

**ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH**  
**ĐẠI HỌC BÁCH KHOA**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN: Lập trình hệ thống nhúng**

**Đề tài : Chạy đo giám sát tình trạng cây trồng**

**Giảng viên hướng dẫn:** Nguyễn Phan Hải Phú

**Sinh viên thực hiện:**

Số thứ tự	Họ và tên	MSSV	Đóng góp	Hoàn thành (%)
1	Nguyễn Anh Tuấn	2115174	Giao diện người dùng (Web Server), vẽ Lưu đồ giải thuật	100%
2	Nguyễn Hoàng Phú Lộc	2113964	Loadcell + Máy Bơm, vẽ Máy trạng thái, Word	100%
3	Phan Tuấn Nguyên	2114237	Cảm biến (Cảm biến độ ẩm đất, DHT22, Cảm biến ánh sáng)	100%

Thành phố Hồ Chí Minh 11/2024

## LỜI CẢM ƠN

---

Trước hết, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy Nguyễn Phan Hải Phú đã luôn tận tình giúp đỡ, định hướng và cung cấp những kiến thức quý báu trong suốt quá trình thực hiện Đề tài. Nhờ sự hỗ trợ, giảng dạy và hướng dẫn của thầy, chúng em đã có thể hoàn thành tốt đề tài “Châu đo giám sát tình trạng câu trồng”

Chúng em cũng xin gửi lời cảm ơn đến ban giám hiệu, các thầy cô trong khoa đã tạo điều kiện thuận lợi cho tôi trong suốt quá trình học tập và nghiên cứu. Những kiến thức mà tôi được học từ các thầy cô là nền tảng quan trọng giúp tôi thực hiện đề tài này.

Cuối cùng, xin chân thành cảm ơn gia đình và bạn bè đã luôn động viên, ủng hộ tinh thần để chúng em vượt qua những khó khăn trong quá trình thực hiện đề tài này. Nhờ sự đồng hành của mọi người, chúng em đã có thêm động lực để hoàn thành công việc một cách tốt nhất.

Xin chân thành cảm ơn!

# MỤC LỤC

---

LỜI CẢM ƠN.....	2
MỤC LỤC .....	3
MỤC LỤC HÌNH ẢNH.....	5
LỜI MỞ ĐẦU .....	6
NỘI DUNG .....	7
CHƯƠNG 1 CO SỞ LÝ THUYẾT .....	7
1. Internet of Things (IoT).....	7
1.1. Khái niệm IOT .....	7
1.2. Kiến trúc IOT .....	7
1.3. Ứng dụng IOT trong nông nghiệp.....	8
2. Cảm biến và thiết bị trong hệ thống .....	9
2.1. ESP32C6 .....	9
2.2. Cảm biến Độ ẩm đất .....	11
2.3. Cảm biến Nhiệt độ, Độ ẩm môi trường DHT22 .....	12
2.4. Cảm biến Cường độ ánh sáng GY-30 .....	13
2.5. Loadcell và mạch chuyển đổi ADC 24 bit HX711 .....	14
2.6. Một số linh kiện khác .....	16
3. Lưu đồ giải thuật và máy trạng thái của hệ thống.....	18
3.1. Lưu đồ giải thuật .....	18
3.2. Máy trạng thái .....	20
CHƯƠNG 2 THIẾT KẾ VÀ THỰC NGHIỆM.....	21
1. Thiết kế phần mềm .....	21
1.1. Giao tiếp ngoại vi .....	21
1.2. Giao diện người dùng .....	25
1.3. Setup hàm Main .....	26
2. Thiết kế phần cứng .....	29

CHƯƠNG 3	KẾT LUẬN.....	30
TÀI LIỆU THAM KHẢO.....	31	

# MỤC LỤC HÌNH ẢNH

---

Figure 1: Ảnh minh họa 01 .....	7
Figure 2: Ảnh minh họa 02 .....	8
Figure 3: Sơ đồ chân Esp32 .....	9
Figure 4: Cảm biến độ ẩm đất .....	11
Figure 5: Cảm biến DHT22.....	12
Figure 6: Cảm biến ánh sáng GY-30.....	13
Figure 7: Loadcell.....	14
Figure 8: Sơ đồ kết nối Loadcell 5kg và HX711 .....	15
Figure 9: Mạch cấp nguồn DC-DC .....	16
Figure 10: Bơm mini 5v.....	16
Figure 11: Led .....	17
Figure 12: Mạch công suất mosfet 1 kênh F5305S 20A.....	17
Figure 13: Lưu đồ giải thuật 01 .....	18
Figure 14: Lưu đồ giải thuật 02.....	19
Figure 15: Máy trạng thái Mealy cho Hệ thống giám sát tình trạng cây trồng .....	20

# LỜI MỞ ĐẦU

---

Ngày nay, với sự phát triển nhanh chóng của khoa học công nghệ, các giải pháp ứng dụng Internet of Things (IoT) trong nông nghiệp đang trở thành xu hướng tất yếu để nâng cao hiệu quả sản xuất và tối ưu hóa nguồn tài nguyên. Trong bối cảnh biến đổi khí hậu và nhu cầu lương thực ngày càng gia tăng, việc áp dụng công nghệ thông minh vào giám sát và chăm sóc cây trồng không chỉ mang lại lợi ích kinh tế mà còn góp phần bảo vệ môi trường và duy trì sự bền vững trong sản xuất nông nghiệp.

Dự án "Chân đế giám sát tình trạng cây trồng" là một giải pháp tích hợp IoT, tập trung vào việc giám sát và điều chỉnh môi trường chăm sóc cây trồng một cách tự động và hiệu quả. Hệ thống được thiết kế với các cảm biến hiện đại, bao gồm cảm biến nhiệt độ, độ ẩm đất và cường độ ánh sáng, nhằm cung cấp dữ liệu chính xác về tình trạng cây trồng trong thời gian thực. Ngoài ra, hệ thống còn tích hợp máy bơm nước và loadcell để tự động hóa việc tưới tiêu, đảm bảo cây trồng được cung cấp đủ nước một cách hợp lý.

Bên cạnh đó, hệ thống còn được kết nối với một webserver, cho phép người dùng theo dõi dữ liệu trực tuyến thông qua các thiết bị như điện thoại hoặc máy tính. Điều này không chỉ mang lại sự tiện lợi trong việc quản lý cây trồng từ xa mà còn cung cấp các thông tin hữu ích để người dùng đưa ra các quyết định kịp thời nhằm nâng cao hiệu suất canh tác.

Đề tài này không chỉ có ý nghĩa về mặt ứng dụng thực tiễn mà còn mở ra cơ hội nghiên cứu và phát triển các giải pháp IoT tiên tiến hơn cho nông nghiệp trong tương lai. Với việc tích hợp công nghệ IoT, dự án hứa hẹn sẽ góp phần giải quyết các thách thức hiện tại trong lĩnh vực chăm sóc cây trồng, hướng đến một nền nông nghiệp hiện đại và bền vững.

# NỘI DUNG

## CHƯƠNG 1 CƠ SỞ LÝ THUYẾT

### 1. Internet of Things (IoT)

#### 1.1. Khái niệm IoT

IoT (Internet of Things) hay "Internet vạn vật" là một mạng lưới các thiết bị được kết nối với nhau qua internet, có khả năng thu thập, trao đổi và xử lý dữ liệu mà không cần sự can thiệp trực tiếp của con người. Các thiết bị này có thể là cảm biến, máy móc, điện thoại, xe cộ, hoặc bất kỳ thiết bị thông minh nào khác. Mục tiêu của IoT là tạo ra một hệ thống tự động hóa và tối ưu hóa, mang lại sự tiện lợi, hiệu quả và khả năng kiểm soát tốt hơn trong các lĩnh vực như sản xuất, nông nghiệp, chăm sóc sức khỏe, và cuộc sống hàng ngày.



Figure 1: Ảnh minh họa 01

#### 1.2. Kiến trúc IoT

Kiến trúc của IoT thường được chia thành ba lớp chính: lớp cảm nhận, lớp truyền thông và lớp ứng dụng. Lớp cảm nhận bao gồm các thiết bị phần cứng như cảm biến và bộ điều khiển, đóng vai trò thu thập thông tin từ môi trường hoặc thiết bị. Lớp truyền thông chịu trách nhiệm truyền dữ liệu giữa các thiết bị và hệ thống lưu trữ thông qua các giao thức mạng như Wi-Fi, Bluetooth hoặc mạng di động. Cuối cùng, lớp ứng dụng sử dụng dữ liệu để phân tích, quản lý và hiển thị thông tin qua các nền tảng web, phần mềm hoặc ứng dụng di động, từ đó cung cấp các giải pháp và dịch vụ thông minh.

### 1.3. Ứng dụng IOT trong nông nghiệp

IoT đã và đang tạo ra những thay đổi to lớn trong nhiều lĩnh vực của cuộc sống và công nghiệp. Trong **ngành công nghiệp**, IoT được sử dụng để giám sát và bảo trì thiết bị từ xa, quản lý dây chuyền sản xuất và tối ưu hóa vận hành. Trong **y tế**, các thiết bị IoT giúp theo dõi sức khỏe từ xa, quản lý bệnh nhân và thậm chí hỗ trợ phẫu thuật từ xa. Giao thông thông minh là một lĩnh vực nổi bật khác, với các ứng dụng như hệ thống đỗ xe tự động, quản lý giao thông theo thời gian thực và xe tự lái. Trong đời sống hàng ngày, các thiết bị như loa thông minh, đèn chiếu sáng tự động, hoặc hệ thống quản lý năng lượng gia đình đã mang lại sự tiện nghi và hiệu quả cho con người.



Figure 2: Ảnh minh họa 02

Trong nông nghiệp, IoT cho phép giám sát độ ẩm đất, nhiệt độ, và điều kiện môi trường trong thời gian thực để tự động hóa việc tưới tiêu. Các cảm biến được đặt trong đất có thể đo độ ẩm và truyền dữ liệu về hệ thống trung tâm. Dựa trên dữ liệu này, hệ thống có thể tự động bật/tắt máy bơm nước hoặc điều chỉnh lưu lượng nước tưới, giúp tiết kiệm nước và tối ưu hóa việc chăm sóc cây trồng. Trong chăn nuôi, IoT được sử dụng để theo dõi tình trạng sức khỏe của vật nuôi thông qua các thiết bị đeo hoặc cảm biến. Các thiết bị này có thể ghi nhận nhịp tim, nhiệt độ cơ thể và vị trí của vật nuôi, từ đó đưa ra cảnh báo sớm nếu phát hiện dấu hiệu bất thường. IoT cũng hỗ trợ kiểm soát và tự động hóa các điều kiện trong nhà kính như nhiệt độ, độ ẩm và ánh sáng. Thông qua các thiết bị điều khiển từ xa, nông dân có thể thiết lập và duy trì môi trường tối ưu cho cây trồng mà không cần phải có mặt trực tiếp. Với sự phát triển của các công nghệ mới như trí tuệ nhân tạo (AI), dữ liệu lớn (Big Data), và mạng 5G, IoT sẽ ngày càng được tích hợp sâu hơn vào nông nghiệp. Từ việc dự đoán mùa vụ, tối ưu hóa quy trình sản xuất đến việc đưa ra các giải pháp canh tác bền vững, IoT sẽ là yếu tố cốt lõi trong việc hiện đại hóa nông nghiệp trên toàn cầu.

## 2. Cảm biến và thiết bị trong hệ thống

### 2.1. ESP32C6

ESP32-C6 là một sản phẩm vi điều khiển tiên tiến thuộc dòng ESP32 của Espressif Systems, nổi tiếng với khả năng kết nối không dây mạnh mẽ và tích hợp nhiều tính năng hiện đại. Được ra mắt như một giải pháp tối ưu cho các ứng dụng IoT thế hệ mới, ESP32-C6 không chỉ kế thừa những điểm mạnh của các sản phẩm trước đó mà còn được trang bị thêm các tính năng độc đáo để đáp ứng nhu cầu ngày càng cao của thị trường công nghệ.

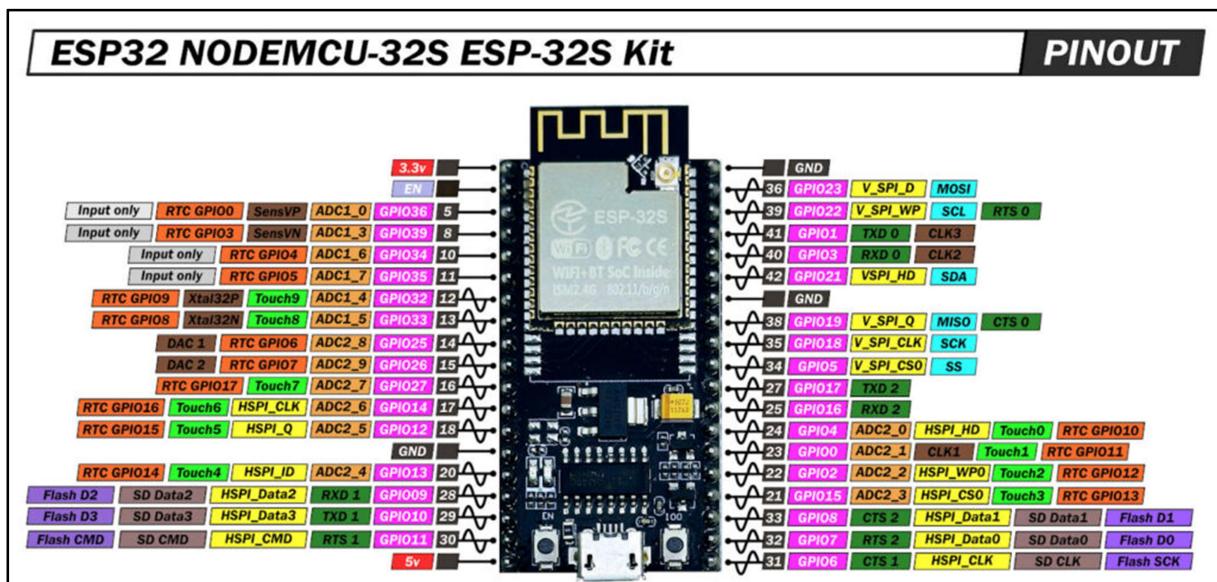


Figure 3: Sơ đồ chân Esp32

ESP32 được trang bị vi xử lý Xtensa Dual-Core LX6 32-bit, với khả năng chạy ở tốc độ từ 160 MHz đến tối đa 240 MHz, đảm bảo hiệu năng xử lý mạnh mẽ. Tốc độ xung nhịp đọc của chip flash có thể tùy chỉnh từ 40 MHz đến 80 MHz khi lập trình. Vi điều khiển này còn sở hữu 520 KB SRAM tích hợp trên chip, bao gồm 8 KB RAM RTC tốc độ cao và 8 KB RAM RTC tốc độ thấp, hỗ trợ hoạt động trong chế độ Deep Sleep để tiết kiệm năng lượng. ESP32 hỗ trợ hai giao thức không dây là Wi-Fi chuẩn 802.11 b/g/n/e/i và Bluetooth v4.2 với các chế độ BR/EDR và BLE, mang lại khả năng kết nối linh hoạt cho các ứng dụng IoT.

Về giao tiếp, ESP32 tương thích với hầu hết các chuẩn phổ biến, bao gồm 2 cổng DAC (chuyển đổi số sang tương tự), 16 cổng ADC (12-bit), 2 cổng I<sup>2</sup>C, 3 cổng UART, 3 cổng SPI (trong đó 1 cổng dành riêng cho chip flash), 2 cổng I<sup>2</sup>S và các giao tiếp hỗ trợ thẻ SD, SDIO/MMC host, và slave (SDIO/SPI). Ngoài ra,

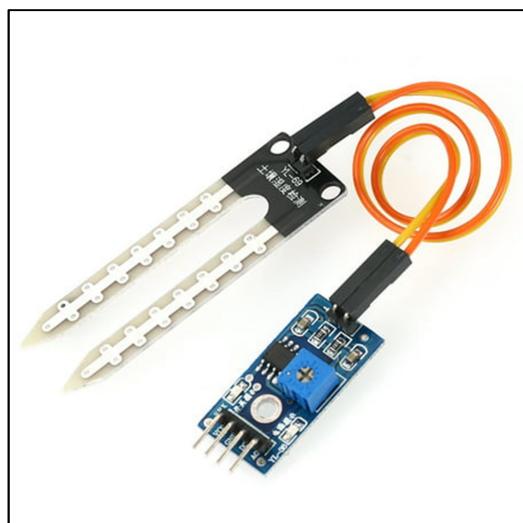
ESP32 còn tích hợp Ethernet MAC với DMA chuyên dụng và hỗ trợ chuẩn IEEE 1588, giao diện CAN bus 2.0, IR (TX/RX), cùng tính năng băm xung PWM trên tất cả các chân GPIO. Đặc biệt, ESP32 được trang bị bộ khuếch đại tương tự tiền khuếch đại (Ultra Low Power Analog Pre-Amplifier), phù hợp cho các ứng dụng cần tiết kiệm năng lượng và xử lý tín hiệu chính xác.

ESP32-C6 được tích hợp bộ vi xử lý RISC-V 32-bit tốc độ cao, cho phép thực hiện các tác vụ phức tạp với hiệu năng ấn tượng. Nó hỗ trợ kết nối Wi-Fi 6 (802.11ax), mang lại tốc độ truyền dữ liệu nhanh hơn, độ trễ thấp hơn và khả năng hoạt động tốt trong môi trường có mật độ thiết bị cao. Đây là một bước tiến lớn so với các sản phẩm trước đó, đặc biệt là khi nhu cầu sử dụng các thiết bị IoT trong môi trường đô thị ngày càng tăng. Ngoài ra, ESP32-C6 cũng hỗ trợ kết nối Bluetooth 5.0 (LE) và Zigbee 3.0, mang đến sự linh hoạt trong việc tích hợp với các giao thức không dây khác nhau. Điều này giúp vi điều khiển trở thành lựa chọn lý tưởng cho các ứng dụng IoT cần giao tiếp đa dạng, chẳng hạn như hệ thống nhà thông minh, cảm biến công nghiệp, và các thiết bị đeo tay. Vi điều khiển ESP32-C6 được trang bị bộ nhớ RAM và flash đủ lớn để chạy các ứng dụng phức tạp. Nó có nhiều chân GPIO hỗ trợ giao tiếp với các cảm biến, module và thiết bị ngoại vi khác, giúp dễ dàng xây dựng các hệ thống IoT toàn diện. Ngoài ra, ESP32-C6 còn tích hợp các tính năng bảo mật cao cấp như mã hóa phần cứng AES-256 và khả năng khởi động bảo mật, giúp bảo vệ dữ liệu người dùng trước các nguy cơ tấn công mạng.

Một trong những điểm mạnh đáng chú ý của ESP32-C6 là khả năng tiêu thụ năng lượng thấp. Với các chế độ tiết kiệm năng lượng linh hoạt, nó rất phù hợp cho các thiết bị IoT chạy bằng pin như cảm biến không dây hoặc các thiết bị giám sát môi trường. Việc tối ưu hóa hiệu suất năng lượng giúp thiết bị hoạt động bền bỉ hơn trong các ứng dụng thực tế. Nhờ vào tính năng Wi-Fi 6 và hỗ trợ các giao thức không dây khác, ESP32-C6 mở ra nhiều tiềm năng ứng dụng trong các lĩnh vực khác nhau. Trong lĩnh vực nhà thông minh, nó có thể được sử dụng để điều khiển đèn, khóa cửa hoặc hệ thống điều hòa không khí. Trong công nghiệp, ESP32-C6 giúp xây dựng các hệ thống giám sát từ xa, phát hiện sự cố hoặc tối ưu hóa quy trình sản xuất. Ngoài ra, trong nông nghiệp thông minh, nó có thể kết nối các cảm biến để thu thập và phân tích dữ liệu môi trường như nhiệt độ, độ ẩm và ánh sáng, từ đó tối ưu hóa quá trình chăm sóc cây trồng.

## 2.2. Cảm biến Độ ẩm đất

Cảm biến độ ẩm đất là một thiết bị quan trọng trong ứng dụng nông nghiệp thông minh, giúp đo lường lượng nước trong đất để tối ưu hóa việc tưới tiêu. Có hai loại cảm biến phổ biến là cảm biến điện trở và cảm biến điện dung. Cảm biến điện dung, như loại SEN0193, hoạt động dựa trên sự thay đổi điện dung giữa các vật đồng trên mạch khi đất chứa độ ẩm khác nhau. Loại cảm biến này vượt trội hơn cảm biến điện trở thông thường ở khả năng chống ăn mòn và độ bền cao vì không sử dụng điện trực tiếp để đo lường. Điện áp đầu ra của nó thường nằm trong khoảng từ 0-3V, tương ứng với mức độ ẩm của đất. Đặc biệt, các phiên bản nâng cao còn có khả năng chống nước, giúp duy trì hiệu suất trong môi trường ẩm ướt.



**Figure 4: Cảm biến độ ẩm đất**

Cảm biến điện trở hoạt động bằng cách đo độ dẫn điện qua đất. Tuy nhiên, nhược điểm lớn của loại này là dễ bị oxi hóa tại các đầu dò do tiếp xúc trực tiếp với đất ẩm. Vì lý do này, chúng ít được ưa chuộng hơn trong các ứng dụng dài hạn.

Khi tích hợp với các nền tảng IoT như Arduino hoặc ESP32, cảm biến độ ẩm đất giúp thu thập dữ liệu theo thời gian thực và gửi về hệ thống quản lý để điều khiển tưới tiêu tự động. Độ chính xác của cảm biến phụ thuộc vào cách triển khai, như khoảng cách dây tín hiệu ngắn để giảm nhiễu và nguồn cấp ổn định để đảm bảo hiệu quả hoạt động.

### 2.3. Cảm biến Nhiệt độ, Độ ẩm môi trường DHT22

Cảm biến DHT22 (hay còn gọi là AM2302) là một loại cảm biến kỹ thuật số chuyên dụng, được sử dụng rộng rãi để đo nhiệt độ và độ ẩm trong các ứng dụng IoT, hệ thống giám sát môi trường, và tự động hóa. Đây là một phiên bản nâng cấp so với cảm biến DHT11, với độ chính xác và phạm vi đo lường tốt hơn, giúp nó trở thành sự lựa chọn hàng đầu cho các dự án cần độ tin cậy cao.



Figure 5: Cảm biến DHT22

DHT22 có khả năng đo nhiệt độ trong khoảng từ -40°C đến +80°C với sai số  $\pm 0.5^{\circ}\text{C}$  và độ ẩm trong khoảng 0–100% RH với sai số  $\pm 2\text{--}5\%$ . Tín hiệu đầu ra là dạng số, đảm bảo dữ liệu truyền tải chính xác và dễ dàng tích hợp với các vi điều khiển như Arduino, ESP32, hoặc Raspberry Pi. Tốc độ truyền dữ liệu của cảm biến phù hợp với các ứng dụng theo thời gian thực, dù chu kỳ cập nhật dữ liệu khoảng 2 giây.

Cảm biến DHT22 bao gồm một cảm biến độ ẩm điện dung và một nhiệt điện trở tích hợp, cùng với một bộ xử lý tín hiệu bên trong. Nó giao tiếp với vi điều khiển thông qua giao thức 1-Wire đơn giản, chỉ yêu cầu một dây để truyền dữ liệu, giúp giảm chi phí và tiết kiệm chân GPIO. DHT22 thường được sử dụng trong các hệ thống giám sát nhà kính, hệ thống HVAC (sưởi, thông gió, điều hòa không khí), thiết bị theo dõi sức khỏe và các ứng dụng IoT cần đo nhiệt độ và độ ẩm chính xác. Với hiệu năng cao và độ bền tốt, DHT22 là một giải pháp lý tưởng cho nhiều dự án công nghệ.

## 2.4. Cảm biến Cường độ ánh sáng GY-30

Cảm biến ánh sáng GY-30, hay còn được biết đến với tên gọi BH1750, là một module cảm biến chuyên dụng để đo cường độ ánh sáng trong các ứng dụng IoT, tự động hóa và giám sát môi trường. Cảm biến này sử dụng chip BH1750 của hãng Rohm Semiconductor, nổi bật với khả năng đo cường độ ánh sáng chính xác và giao tiếp đơn giản qua giao thức I<sup>2</sup>C.

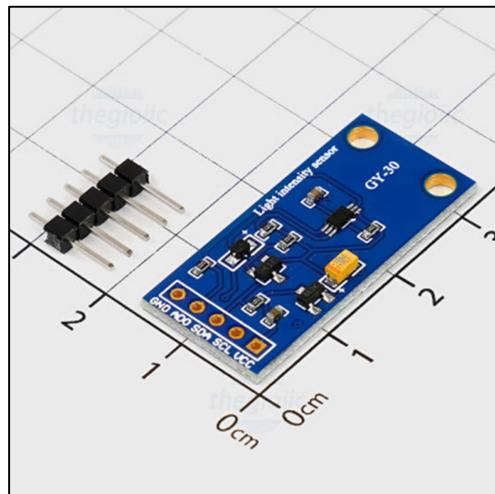


Figure 6: Cảm biến ánh sáng GY-30

GY-30 được thiết kế để đo cường độ ánh sáng trong dải từ 0 đến 65535 lux, với độ nhạy cao và sai số thấp. Đặc biệt, cảm biến này có khả năng đo ánh sáng tự nhiên và nhân tạo, giúp nó phù hợp với nhiều môi trường khác nhau, từ trong nhà đến ngoài trời. GY-30 hoạt động ổn định trong khoảng điện áp từ 3V đến 5V, cho phép nó tương thích với nhiều loại vi điều khiển như Arduino, ESP32 hoặc Raspberry Pi. Cảm biến GY-30 sử dụng giao thức I<sup>2</sup>C, giúp việc giao tiếp với các vi điều khiển trở nên đơn giản và chỉ cần hai chân SDA và SCL. Nó có khả năng chuyển đổi tín hiệu ánh sáng trực tiếp thành dữ liệu số, loại bỏ sự cần thiết của các phép tính phức tạp, đồng thời tiêu thụ năng lượng thấp, rất phù hợp cho các thiết bị chạy bằng pin.

GY-30 thường được sử dụng trong các dự án giám sát cường độ ánh sáng, như hệ thống điều chỉnh độ sáng đèn tự động, thiết bị giám sát cây trồng trong nông nghiệp thông minh, và các ứng dụng theo dõi môi trường. Khả năng đo sáng chính xác và ổn định của cảm biến giúp tối ưu hóa hiệu suất sử dụng ánh sáng, tiết kiệm năng lượng và tăng cường hiệu quả hoạt động của hệ thống.

## 2.5. Loadcell và mạch chuyển đổi ADC 24 bit HX711

### 2.5.1. Loadcell 5kg

Loadcell 5kg sử dụng nguyên lý biến dạng điện trở (strain gauge), trong đó lực tác động lên loadcell làm thay đổi điện trở của các cảm biến tích hợp bên trong, từ đó tạo ra tín hiệu điện tỷ lệ với tải trọng. Loadcell này có khả năng đo lực tối đa lên đến 5kg, với độ nhạy và độ chính xác cao, phù hợp cho các ứng dụng nhỏ gọn.

Thiết kế của loadcell thường gồm một thanh kim loại chắc chắn với các strain gauge được gắn cố định. Nó có bốn dây kết nối: hai dây nguồn (E+ và E-) và hai dây tín hiệu (S+ và S-). Tuy nhiên, tín hiệu từ loadcell thường rất nhỏ (dạng mV), cần một mạch khuếch đại và chuyển đổi tín hiệu để đọc dữ liệu chính xác, và đó là vai trò của HX711, phù hợp cho các ứng dụng đo trọng lượng nhỏ gọn như cân điện tử hoặc giám sát tải trọng.

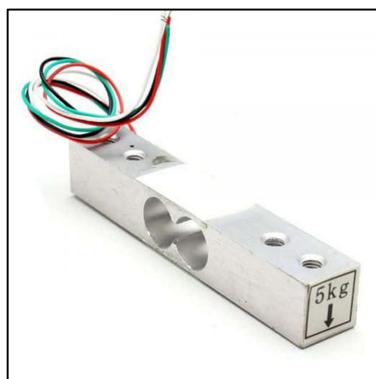


Figure 7: Loadcell

#### Thông số kỹ thuật cơ bản:

- Tải trọng tối đa: 5kg.
- Tải trọng tối thiểu: 0.01kg (tùy thuộc vào mạch xử lý).
- Điện áp kích thích (Excitation Voltage): 5V DC hoặc 10V DC (tối đa 15V DC).
- Dải tín hiệu đầu ra (Output Sensitivity): 1.0mV/V đến 2.0mV/V (tỷ lệ tín hiệu với tải trọng).
- Điện trở đầu vào:  $\sim 400\Omega$ .
- Điện trở đầu ra:  $\sim 350\Omega$ .
- Sai số toàn phần:  $\pm 0.02\%$  FS (Full Scale).
- Độ nhạy:  $\sim 2\text{mV/V}$  (tăng tín hiệu điện áp tỷ lệ với tải trọng).
- Vật liệu: Nhôm hoặc hợp kim thép không gỉ (tăng độ bền cơ học).

### 2.5.2. Mạch đọc ADC 24bit HX711

HX711 là một module khuếch đại và chuyển đổi tín hiệu analog sang số (ADC) với độ phân giải 24-bit, được thiết kế đặc biệt cho các ứng dụng đo lường sử dụng loadcell. Mạch này khuếch đại tín hiệu đầu vào từ loadcell lên mức dễ đọc, sau đó chuyển đổi thành dữ liệu số để truyền đến vi điều khiển qua giao thức 2 dây (SCK và DT). Ưu điểm nổi bật của HX711 là độ nhạy cao, tiêu thụ điện năng thấp và khả năng lọc nhiễu tốt, đảm bảo tín hiệu đầu ra ổn định và chính xác. Nó hỗ trợ hai kênh đầu vào tín hiệu (A và B), cho phép linh hoạt trong việc kết nối nhiều loại loadcell.

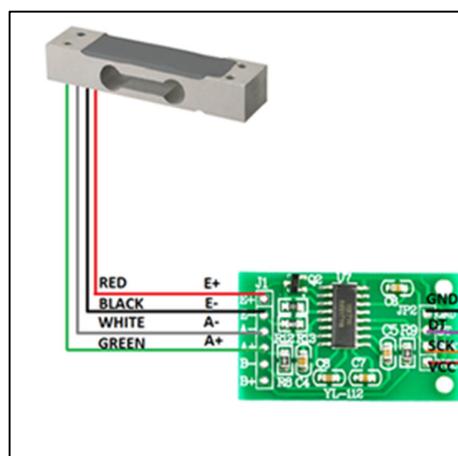


Figure 8: Sơ đồ kết nối Loadcell 5kg và HX711

#### Thông số kỹ thuật cơ bản:

- Độ phân giải ADC: 24-bit.
- Tần số lấy mẫu:
  - 10 SPS (samples per second) với độ nhiễu thấp.
  - 80 SPS trong trường hợp cần phản hồi nhanh hơn.
- Nguồn cấp: 2.7V–5.5V DC (thường sử dụng 3.3V hoặc 5V).
- Dải điện áp đầu vào:  $\pm 20\text{mV}$  (kênh A, với khuếch đại x128),  $\pm 40\text{mV}$  (kênh B).
- Độ lợi khuếch đại (Gain): 32x, 64x, 128x (chọn được trên kênh A hoặc B).
- Tiêu thụ dòng điện:  $<1.5\text{mA}$  khi hoạt động,  $<1\mu\text{A}$  ở chế độ chờ.
- Nhiễu tín hiệu (Noise):  $<50\text{nV}$  khi tần số lấy mẫu 10 SPS.
- Giao tiếp: Serial Clock Input (SCK) và Data Output (DT).
- Kích thước module:  $\sim 35\text{mm} \times 20\text{mm}$  (nhỏ gọn, dễ tích hợp).

## 2.6. Một số linh kiện khác

### 2.6.1. Mạch cấp nguồn DC-DC nhiều đầu ra 3.3V 5V 12V

Mạch nguồn DC-DC với nhiều đầu ra là thiết bị chuyển đổi điện áp từ một nguồn cấp duy nhất (thường là 12V hoặc 24V) sang nhiều mức điện áp khác nhau như 3.3V, 5V và 12V. Loại mạch này thường sử dụng công nghệ chuyển đổi xung (switching), đảm bảo hiệu suất cao (lên tới 90-95%) và giảm nhiệt độ tỏa ra. Nó được dùng để cung cấp năng lượng ổn định cho các cảm biến, vi điều khiển và các thiết bị ngoại vi trong hệ thống IoT. Mạch có thiết kế nhỏ gọn, tích hợp các chức năng bảo vệ như chống ngắn mạch và quá tải, giúp đảm bảo an toàn trong vận hành.



Figure 9: Mạch cấp nguồn DC-DC

### 2.6.2. Bơm 5v

Bơm nước mini 5V là thiết bị thường được sử dụng trong các ứng dụng IoT như tưới tiêu tự động, hệ thống làm mát hoặc bơm tuần hoàn. Được thiết kế nhỏ gọn và dễ tích hợp, bơm 5V hoạt động với nguồn điện một chiều (DC), với lưu lượng nước trung bình từ 80-120 lít/giờ, tùy thuộc vào model. Bơm có thể hoạt động ở dòng điện khoảng 200-350mA, phù hợp với các hệ thống sử dụng pin hoặc nguồn cấp nhỏ. Ưu điểm của loại bơm này là tiết kiệm năng lượng, ít tiếng ồn và tuổi thọ cao. Tuy nhiên, lưu ý cần sử dụng đúng áp suất và độ cao tối đa (thường khoảng 1-1.5m) để đảm bảo hiệu suất tốt nhất.



Figure 10: Bơm mini 5v

### 2.6.3. Led

LED là một thành phần không thể thiếu trong các dự án IoT, từ việc hiển thị trạng thái hoạt động đến chiếu sáng. LED thông thường hoạt động với điện áp thấp (khoảng 2-3.3V) và dòng điện khoảng 20mA, giúp tiết kiệm năng lượng và dễ dàng điều khiển. Ngoài LED đơn sắc, còn có LED RGB và LED ma trận, cho phép tạo ra nhiều hiệu ứng màu sắc và hiển thị thông tin trực quan hơn. Khi tích hợp với các vi điều khiển, LED thường được điều khiển thông qua các chân GPIO hoặc thông qua giao thức PWM để thay đổi độ sáng và màu sắc.

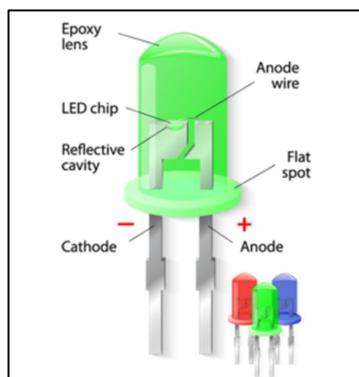


Figure 11: Led

#### 2.6.1. Mạch công suất mosfet 1 kênh F5305S 20A

Mạch công suất MOSFET F5305S là thiết bị chuyển mạch điện tử một kênh, cho phép điều khiển dòng điện tải lớn (lên tới 20A) từ các vi điều khiển như Arduino hoặc ESP32. MOSFET F5305S có điện áp điều khiển công (gate) thấp, thường hoạt động hiệu quả với tín hiệu điều khiển từ 3.3V hoặc 5V. Loại mạch này được sử dụng để điều khiển động cơ, LED công suất, hoặc các thiết bị tải nặng khác. MOSFET F5305S có ưu điểm về khả năng đóng/cắt nhanh, tổn hao nhiệt thấp và thiết kế bảo vệ chống ngắn mạch. Một ứng dụng điển hình của mạch này là điều khiển bơm nước 5V trong hệ thống tưới tiêu tự động.



Figure 12: Mạch công suất mosfet 1 kênh F5305S 20A

### 3. Lưu đồ giải thuật và máy trạng thái của hệ thống

#### 3.1. Lưu đồ giải thuật

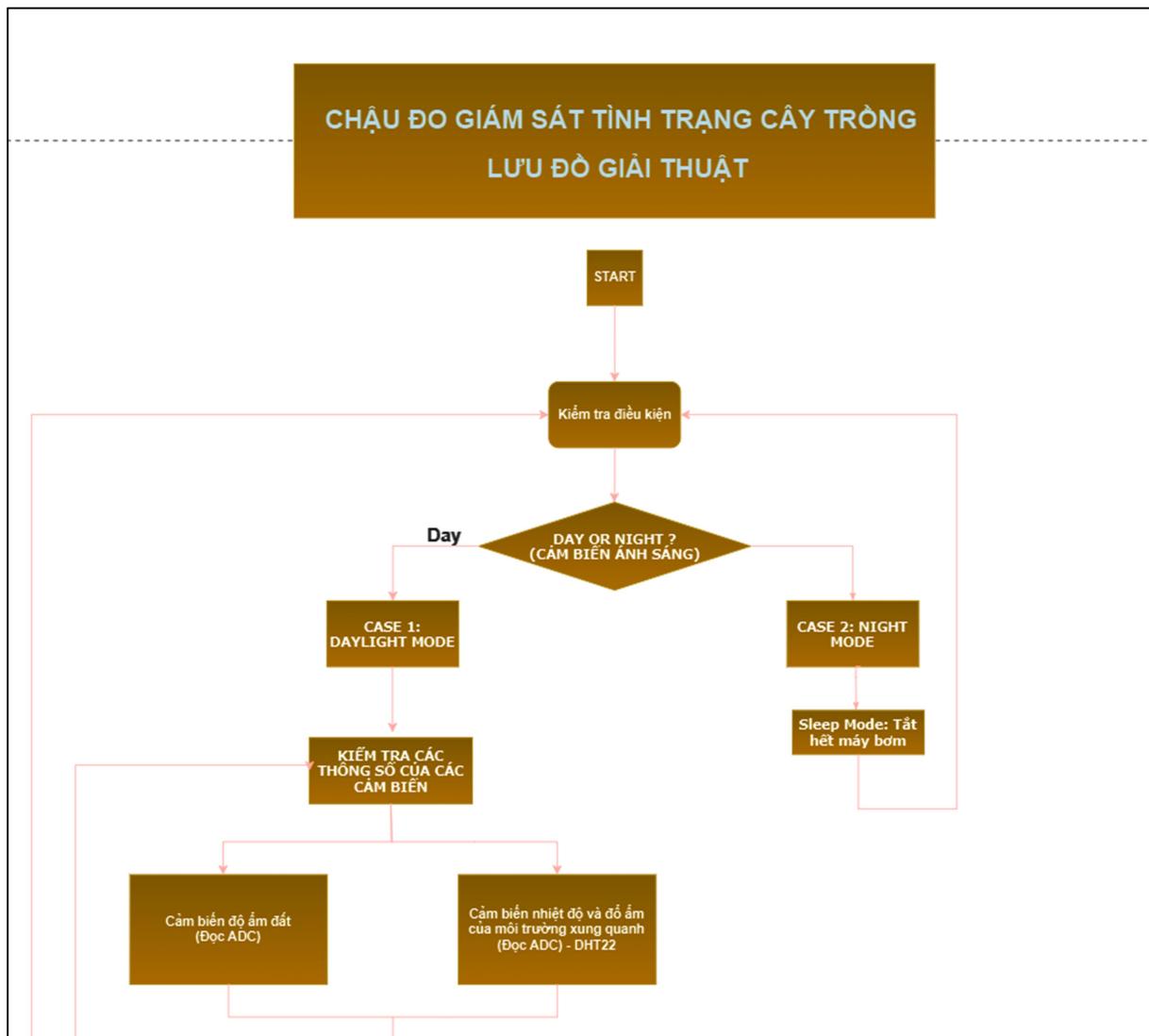


Figure 13: Lưu đồ giải thuật 01

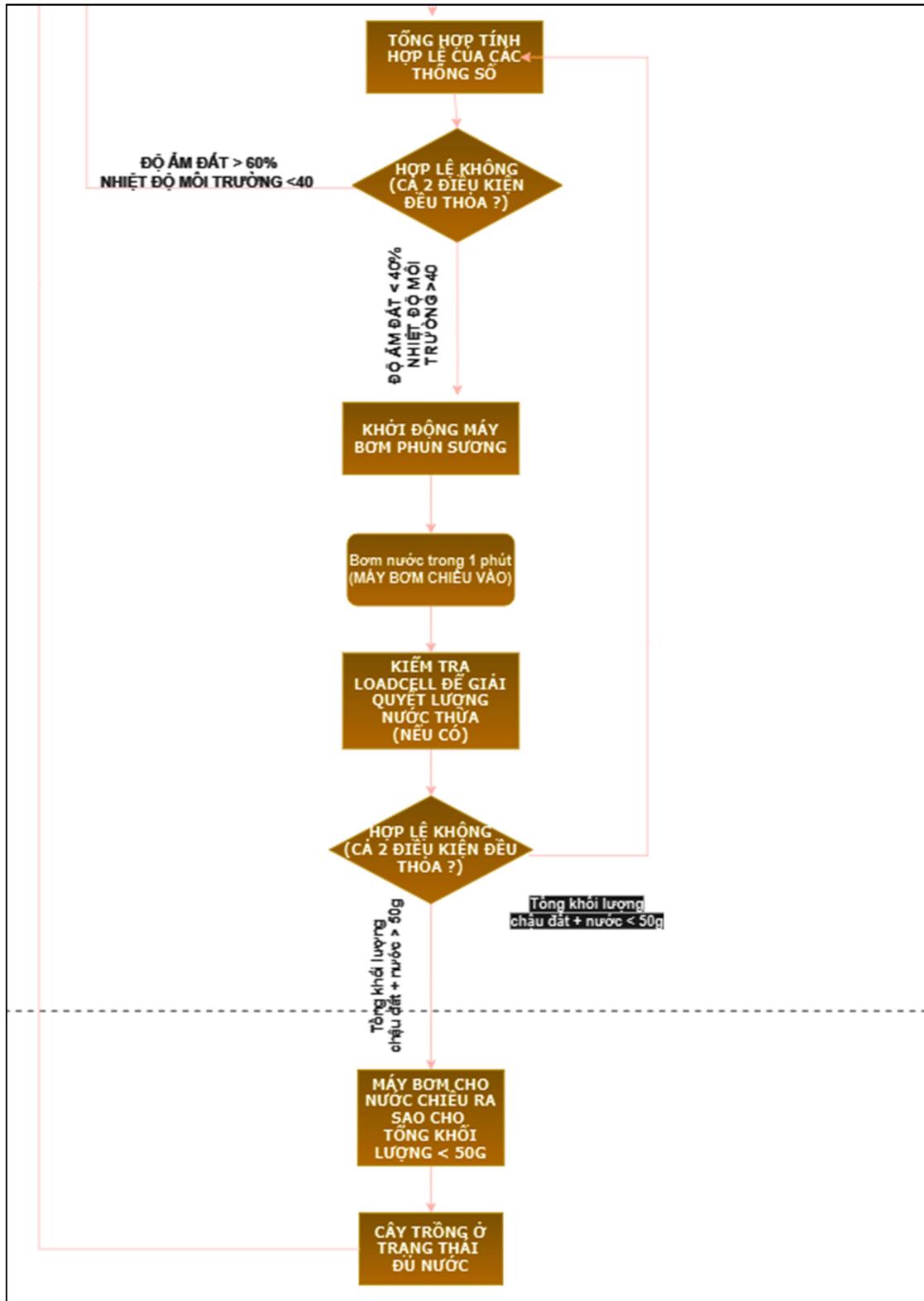


Figure 14: Lưu đồ giải thuật 02

### 3.2. Máy trạng thái

Mô tả máy trạng thái Mealy: Chân đõ theo dõi tình trạng cây trồng

Input lần lượt là X1 (Cảm biến ánh sáng), X2(Cảm biến nhiệt độ), X3(Cảm biến độ ẩm đất) và X4(Loadcell). X1 = 1 khi trời sáng (cảm biến cho giá trị  $>400$  lux), ngược lại X1 = 0. X2 = 1 khi nhiệt độ ngoài trời  $> 40^{\circ}\text{C}$ , ngược lại X2 =0. X3 = 1 khi độ ẩm đất thấp ( $< 40\%$ ), ngược lại X3 = 0. X4 = 1 khi Loadcell đọc giá trị nước thừa  $> 50\text{ml}$ , ngược lại X4 = 0.

Output lần lượt là 2 máy bơm để bơm nước vào tưới cho cây và bơm nước đong lại ra ngoài chân đõ, Led xanh để báo hiệu hệ thống vào trạng thái gửi dữ liệu lên WebServer, Led đỏ báo hiệu máy đang ở trạng thái chờ (Chờ tín hiệu từ các Cảm biến)

Các trạng thái lần lượt là S1: Chờ dữ liệu được gửi từ các Cảm biến khi trời sáng, khi trời tối lập tức chuyển về S2 (chế độ ngủ) và tắt hết các cảm biến. Khi nhận tín hiệu từ Cảm biến nhiệt độ và Độ ẩm, nước vào đến khi đủ nước rồi chuyển về trạng thái Chờ S1. Tương tự với khi nhận tín hiệu từ Loadcell, chuyển sang S4 bơm nước ra đến khi nước đong  $< 50\text{ml}$  thì chuyển về trạng thái chờ. Khi nhận tín hiệu từ các cảm biến thì chuyển sang S5 để gửi dữ liệu lên WebServer

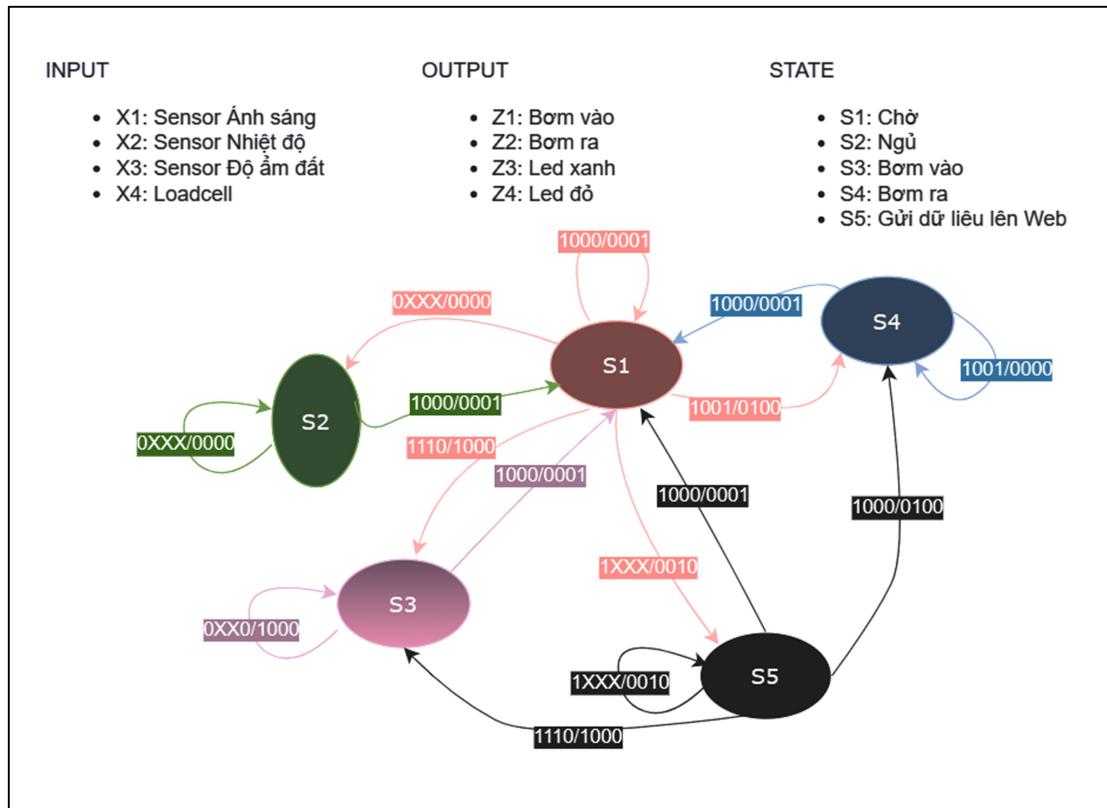


Figure 15: Máy trạng thái Mealy cho Hệ thống giám sát tình trạng cây trồng

## CHƯƠNG 2 THIẾT KẾ VÀ THỰC NGHIỆM

### 1. Thiết kế phần mềm

#### 1.1. Giao tiếp ngoại vi

##### 1.1.1. Các khai báo cơ bản

```
// khai báo các giá trị dự báo tình trạng cây trồng
static char my_string_1[100] = "Binh Thuong";
static char my_string_2[100] = "Kho Heo";
static char my_string_3[100] = "Ung Nuoc";
static char my_string_4[100] = "Thieu Anh Sang";
static char my_string_5[100] = "May bom vao dang bat";
static char my_string_6[100] = "May bom vao dang tat";
static char my_string_7[100] = "May bom ra dang bat";
static char my_string_8[100] = "May bom ra dang tat";

// Khai bao thong so Loadcell doc tu HX711
float zero = 838752.765;
float load      ;
float weight = 0;

//Khai bao semaphore
static SemaphoreHandle_t data_moisure_sensor_semaphore;
static SemaphoreHandle_t lux_semaphore;
static SemaphoreHandle_t humid_semaphore;
static SemaphoreHandle_t temp_semaphore;

void delay_us(uint32_t us){
    uint32_t start = esp_timer_get_time();
    while ((esp_timer_get_time() - start) < us) {
        // Busy wait
    }
}
```

### 1.1.2. Cảm biến Ánh sáng

```
void i2c_start() {
    gpio_set_level(I2C_MASTER_SDA_IO, 1);
    gpio_set_level(I2C_MASTER_SCL_IO, 1);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SDA_IO, 0);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SCL_IO, 0);
}

void i2c_stop() {
    gpio_set_level(I2C_MASTER_SDA_IO, 0);
    gpio_set_level(I2C_MASTER_SCL_IO, 1);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SDA_IO, 1);
    delay_us(5);
}

void i2c_write_bit(int bit) {
    gpio_set_level(I2C_MASTER_SDA_IO, bit);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SCL_IO, 1);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SCL_IO, 0);
    delay_us(5);
}

int i2c_read_bit() {
    gpio_set_direction(I2C_MASTER_SDA_IO, GPIO_MODE_INPUT);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SCL_IO, 1);
    int bit = gpio_get_level(I2C_MASTER_SDA_IO);
    delay_us(5);
    gpio_set_level(I2C_MASTER_SCL_IO, 0);
    gpio_set_direction(I2C_MASTER_SDA_IO, GPIO_MODE_OUTPUT);
    return bit;
}
```

### 1.1.3. Cảm biến Nhiệt độ và Độ ẩm đất

```
void read_dht22(int16_t *temperature, int16_t *humidity) {
    uint8_t data[5] = {0};
    // Send start signal
    gpio_set_direction(DHT_PIN, GPIO_MODE_OUTPUT);
    gpio_set_level(DHT_PIN, 0);
    delay_us(20000); // 20ms
    gpio_set_level(DHT_PIN, 1);
    delay_us(40); // 40us
    gpio_set_direction(DHT_PIN, GPIO_MODE_INPUT);
    // Wait for response
    while (gpio_get_level(DHT_PIN) == 1);
    while (gpio_get_level(DHT_PIN) == 0);
    while (gpio_get_level(DHT_PIN) == 1);
    // Read data
    for (int i = 0; i < 40; i++) {
        while (gpio_get_level(DHT_PIN) == 0);
        delay_us(28);
        if (gpio_get_level(DHT_PIN) == 1) {
            data[i / 8] |= (1 << (7 - (i % 8)));
        }
        while (gpio_get_level(DHT_PIN) == 1);
    }
    // Checksum
    if (data[4] == ((data[0] + data[1] + data[2] + data[3]) & 0xFF)) {
        *humidity = data[0] << 8 | data[1];
        *temperature = data[2] << 8 | data[3];
    } else {
        printf("Checksum failed\n");
    }
}
```

#### 1.1.4. Loadcell

```
unsigned long HX711_read()
{
    gpio_set_level(GPIO_PD_SCK, LOW);
    // wait for the chip to become ready
    while (HX711_is_ready())
    {
        vTaskDelay(10 / portTICK_PERIOD_MS);
    }

    unsigned long value = 0;

    //--- Enter critical section ---
    portDISABLE_INTERRUPTS();

    for(int i=0; i < 24 ; i++)
    {
        gpio_set_level(GPIO_PD_SCK, HIGH);
        ets_delay_us(CLOCK_DELAY_US);
        value = value << 1;
        gpio_set_level(GPIO_PD_SCK, LOW);
        ets_delay_us(CLOCK_DELAY_US);

        if(gpio_get_level(GPIO_DOUT))
            value++;
    }

    // set the channel and the gain factor for the next reading using the
    // clock pin
    for (unsigned int i = 0; i < GAIN; i++)
    {
        gpio_set_level(GPIO_PD_SCK, HIGH);
        ets_delay_us(CLOCK_DELAY_US);
        gpio_set_level(GPIO_PD_SCK, LOW);
        ets_delay_us(CLOCK_DELAY_US);
    }
    portENABLE_INTERRUPTS();
    //--- Exit critical section ---

    value =value^0x800000;

    return (value);
}
```

## 1.2. Giao diện người dùng

```
<body>
    <h1>ESP32 Monitoring System</h1>
    <h3>L01 - GROUP 15</h3>

    <table>
        <tr>
            <th>Parameter</th>
            <th>Value</th>
        </tr>
        <tr>
            <td class="parameter">Soil Moisture</td>
            <td class="value">%d &#37;</td>
        </tr>
        <tr>
            <td class="parameter">Water Load (Weight)</td>
            <td class="value">%f kg</td>
        </tr>
        <tr>
            <td class="parameter">Light Intensity</td>
            <td class="value">%d lux</td>
        </tr>
        <tr>
            <td class="parameter">Temperature</td>
            <td class="value">%d &deg;C</td>
        </tr>
        <tr>
            <td class="parameter">Humidity</td>
            <td class="value">%d &#37;</td>
        </tr>
        <tr>
            <td class="parameter">Plant Status</td>
            <td class="value">%s</td>
        </tr>
        <tr>
            <td class="parameter">Inlet Pump Status</td>
            <td class="value">%s</td>
        </tr>
        <tr>
            <td class="parameter">Outlet Pump Status</td>
            <td class="value">%s</td>
        </tr>
    </table>
</body>
```

### 1.3. Setup hàm Main

```
void app_main(void)
{
    static httpd_handle_t server = NULL;
    ESP_ERROR_CHECK(nvs_flash_init());
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT,
IP_EVENT_STA_GOT_IP, &connect_handler, &server));
    ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT,
WIFI_EVENT_STA_DISCONNECTED, &disconnect_handler, &server));
    ESP_ERROR_CHECK(example_connect());
    nvs_flash_init();
    // initialise_weight_sensor();
    // setup Loadcell
    HX711_init(GPIO_DATA,GPIO_SCLK,eGAIN_128);
    nvs_flash_init();
    //setup bom
    gpio_config_t GPIO_Config = {};
    GPIO_Config.pin_bit_mask = (1<<23);
    GPIO_Config.mode = GPIO_MODE_OUTPUT;
    GPIO_Config.pull_up_en = GPIO_PULLUP_DISABLE;
    GPIO_Config.pull_down_en = GPIO_PULLDOWN_DISABLE;
    GPIO_Config.intr_type = GPIO_INTR_DISABLE;
    gpio_config(&GPIO_Config);
    if (unit == ADC_UNIT_1)
    {
        adc1_config_width(ADC_WIDTH_BIT_12);
        adc1_config_channel_atten(channel, atten);
    }
    else
    {
        adc2_config_channel_atten((adc2_channel_t)channel, atten);
    }
    adc_chars = calloc(1, sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_characterize(unit, atten, ADC_WIDTH_BIT_12, DEFAULT_VREF,
    adc_chars);
    // Tạo semaphore cho cảm biến độ ẩm đất
    data_moisure_sensor_semaphore = xSemaphoreCreateMutex();
    if (data_moisure_sensor_semaphore == NULL)
    {
        ESP_LOGE(TAG, "Failed to create semaphore");
        return;
    }
    //BH1750 VS DHT22
    init_adc();
    gpio_set_direction(I2C_MASTER_SCL_IO, GPIO_MODE_OUTPUT);
    gpio_set_direction(I2C_MASTER_SDA_IO, GPIO_MODE_OUTPUT);
```

```

while (1) {
    vTaskDelay(pdMS_TO_TICKS(10)); // Nhường CPU 10ms
    read_dht22(&temperature, &humidity);
    printf("ĐỘ ẨM MÔI TRƯỜNG: %d %% \nNHIỆT ĐỘ MÔI TRƯỜNG: %d°C\n",
humidity / 10, temperature / 10);

    int adc_reading_1 = read_adc();
    int voltage_1 = esp_adc_cal_raw_to_voltage(adc_reading_1, adc_chars);
    int value = (100 - ((float)(adc_reading_1 - 1800) / (4095 - 1800)) * 100);
    printf("ĐỘ ẨM ĐẤT:           %d %%\n", value);
    i2c_start();
    i2c_write_byte(0x46); // Địa chỉ I2C của BH1750 với lệnh ghi
    i2c_write_byte(0x10); // Lệnh khởi động đo ánh sáng
    i2c_stop();
    vTaskDelay(180 / portTICK_PERIOD_MS); // Đợi cảm biến đo xong
    i2c_start();
    i2c_write_byte(0x47); // Địa chỉ I2C của BH1750 với lệnh đọc
    uint8_t msb = i2c_read_byte(1); // Đọc byte cao
    uint8_t lsb = i2c_read_byte(0); // Đọc byte thấp
    i2c_stop();

    lux = (msb << 8) | lsb;
    printf("ÁNH SÁNG:           %d lx\n\n", lux);

    vTaskDelay(pdMS_TO_TICKS(1000));

    vTaskDelay(pdMS_TO_TICKS(10)); // Nhường CPU 10ms
//CAMBIEN
    int adc_reading_2 = 0;
    for (int i = 0; i < NO_OF_SAMPLES; i++)
    {
        if (unit == ADC_UNIT_1)
        {
            adc_reading_2 += adc1_get_raw((adc1_channel_t)channel);
        }
        else
        {
            int raw;
            adc2_get_raw((adc2_channel_t)channel, ADC_WIDTH_BIT_12, &raw);
            adc_reading_2 += raw;
        }
    }
    adc_reading_2 /= NO_OF_SAMPLES;
    int voltage = esp_adc_cal_raw_to_voltage(adc_reading_2, adc_chars);
    printf("Raw: %d\tVoltage: %d mV\n", adc_reading_2, voltage);
}

```

```

        vTaskDelay(pdMS_TO_TICKS(10)); // Nhờng CPU 10ms
        // Tính toán độ ẩm
        int value_1 = (100 - ((float)(adc_reading_2 - 1400) / (4095 - 1400))
* 100);
        value = (value < 0) ? 0 : ((value > 100) ? 100 : value);

        printf("Độ ẩm: %d %%\n", value_1);

        vTaskDelay(pdMS_TO_TICKS(10)); // Nhờng CPU 10ms
        // Cập nhật data_moisure_sensor một cách an toàn
        if (xSemaphoreTake(data_moisure_sensor_semaphore,
pdMS_TO_TICKS(100)) == pdTRUE)
        {
            data_moisure_sensor = value;
            xSemaphoreGive(data_moisure_sensor_semaphore);
        }

        vTaskDelay(pdMS_TO_TICKS(1000));
        vTaskDelay(pdMS_TO_TICKS(10)); // Nhờng CPU 10ms
        //Loadcell
        weight =0;
        for (char i = 0; i < 50; i++)
        {
            weight += HX711_read();
        }
        load = ((weight/500) - zero)/ 37.55 ;
        ESP_LOGI(TAG, "***** Loadcell = %f *****\n ", load);
        vTaskDelay(pdMS_TO_TICKS(9));
        vTaskDelay(pdMS_TO_TICKS(10)); // Nhờng CPU 10ms
        //Bom
        gpio_set_level(23,0);
        vTaskDelay(100/ portTICK_PERIOD_MS);
        bomnuoc_1 =1;
        gpio_set_level(23,1);
        vTaskDelay(100/ portTICK_PERIOD_MS);
        vTaskDelay(pdMS_TO_TICKS(10)); // Nhờng CPU 10ms
        vTaskDelay(pdMS_TO_TICKS(5000)); // Nhờng CPU
    }
}

```

## 2. Thiết kế phần cứng

## **CHƯƠNG 3      KẾT LUẬN**

# TÀI LIỆU THAM KHẢO

---

- [1]
- [2] Espressif, “Esp Datasheet” từ nguồn : ”ALLDATASHEET.com”
- [3] IC đây rồi, “Cảm biến cường độ ánh sáng GY-30 BH1750FVI” từ nguồn <https://icdayroi.com/>
- [4] BaoAn automation, “Cảm biến độ ẩm là gì? Kiến thức chung về cảm biến độ ẩm”, từ nguồn : [“https://baa.vn/vn/tin-tuc/cam-bien-do-am-la-gi-kien-thuc-chung-ve-cam-bien-do-am\\_52531?srsltid=AfmBOopSGvxV1b0d6qwBDISlcunpWmqj59ZXWSG4xm7UtsU\\_oLXmysTY”](https://baa.vn/vn/tin-tuc/cam-bien-do-am-la-gi-kien-thuc-chung-ve-cam-bien-do-am_52531?srsltid=AfmBOopSGvxV1b0d6qwBDISlcunpWmqj59ZXWSG4xm7UtsU_oLXmysTY)
- [5] Nshop, “Cảm Biến Loadcell 5kg” từ nguồn “icdayroi.com”
- [6] Adruno, “Question about Soil Moisture Sensor”, từ nguồn: [“https://forum.arduino.cc/”](https://forum.arduino.cc/)
- [7] Bùi Quốc Bảo, “Giáo trình Lập trình hệ thống nhúng”
- [8] Ádf
- [9]