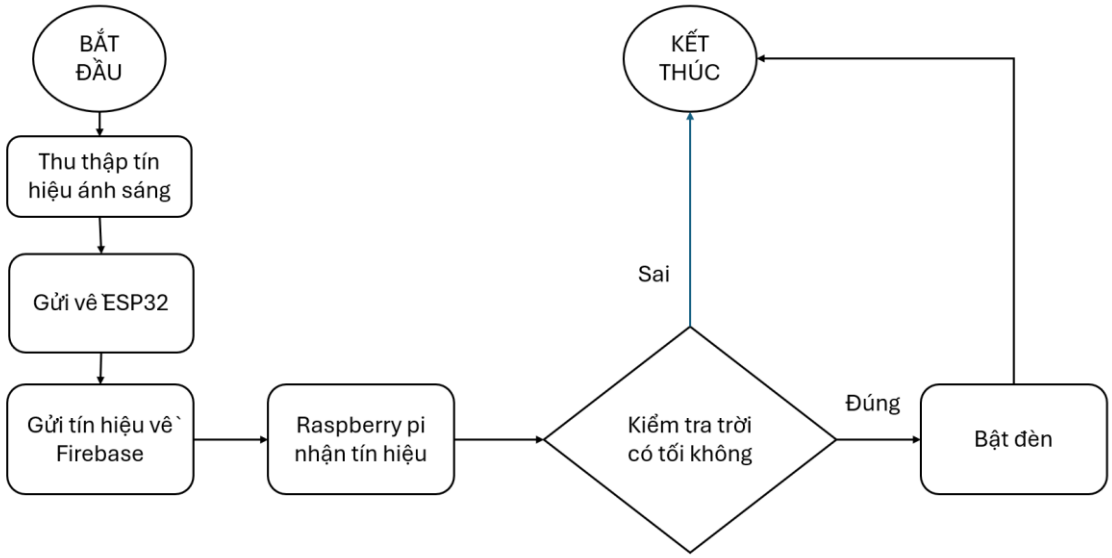
2.3.3 Lưu đồ thuật toán cảm biến nhiệt độ



Hình 2.18 Lưu đồ thuật toán cảm biến nhiệt độ

Hình 2.18 là lưu đồ cảm biến nhiệt độ: quá trình tương tự như cảm biến độ ẩm, nhưng thay vì kích hoạt bơm, hệ thống sẽ bật quạt để làm mát khi nhiệt độ vượt ngưỡng.

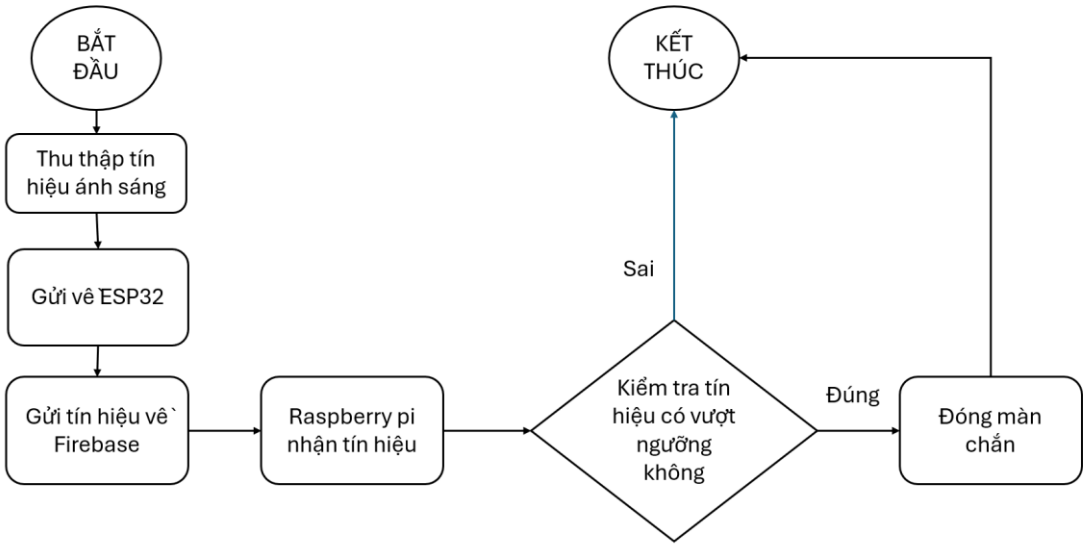
2.3.4 Lưu đồ thuật toán cảm biến ánh sáng quang trở



Hình 2.19: Lưu đồ thuật toán cảm biến ánh sáng quang trở

Hình 2.19 thể hiện thuật toán cảm biến ánh sáng quang trở: dữ liệu ánh sáng được thu thập, gửi qua ESP32 và Firebase, sau đó Raspberry Pi kiểm tra điều kiện trời tối để bật đèn khi cần thiết.

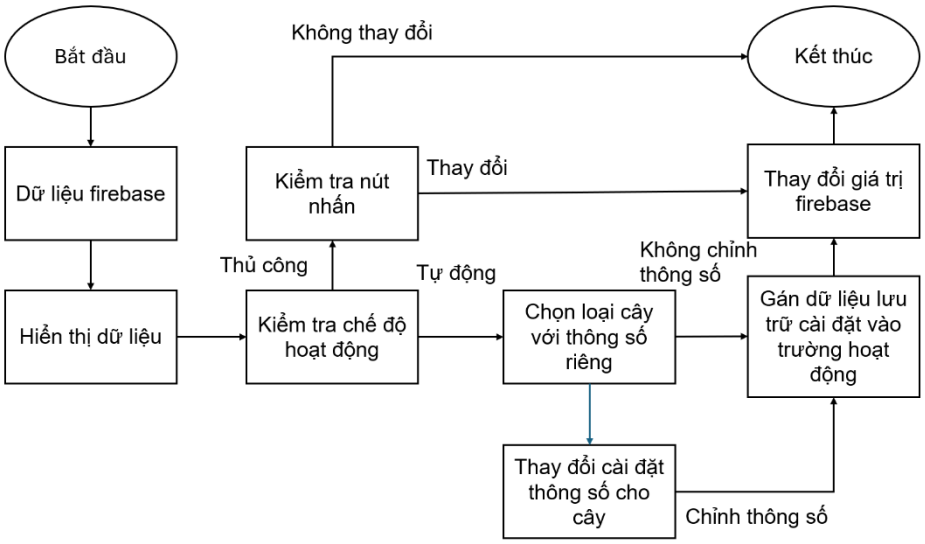
2.3.5 Lưu đồ thuật toán cảm biến ánh sáng theo lux



Hình 2.20: Lưu đồ thuật toán cảm biến ánh sáng theo lux

Hình 2.20 là thuật toán cảm biến ánh sáng theo đơn vị lux: tương tự quy trình trên, nếu giá trị ánh sáng vượt ngưỡng, hệ thống sẽ điều khiển đóng màn chắn.

2.3.6 Lưu đồ thuật toán của ứng dụng di động



Hình 2.21: Lưu đồ thuật toán của ứng dụng di động

Hình 2.21 mô tả thuật toán hoạt động của ứng dụng di động: ứng dụng lấy dữ liệu từ Firebase, hiển thị cho người dùng và cho phép thay đổi thông số hoạt động thủ công hoặc tự động theo từng loại cây. Các điều chỉnh sẽ được lưu lại và đồng bộ lên Firebase để cập nhật hệ thống điều khiển.

2.4 Cơ sở dữ liệu Firebase

Firebase Realtime Database là một dịch vụ cơ sở dữ liệu NoSQL do Google cung cấp, cho phép lưu trữ và đồng bộ dữ liệu theo thời gian thực giữa client và server.

Đặc điểm chính:

- Dữ liệu dạng JSON: Cơ sở dữ liệu được lưu dưới dạng cây JSON, phù hợp với các ứng dụng cần đồng bộ nhanh.
- Thời gian thực (Realtime): Khi dữ liệu thay đổi, tất cả client đang kết nối sẽ nhận được bản cập nhật ngay lập tức.
- Đồng bộ hai chiều: Người dùng có thể đọc và ghi dữ liệu từ cả ứng dụng web, Android, iOS.
- Truy cập bằng REST API hoặc SDK của Firebase.

Cấu trúc dữ liệu:

- Tổ chức dưới dạng cây phân cấp (hierarchical JSON tree).
- Dữ liệu được truy cập thông qua các đường dẫn (path).

Quyền truy cập và bảo mật:

- Firebase sử dụng Firebase Rules để quản lý quyền đọc/ghi dữ liệu.
- Có thể giới hạn theo người dùng, thời gian, điều kiện,...

Ưu điểm:

- Tốc độ cao: Gửi/nhận dữ liệu gần như tức thời.
- Dễ tích hợp: Với các ứng dụng Android, iOS và web.
- Miễn phí (ở mức giới hạn): Phù hợp với các dự án vừa và nhỏ.

Ứng dụng thực tế:

- Ứng dụng chat, theo dõi cảm biến IoT, thông báo trạng thái, quản lý người dùng online, đồng bộ dữ liệu trên nhiều thiết bị

2.5 Phần mềm Thunkable

Thunkable là một nền tảng lập trình kéo-thả giúp người dùng tạo ra ứng dụng di động cho Android và iOS mà không cần viết mã phức tạp.

Đặc điểm chính:

- Lập trình không cần code (No-code): Sử dụng giao diện kéo – thả trực quan dựa trên các khối logic.
- Tạo ứng dụng đa nền tảng: Viết một lần, xuất bản lên cả Android và iOS.
- Hỗ trợ thời gian thực: Có thể xem trước giao diện và chức năng ngay khi thiết kế.
- Tích hợp dịch vụ đám mây: Dễ dàng kết nối với Firebase, Google Sheets, API bên ngoài.

Chức năng nổi bật:

- Thiết kế giao diện bằng cách kéo các thành phần (nút, văn bản, hình ảnh, v.v.).
- Cho phép người dùng thử nghiệm trực tiếp trên điện thoại qua ứng dụng Thunkable Live.

Ưu điểm:

- Dễ học, phù hợp cho người mới bắt đầu hoặc học sinh, sinh viên.
- Không cần cài đặt phần mềm (chạy trực tuyến trên trình duyệt).
- Có phiên bản miễn phí và nhiều tài nguyên học tập.

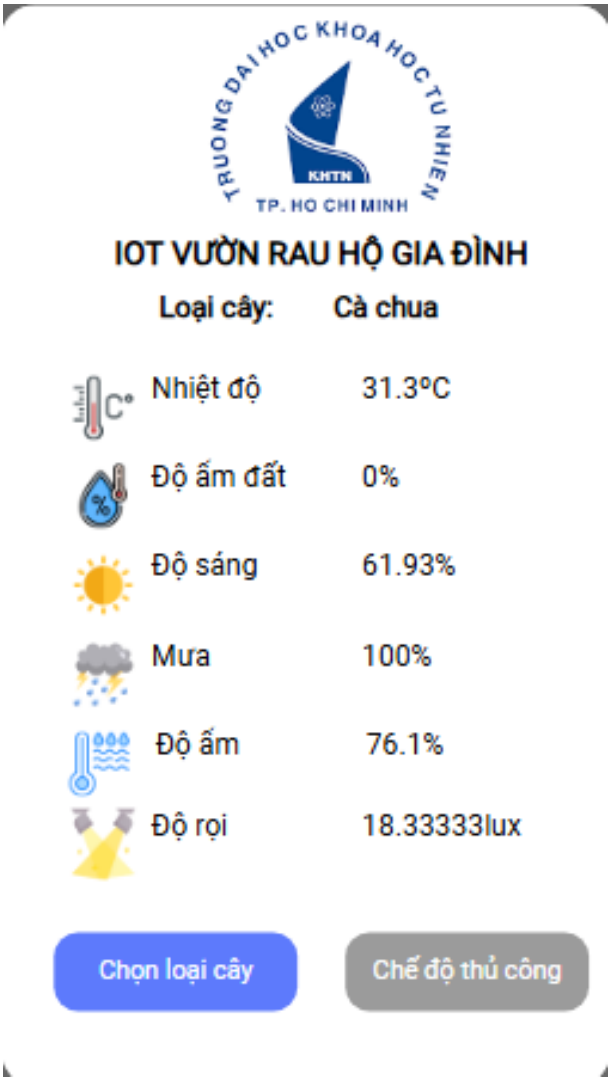
Ứng dụng:

- Phát triển ứng dụng IoT, trò chơi đơn giản, ứng dụng giáo dục, quản lý cơ sở dữ liệu..

CHƯƠNG 3: THỰC HIỆN VÀ LẮP ĐẶT MÔ HÌNH

3.1 Thiết kế giao diện phần mềm điều khiển ứng dụng di động

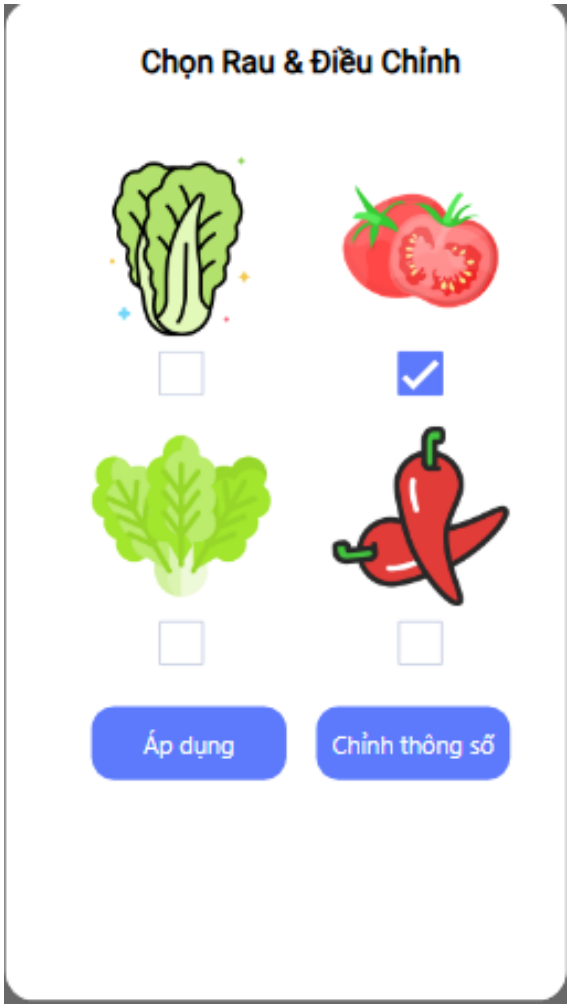
Phần mềm xây dựng trên nền tảng Thunkable:



Hình 3.1: Màn hình chính của ứng dụng hiển thị thông tin từ cảm biến

Màn hình chính của ứng dụng được thiết kế để hiển thị đầy đủ thông tin về tình trạng hoạt động của hệ thống, bao gồm loại cây đang được canh tác và dữ liệu thu thập từ các cảm biến môi trường như nhiệt độ, độ ẩm không khí, cường độ ánh sáng, độ ẩm đất và tình trạng mưa. Nhờ giao diện trực quan này, người dùng có thể dễ dàng theo dõi tình hình thực tế của vườn rau một cách nhanh chóng và chính xác.

Tại thời điểm được minh họa trong hình, hệ thống đang hoạt động ở chế độ tự động. Có nghĩa là các thiết bị điều khiển (như bơm nước, đèn chiếu sáng, quạt gió,...) sẽ được kích hoạt dựa trên ngưỡng giá trị cảm biến đã cài đặt sẵn, giúp tiết kiệm công sức giám sát và can thiệp thủ công từ người dùng.



Hình 3.2 : Màn hình lựa chọn loại cây trồng với được cài đặt

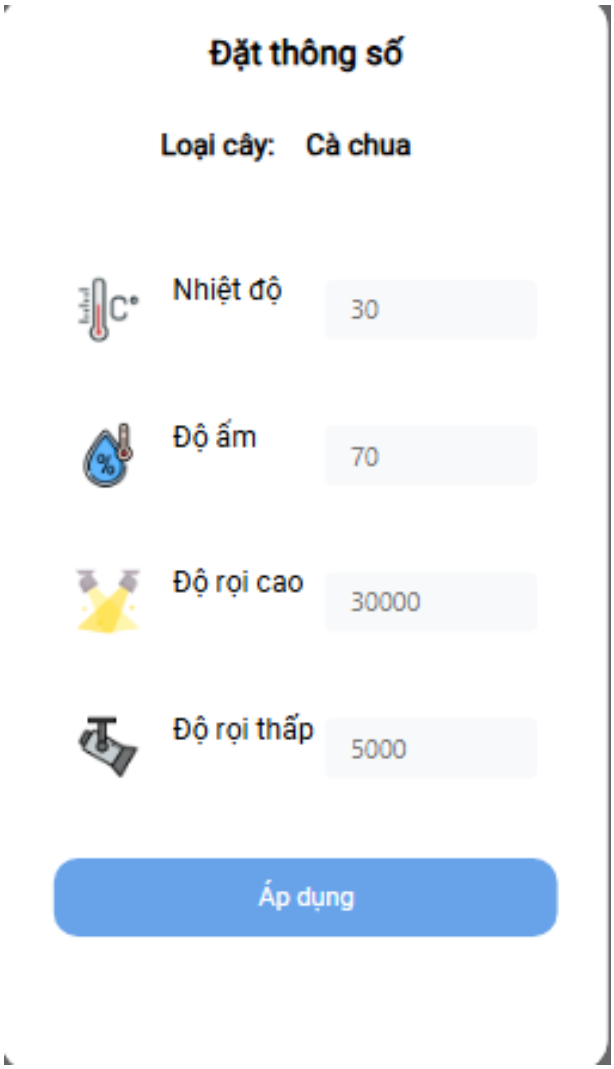
Màn hình này sẽ được mở ra ngay sau khi người dùng nhấn vào nút chọn loại cây trong giao diện chính của ứng dụng. Tại đây, người dùng có thể dễ dàng lựa chọn một trong các loại cây đã được hệ thống định sẵn. Mỗi loại cây đều đi kèm với các thông số môi trường tối ưu như: nhiệt độ, độ ẩm đất, độ rọi ánh sáng... Các thông số này được thiết lập mặc định theo khuyến nghị, tuy nhiên người dùng hoàn toàn có thể tùy chỉnh lại để phù hợp với điều kiện thực tế hoặc nhu cầu riêng.

Giao diện cung cấp 4 loại cây phổ biến đã được tích hợp để người dùng lựa chọn nhanh chóng. Ngoài ra, hệ thống hỗ trợ hai chức năng chính:

- **Nút "Áp dụng"**: Dùng để kích hoạt ngay thông số mặc định của loại cây đã chọn, áp dụng cho chế độ tự động vận hành hệ thống.

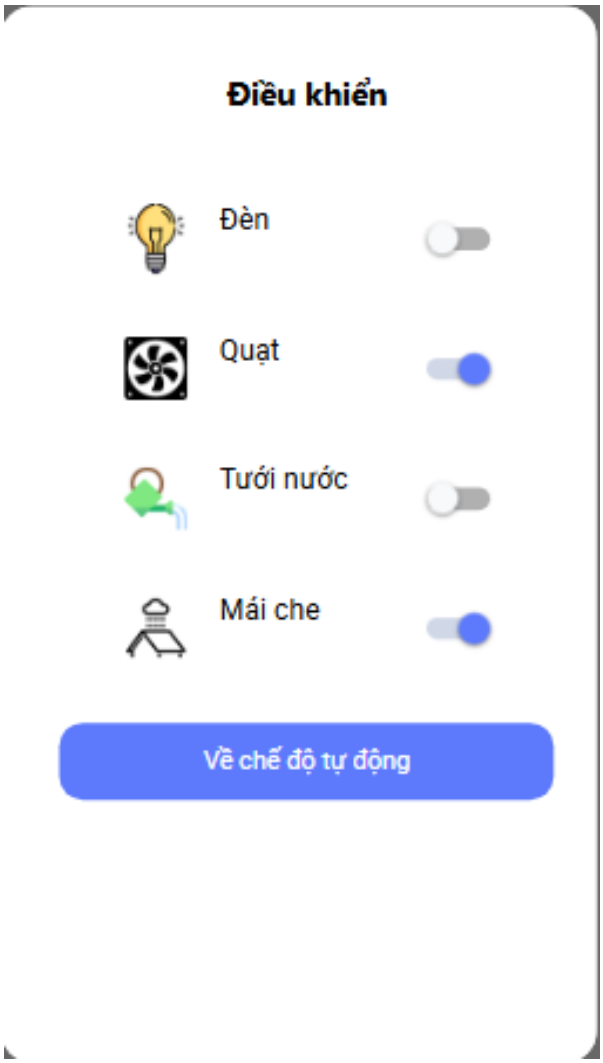
- **Nút "Chỉnh thông số"**: Cho phép người dùng vào phần cài đặt chi tiết để điều chỉnh lại các thông số như nhiệt độ, độ ẩm, độ sáng... của loại cây đã chọn một cách linh hoạt.

Chức năng này không chỉ giúp việc chăm sóc cây trồng trở nên dễ dàng hơn mà còn phù hợp với nhiều nhu cầu khác nhau, từ người mới bắt đầu đến người dùng có kinh nghiệm.



Hình 3.3: Màn hình cài đặt thông số cho các loại rau

Màn hình hiển thị sau khi người dùng nhấn vào chỉnh thông số. Ở đây hiển thị thông số đã được cài đặt trước đó để dễ quan sát. Người dùng có thể thay đổi và ấn vào áp dụng để lưu thông số.



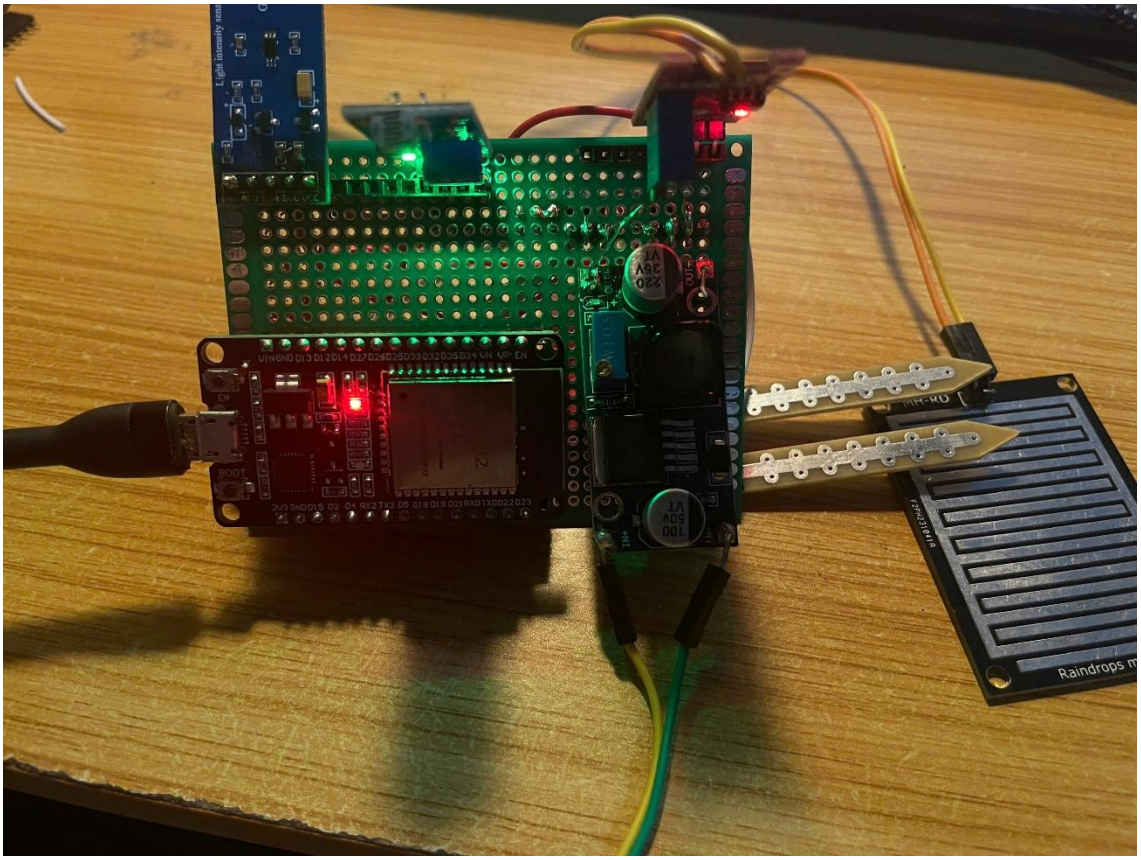
Hình 3.4: Màn hình điều khiển thủ công các thiết bị

Màn hình mở ra khi ấn vào nút chế độ bằng tay. Ở đây người dùng có thể bật tắt các chức năng như đèn, quạt, tưới nước cho cây, mái che.

Phần mềm điều khiển được thiết kế trên nền tảng Thunkable với giao diện trực quan, thân thiện, hỗ trợ các chức năng:

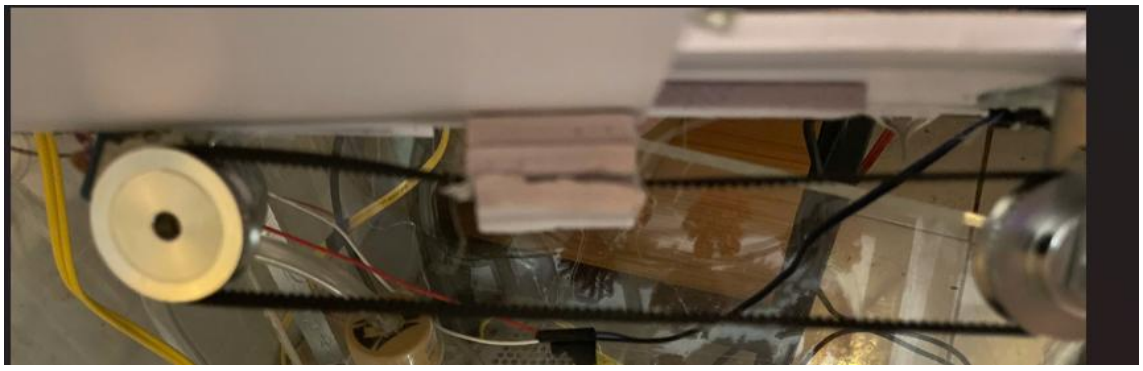
- Hiển thị thông tin cảm biến: nhiệt độ, độ ẩm, cường độ ánh sáng và tình trạng hệ thống.
- Lựa chọn loại cây trồng và cài đặt thông số môi trường phù hợp.
- Điều khiển thủ công: bật/tắt đèn, bơm nước, quạt thông gió.

3.2 Thiết kế Board mạch

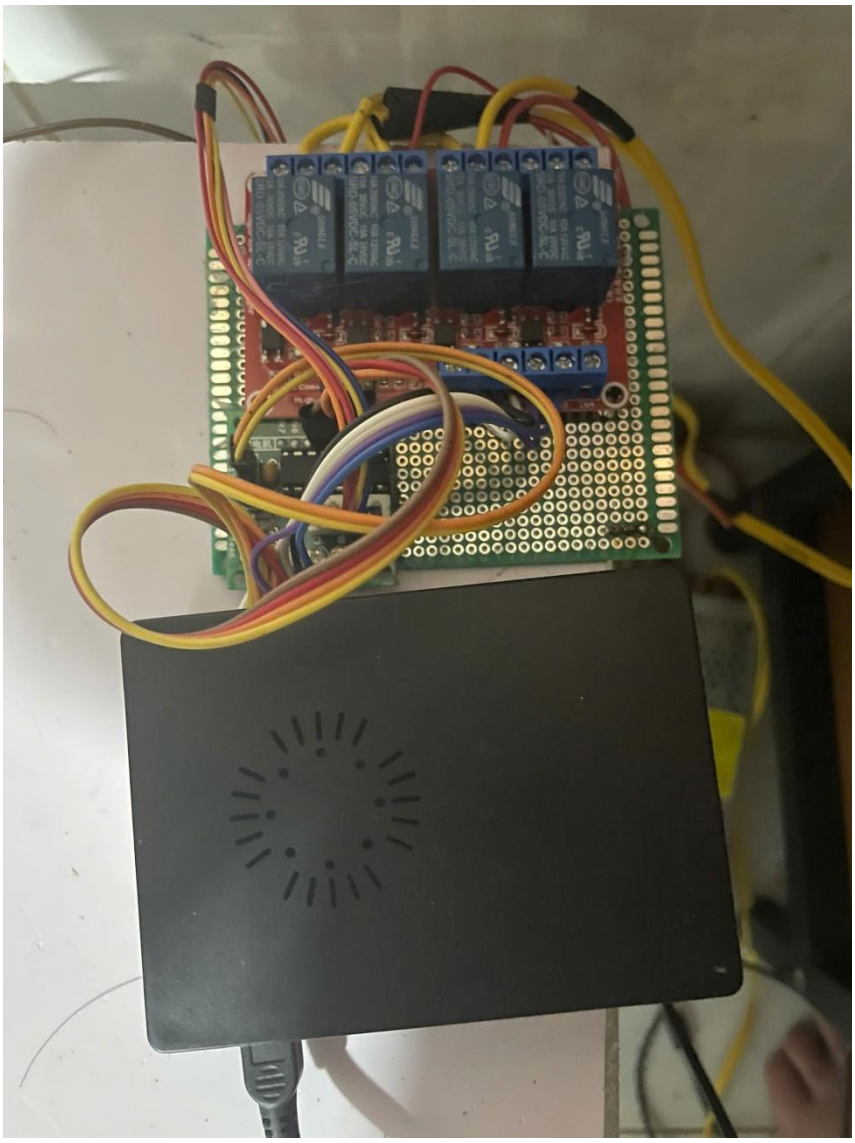


Hình 3.5: Board thu thập dữ liệu từ cảm biến sử dụng Esp32

Trên Board bao gồm: Esp32 là MCU chính và các ngoại vi bao gồm các cảm biến: ánh sáng (lux), cảm biến mưa, cảm biến ánh sáng (quang trở), cảm biến nhiệt độ độ ẩm không khí (DHT11), cảm biến độ ẩm đất và mạch hạ điện áp 12v-5v.



Hình 3.6 Bộ điều khiển cửa thông qua mô tơ step, puly và dây đai truyền động



Hình 3.7: Raspberry Pi3b+ điều khiển relay và driver động cơ step

Trong hình 3.7, mô hình sử dụng mạch điều khiển trung tâm là Raspberry Pi 3B+ để điều khiển các thiết bị phụ trợ thông qua mạch relay và mạch driver động cơ bước. Cụ thể, Raspberry Pi kết nối với một board mạch mở rộng, trong đó bao gồm:

- **Mạch driver điều khiển động cơ bước**: Được sử dụng để điều khiển cơ cấu đóng mở cửa tự động (như cửa che mưa hoặc thông gió).
- **4 relay**: Có vai trò như các công tắc điện tử, relay này cho phép Raspberry Pi bật/tắt tự động các thiết bị điện như: hệ thống đèn chiếu sáng, quạt thông gió, và bơm nước.

Hệ thống kết hợp giữa Raspberry Pi và các linh kiện phụ trợ này giúp việc điều khiển trở nên chính xác, linh hoạt và hoàn toàn có thể tự động hóa theo chương trình đã lập trình sẵn.

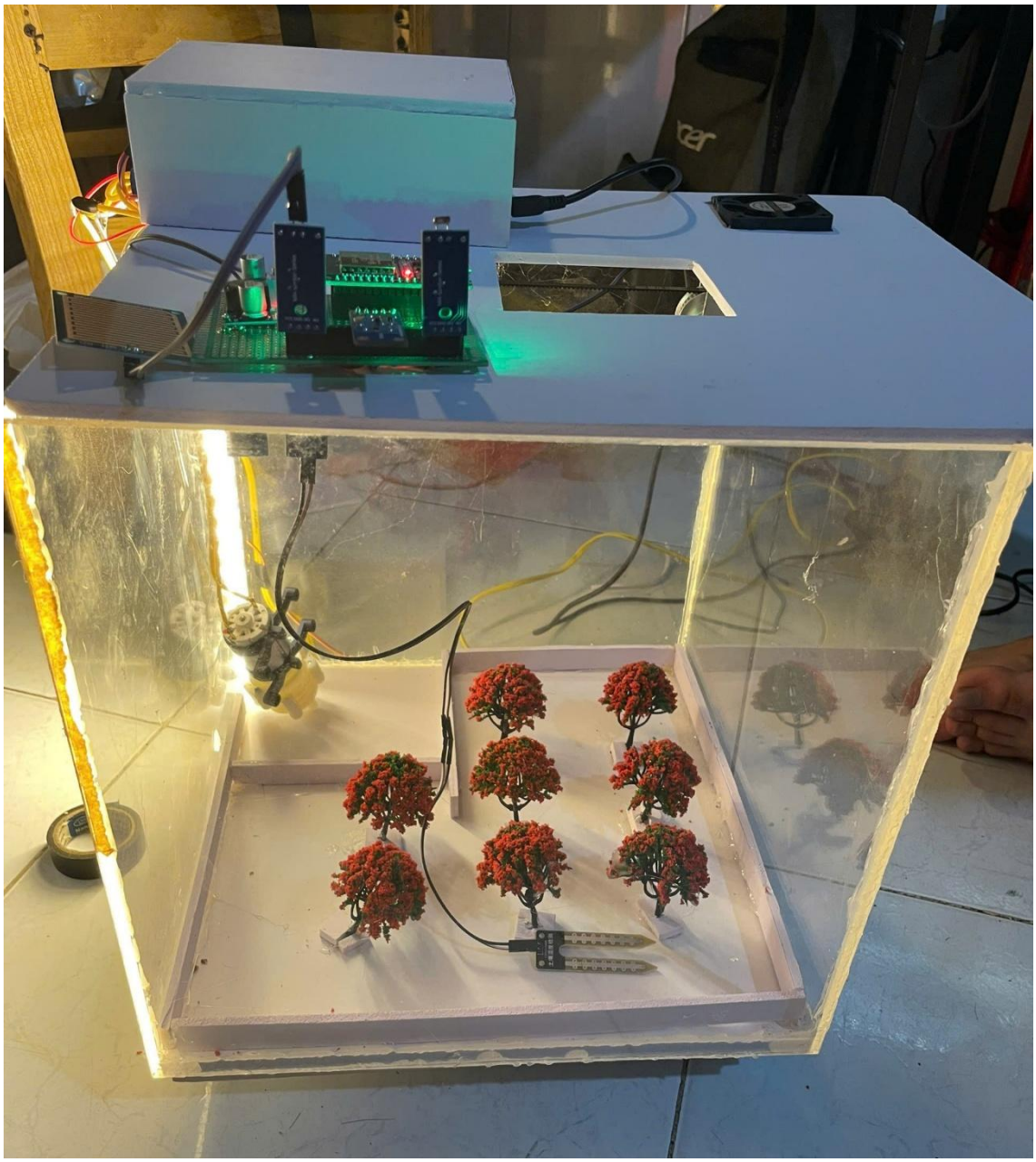
Tổng quan mô hình thực tế:

Hình 3.8 Tổng quan mô hình nhìn từ trên xuống

Hình ảnh mô tả tổng thể bố trí các thành phần chính trong hệ thống khi nhìn từ trên xuống. Các bộ phận được sắp xếp khoa học, đảm bảo hoạt động hiệu quả và thuận tiện trong việc giám sát, điều khiển. Cụ thể như sau:

- **Phần cửa tự động**: Được bố trí ở phía trên, có chức năng đóng/mở dựa trên tín hiệu điều khiển thủ công từ người dùng hoặc tự động dựa vào các giá trị cảm biến môi trường.
- **Khu vực bên trái**: Là nơi đặt **khối cảm biến**, bao gồm mạch ESP32 và các cảm biến như nhiệt độ, độ ẩm, ánh sáng, mưa, độ ẩm đất,... Những dữ liệu thu thập được từ các cảm biến sẽ được xử lý và chuyển về cơ sở dữ liệu để phân tích.
- **Khu vực bên phải**: Chứa **khối chấp hành**, gồm Raspberry Pi 3B+ kết hợp với các relay để điều khiển đèn, quạt, bơm nước và động cơ đóng mở cửa.
- **Phía dưới**: Là **khối nguồn**, bao gồm các thiết bị cung cấp năng lượng cho toàn bộ hệ thống, đảm bảo hoạt động liên tục và ổn định.

- **Hệ thống quạt thông gió**: Được tích hợp để điều hòa nhiệt độ bên trong khu vực trồng cây, tránh tình trạng quá nóng hoặc thiếu lưu thông không khí.



Hình 3.9 Tổng quan mô hình nhìn từ bên hông

- Board thu thập dữ liệu: Sử dụng ESP32 làm vi điều khiển chính để kết nối các cảm biến đo lường như ánh sáng (lux), cảm biến nhiệt độ và độ ẩm không khí (DHT11), cảm biến độ ẩm đất, cảm biến mưa và module hạ áp 12V-5V.
- Board điều khiển: Raspberry Pi 3 Model B+ kết nối với relay 4 kênh để bật/tắt đèn, quạt, máy bơm nước và driver động cơ bước.

3.3 Chu trình hoạt động của hệ thống

Hệ thống IoT giám sát trồng rau trong hộ gia đình hoạt động theo quy trình tự động hóa và bán tự động:

3.3.1 Thu thập dữ liệu

- Các cảm biến đo các thông số môi trường như nhiệt độ, độ ẩm, ánh sáng và độ ẩm đất. Các tín hiệu này được gửi về bộ điều khiển ESP32 qua giao thức I2C hoặc tín hiệu số/analog.
- Dữ liệu được xử lý và gửi đến Firebase Realtime Database để lưu trữ và đồng bộ.

3.3.2 Xử lý và điều khiển

- Điều khiển tự động: Dựa trên dữ liệu cảm biến và các ngưỡng giá trị thiết lập trước, hệ thống tự động kích hoạt các thiết bị:
    - Đèn LED bật khi cường độ ánh sáng thấp hơn ngưỡng tối thiểu.
    - Máy bơm nước hoạt động khi độ ẩm đất giảm dưới ngưỡng cài đặt.
    - Quạt thông gió bật khi nhiệt độ vượt ngưỡng tối đa.
    - Màn chắn tự động đóng/mở dựa trên mức độ ánh sáng để bảo vệ cây trồng.
- Điều khiển thủ công: Người dùng có thể can thiệp trực tiếp qua ứng dụng di động để điều khiển màn chắn, bật/tắt đèn, quạt hoặc máy bơm.

3.3.3 Giám sát và điều khiển từ xa

- Dữ liệu thời gian thực được đồng bộ lên Firebase, cho phép người dùng theo dõi tình trạng hệ thống trên ứng dụng di động.
- Các cài đặt ngưỡng có thể điều chỉnh thông qua giao diện ứng dụng để tối ưu hóa điều kiện trồng cây.

CHƯƠNG 4: KẾT LUẬN

4.1 Kết quả đạt được

Đề tài đã hoàn thiện mô hình IoT giám sát trồng rau tại hộ gia đình với các kết quả sau:

- Xây dựng thành công hệ thống phần cứng gồm cảm biến (độ ẩm đất, nhiệt độ, ánh sáng, mưa...) và các thiết bị chấp hành (bơm, quạt, đèn...).
- Ứng dụng điều khiển trên điện thoại được thiết kế bằng Thunkable, giao diện thân thiện, hỗ trợ giám sát và điều khiển từ xa.
- Dữ liệu được đồng bộ theo thời gian thực qua Firebase, giúp hệ thống tự động điều chỉnh môi trường trồng rau hiệu quả.
- Mô hình hoạt động ổn định, dễ mở rộng và phù hợp với nhu cầu trồng rau sạch trong đô thị.

4.2 Phạm vi ứng dụng của kết quả đề tài

Kết quả của đề tài có thể được ứng dụng rộng rãi trong các lĩnh vực sau:

- Hộ gia đình đô thị: Hỗ trợ người dân trồng rau sạch tại nhà một cách tự động, tiện lợi và tiết kiệm thời gian.
- Mô hình giáo dục: Là tài liệu tham khảo và thực hành cho sinh viên trong lĩnh vực IoT, hệ thống nhúng và tự động hóa.
- Nông nghiệp quy mô nhỏ: Có thể mở rộng để áp dụng trong vườn rau, nhà kính, hoặc trang trại thông minh.
- Các dự án khởi nghiệp: Là nền tảng phát triển sản phẩm thương mại hướng đến thị trường nông nghiệp thông minh, nhà thông minh.

4.3 Hạn chế và hướng phát triển của đề tài

Hạn chế:

- Mô hình hiện tại chỉ áp dụng cho quy mô nhỏ, chưa phù hợp với hệ thống trồng trọt diện rộng.
- Tính năng tự động còn đơn giản, chủ yếu dựa trên ngưỡng cố định, chưa có thuật toán học máy.
- Giao diện ứng dụng còn hạn chế về mặt trực quan và chưa hỗ trợ đa người dùng.

Hướng phát triển:

- Tích hợp thêm cảm biến đo pH, dinh dưỡng đất, khí CO_2... để theo dõi môi trường đầy đủ hơn.
- Phát triển thuật toán thông minh dựa trên học máy (AI) để tối ưu việc chăm sóc cây.
- Cải tiến ứng dụng di động với giao diện tốt hơn, bổ sung tính năng cảnh báo và điều khiển bằng giọng nói.
- Mở rộng quy mô hệ thống cho nhà kính, vườn canh tác nhỏ và hợp tác xã nông nghiệp.

Tài liệu tham khảo

[1] Google, “Firebase Documentation,” Firebase, 2024. [Online]. Available: https://firebase.google.com/docs.

[2] Thunkable Team, “Connect a Firebase Realtime Database,” *Thunkable Documentation*, 2024. [Online]. Available:

[3] Kênh Khám Phá Công Nghệ, “@kapakhamphacongnghe7974,” *YouTube Channel*, YouTube, 2025. [Online].

[4] ETC2 (List of Unclassified Manufacturers), “DHT11 Humidity & Temperature Sensor Datasheet,” AllDataSheet.

[5] ROHM Co., Ltd., “BH1750FVI Digital 16-bit Serial Output Ambient Light Sensor IC (Rev.D, Nov. 2011),” Mouser Electronic.

[6] STMicroelectronics, “ULN2003 Darlington Array Datasheet,” AllDataSheet.

[7] Raspberry Pi Ltd, “Raspberry Pi 3 Model B+ Product Brief,” Raspberry Pi Ltd, Jan. 2025.

[8] Thunkable Inc., “Thunkable Official Website,” Thunkable Inc., Jul. 2025. [Online]. Available: https://thunkable.com/.

[9] Espressif Systems, “ESP32 Series Datasheet,” *Espressif Systems*, ver. 4.9, 2020. [Online].https://www.espressif.com/sites/default/files/documentation/esp32datasheet_en.pdf

[10] Nguyễn Hữu Phước, “Cách viết và chạy chương trình Python trên Raspberry,” *Điện Tử Việt*, ngày 24 tháng 2, 2021. [Online]. Available: https://dientuviet.com/cach-viet-va-chay-chuong-trinh-python-tren-raspberry/

Phụ lục

A.Code cho Esp32

```
#include <Wire.h>
#include <BH1750.h>
#include <Arduino.h>
#include "DHT.h"
#include "WiFi.h"
#include "FirebaseESP32.h"
#include <ArduinoJson.h>

// Cấu hình cảm biến và Wi-Fi0
#define DHTPIN 4
#define DHTTYPE DHT11
#define WIFI_SSID "LAPTOP"
#define WIFI_PASSWORD "20012002"

// Cấu hình Firebase
#define FIREBASE_HOST "https://iot-vuonrau-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "sVND1E8RPBOhjj52fHnMok25BKHABxU3iF1EiHgS"

// Khởi tạo các đối tượng
DHT dht(DHTPIN, DHTTYPE);
BH1750 lightMeter;
FirebaseData firebaseData;
FirebaseConfig config;
FirebaseAuth auth;

// Biến lưu trữ dữ liệu
float value_l, real_value_l;
float value_r, real_value_r, real_value_m;
float rain, light, moisture;
```

```
// Biến thời gian và đường dẫn Firebase
unsigned long t1 = 0;
String path = "/";

void setup() {
Serial.begin(9600);

// Khởi tạo I2C và cảm biến ánh sáng
Wire.begin();
lightMeter.begin();
dht.begin();

// Kết nối Wi-Fi
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(300);
}
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());

// Cấu hình Firebase
config.host = FIREBASE_HOST;
config.signer.tokens.legacy_token = FIREBASE_AUTH;

// Kết nối Firebase
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
```

```
        Serial.println("Firebase initialized successfully.");
}

void loop() {
        // Đọc dữ liệu cảm biến ánh sáng
        float lux = lightMeter.readLightLevel();
        Serial.print("Light: ");
        Serial.print(lux);
        Serial.println(" lx");

        // Đọc dữ liệu nhiệt độ và độ ẩm từ DHT11
        float h = dht.readHumidity();
        float t = dht.readTemperature();
        Serial.print("Humidity: ");
        Serial.print(h);
        Serial.println(" %");
        Serial.print("Temperature: ");
        Serial.print(t);
        Serial.println(" *C");

        // Đọc dữ liệu từ cảm biến khác
        real_value_m = analogRead(19);
        moisture = 100 - (real_value_m / 4095) * 100;

        real_value_l = analogRead(35);
        light = 100 - (real_value_l / 4095) * 100;

        real_value_r = analogRead(32);
        rain = 100 - (real_value_r / 4095) * 100;

        Serial.print("Rain sensor: ");
        Serial.println(rain);
```

```
        Serial.print("Light sensor: ");
        Serial.println(light);
        Serial.print("Moisture sensor: ");
        Serial.println(moisture);

        // Gửi dữ liệu lên Firebase mỗi 500ms
        if (millis() - t1 > 500) {
        float light1 = round(light * 100) / 100;
        float rain1 = round(rain * 100) / 100;
        float moisture1 = round(moisture * 100) / 100;

        Firebase.setFloat(firebaseData, path + "/light", light1);
        Firebase.setFloat(firebaseData, path + "/Humidity", h);
        Firebase.setFloat(firebaseData, path + "/rain", rain1);
        Firebase.setFloat(firebaseData, path + "/Temperature", t);
        Firebase.setFloat(firebaseData, path + "/illu", lux);
        Firebase.setFloat(firebaseData, path + "/moisture", moisture1);



        t1 = millis();
        }
}
```

B. Code cho Raspberry Pi

```
import pyrebase
import RPi.GPIO as gpio
from time import sleep

# ---------------- CẤU HÌNH FIREBASE ----------------
config = {
```

```
    "apiKey": "AIzaSyDh7evzo18VgSTXMQOBQmvPOpOIO87Gs2A",
    "authDomain": "iot-vuonrau.firebaseapp.com",
    "databaseURL": "https://iot-vuonrau-default-rtdb.firebaseio.com",
    "projectId": "iot-vuonrau",
    "storageBucket": "iot-vuonrau.firebasestorage.app",
    "messagingSenderId": "786889359852",
    "appId": "1:786889359852:web:9adb2a628c73a307fec0c9",
    "measurementId": "G-Z14B2QP8K9"
}

# ----------------- KHAI BÁO CHÂN GPIO -----------------
MOTOR_PINS = [17, 18, 27, 22]  # IN1, IN2, IN3, IN4
LIGHT = 5     # Đèn
FAN = 23      # Quạt
PUMP = 24     # Máy bơm

# -- --------------- KHỞI TẠO GPIO -----------------
gpio.setmode(gpio.BCM)
for pin in MOTOR_PINS:
    gpio.setup(pin, gpio.OUT)
    gpio.output(pin, 0)

gpio.setup(LIGHT, gpio.OUT)
gpio.setup(PUMP, gpio.OUT)
gpio.setup(FAN, gpio.OUT)

# ----------------- KHỞI TẠO FIREBASE -----------------
firebase = pyrebase.initialize_app(config)
db = firebase.database()

# ----------------- MA TRẬN NỬA BƯỚC -----------------
halfstep_seq = [
```

```
    [1,0,0,0],
    [1,1,0,0],
    [0,1,0,0],
    [0,1,1,0],
    [0,0,1,0],
    [0,0,1,1],
    [0,0,0,1],
    [1,0,0,1]
]

# ---------------- HÀM TIỆN ÍCH ----------------
def get_float_value(node):
    try:
        return float(db.child(node).get().val())
    except (ValueError, TypeError):
        print(f"[LỖI] Không thể chuyển {node} thành float.")
        return 0.0

def get_int_value(node):
    try:
        return int(db.child(node).get().val())
    except (ValueError, TypeError):
        print(f"[LỖI] Không thể chuyển {node} thành int.")
        return 0

# ---------------- HÀM ĐIỀU KHIỂN ----------------
def move_motor(direction, steps):
    """Điều khiển động cơ bước 28BYJ-48 với ULN2003"""
    if direction == 0:
        seq = halfstep_seq
    else:
        seq = list(reversed(halfstep_seq))
```

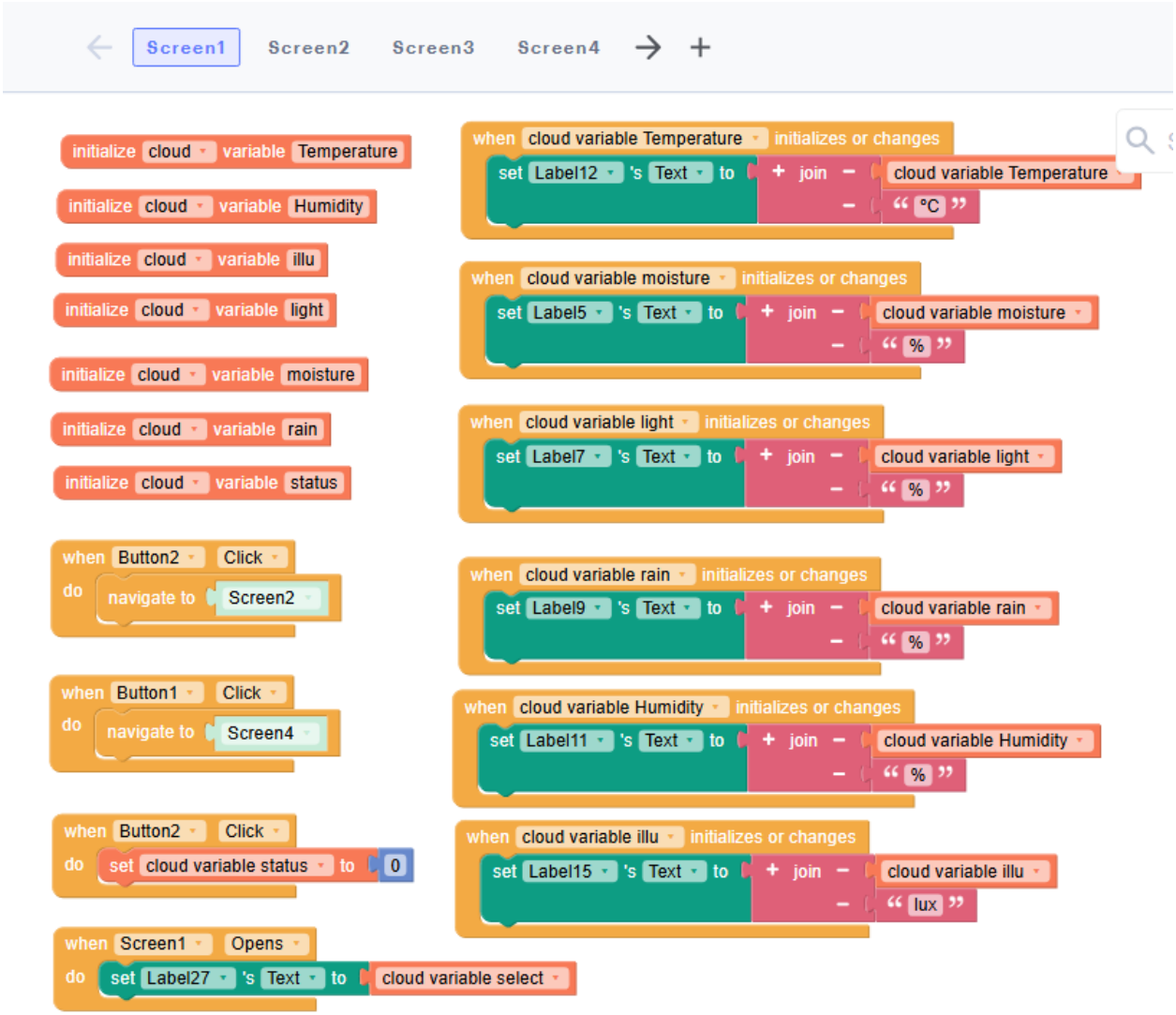
```
    for _ in range(steps):
        for halfstep in seq:
            for pin in range(4):
                gpio.output(MOTOR_PINS[pin], halfstep[pin])
            sleep(0.001)

def control_light():
    light_val = get_int_value("den")
    gpio.output(LIGHT, light_val)
    print("Đèn:", "Bật" if light_val else "Tắt")

def control_fan():
    fan_val = get_int_value("fan")
    gpio.output(FAN, fan_val)
    print("Quạt:", "Bật" if fan_val else "Tắt")

def control_pump():
    pump_val = get_int_value("pump")
    gpio.output(PUMP, pump_val)
    print("Bơm:", "Bật" if pump_val else "Tắt")

def manual_shield():
    desired_state = get_int_value("manchan")   # 1: mở, 0: đóng
    current_state = get_int_value("vtmc")
    if desired_state == 1 and current_state == 0:
        move_motor(0, 512)  # quay ngược (mở)
        db.update({"vtmc": 1})
        print("Màn chắn mở (thủ công)")
    elif desired_state == 0 and current_state == 1:
        move_motor(1, 512)  # quay thuận (đóng)
        db.update({"vtmc": 0})
```

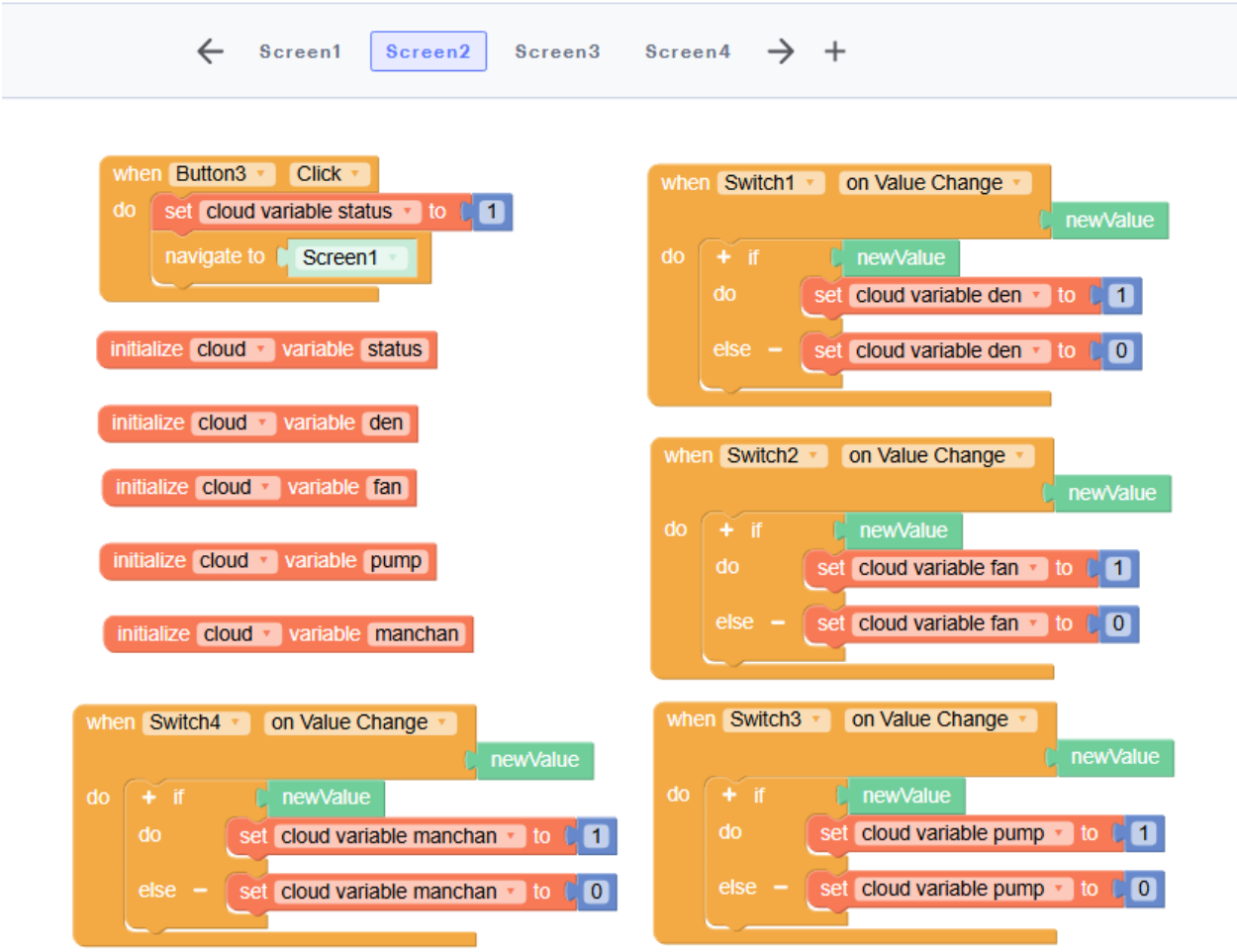
```
        print("Màn chắn đóng (thủ công)")

# ---------------- CHẾ ĐỘ TỰ ĐỘNG ----------------
def auto_light():
    current_light = get_float_value("illu")
    light_threshold = get_float_value("illu_set_low")
    if current_light < light_threshold:
        gpio.output(LIGHT, 1)
        db.update({"den": 1})
        print("Đèn tự động bật")
    else:
        gpio.output(LIGHT, 0)
        db.update({"den": 0})
        print("Đèn tự động tắt")

def auto_fan():
    current_temp = get_float_value("Temperature")
    temp_threshold = get_float_value("temp_set")
    if current_temp > temp_threshold:
        gpio.output(FAN, 1)
        db.update({"fan": 1})
        print("Quạt tự động bật")
    else:
        gpio.output(FAN, 0)
        db.update({"fan": 0})
        print("Quạt tự động tắt")

def auto_pump():
    soil_moisture = get_float_value("moisture")
    moisture_threshold = get_float_value("moisture_set")
    if soil_moisture < moisture_threshold:
        gpio.output(PUMP, 1)
```

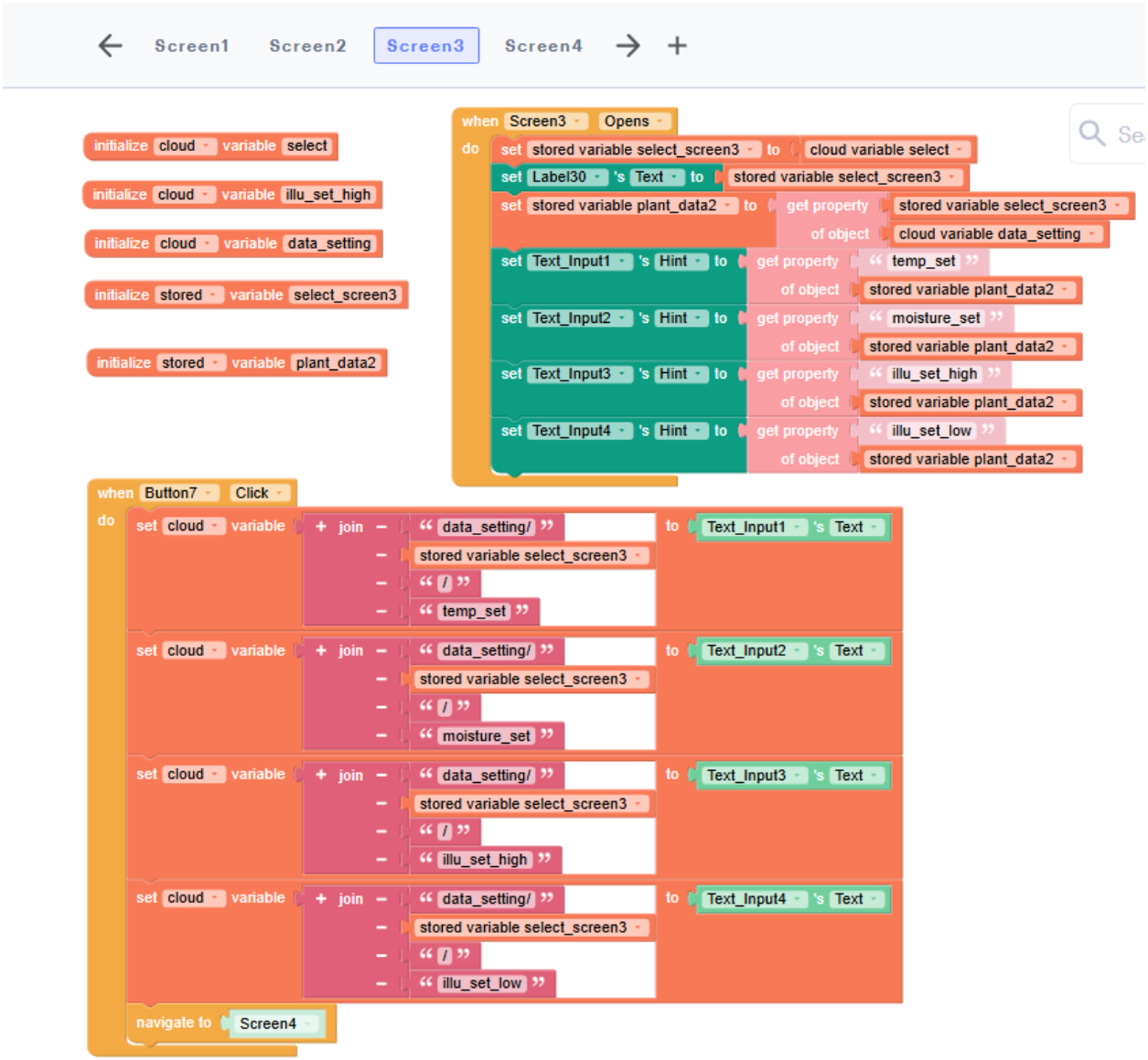
```
        db.update({"pump": 1})
        print("Bơm tự động bật")
    else:
        gpio.output(PUMP, 0)
        db.update({"pump": 0})
        print("Bơm tự động tắt")

def auto_shield():
    current_light = get_float_value("illu")
    light_high_threshold = get_float_value("illu_set_high")
    vtmc_val = get_int_value("vtmc")
    if current_light > light_high_threshold and vtmc_val == 0:
        move_motor(0, 512)
        db.update({"vtmc": 1})
        print("Màn chắn tự động mở")
    elif current_light <= light_high_threshold and vtmc_val == 1:
        move_motor(1, 4096)
        db.update({"vtmc": 0})
        print("Màn chắn tự động đóng")

# ----------------- CHƯƠNG TRÌNH CHÍNH -----------------
if __name__ == "__main__":
    print("Khởi động hệ thống IoT")
    try:
        while True:
            status = get_int_value("status")  # 0: thủ công, 1: tự động
            if status == 0:
                print("Chế độ THỦ CÔNG")
                control_light()
                control_fan()
                control_pump()
                manual_shield()
```

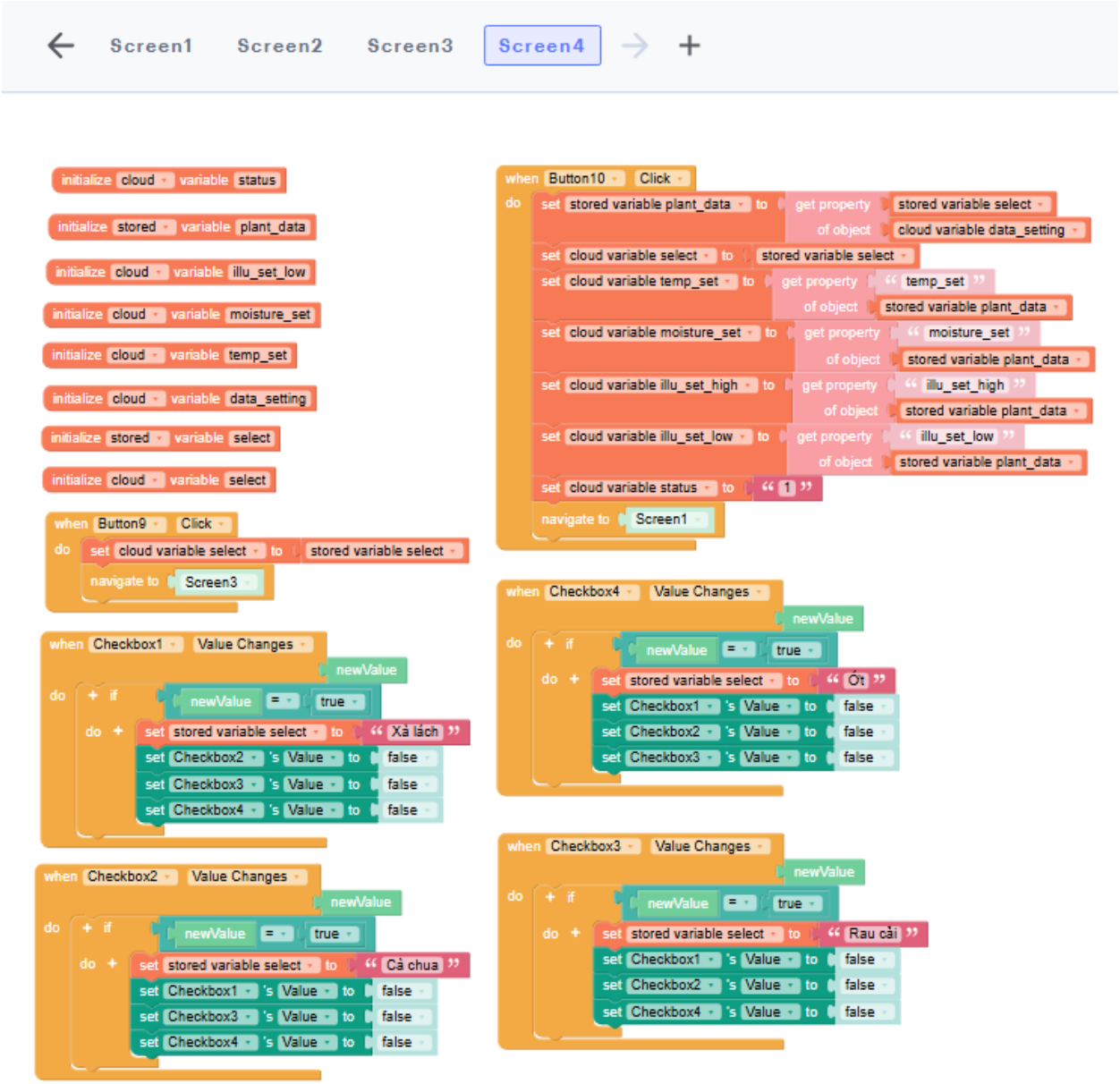
```
        elif status == 1:
            print("Chế độ TỰ ĐỘNG")
            auto_light()
            auto_fan()
            auto_pump()
            auto_shield()
        sleep(1)
    except KeyboardInterrupt:
        print("Dừng hệ thống, dọn dẹp GPIO")
        gpio.cleanup()
```

C. Code Thunkable


Screen1
Screen2
Screen3
Screen4
initialize cloud variable Temperature
initialize cloud variable Humidity
initialize cloud variable illu
initialize cloud variable light
initialize cloud variable moisture
initialize cloud variable rain
initialize cloud variable status
when Button2 Click
do navigate to Screen2
when Button1 Click
do navigate to Screen4
when Button2 Click
do set cloud variable status to 0
when Screen1 Opens
do set Label27 's Text to cloud variable select
when cloud variable Temperature initializes or changes
set Label12 's Text to join cloud variable Temperature
"°C"
when cloud variable moisture initializes or changes
set Label5 's Text to join cloud variable moisture
"%"
when cloud variable light initializes or changes
set Label7 's Text to join cloud variable light
"%"
when cloud variable rain initializes or changes
set Label9 's Text to join cloud variable rain
"%"
when cloud variable Humidity initializes or changes
set Label11 's Text to join cloud variable Humidity
"%"
when cloud variable illu initializes or changes
set Label15 's Text to join cloud variable illu
"lux"


Screen1
Screen2
Screen3
Screen4
when Button3 Click
do set cloud variable status to 1
navigate to Screen1
initialize cloud variable status
initialize cloud variable den
initialize cloud variable fan
initialize cloud variable pump
initialize cloud variable manchan
when Switch1 on Value Change
newValue
do if newValue
do set cloud variable den to 1
else set cloud variable den to 0
when Switch2 on Value Change
newValue
do if newValue
do set cloud variable fan to 1
else set cloud variable fan to 0
when Switch4 on Value Change
newValue
do if newValue
do set cloud variable manchan to 1
else set cloud variable manchan to 0
when Switch3 on Value Change
newValue
do if newValue
do set cloud variable pump to 1
else set cloud variable pump to 0


Screen1
Screen2
Screen3
Screen4
initialize cloud variable select
initialize cloud variable illu_set_high
initialize cloud variable data_setting
initialize stored variable select_screen3
initialize stored variable plant_data2
when Screen3 Opens
do
set stored variable select_screen3 to cloud variable select
set Label30 's Text to stored variable select_screen3
set stored variable plant_data2 to get property stored variable select_screen3
of object cloud variable data_setting
set Text_Input1 's Hint to get property "temp_set"
of object stored variable plant_data2
set Text_Input2 's Hint to get property "moisture_set"
of object stored variable plant_data2
set Text_Input3 's Hint to get property "illu_set_high"
of object stored variable plant_data2
set Text_Input4 's Hint to get property "illu_set_low"
of object stored variable plant_data2
when Button7 Click
do
set cloud variable join "data_setting/" stored variable select_screen3 "/" "temp_set" to Text_Input1 's Text
set cloud variable join "data_setting/" stored variable select_screen3 "/" "moisture_set" to Text_Input2 's Text
set cloud variable join "data_setting/" stored variable select_screen3 "/" "illu_set_high" to Text_Input3 's Text
set cloud variable join "data_setting/" stored variable select_screen3 "/" "illu_set_low" to Text_Input4 's Text
navigate to Screen4


Screen1
Screen2
Screen3
Screen4
initialize cloud variable status
initialize stored variable plant_data
initialize cloud variable illu_set_low
initialize cloud variable moisture_set
initialize cloud variable temp_set
initialize cloud variable data_setting
initialize stored variable select
initialize cloud variable select
when Button9 Click
do set cloud variable select to stored variable select
navigate to Screen3
when Checkbox1 Value Changes
newValue
do if newValue = true
do set stored variable select to "Xà lách"
set Checkbox2 's Value to false
set Checkbox3 's Value to false
set Checkbox4 's Value to false
when Checkbox2 Value Changes
newValue
do if newValue = true
do set stored variable select to "Cà chua"
set Checkbox1 's Value to false
set Checkbox3 's Value to false
set Checkbox4 's Value to false
when Button10 Click
do set stored variable plant_data to get property stored variable select
of object cloud variable data_setting
set cloud variable select to stored variable select
set cloud variable temp_set to get property "temp_set"
of object stored variable plant_data
set cloud variable moisture_set to get property "moisture_set"
of object stored variable plant_data
set cloud variable illu_set_high to get property "illu_set_high"
of object stored variable plant_data
set cloud variable illu_set_low to get property "illu_set_low"
of object stored variable plant_data
set cloud variable status to "1"
navigate to Screen1
when Checkbox4 Value Changes
newValue
do if newValue = true
do set stored variable select to "Ớt"
set Checkbox1 's Value to false
set Checkbox2 's Value to false
set Checkbox3 's Value to false
when Checkbox3 Value Changes
newValue
do if newValue = true
do set stored variable select to "Rau cải"
set Checkbox1 's Value to false
set Checkbox2 's Value to false
set Checkbox4 's Value to false