ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ
----------o0o----------

ĐỒ ÁN MÔN HỌC

THIẾT KẾ BỘ TƯỚI TỰ ĐỘNG CHO VƯỜN CÂY PHỤ THUỘC VÀO NHIỆT ĐỘ VÀ ĐỘ ẨM

GVHD: Vũ Quang Thời
SVTH: Nguyễn Nhựt Linh
MSSV: 1812819

TP. HỒ CHÍ MINH, THÁNG 05 NĂM 2022

LỜI CẢM ƠN

Em xin gửi lời chân thành đến thầy Vũ Quang Thời giảng viên bộ mộn điện tử trường Đại học Bách Khoa - Đại học Quốc gia TP.HCM đã trang bị giúp em những kỹ năng cơ bản và kiến thức để hoàn thành được đồ án này.

Tuy nhiên, trong quá trình làm đồ án do kiến thức chuyên ngành của em còn hạn chế nên không thể tránh khỏi một vài thiết sót khi trình bày và đánh giá vấn đề. Rất mong nhận được sự góp ý, đánh giá của thầy cô bộ môn để đề tài của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Tp. Hồ Chí Minh, ngày 26 tháng 05 năm 2022 .

Sinh viên

TÓM TẮT ĐỒ ÁN

Đồ án này trình bày về thiết kế bộ tưới cây tự động cho vườn cây phụ thuộc vào độ ẩm và nhiệt độ không khí. Yêu cầu được đặt ra cho đề tài phải có analog, vi xử lí và truyền thông ioT. Sử dụng vi điều khiển Pic để điều khiển hệ thống, có cảm biến nhiệt độ và độ ẩm để cập nhật nhiêt độ và độ ảm trong môi trường, và điều khiển qua bằng điện thoại thông minh. Bộ tưới sẽ đóng ngắt máy bơm theo điều khiển nhiệt đô và độ ẩm. Mục tiêu đồ án là giúp sinh viên nắm rõ hơn về lập trình Pic, kết nối ngoại vi,… để hoàn thiện kiến thức cho bản thân trong tương lai

MỤC LỤC

DANH SÁCH HÌNH MINH HỌA

DANH SÁCH BẢNG SỐ LIỆU

1. GIỚI THIỆU

1.1 Tổng quan

Trong một xã hội hiện đại, sự phát triển của ngành của ngành điện tử viễn thông là một yêu cầu không thể thiếu để thúc đẩy nền kinh tế phát triển và góp phần nâng cao đời sống xã hội.

Qua những thông tin về nền nông nghiệp, công nghiệp của Việt Nam và của các nước phát triển trên thế giới nên trên. Chúng ta thấy rằng không áp dụng công nghệ cao vào sản xuất nông nghiệp ở Việt Nam là một thiếu sót rất lớn đối với một đất nước chủ yếu về nông nghiệp. Tuy nhiên để áp dụng công nghệ hiện đại như ở Israel hay Nhật Bản vào nền nông nghiệp nước ta ngay lúc này là điều rất khó khăn vì những hệ thống đó đòi hỏi phải đầu tư rất cao với những người nông dân ở Việt Nam. Từ thực tế đó chúng em thấy rằng mình hoàn toàn có thể học hỏi và thiết kế những hệ thống tự động hóa đơn giản trong nông nghiệp với giá thành rẻ hơn nhưng vẫn đáp ứng được yêu cầu của một hệ thống thông minh. Ngày nay, lĩnh vực điều khiển đã được ứng dụng nhiều trong các thiết bị, sản phẩm phục vụ cho nhu cầu sinh hoạt hàng ngày làm cho đời sống của chúng ta ngày càng hiện đại và tiện nghi hơn.…

Với mục tiêu nêu trên và xuất phát từ những yêu cầu thực tế, trọng tâm của đề tài này sẽ đi sâu nghiên cứu “Thiết kế bộ tưới cây tự động cho vương cây phụ thuộc vào nhiệt độ và độ ẩm”. Với mong muốn đưa hệ thống của mình vào úng dụng trong cuộc sống hàng ngày.

1.2 Nhiệm vụ đề tài

Các loại vi mạch này xử lý nhanh hơn rất nhiều so với các vi mạch trước và đặc biệt có thể ghi/xóa dữ liệu 1 cách dễ dàng. Vì thế, nó được sử dụng nhiều trong các thiết bị điện tử. Với sự ra đời của IC dòng mới làm thúc đẩy sự phát triển của những module cảm biến như: module cảm biến hồng ngoại…

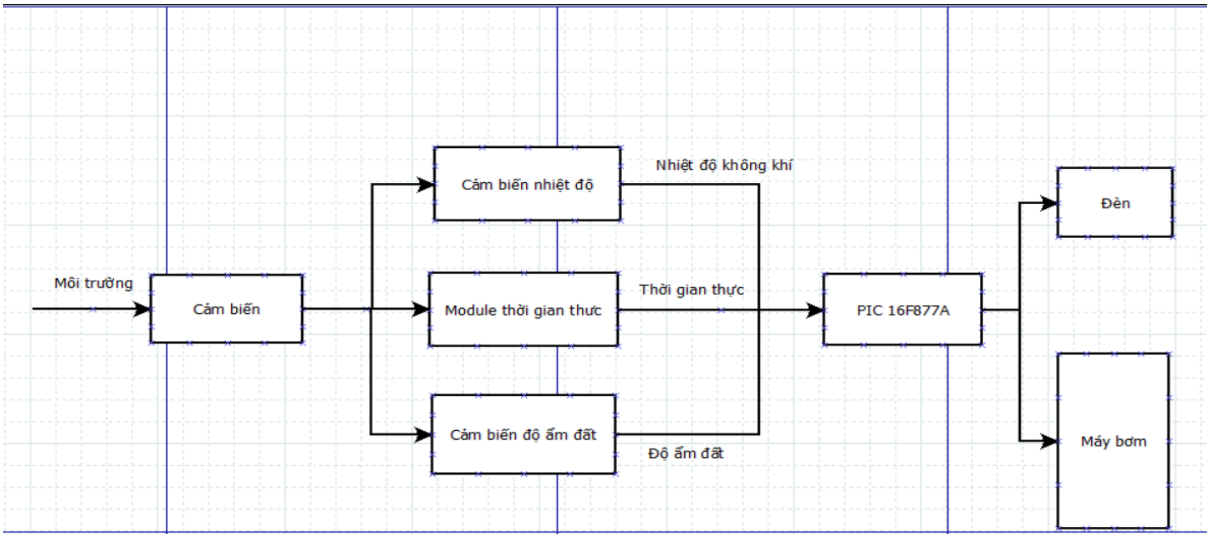
Bên cạnh sự phát triển của khoa học kỹ thuật đã góp phần nâng cao đời sống con người, máy móc có thể hoạt động làm giảm đi sức người, sức của. Cũng chính vì thế mà con người muốn tìm kiếm những điều mới mẻ, tiện lợi, dễ dàng giám sát và kiểm tra. Sự lựa chọn cấp thiết hiện giờ chính là một hệ thống tưới cây tự động.

Chúng ta sẽ bắt đầu tìm hiểu nguyên lí của một bộ tưới cây tự động hoạt động như thế nào từ đó sẽ thiết kế một bộ tưới cây tự động dựa trên vi điều khiển pic và cảm biến nhiệt đô

và độ ẩm để đưa ra các chế độ hoạt động phù hợp nhất. Và ở đây chúng ta dùng PIC 16F877A để làm đồ án này.

2. LÝ THUYẾT

2.1 Xây dựng sơ đồ khối ban đầu:



Hình 2. 1 Sơ đồ khối ban đầu hình thành

Trong sơ đồ khối trên gồm các khối:

-Khối điều khiển trung tâm PIC 16F877A: Điều khiển toàn bộ chức năng của mạch, nhận dữ liệu giải mã tín hiệu nhiệt độ, độ ẩm. Đưa hiển thị lên LCD sau đó đưa ra tín hiệu điều khiển bật/ tắt máy bơm.

-Khối cảm biến nhiệt độ LM35: Cập nhật nhiệt độ của môi trường gửi về cho vi điều khiển

-Khối cảm biến độ ẩm đất Soil Moisture: Cập nhật độ ẩm đất gửi về cho vi xử lí

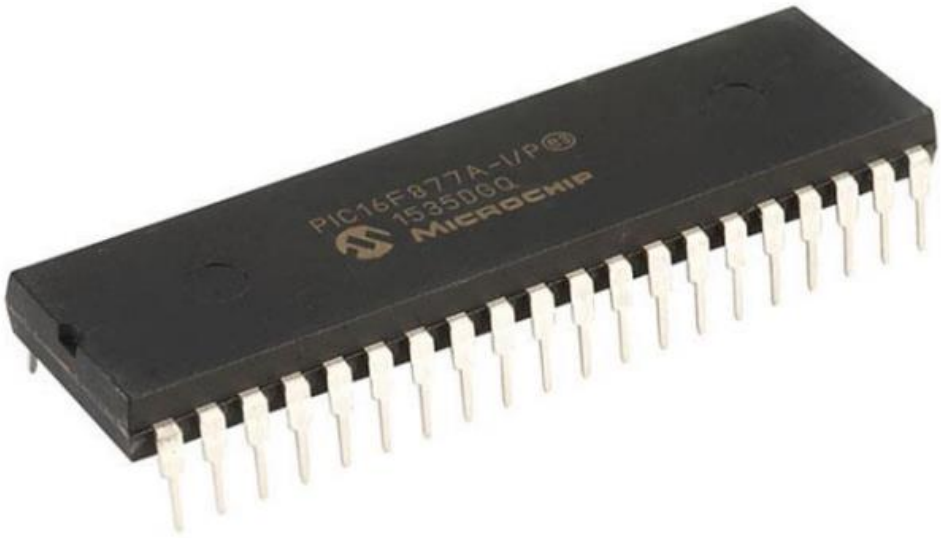
-Khối hiển thị LCD: Là LCD 4 dòng 20 kí tự để hiển thị nhiệt độ, độ ẩm và chế độ hoạt động

-Khối thiết bị: Máy bơm, Đèn có relay 5VDC dùng để nhận lệnh đóng ngắt máy bơm từ vi điều khiển.

-Khối module thời gian thực: cập nhật thời gian thực cho vi điều khiển xử lí để điều khiển tính hiệu đèn.

2.2 Giới thiệt tổng quan và chức năng của từng khối:

2.2.1 Khối vi điều khiển Pic 16F877A:



Hình 2. 2 Pic 16F877A

a. Tổng quan về Pic 16F877A:

- Đây là vi điều khiển thuộc họ PIC16Fxxx với tập lệnh gồm 35 lệnh có độ dài 14 bit. Mỗi lệnh đều được thực thi trong một chu kì xung clock. Tốc độ hoạt động tối đa cho phép là 20 MHz với một chu kì lệnh là 200ns. Bộ nhớ chương trình 8Kx14 bit, bộ nhớ dữ liệu 368-byte RAM và bộ nhớ dữ liệu EEPROM với dung lượng 256 byte. Số PORT I/O là 5 với 33 pin I/O.

- Các đặc tính ngoại vi bao gồm các khối chức năng sau:

- Timer0: bộ đếm 8-bit với bộ chia tần số 8 bit.

- Timer1: bộ đếm 16-bit với bộ chia tần số, có thể thực hiện chức năng đếm dựa vào xung clock ngoại vi ngay khi vi điều khiển hoạt động ở chế độ sleep.

- Timer2: bộ đếm 8-bit với bộ chia tần số, bộ postcaler.

- Hai bộ Capture/so sánh/điều chế độ rông xung

- Các chuẩn giao tiếp nối tiếp SSP (Synchronous Serial Port), SPI và I2C.

- Chuẩn giao tiếp nối tiếp USART với 9-bit địa chỉ.

- Cổng giao tiếp song song PSP (Parallel Slave Port) với các chân điều khiển RD, WR, CS ở bên ngoài.

- Các đặc tính Analog:

- 8 kênh chuyển đổi ADC 10 bit

- Hai bộ so sánh.

- Bên cạnh đó là một vài đặc tính khác của vi điều khiển như:

- Bộ nhớ flash với khả năng ghi xóa được 100.000 lần.

- Bộ nhớ EEPROM với khả năng ghi xóa được 1.000.000 lần

- Dữ liệu bộ nhớ EEPROM có thể lưu trữ trên 40 năm.

- Khả năng tự nạp chương trình với sự điều khiển của phần mềm.

- Nạp được chương trình ngay trên mạch điện ICSP (In Circuit Serial Programming)

- Watchdog Timer với bộ dao động trong.

- Chức năng bảo mật mã chương trình.

- Chế độ Sleep.

- Có thể hoạt động với nhiều dạng Oscillator khác nhau

- Tổ chức bộ nhớ

- Cấu trúc bộ nhớ của vi điều khiển PIC16F877A bao gồm bộ nhớ chương trình (Program memory) và bộ nhớ dữ liệu (Data Memory).

+ Bộ nhớ chương trình:

Bộ nhớ chương trình của vi điều khiển PIC16F877A là bộ nhớ flash, dung lượng bộ nhớ 8K word (1 word = 14 bit) và được phân thành nhiều trang (từ page0 đến page 3). Như vậy bộ nhớ chương trình có khả năng chứa được 8*1024 = 8192 lệnh (vì một lệnh sau khi mã hóa sẽ có dung lượng 1 word (14 bit) Khi vi

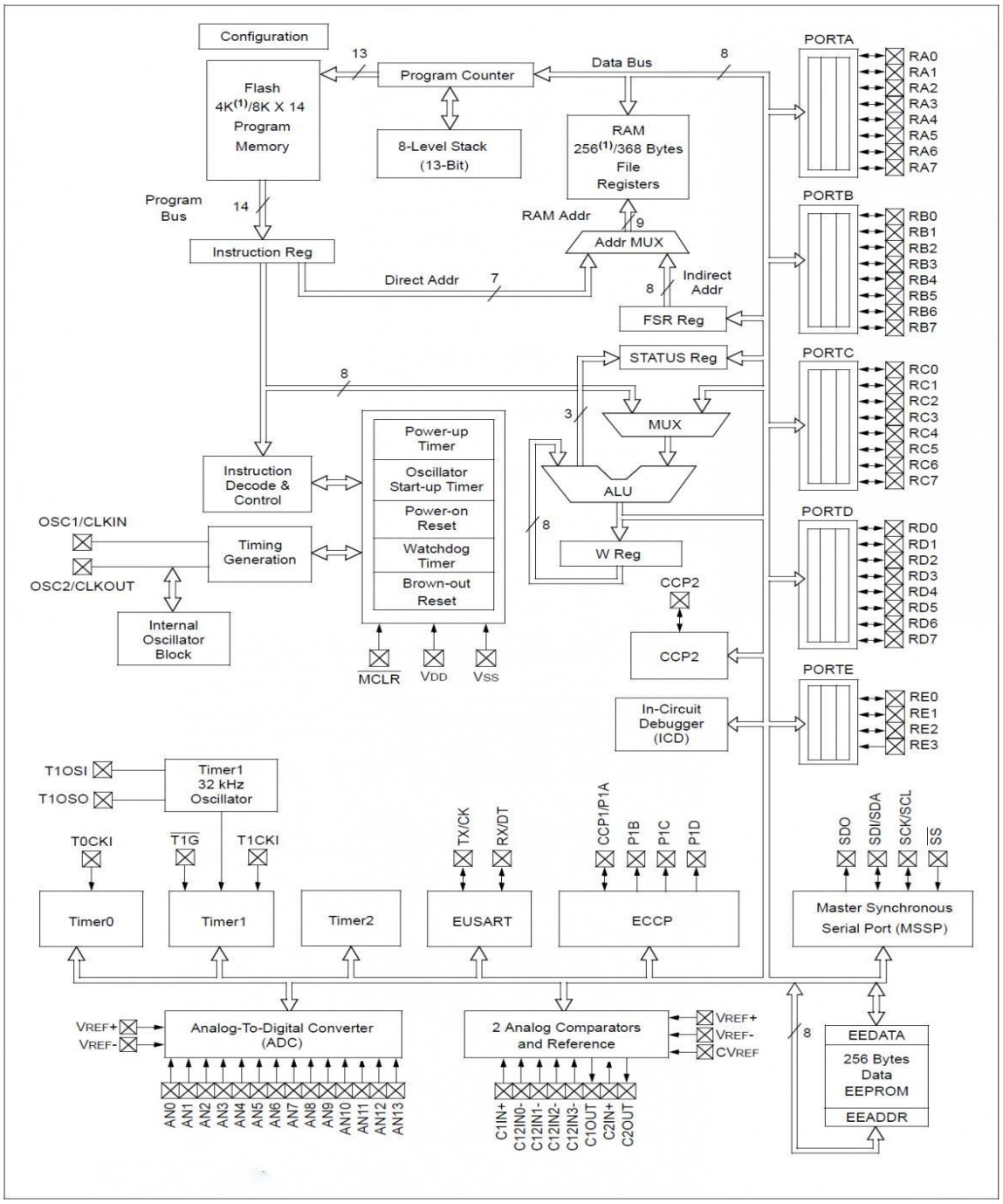
điều khiển được reset, bộ đếm chương trình sẽ chỉ đến địa chỉ 0000h (Reset vector). Khi có ngắt xảy ra, bộ đếm chương trình sẽ chỉ đến địa chỉ 0004h (Interrupt vector). Bộ nhớ chương trình không bao gồm bộ nhớ stack và không được địa chỉ hóa bởi bộ đếm chương trình.

+ Bộ nhớ dữ liệu:

Bộ nhớ dữ liệu của PIC là bộ nhớ EEPROM được chia ra làm nhiều bank. Đối với PIC16F877A bộ nhớ dữ liệu được chia ra làm 4 bank. Mỗi bank có dung lượng 128 byte, bao gồm các thanh ghi có chức năng đặc biệt SFG (Special Function Register) nằm ở các vùng địa chỉ thấp và các thanh ghi mục đích chung GPR (General Purpose Register) nằm ở vùng địa chỉ còn lại trong bank. Các thanh ghi SFR thường xuyên được sử dụng (ví dụ như thanh ghi STATUS) sẽ được đặt ở tất cà các bank của bộ nhớ dữ liệu giúp thuận tiện trong quá trình truy xuất và làm giảm bớt lệnh của chương trình.

b. Tổng quan về Pic 16F877A:

➢ Sơ đồ khối của PIC 16F877A:



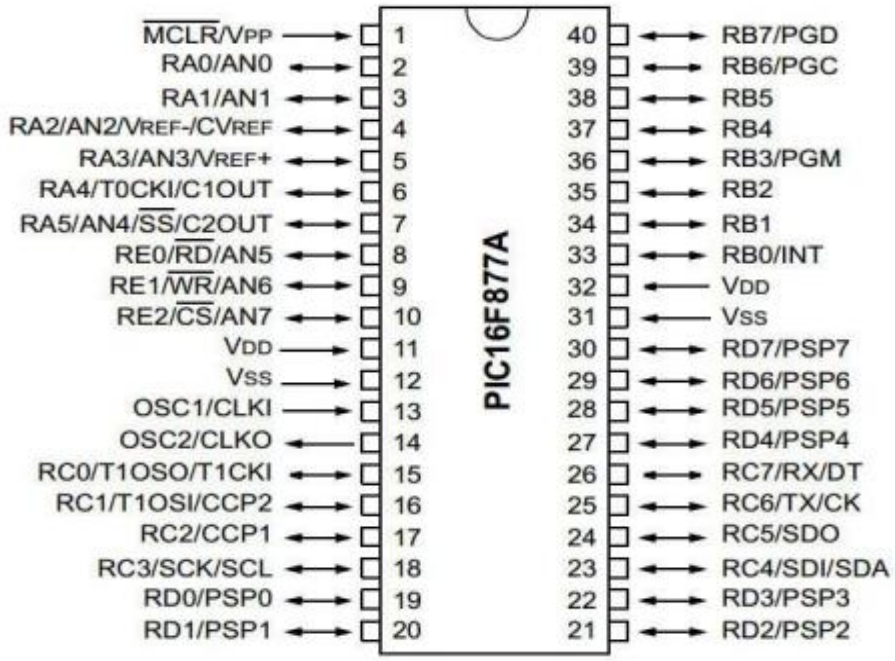
Hình 2. 3 Sơ đồ khối Pic 16F877A

Cấu hình bên trong của Pic 16F887A:

- Có thanh ghi trạng thái (status register) cho biết trạng thái sau khi tính toán của khối ALU.

- Có thanh ghi FSR
- Có khối ALU cùng với thanh ghi working hay thanh ghi A để xử lý dữ liệu.
- Có khối giải mã lệnh và điều khiển (Instruction Decode and Control).
- Có khối dao động nội (Internal Oscillator Block).
- Có khối dao động kết nối với 2 ngõ vào OSC1 và OSC2 để tạo dao động.
- Có khối các bộ định thời khi cấp điện PUT, có bộ định thời chờ dao động ổn định, có mạch reset khi có điện, có bộ định thời giám sát watchdog, có mạch reset khi phát hiện sụt giảm nguồn.
- Có khối bộ dao động cho timer1 có tần số 32kHz kết nối với 2 ngõ vào T1OSI và T1OSO.
- Có khối CCP2 và ECCP.
- Có khối mạch gỡ rối (In-Circuit Debugger IDC).
- Có khối timer0 với ngõ vào xung đếm từ bên ngoài là T0CKI.
- Có khối truyền dữ liệu đồng bộ/bất đồng bộ nâng cao.
- Có khối truyền dữ liệu đồng bộ MSSP cho SPI và I2C.
- Có khối bộ nhớ Eeprom 256-bytes và thanh ghi quản lý địa chỉ EEADDR và thanh ghi dữ liệu EEDATA.
- Có khối chuyển đổi tín hiệu tương tự sang số ADC.
- Có khối 2 bộ so sánh với nhiều ngõ vào ra và điện áp tham chiếu.
- Có khối các port A, B, C, E và D

➢ Sơ đồ chân của PIC 16F877A:



Hình 2. 4 Sơ đồ chân PIC 16F877A

❖ Chức năng chân của port A:

- Chân RA0/AN0/ULPWU/C12IN0- (2): có 4 chức năng:
  + RA0: xuất/ nhập số – bit thứ 0 của port A.
  + AN0: ngõ vào tương tự của kênh thứ 0.
- Chân RA1/AN1/C12IN1- (3): có 3 chức năng:
  + RA1: xuất/nhập số – bit thứ 1 của port A.
  + AN1: ngõ vào tương tự của kênh thứ 1
- Chân RA2/AN2/VREF-/CVREF/C2IN+ (4): có 5 chức năng:
  + RA2: xuất/nhập số – bit thứ 2 của port A.
  + AN2: ngõ vào tương tự của kênh thứ 2.
  + VREF-: ngõ vào điện áp chuẩn (thấp) của bộ ADC.
  + CVREF: điện áp tham chiếu VREF ngõ vào bộ so sánh.
- Chân RA3/AN3/VREF+/C1IN+ (5): có 4 chức năng:
  + RA3: xuất/nhập số – bit thứ 3 của port A.
  + AN3: ngõ vào tương tự kênh thứ 3.
  + VREF+: ngõ vào điện áp chuẩn (cao) của bộ A/D.

+ C1IN+: ngõ vào dương của bộ so sánh C1.

- Chân RA4/TOCKI/C1OUT (6): có 3 chức năng:
  + RA4: xuất/nhập số – bit thứ 4 của port A.
  + TOCKI: ngõ vào xung clock từ bên ngoài cho Timer0.
  + C1OUT: ngõ ra bộ so sánh 1.
- Chân RA5/AN4/ SS / C2OUT (7): có 4 chức năng:
  + RA5: xuất/nhập số – bit thứ 5 của port A.
  + AN4: ngõ vào tương tự kênh thứ 4.
  + SS: ngõ vào chọn lựa SPI tớ (Slave SPI device).
  + C2OUT: ngõ ra bộ so sánh 2.
- Chân RA6/OSC2/CLKOUT (14): có 3 chức năng:
  + RA6: xuất/nhập số – bit thứ 6 của port A.
  + OSC2: ngõ ra dao động thạch anh. Kết nối đến thạch anh hoặc bộ cộng hưởng.
- Chân RA7/OSC1/CLKIN (13): có 3 chức năng:
  + RA7: xuất/nhập số – bit thứ 7 của port A.
  + OSC1: ngõ vào dao động thạch anh hoặc ngõ vào nguồn xung ở bên ngoài.

❖ Chức năng chân của port B:

- Chân RB0/AN12/INT (33): có 3 chức năng:
  + RB0: xuất/nhập số – bit thứ 0 của port B.
  + AN12: ngõ vào tương tự kênh thứ 12.
  + INT: ngõ vào nhận tín hiệu ngắt ngoài.
- Chân RB1/AN10/C12IN3- (34): có 3 chức năng:
  + RB1: xuất/nhập số – bit thứ 1 của port B.
  + AN10: ngõ vào tương tự kênh thứ 10.
  + C12IN3-: ngõ vào âm thứ 3 của bộ so sánh C1 hoặc C2
- Chân RB2/AN8 (35): có 2 chức năng:
  + RB2: xuất/nhập số – bit thứ 2 của port B.
  + AN8: ngõ vào tương tự kênh thứ 8.
- Chân RB3/AN9/PGM/C12IN2 (36): có 4 chức năng:
  + RB3: xuất/nhập số – bit thứ 3 của port B.
  + AN9: ngõ vào tương tự kênh thứ 9.

+ PGM: Chân cho phép lập trình điện áp thấp ICSP.

+ C12IN1-: ngõ vào âm thứ 2 của bộ so sánh C1 hoặc C2

- Chân RB4/AN11 (37): có 2 chức năng:

  + RB4: xuất/nhập số – bit thứ 4 của port B.

  + AN11: ngõ vào tương tự kênh thứ 11.

- Chân RB5/ AN13/T1G (38): có 3 chức năng:

  + RB5: xuất/nhập số – bit thứ 5 của port B.

  + AN13: ngõ vào tương tự kênh thứ 13.

  + T1G (Timer1 gate input): ngõ vào Gate cho phép time1 đếm dùng để đếm độ rộng xung.

- Chân RB6/ICSPCLK (39): có 2 chức năng:

  + RB6: xuất/nhập số.

  + ICSPCLK: xung clock lập trình nối tiếp.

- Chân RB7/ICSPDAT (40): có 2 chức năng:

  + RB7: xuất/nhập số.

  + ICSPDAT: ngõ xuất nhập dữ liệu lập trình nối tiếp.

❖ Chức năng chân của port C:

- Chân RC0/T1OSO/T1CKI (15): có 3 chức năng:

  + RC0: xuất/nhập số – bit thứ 0 của port C.

  + T1OSO: ngõ ra của bộ dao động Timer1.

  + T1CKI: ngõ vào xung clock từ bên ngoài Timer1.

- Chân RC1/T1OSI/CCP2 (16): có 3 chức năng:

  + RC1: xuất/nhập số – bit thứ 1 của port C.

  + T1OSI: ngõ vào của bộ dao động Timer1.

  + CCP2: ngõ vào Capture2, ngõ ra compare2, ngõ ra PWM2.

- Chân RC2 /P1A/CCP1 (17): có 3 chức năng:

  + RC2: xuất/nhập số – bit thứ 2 của port C.

  + P1A: ngõ ra PWM.

  + CCP1: ngõ vào Capture1, ngõ ra compare1, ngõ ra PWM1.

- Chân RC3/SCK/SCL (18): có 3 chức năng:

  + RC3: xuất/nhập số – bit thứ 3 của port C.

  + SCK: ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ SPI.

+ SCL: ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ I2C.

- Chân RC4/SDI/SDA (23): có 3 chức năng:
  + RC4: xuất/nhập số – bit thứ 4 của port C.
  + SDI: ngõ vào dữ liệu trong truyền dữ liệu kiểu SPI.
  + SDA: xuất/nhập dữ liệu I2C.
- Chân RC5/SDO (24): có 2 chức năng:
  + RC5: xuất/nhập số – bit thứ 5 của port C.
  + SDO: ngõ xuất dữ liệu trong truyền dữ liệu kiểu SPI.
- Chân RC6/TX/CK (25): có 3 chức năng:
  + RC6: xuất/nhập số – bit thứ 6 của port C.
  + TX: ngõ ra phát dữ liệu trong chế độ truyền bất đồng bộ USART.
  + CK: ngõ ra cấp xung clock trong chế độ truyền đồng bộ USART.
- Chân RC7/RX/DT (26): có 3 chức năng:
  + RC7: xuất/nhập số – bit thứ 7 của port C.
  + RX: ngõ vào nhận dữ liệu trong chế độ truyền bất đồng bộ EUSART.
  + DT: ngõ phát và nhận dữ liệu ở chế độ truyền đồng bộ EUSART.

❖ Chức năng chân của port D:

- Chân RD0 (19): có 1 chức năng:
  + RD0: xuất/nhập số – bit thứ 0 của port D.
- Chân RD1 (20): có 1 chức năng:
- + RD1: xuất/nhập số – bit thứ 1 của port D.
- Chân RD2 (21): có 1 chức năng:
  + RD2: xuất/nhập số – bit thứ 2 của port D.
- Chân RD3 (22): có 1 chức năng:
  + RD3: xuất/nhập số – bit thứ 3 của port D.
- Chân RD4 (27): có 1 chức năng:
  + RD4: xuất/nhập số – bit thứ 4 của port D.
- Chân RD5/ P1B (28): có 2 chức năng:
  + RD5: xuất/nhập số – bit thứ 5 của port D.
  + P1B: ngõ ra PWM.
- Chân RD6/ P1C (29): có 2 chức năng:
  + RD6: xuất/nhập số – bit thứ 6 của port D.

+ P1C: ngõ ra PWM.

- Chân RD7/P1D (30): có 2 chức năng:

  + RD7: xuất/nhập số – bit thứ 7 của port D.

  + P1D: ngõ ra tăng cường CPP1

❖ Chức năng chân của port E:

- Chân RE1/AN6 (9): có 2 chức năng:

  + RE0: xuất/nhập số.

  + AN5: ngõ vào tương tự 5.

- Chân RE2/AN7 (10): có 2 chức năng:

  + RE2: xuất/nhập số.

  + AN7: ngõ vào tương tự kênh thứ 7.

- Chân RE3/ MCLR /VPP (1): có 3 chức năng:

  + RE3: xuất/nhập số – bit thứ 3 của port E.

  + MCLR: là ngõ vào reset tích cực mức thấp.

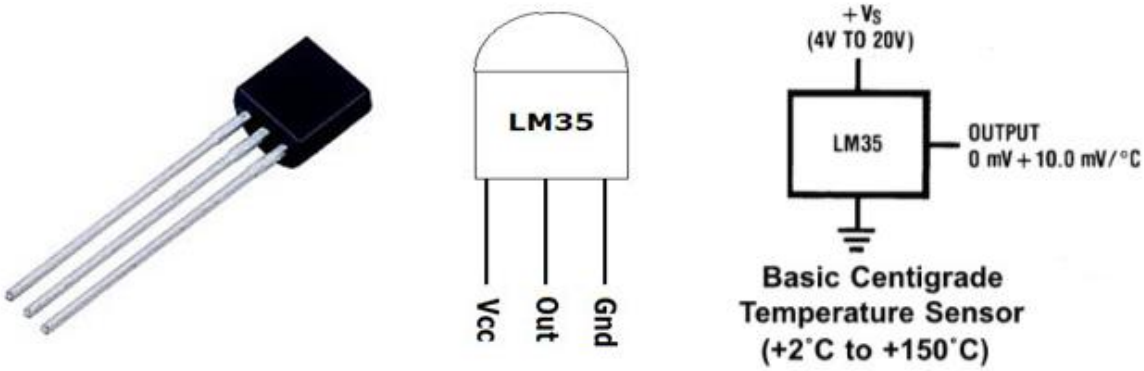
  + VPP: ngõ vào nhận điện áp khi ghi dữ liệu vào bộ nhớ nội flash.

  + Chân VDD (11), (32):

  + Nguồn cung cấp dương từ 2V đến 5V. + Chân VSS (12), (31):

  + Nguồn cung cấp 0V.

2.2.2 Khối cảm biến nhiệt độ LM35:



Hình 2. 5 Cảm biến nhiệt độ LM35

LM35 là một cảm biến nhiệt độ được sử dụng rộng rãi. Nó hiển thị các giá trị dưới dạng điện áp đầu ra thay vì độ C.

LM35 hiển thị giá trị điện áp cao hơn cặp nhiệt điện và có thể không cần khuếch đại điện áp đầu ra.

Điện áp đầu ra của LM35 tỷ lệ với nhiệt độ C. Hệ số thang đo là 0,01 V / ° C.

Một đặc điểm quan trọng nhất là nó chỉ lấy 60 micromps từ nguồn và có khả năng tự gia nhiệt thấp.

Cảm biến nhiệt độ LM35 có nhiều gói khác nhau như gói giống transistor kim loại T0-46, gói giống transistor nhựa TO-92, gói dán 8 chân SO-8.

Số chân	Tên chân	Chức năng
1	V_{CC} hay $+V_S$	Chân cấp nguồn với điện áp từ 4V đến 30V
2	V_{OUT}	Chân lấy điện áp ra, điện áp ở chân này thay đổi 10mV/^oC
3	GND	Chân nối đất

Bảng 2. 1 Tên chân và chức năng của LM35

Thông số:

- Hiệu chuẩn trực tiếp theo ^oC
- Điện áp hoạt động: 4-30VDC
- Dòng điện tiêu thụ: khoảng 60uA
- Nhiệt độ thay đổi tuyến tính: 10mV/°C
- Khoảng nhiệt độ đo được: -55°C đến 150°C
- Điện áp thay đổi tuyến tính theo nhiệt độ: 10mV/°C
- Độ tự gia nhiệt thấp, 0,08^oC trong không khí tĩnh
- Sai số: 0,25°C
- Trở kháng ngõ ra nhỏ, 0,2Ω với dòng tải 1mA
- Kiểu chân: TO92
- Kích thước: 4.3 × 4.3mm

LM35 có thể đo nhiệt độ trong phạm vi từ -55^oC đến 150^oC. Độ chính xác thực tế của cảm biến: ±1/4°C ở nhiệt độ phòng và ±3/4°C trong phạm vi nhiệt độ từ -55°C đến 150°C. Việc chuyển đổi điện áp đầu ra sang ^oC cũng dễ dàng và trực tiếp.

Trở kháng đầu ra nhỏ, đầu ra tuyến tính và hiệu chuẩn chính xác là những đặc tính vốn có của LM35, giúp tạo giao tiếp để đọc hoặc điều khiển mạch rất dễ dàng.

Điện áp cung cấp cho cảm biến LM35 hoạt động có thể từ +4 V đến 30 V. Nó tiêu thụ dòng điện khoảng 60µA. LM35 có nhiều họ là LM35A, LM35CA, LM35D, LM135, LM135A, LM235, LM335. Tất cả các thành viên trong họ LM35 đều hoạt động theo nguyên tắc giống nhau nhưng khả năng đo nhiệt độ khác nhau và chúng cũng có nhiều kiểu chân khác nhau (SOIC, TO-220, TO-92, TO).

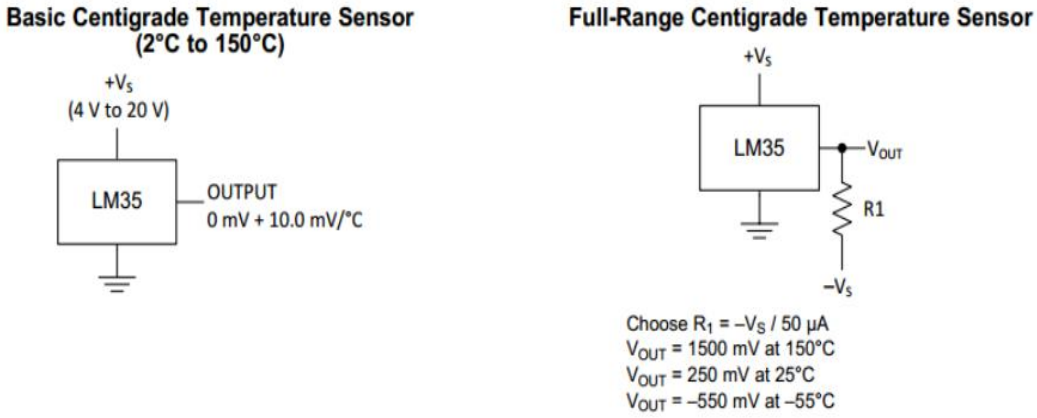
Nguyên lí hoạt động của cảm biến nhiệt độ LM35:

Cảm biến LM35 hoạt động bằng cách cho ra một giá trị điện áp nhất định tại chân V_{OUT} (chân giữa) ứng với mỗi mức nhiệt độ. Như vậy, bằng cách đưa vào chân bên trái của cảm biến LM35 điện áp 5V, chân phải nối đất, đo hiệu điện thế ở chân giữa, bạn sẽ có được nhiệt độ (0-100ºC) tương ứng với điện áp đo được.

Vì điện áp ngõ ra của cảm biến tương đối nhỏ nên thông thường trong các mạch ứng dụng thực tế, chúng ta thường dùng Op-Amp để khuếch đại điện áp ngõ ra này.

Các dạng mạch đo nhiệt độ:

LM35 có thể được sử dụng một trong hai cấu hình mạch như hình bên dưới. Cả hai đều mang lại kết quả khác nhau.



Hình 2. 6 Các dạng mạch đo nhiệt độ

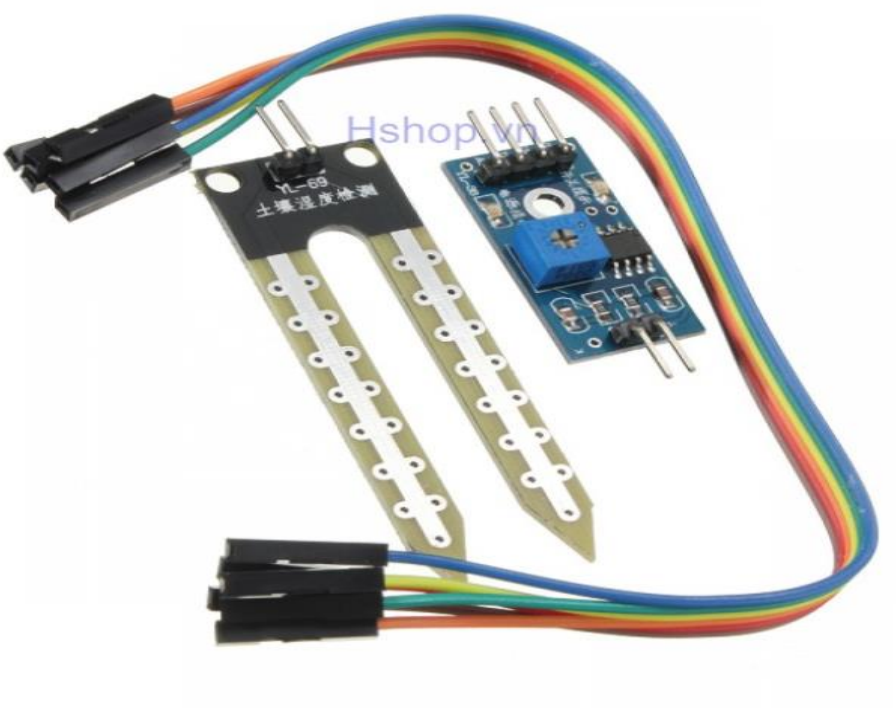
Trong cấu hình mạch phía bên trái, cảm biến chỉ có thể đo nhiệt độ dương từ 2 ºC đến 150 ºC. Theo cấu hình mạch này, chúng ta chỉ cần cấp nguồn cho LM35 và kết nối đầu ra trực tiếp với bộ chuyển đổi tương tự sang số.

Trong cấu hình mạch thứ hai, chúng ta có thể đo nhiệt độ toàn dải từ -55 ºC đến 150 ºC. Cấu hình mạch này hơi phức tạp nhưng mang lại kết quả cao. Trong trường hợp này, chúng ta phải kết nối một điện trở bên ngoài (R1) để chuyển mức điện áp âm lên dương. Giá trị điện trở bên ngoài có thể được tính toán theo công thức ghi bên dưới cấu hình mạch.

Mặc dù cấu hình mạch đầu tiên không cần điện trở ở phía đầu ra nhưng tôi khuyên bạn nên kết nối điện trở 80 kΩ đến 100 kΩ giữa chân V_{OUT} và chân GND. Khi tôi thực hiện một số thí nghiệm, tôi nhận thấy rằng các số đọc bị dao động và ngõ ra V_{OUT} có hiện tượng thả nổi. Vì vậy, một điện trở giữa V_{OUT} và GND sẽ cố định chân V_{OUT} ở mức thấp và ngăn không cho chân này bị thả nổi.

Các thông số về độ chính xác cho cả hai cấu hình mạch là khác nhau. Mức độ chính xác trung bình là ± 1 ^{o}C cho cả hai cấu hình. Nhưng mức độ chính xác giảm đối với khoảng nhiệt độ từ 2 ^{o}C đến 25 ^{o}C.

2.2.3 Khối cảm biến độ ẩm đất Soil Moisture:



Hình 2. 7 Cảm biến độ ẩm đất Soil Moisture

Cảm biến độ ẩm đất Soil Moisture Sensor được làm từ vật liệu hữu cơ macromolecule thường được sử dụng trong các mô hình tưới nước tự động, vườn thông minh,..., cảm biến giúp xác định độ ẩm của đất qua đầu dò và trả về giá trị Analog, Digital qua 2 chân tương ứng để giao tiếp với Vi điều khiển để thực hiện vô số các ứng dụng khác nhau.

Sơ đồ chân:

VCC	3.3V ~ 5V
GND	GND của nguồn ngoài
DO	Đầu ra tín hiệu số (mức cao hoặc mức thấp)
AO	Đầu ra tín hiệu tương tự (Analog)

Bảng 2. 2 Sơ đồ chân và chức năng của cảm biến độ ẩm đất

Thông số kĩ thuật:

- Điện áp hoạt động: 3.3~5VDC
- Tín hiệu đầu ra:
  Digital: High hoặc Low, có thể điều chỉnh độ ẩm mong muốn bằng biến trở thông qua mạch so sánh LM393 tích hợp.
  Analog: theo điện áp cấp nguồn tương ứng.
- Kích thước: 3x3.16cm

2.2.4 Khối hiện thị LCD:

Hình 2. 8 LCD 20x4

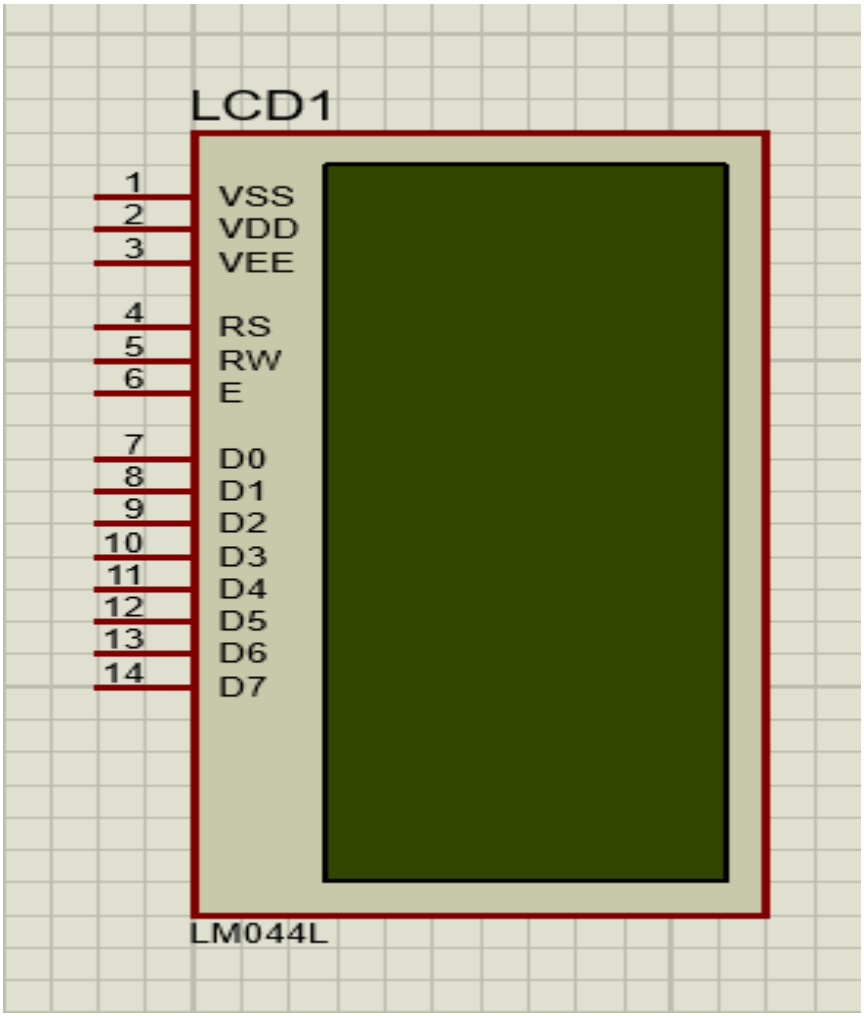
LCD 20x4 là loại màn hình tinh thể lỏng nhỏ dùng để hiển thị chữ hoặc số trong bảng mã ASCII. Mỗi ô của Text LCD bao gồm các chấm tinh thể lỏng, các chấm này kết hợp với nhau theo trình tự "ẩn" hoặc "hiện" sẽ tạo nên các kí tự cần hiển thị và mỗi ô chỉ hiển thị được một kí tự duy nhất.

LCD 20x4 nghĩa là loại LCD có 4 dòng và mỗi dòng chỉ hiển thị được 20 kí tự. Đây là loại màn hình được sử dụng rất phổ biến trong các loại mạch điện.

Thông số kĩ thuật:

- Điện áp hoạt động là 5 V
- Kích thước: 98 x 60 x 13.5 mm
- Chữ đen, nền xanh lá
- Khoảng cách giữa hai chân kết nối là 0.1-inch tiện dụng khi kết nối với Breadboard.
- Tên các chân được ghi ở mặt sau của màn hình LCD hỗ trợ việc kết nối, đi dây điện.
- Có đèn led nền, có thể dùng biến trở hoặc PWM điều chỉnh độ sáng để sử dụng ít điện năng hơn.
- Có thể được điều khiển với 6 dây tín hiệu.
- Giao tiếp với Pic qua chuẩn I2C.

Sơ đồ và chức năng chân của LCD 20x4:



Hình 2. 9 Sơ đồ chân của LCD 20x4

Chân	Ký hiệu	Chức năng
1	V_{ss}	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển.
2	V_{dd}	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với 5V của mạch điều khiển.
3	V_o	Chân này dùng để điều chỉnh độ tương phản của LCD.
		Chân chọn thanh ghi (Register select). Nối chân RS với logic "0" (GND) hoặc logic "1" (Vcc) để chọn thanh ghi.
4	RS	+ Logic "0": Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ "ghi" - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ "đọc" - read) + Logic "1": Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
5	RW	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic "0" để LCD hoạt động ở chế độ ghi, hoặc nối với logic "1" để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào (chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (low-to-high transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0- DB7 khi phát hiện sườn lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7÷14	DB0÷DB7	8 đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này: + Chế độ 8 bit: Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit: Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7.
15	A	Chân Catot, điện áp khoảng U_{ak}=4,2V
16	K	Chân Anot nối đất của đèn Back light

Bảng 2. 3 Bảng chân và chức năng của LCD 20x4

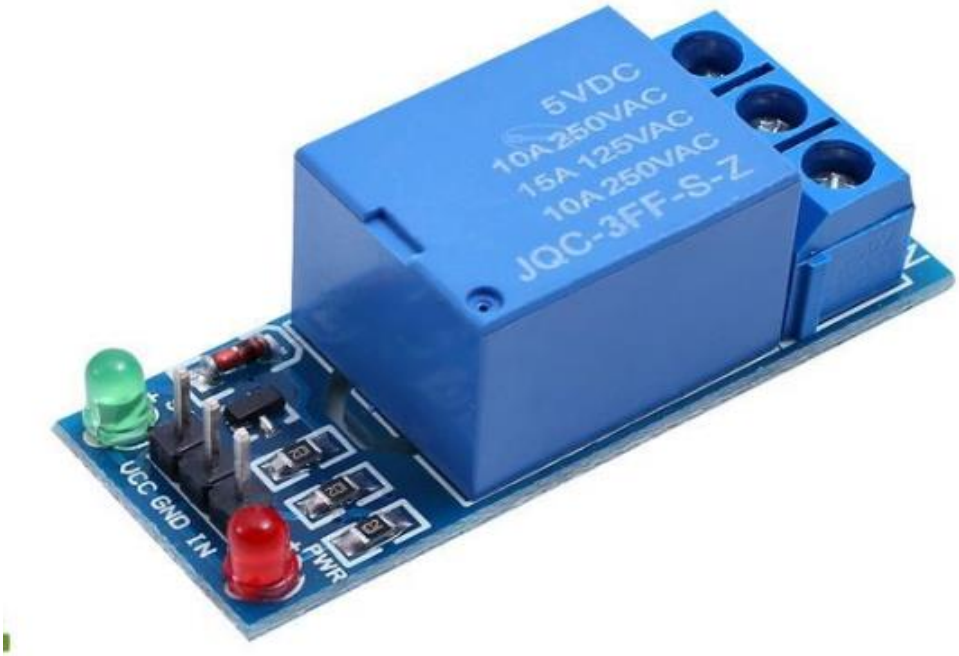
Các chân cấp nguồn: Chân số 1 là chân Vss nối mass (0V), chân thứ 2 là Vdd nối với nguồn +5V. Chân thứ 3 dùng để chỉnh tương phản thường nối với biến trở.

Các chân điều khiển: Chân số 4 là chân RS dùng để điều khiển lựa chọn thanh ghi. Chân R/W dùng để điều khiển quá trình đọc và ghi. Chân E là chân cho phép dạng xung chốt.

Các chân dữ liệu D7÷D0: Chân số 7 đến chân số 14 là 8 chân dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.

2.2.5 Khối thiết bị máy bơm:

Khối thiết bị máy bơm: điển thi có Relay 5VDC làm nhiều vụ nhận lệnh từ vi điều khển để đóng ngắt máy bơm. Ta tìm hiểu sâu hơn về linh kênh này.

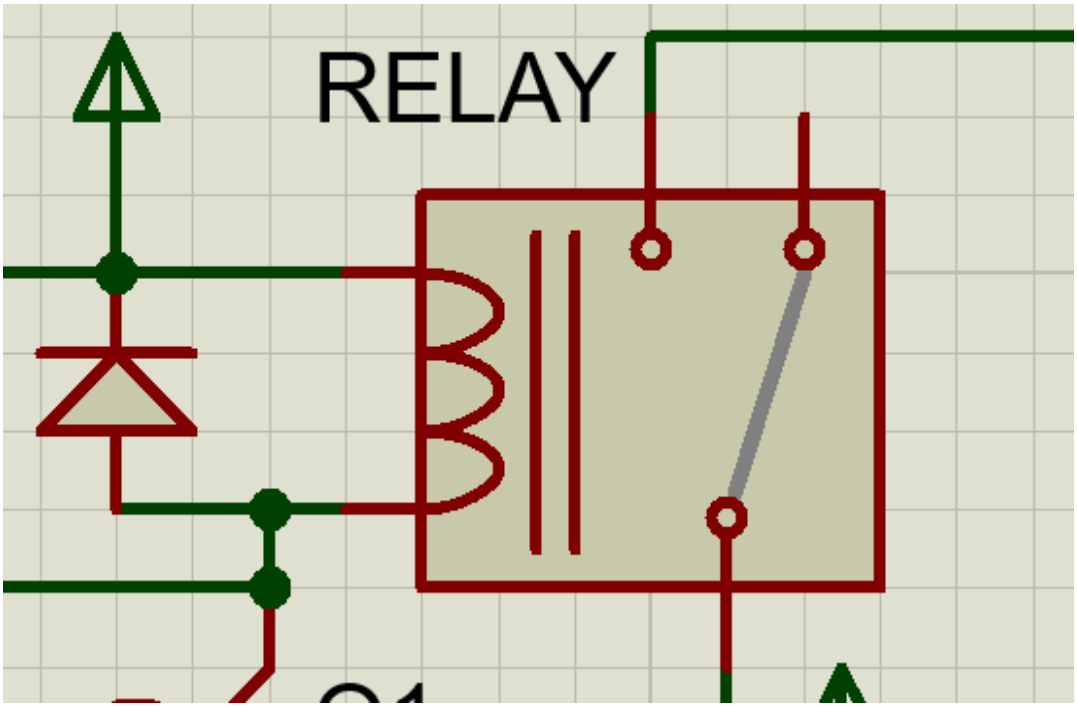


Hình 2. 10 Relay 5V

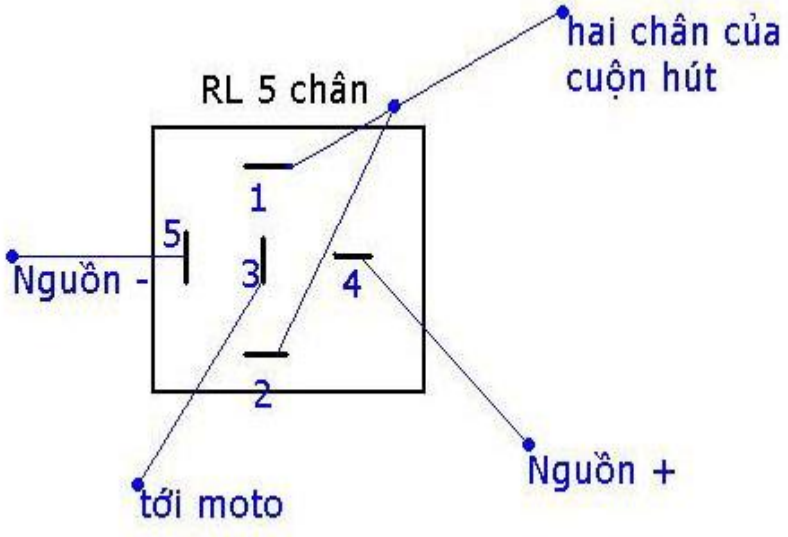
Relay hay còn gọi là rơ – le là tên gọi theo tiếng Pháp, là một công tắc (khóa K) điện từ được vận hành bởi một dòng điện tương đối nhỏ có thể bật hoặc tắt một

dòng điện lớn hơn nhiều. Bản chất của relay là một nam châm điện (một cuộn dây trở thành một nam châm tạm thời khi dòng điện chạy qua nó) và hệ thống các tiếp điểm đóng cắt có thiết kế module hóa dễ dàng lắp đặt. Bạn có thể nghĩ relay sẽ như một loại đòn bẩy điện vậy, khi chúng ta kích nó bằng một dòng điện nhỏ thì nó sẽ bật "đòn bẩy" một thiết bị nào đó đang sử dụng dòng điện lớn hơn nhiều.

Điện áp và dòng điện được relay chuyển mạch sẽ rất khác so với tín hiệu được sử dụng để kích hoạt hoặc cấp điện cho relay. Nói tóm lại rơ-le hay relay là một thiết bị thông dụng, gọn nhẹ, giá thành dễ tiếp cận và được sử dụng rộng rãi trong đời sống hằng ngày của chúng ta.



Hình 2. 11 Sơ đồ chân Relay 5V trong protues



Hình 2. 12 Sơ đồ chân của Relay 5v

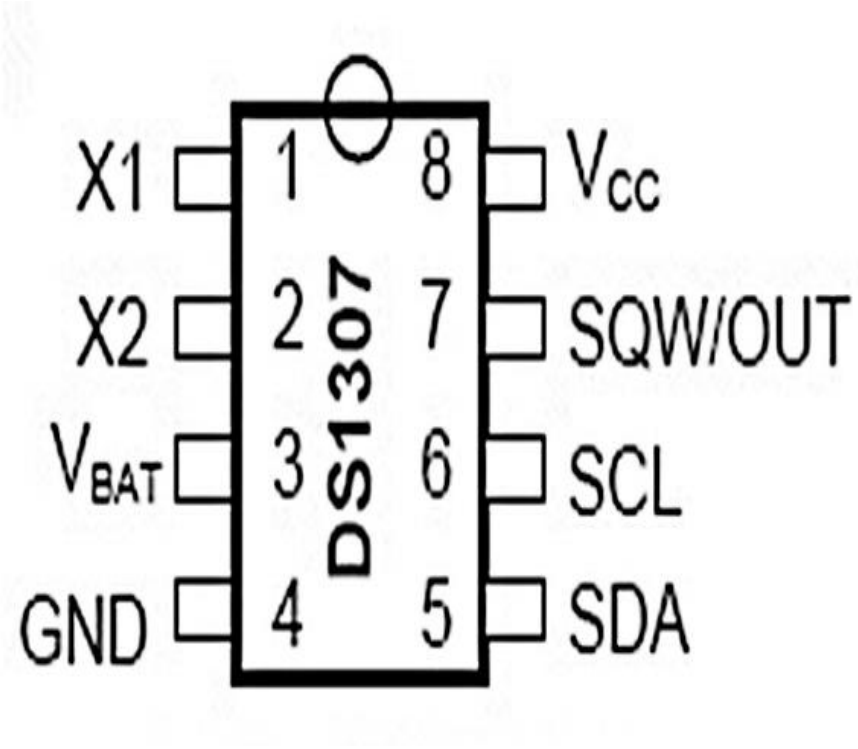
Số chân	Tên chân	Chức năng
1	Coil End 1	Sử dụng kích hoạt (Bật/ Tắt) Relay
2	Coil End 2	Sử dụng kích hoạt (Bật/ Tắt) Relay
3	Common (COM)	Chân kết nối với một đầu bất kỳ của tải.
4	Normally Close (NC)	Chân thường đóng, nếu tải được kết nối với chân NC, tải được kết nối trước khi kích hoạt Relay
5	Normally Open (NO)	Chân thường mở, nếu tải được kết nối với chân NO, tải được không kết nối trước khi kích hoạt Relay

Bảng 2. 4 Tên chân và chức năng của Relay 5V

Thông số kỹ thuật:

- Điện áp kích hoạt (điện áp trên cuộn dây): 5 VDC
- Dòng kích hoạt (dòng điện danh định): 70mA
- Dòng tải AC tối đa: 10A – 250/125 VAC
- Dòng tải DC tối đa: 10A – 30/28 VDC
- Thời gian hoạt động: 10ms
- Chuyển mạch tối đa: 300 lần/phút

2.2.6 Khối thời gian thực:



Hình 2. 13. IC DS07

DS1307 là chip thời gian thực hay RTC (Read time clock), thời gian thực ở đây là tính chính xác về thời gian tuyệt đối cho thời gian mà con người đang sử dụng: Thứ, ngày, tháng, năm, giờ, phút, giây. Thời gian được lưu trữ trong DS1307 cho đến năm 2100.

DS1307 được chế tạo bởi Dallas Semiconductor, chip có cấu tạo bên ngoại khá đơn giản. Chip DS1307 có 8 chân và chúng ta hay dùng là dạng Dip và thứ tự các chân nó được mô tả như hình.

Chip DS1307 có **7 thanh ghi** 8 bit mỗi thanh ghi này chứa: Thứ, ngày, tháng, năm, giờ, phút, giây. DS1307 được đọc thông qua chuẩn truyền thông I2C nên do đó để đọc được và ghi từ DS1307 thông qua chuẩn truyền thông này.

Cấu tạo bên trong DS1307 bao gồm một số thành phần như **mạch nguồn,** mạch dao động, mạch điều khiển logic, **mạch giao diện I2C**, con trỏ địa chỉ và các thanh ghi. Đa số các thành phần bên trong DS1307 là thành phần cứng nên việc sử dụng DS1307 trở nên khá dễ dàng.

Thông số kỹ thuật:

Điện áp cung cấp là 5V chuẩn
X1 và X2cần dao động thạch anh 32.768Khz.
Nguồn nuôi cho chip. Nguồn này từ (2V- 3.5V)
Dải nhiệt độ hoạt động: -40^{o}C ~ 85^{o}C.

2.3 Tổng quan về giao thức I2C

2.3.1 Giới thiệu

Đầu năm 1980 Phillips Semiconductor đã phát triển một chuẩn giao tiếp nối tiếp 2 dây được gọi là I2C. I2C là tên viết tắt của cụm từ InterIntergrated Circuit. Đây là đường Bus giao tiếp giữa các IC với nhau. I2C mặc dù được phát triển bởi Phillips, nhưng nó đã được rất nhiều nhà sản xuất IC trên thế giới sử dụng. I2C trở thành một chuẩn công nghệ cho các giao tiếp điều khiển, có thể kể ra đây một vài tên tuổi ngoài Phillips như: Texas Intrument (TI), MaximDallas, Analog Device, National Semiconductor... Bus I2C được sử dụng là bus giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại vi điều khiển 8051, PIC, AVR, ARM…chip nhớ như: RAM tĩnh (Static Ram), EEPROM, bộ chuyển đổi tương tự số (ADC)…

2.3.2 Đặc điểm

Sau đây là một số đặc điểm quan trọng của giao thức giao tiếp I2C:

- Chỉ cần có hai đường bus (dây) chung để điều khiển bất kỳ thiết bị / IC nào trên mạng I2C

- Không cần thỏa thuận trước về tốc độ truyền dữ liệu như trong giao tiếp UART. Vì vậy, tốc độ truyền dữ liệu có thể được điều chỉnh bất cứ khi nào cần thiết - Cơ chế đơn giản để xác thực dữ liệu được truyền

- Sử dụng hệ thống địa chỉ 7 bit để xác định một thiết bị / IC cụ thể trên bus I2C

-Các mạng I2C dễ dàng mở rộng. Các thiết bị mới có thể được kết nối đơn giản với hai đường bus chung

Cả hai đường bus I2C (SDA, SCL) đều hoạt động như các bộ lái cực máng hở (open drain). Nó có nghĩa là bất kỳ thiết bị / IC trên mạng I2C có thể lái SDA và SCL xuống mức thấp, nhưng không thể lái chúng lên mức cao. Vì vậy, một điện trở kéo lên (khoảng 1 kΩ đến 4,7 kΩ) được sử dụng cho mỗi đường bus, để giữ cho chúng ở mức cao (ở điện áp dương) theo mặc định.

Lý do sử dụng một hệ thống cực máng hở (open drain) là để không xảy ra hiện tượng ngắn mạch, điều này có thể xảy ra khi một thiết bị cố gắng kéo đường dây lên cao và một số thiết bị khác cố gắng kéo đường dây xuống thấp.

Như hình về ta thấy có thể có rất phiều thiết bị (ICs) cùng được kết nối vào một bus I2C, tuy nhiên sẽ không xảy ra chuyện nhầm lẫn giữa các thiết bị bởi môi thiết bị sẽ được nhận ra bởi một địa chỉ duy nhất với một quan hệ chủ tớ (Master/Slave) tồn tại trong thời gian kết nối. Mỗi thiết bị có thể hoạt động như là thiết bị nhận hoặc truyền dữ liệu hay có thể vừa truyền vừa nhận. Hoạt động truyền hay nhận còn phụ thuộc vào việc thiết bị đó là chủ hay tớ.

Một thiết bị hay một IC khi kết nối với bus I2C ngoài một địa chỉ (duy nhất) để phân biệt nó còn được cấu hình là thiết bị chủ hay tớ. Sở dĩ có sự phân biệt này là bởi vì trên một bus I2C thì quyền điều khiển thuộc về thiết bị chủ. Thiết bị chủ nắm vai trò tạo xung đồng hồ cho toàn hệ thống, khi giữa 2 thiết bị chủ - tớ giao tiếp thì thiết bị chủ giữ vai trò chủ động, còn thiết bị tớ giữ vai trò bị động trong việc giao tiếp.

2.3.3 Chế độ hoạt động (tốc độ truyền)

Các bus I2C có thể hoạt động ở ba chế độ, hay nói cách khác các dữ liệu trên bus I2C có thể được truyền trong ba chế độ khác nhau.

- Chế độ tiêu chuẩn (Standard mode):

Chế độ tiểu chuẩn ban đầu được phát hành vào đầu những năm 80. Nó có tốc độ dữ liệu tối đa là 100kbps.

Nó sử dụng 7 bit địa chỉ và 112 bit địa chỉ tớ

- Chế độ nhanh (Fast mode) - Chế độ cao tốc High-Speed (HS) mode:

Tốc độ dữ liệu tối đa được tăng lên đến 400 kbps.

Để ngăn chặn gai tiếng ồn, ngõ vào của thiết bị Fast mode là Schmitt - triggered.

Chân SCL và SDA của một thiết bị tớ I2C ở trạng thái trở kháng cao khi không cấp nguồn.

-Chế độ cao tốc:

Chế độ này được tạo ra chủ yếu để tăng tốc độ dữ liệu lên đèn 36 lần so với chế độ tiêu chuẩn.

Nó cung cấp 1.7Mbps (với Cb = 400pF) và 3.4Mbps (Cb= 100pF). Một bus I2C có thể hoạt động ở nhiều chế độ khác nhau:

Một chủ một tớ (one master - one slave).

Một chủ nhiều tớ (one master- multi slave).

Nhiều chủ nhiều tớ (multi master - multi slave).

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

- **Yêu cầu thiết kế**

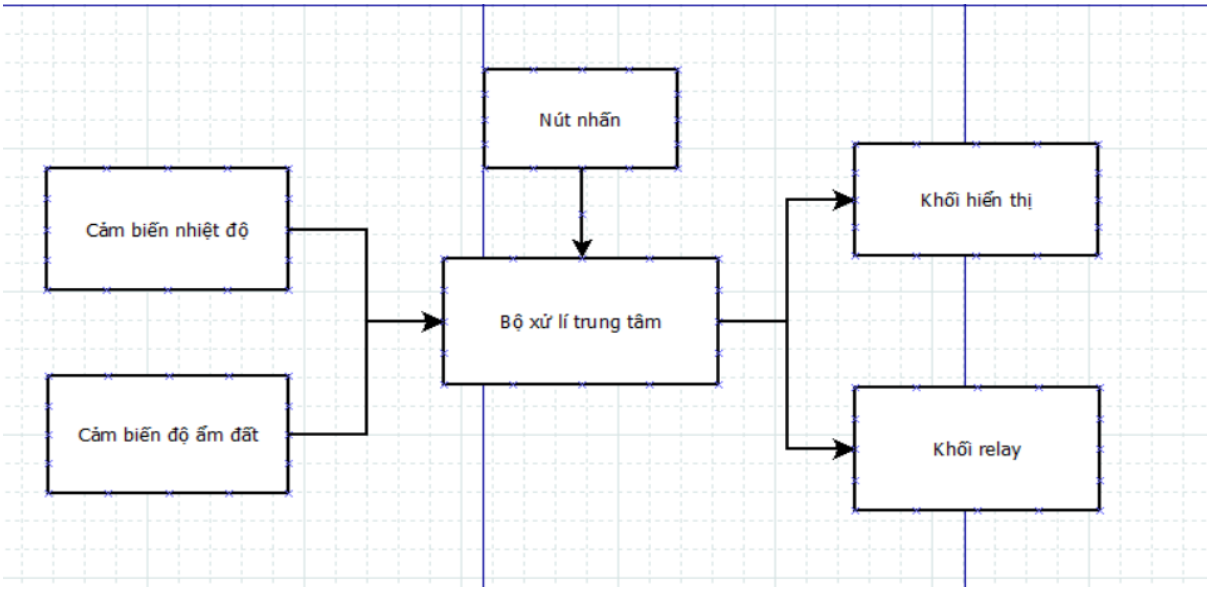
-Bật/tắt được máy bơm nước tự động: Có thể điều khiển máy bơm bằng tay và cả chế độ tự động

-Thiết lập được ngưỡng nhiệt độ và độ ẩm cần tưới nước.

-Hiện thị thông tin, hoạt động màn hình lên LCD: Chế độ hoạt động hiện tại, độ ẩm nhiệt độ thiết độ, độ ẩm nhiệt độ thực tế,…

-Độ chính xác cao và hoạt động ổn định.

- **Sơ đồ khối tổng quát sau khi đã hoàn thành:**



Hình 3. 1 Sơ đồ khối tổng sau khi hoàn thành

Gồm 6 khối cở bản chính:

-Khối cảm biến nhiệt độ: cảm biến nhiệt độ LM35, thu thập dữ liệu nhiệt độ hiện tại và gửi về khối xử lý trung tâm.

-Khối cảm biến độ ẩm đất: cảm biến độ ẩm đất, thu thập dữ liệu độ ẩm hiện tại và gửi về khối xử lý trung tâm.

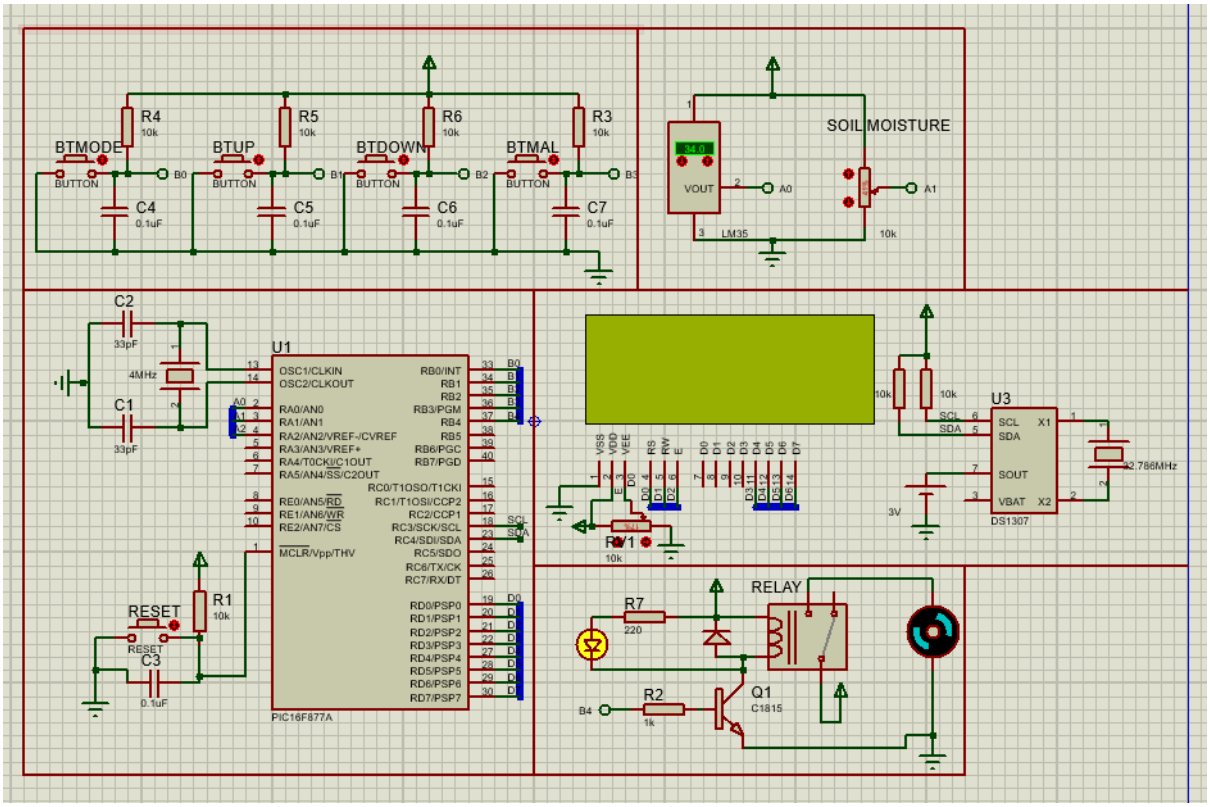
-Khối nút nhấn: gửi yêu cầu điều khiển từ người dùng tới khối xử lý trung tâm

-Khối xử lí trung tâm: Pic16f877a. Làm nhiệm vụ xử lý yêu cầu từ nút nhấn, gửi thông tin nhiệt độ, độ ẩm lên màn hình LCD và điều khiển khối Relay.

-Khối hiển thị: Nhận nhận tin từ bộ xử lí trung tâm rồi hiện thị lên LCD

-Khối relay: Relay nhận tín hiệu điều khiển từ khối xử lý trung tâm và điều khiển bật/tắt thiết bị.

- **Sơ đồ mạch nguyễn lí:**



Hình 3. 2 Sơ đồ mạch nguyên lí

4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM:

Lưu đồ thuật đồ chế độ hoạt động:

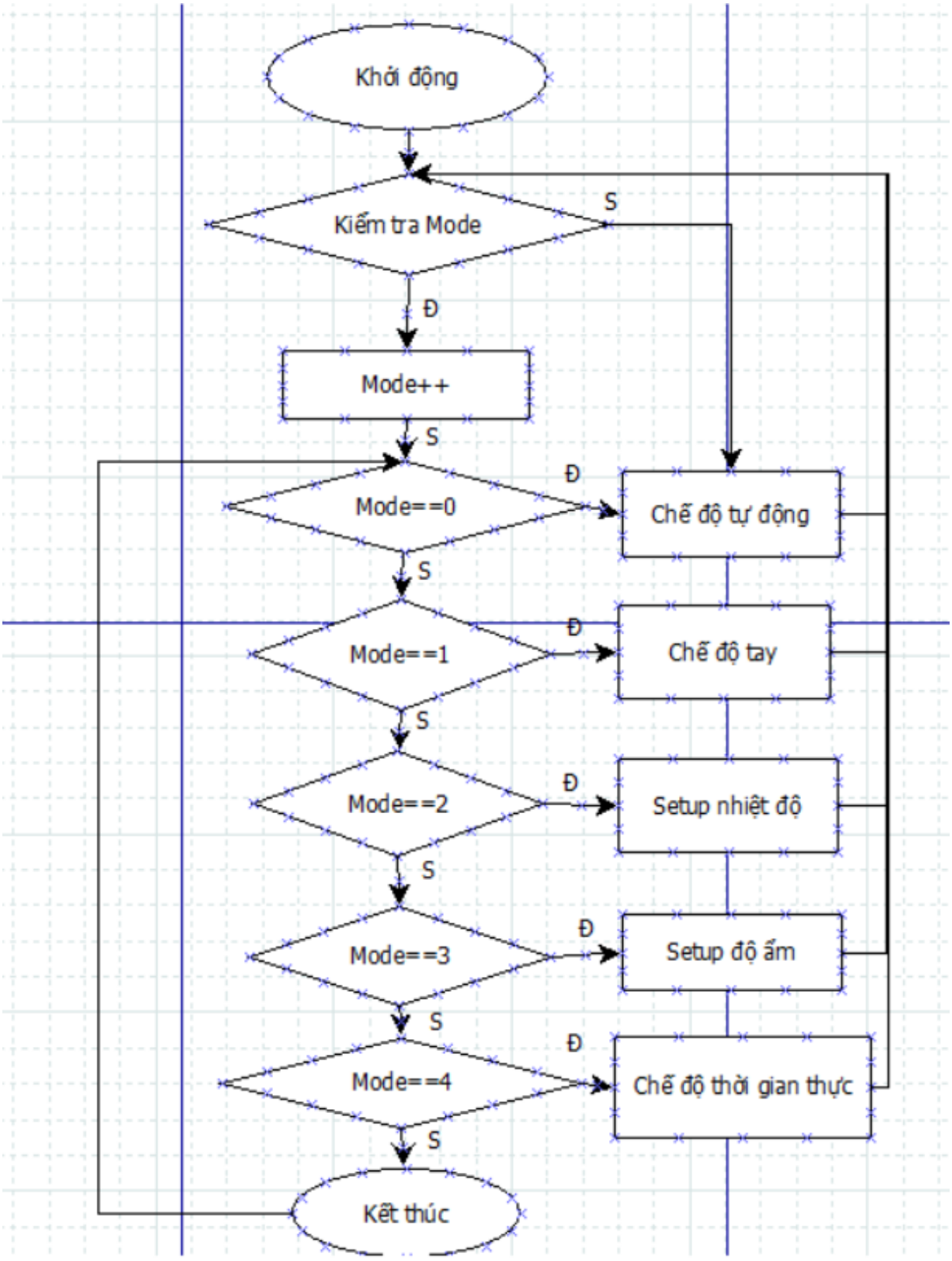
Nhấn nút Mode để thay đổi giữa các chế độ. Khi bắt đầu chương trình, Mode=0, hệ thống sẽ hoạt động theo chết độ tự động. Mỗi lần ấn nút nhấn Mode, mode sẽ tăng 1 đơn vị, cụ thể:

-Khi Mode=0: hoạt động theo chế độ tự động

-Khi Mode=1: hoạt động theo chế độ tay

-Khi Mode=2: chế độ cài đặt nhiệt độ. Lúc này trên LCD sẽ hiện thị yêu cầu thiết lập móc nhiệt độ

-Khi Mode=3: chế độ cài đặt độ ẩm. Lúc này trên LCD sẽ hiện thị yêu cầu thiết lập móc nhiệt độ.

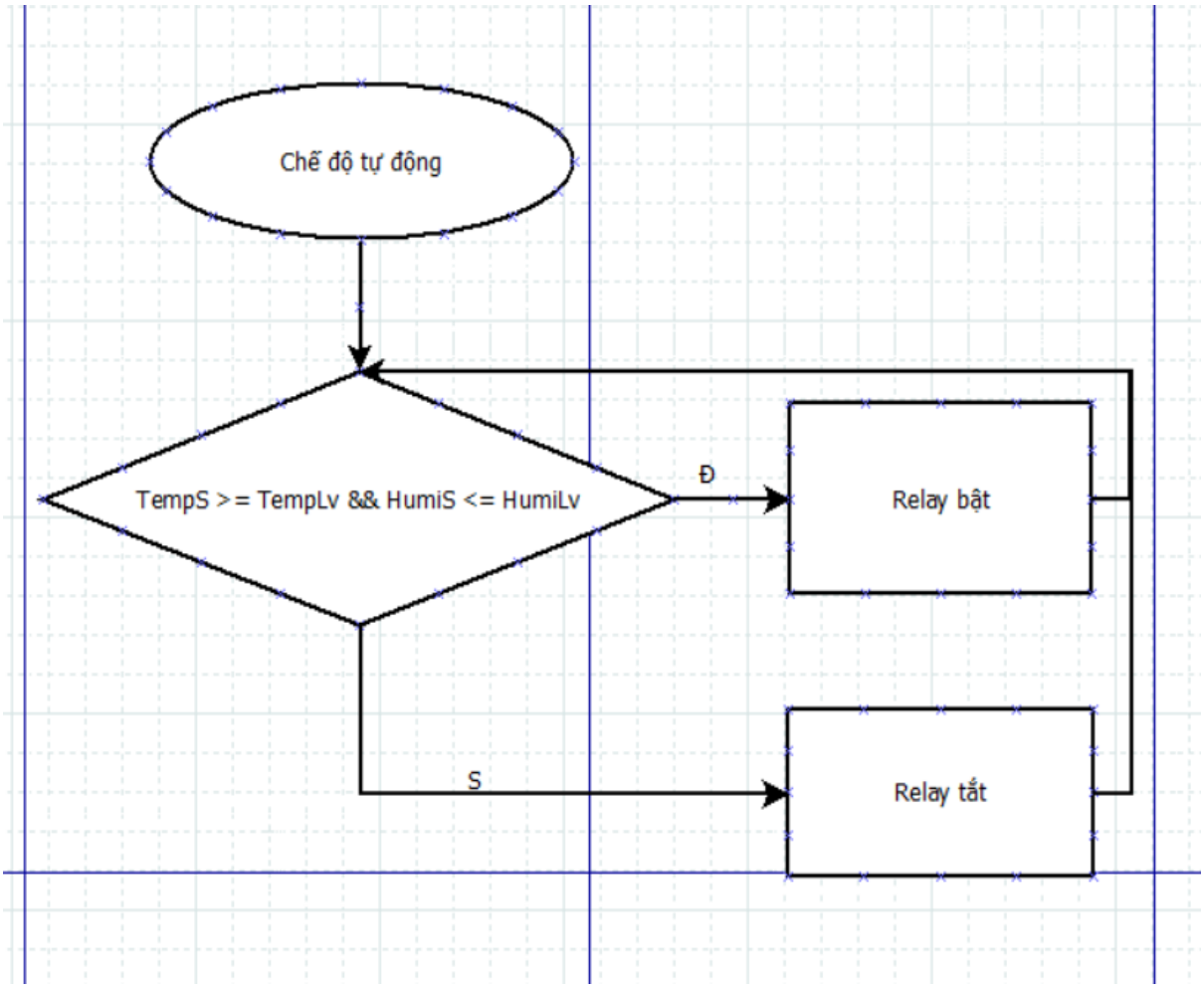


Hình 4. 1 Lưu đồ giải thuật chế độ hoạt động

Ở chế độ tự động: hệ thống sẽ kiểm tra nhiệt độ và độ ẩm cảm biến và nhiệt đô và độ ẩm được setup để bắt hay tắt máy bơm.

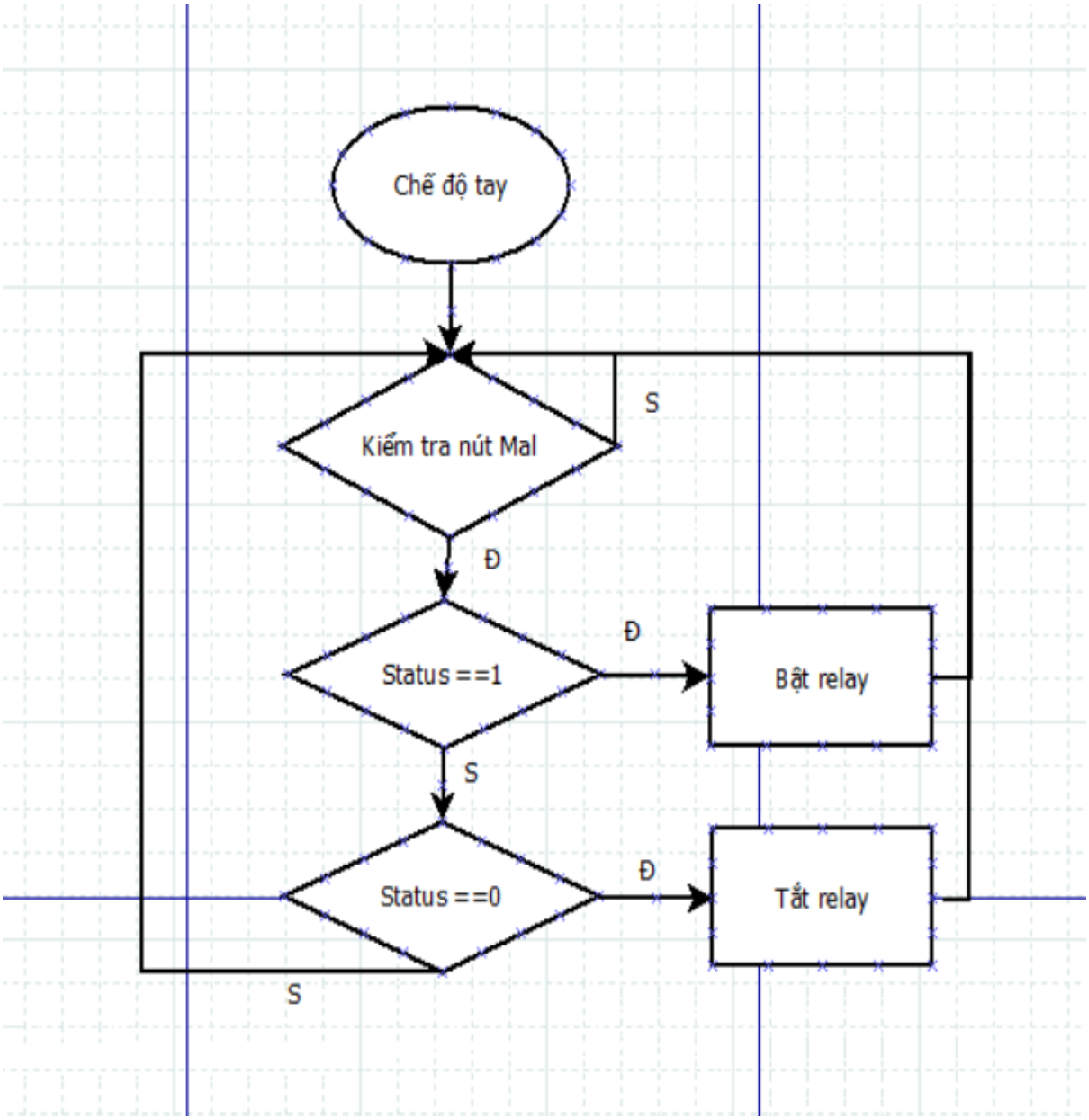
-Nhiệt độ cảm biến lơn hơn nhiệt độ cài đặt hoặc đồ ảm cảm biến bé hơn nhiệt độ cài đặt thì máy bơm sẽ bật.

-Ngược lại, máy bơm tắt.



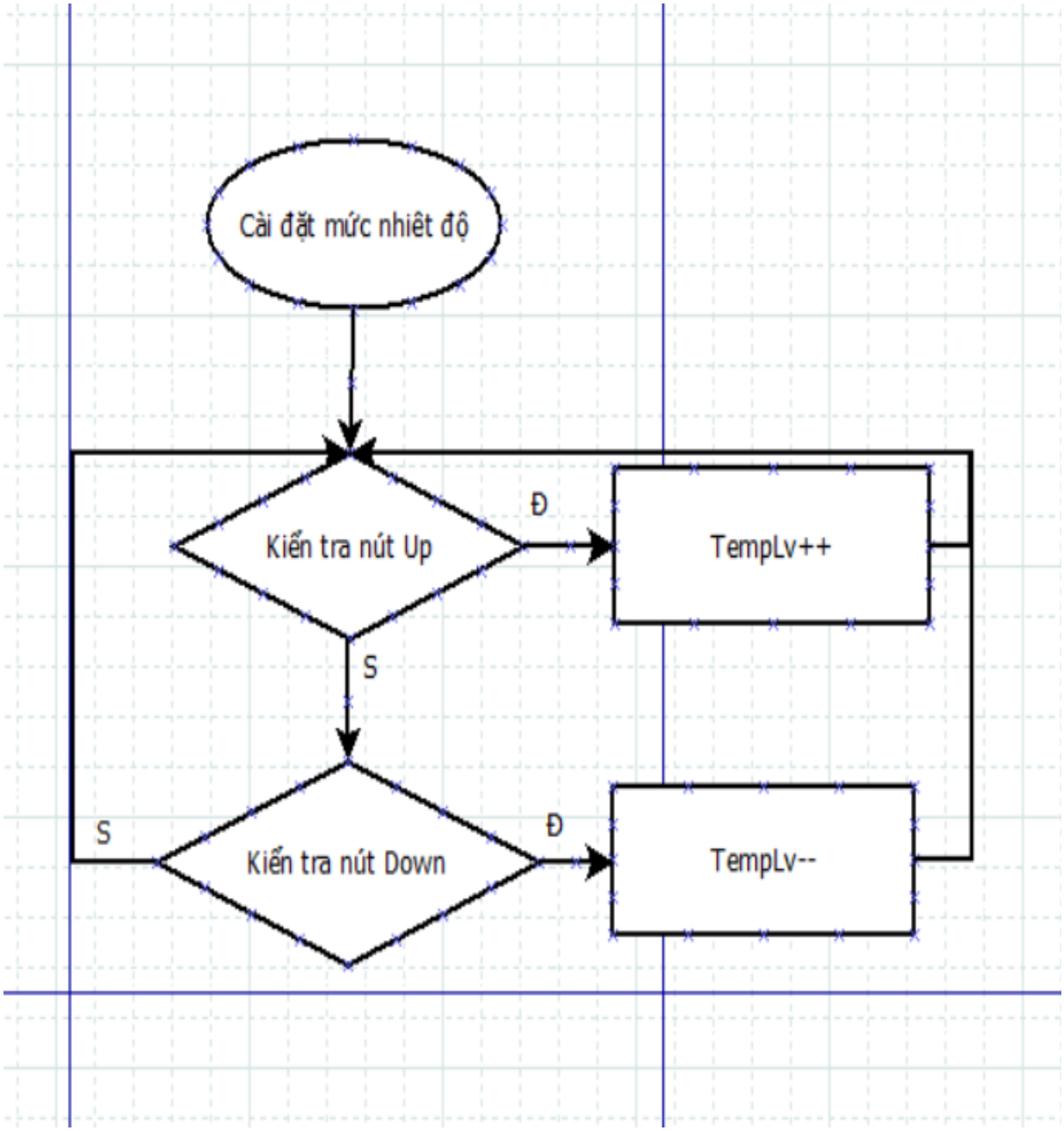
Hình 4. 2 Sờ đồ giải thuật ở chế độ tự động

Ở chế độ tay: hệ thống được điều khiển bật tắt theo nút nhấn MAL: Nhấn Mal để tắt máy bơm, và lần nữa để bât máy bơm.



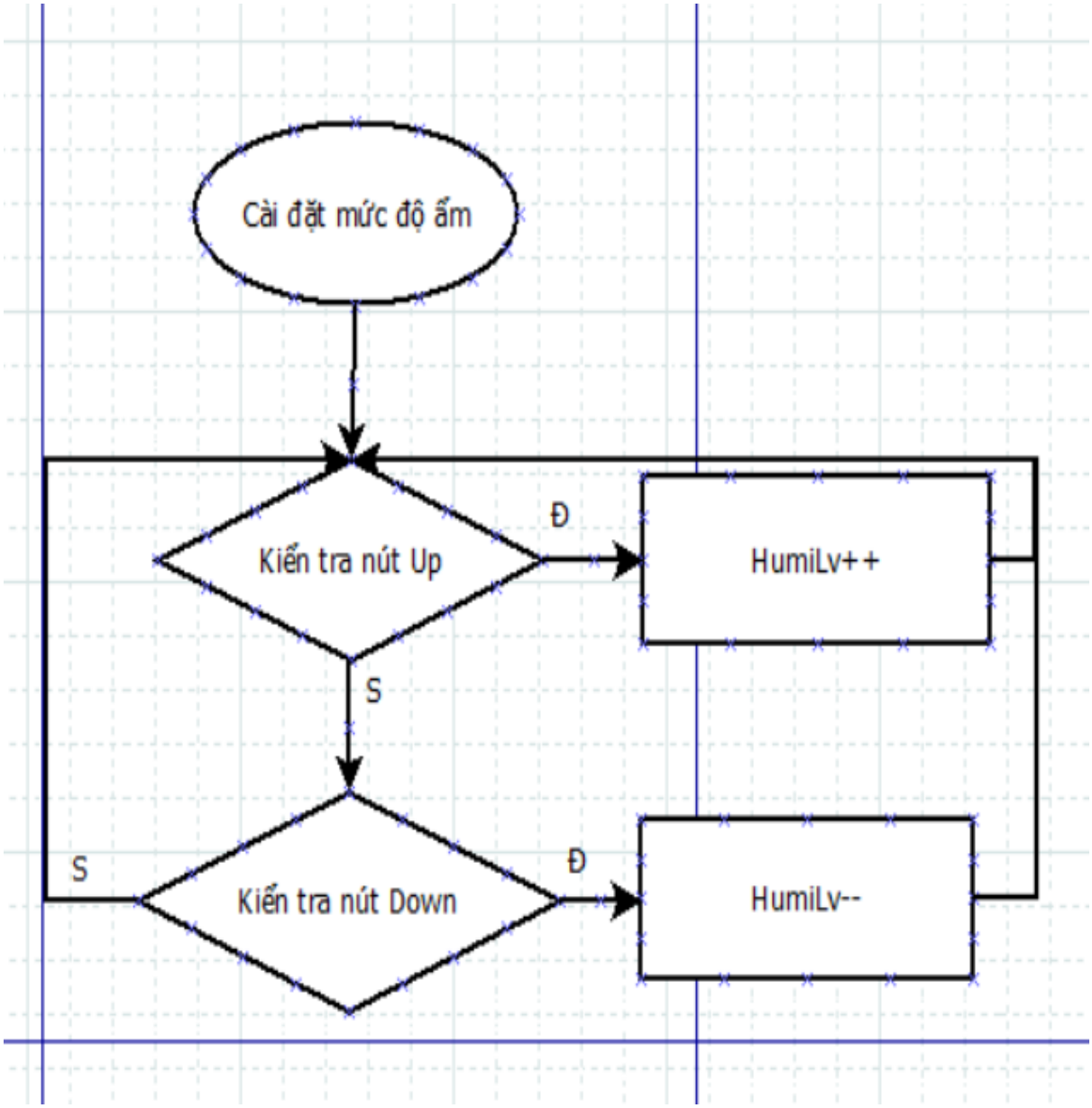
Hình 4. 3 Sờ đồ giải thuật ở chế độ tay

Trong chế độ cài đặt nhiệt độ: Ta ấn nút Up để tăng nhiệt độ cài đặt, mỗi lần ấn tương với 1 độ C, tương tự để nút Down để giảm nhiệt đồ



Hình 4. 4 Sơ đồ giải thuật hoạt động ở chế độ cài đặt nhiệt độ

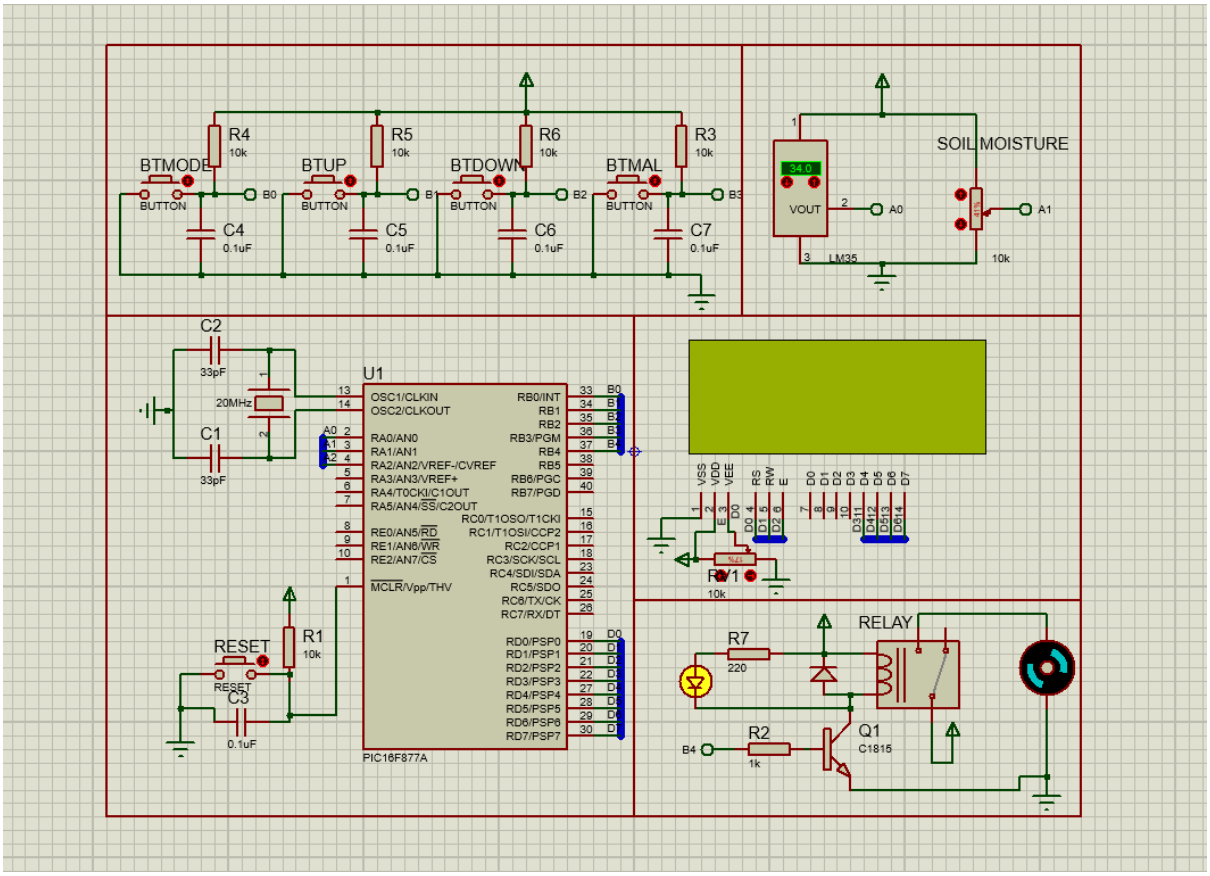
Trong chế độ cài đặt độ ẩm đất: Ta ấn nút Up và Down để tăng giảm nước cài đặt.



Hình 4. 5 Sơ đồ giải thuật hoạt động ở chế độ cài đặt độ ẩm

5. KẾT QUẢ THỰC HIỆN

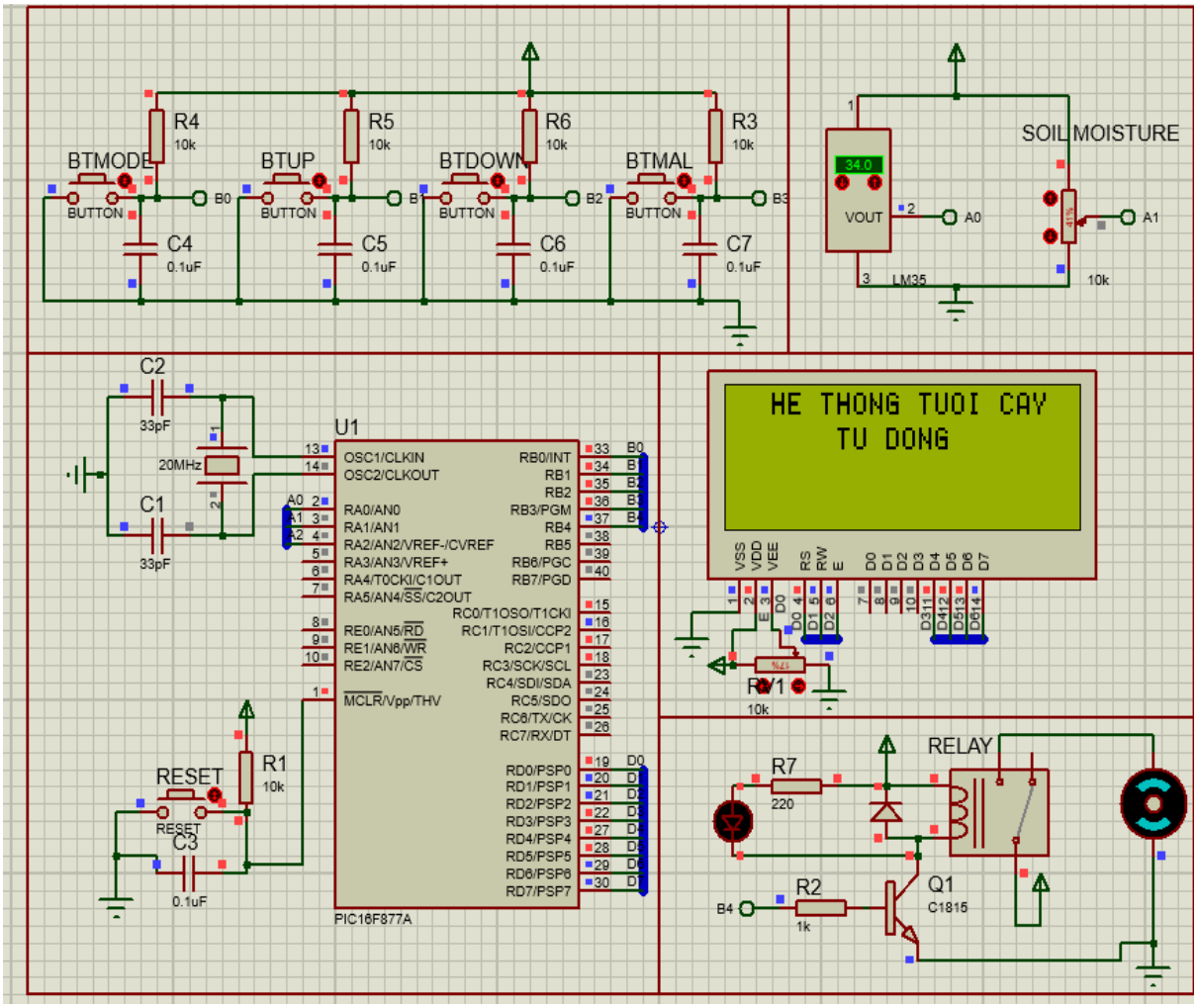
Mạch mô phỏng:



Hình 5. 1 Mạch mô phỏng

Kết quả mô phỏng”

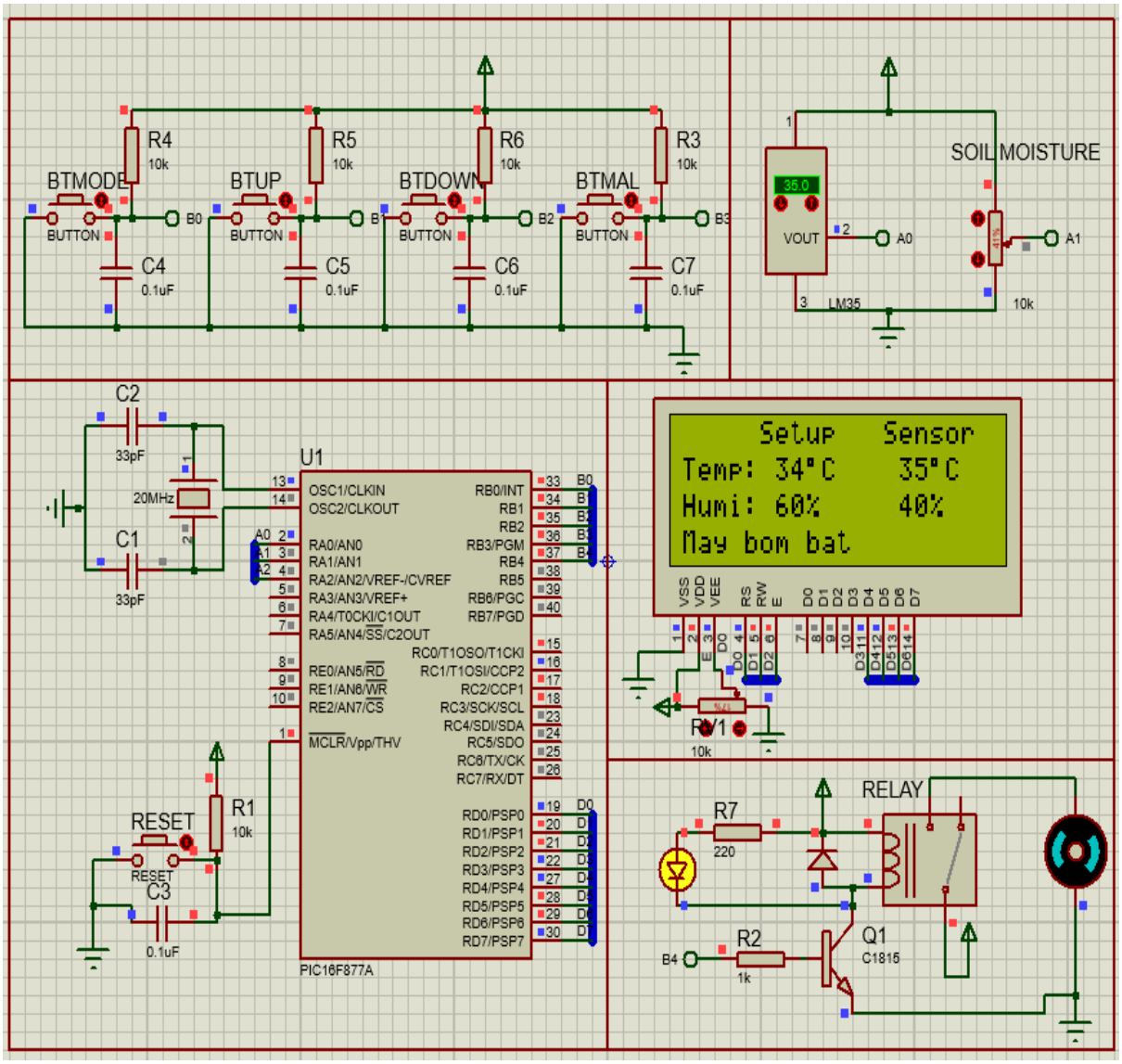
-Khi mạch khởi động



Hình 5. 2 Bộ tưới hoạt động

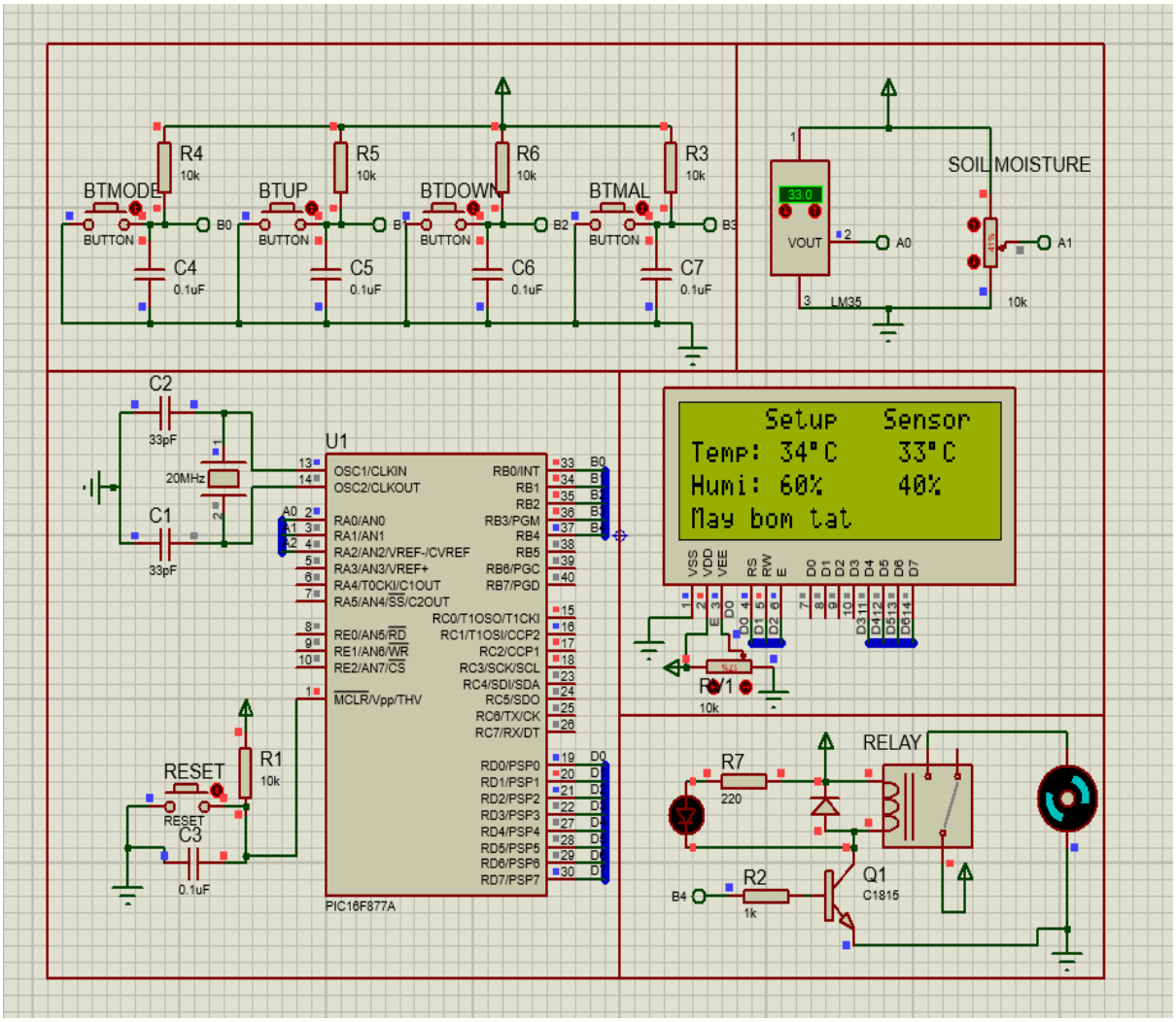
-Chế độ hoạt đông tự động khi Mode=0

+Khi nhiệt độ môi trường lớn hơn hoặc bằng cảm biến hoặc đổ ẩm mội trường lớn hơn hoặc bằng cảm biến thì máy bơm bật



Hình 5. 3 Máy bơm bật ở chế độ tự động

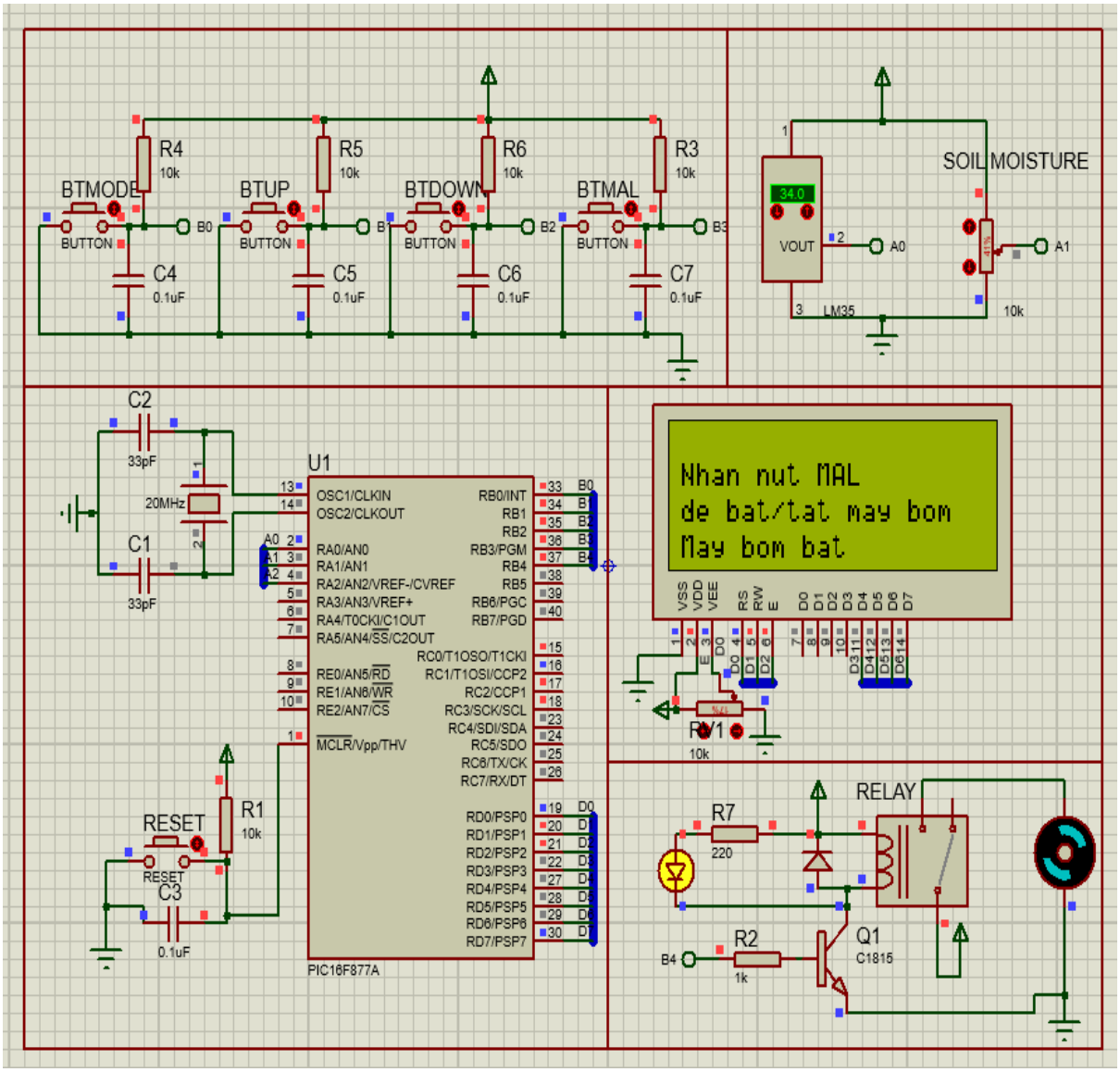
+Khi nhiệt độ cảm biến bé hơn nhiệt độ setup và độ ẩm cảm biến lớn hơn độ ẩm setup thì máy bơm tắt.



Hình 5. 4 Máy bơm tắt ở chế độ tự động

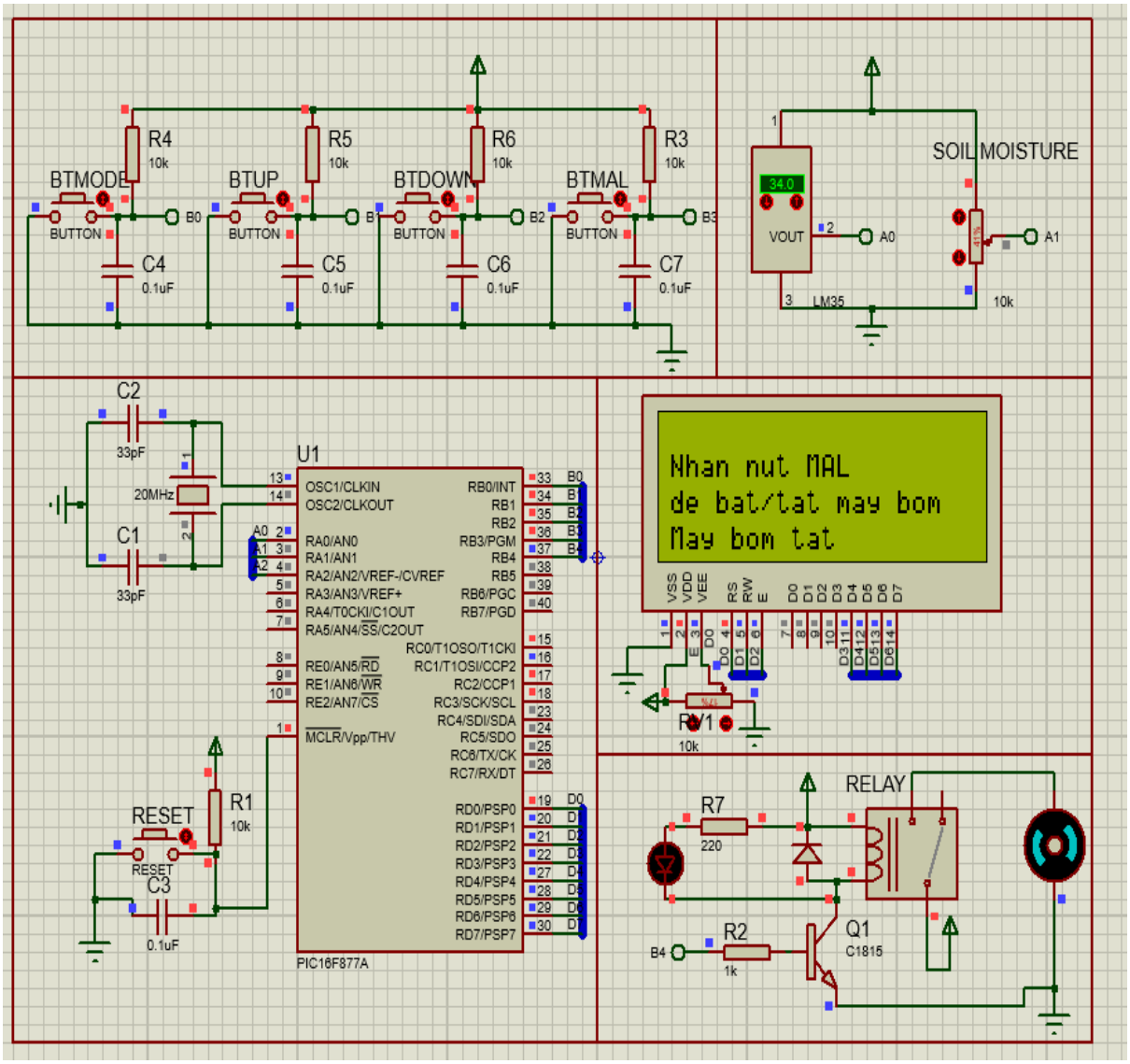
-Chế độ hoạt độ bằng tay khi Mode=1:

+Máy bơm bật



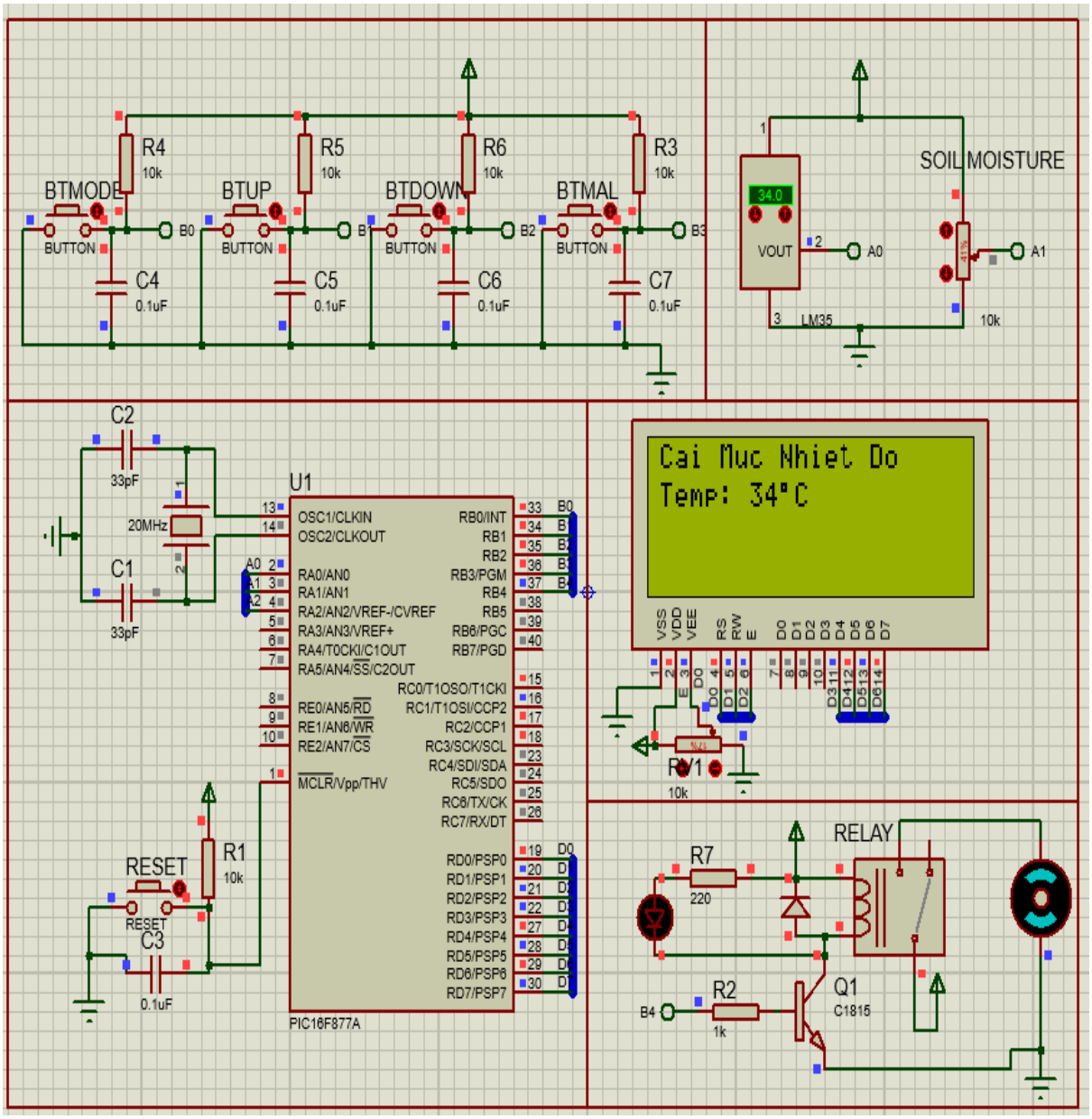
Hình 5. 5 Máy bơm bật ở chế độ tay

+Nhấn Mab, máy bơm tắt:



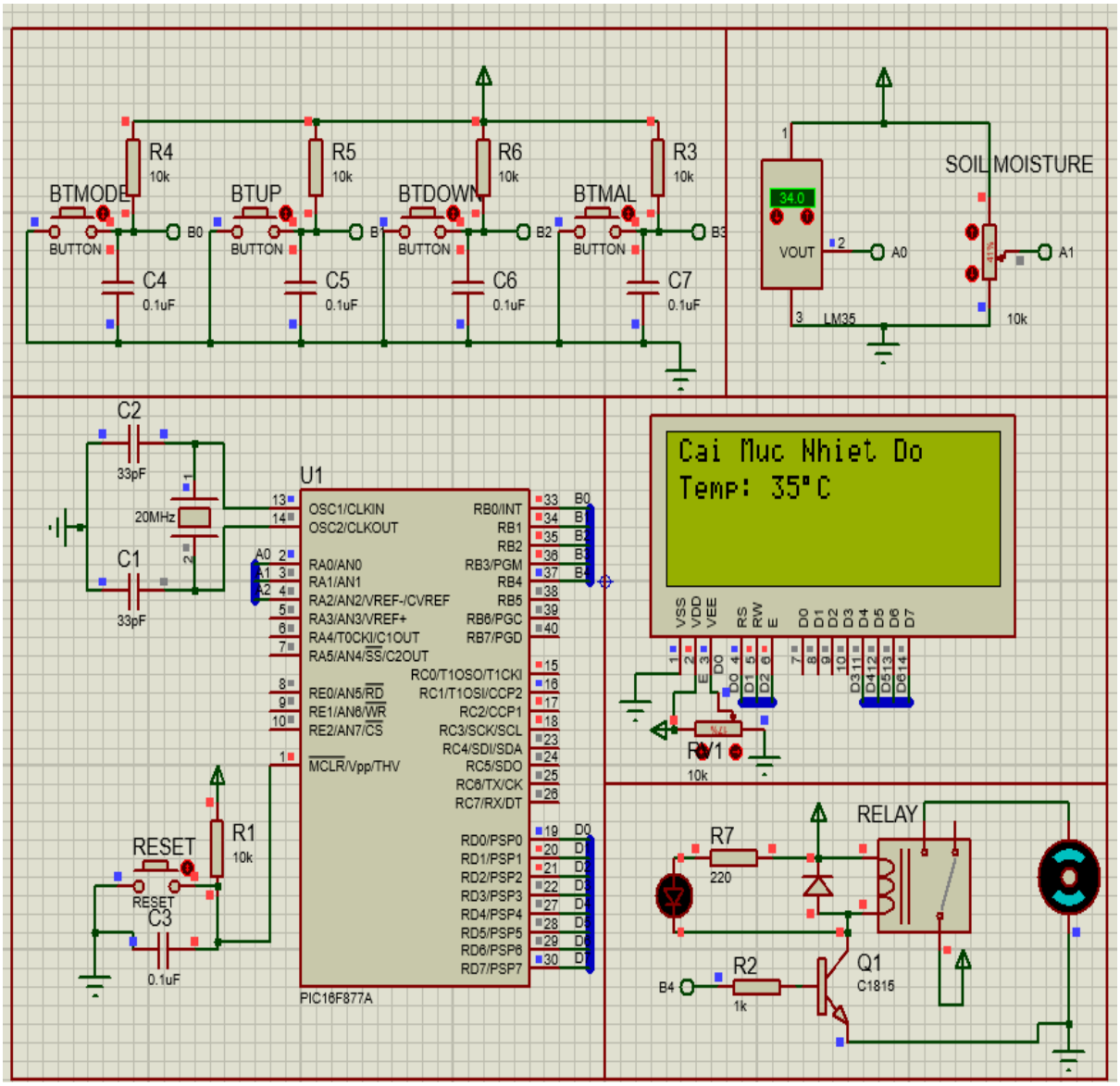
Hình 5. 6 Máy bơm bật ở chế độ tay

Ở chế độ cài đặt nhiệt độ khi mode=2



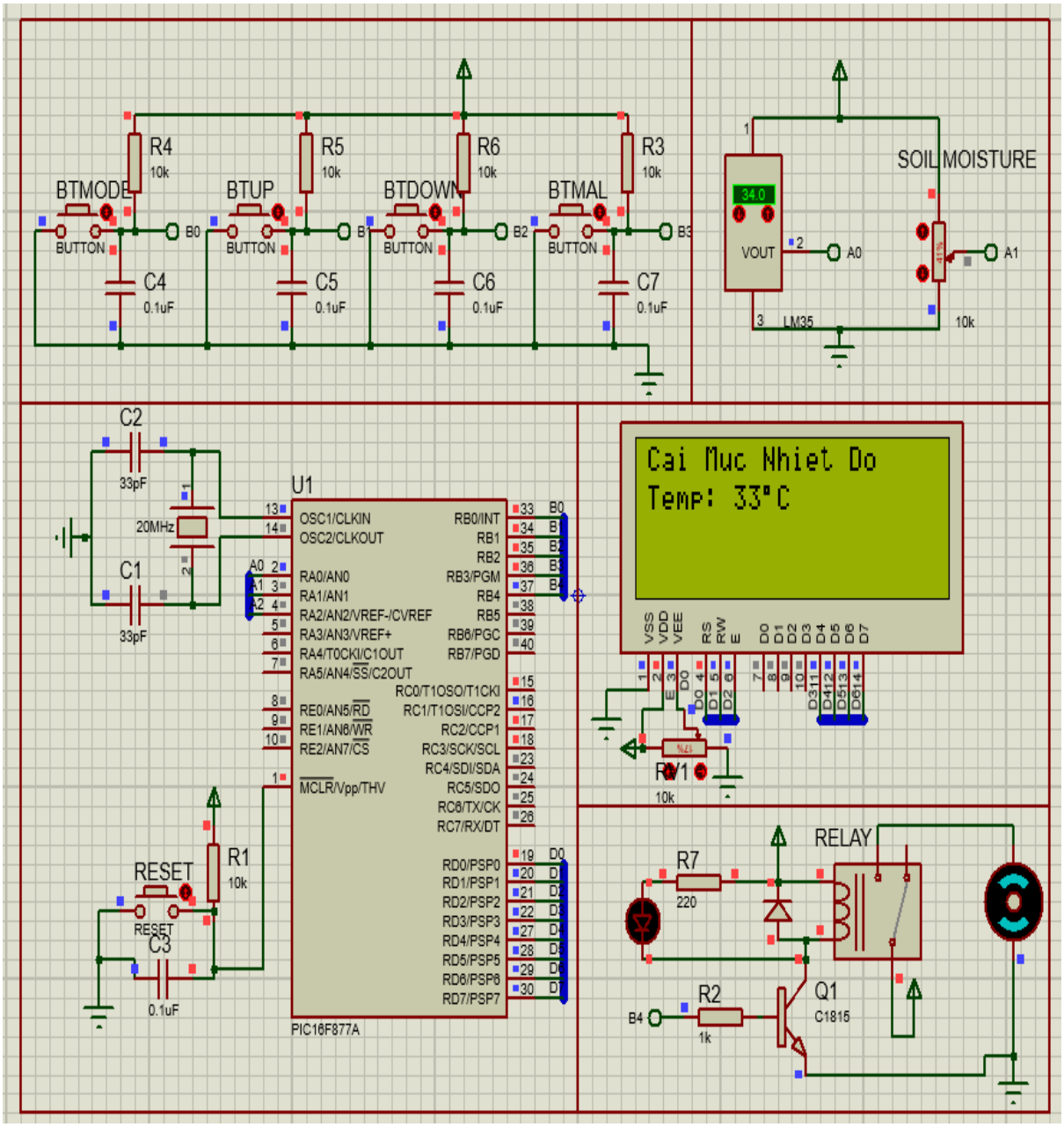
Hình 5. 7 Chế độ cài đặt nhiệt độ

+Nhấn nút UP để tăng nhiệt độ cài đặt lên 1



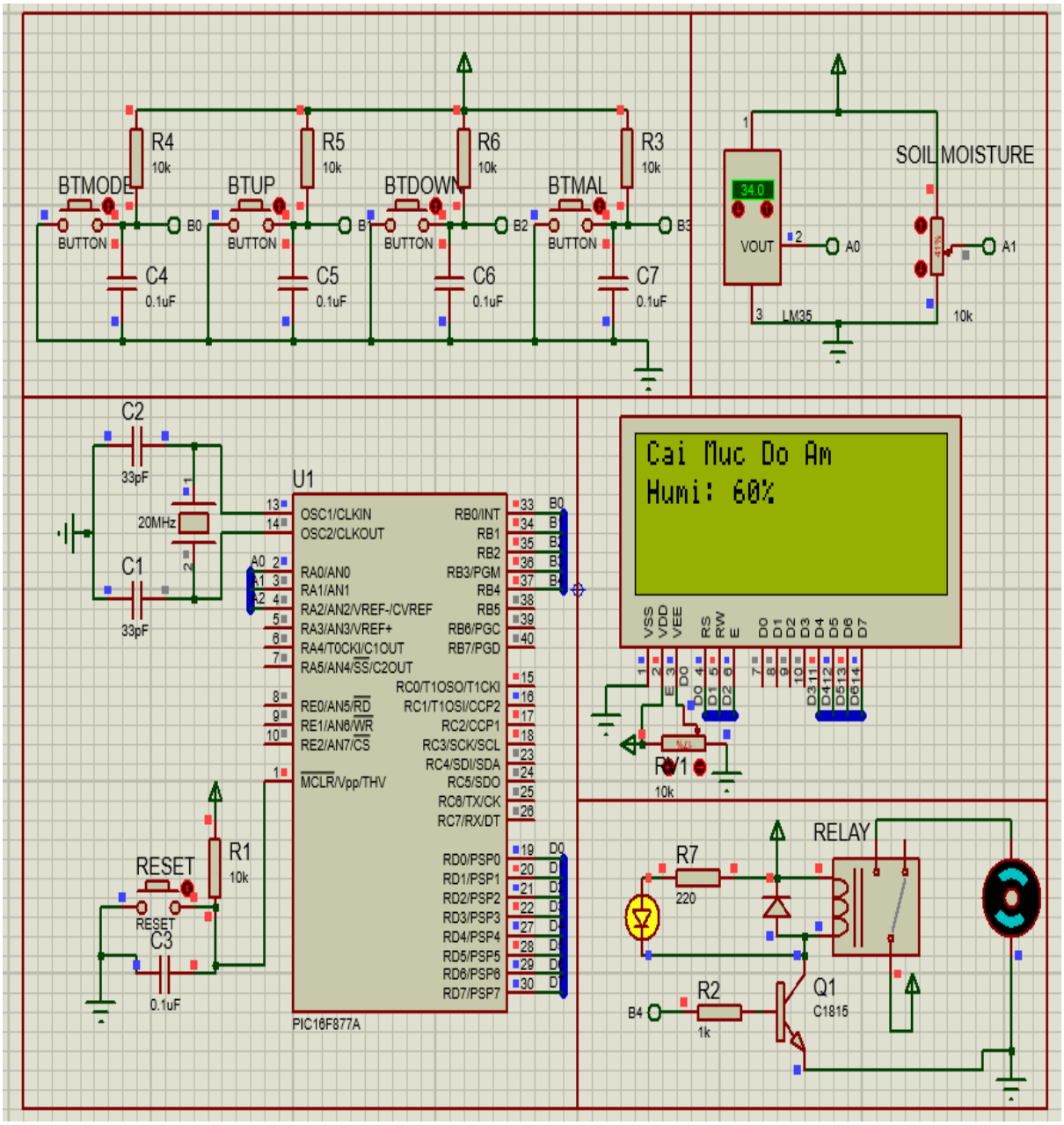
Hình 5. 8 Nhấp Up để tăng nhiệt độ cài đặt lên 1

+Nhấp nút Down để giảm nhiệt độ xuống 1



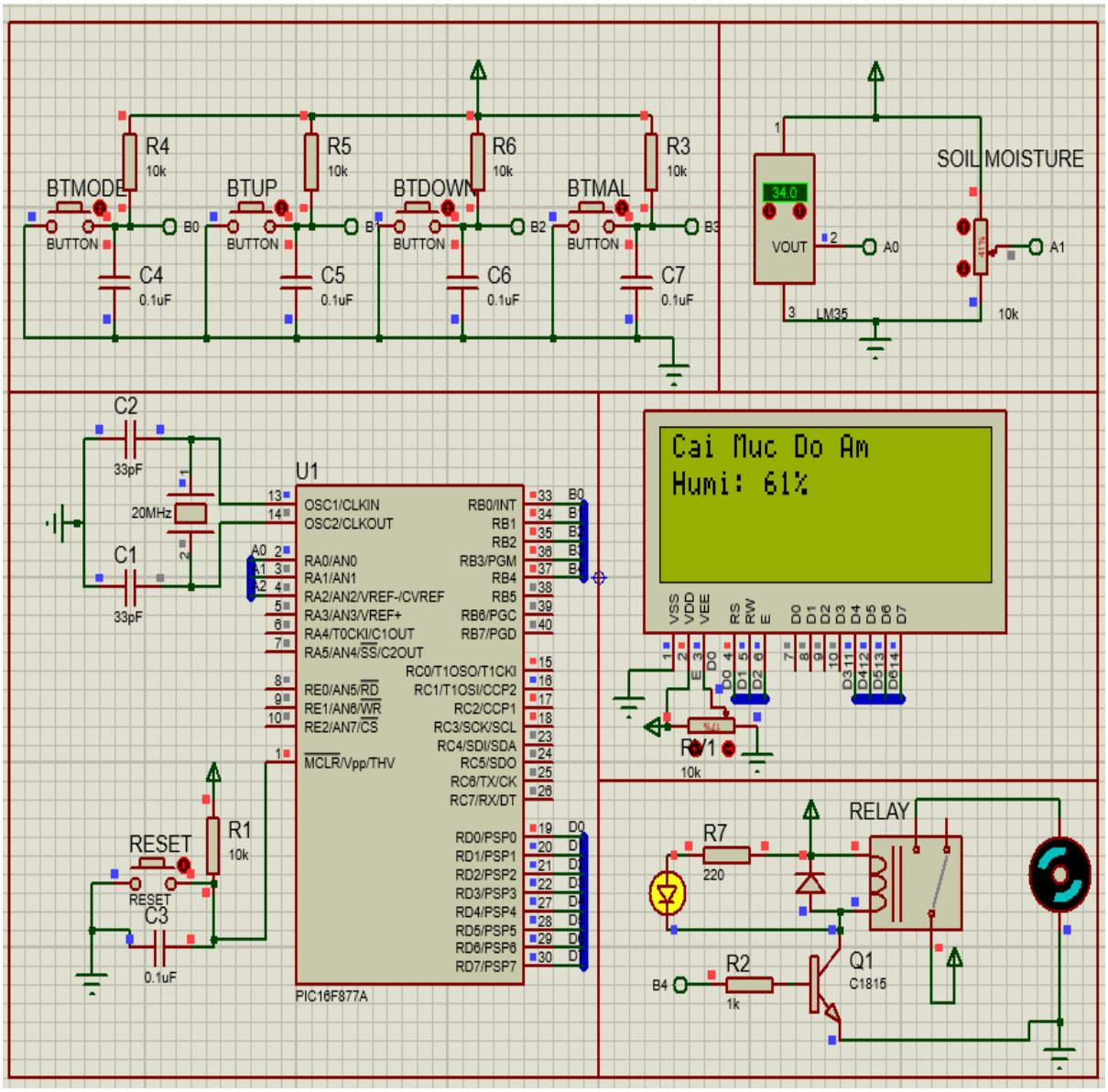
Hình 5. 9 Nhấn DOWN để giảm nhiệt độ cài đặt xuống 1

Ở chế độ cài đặt độ ẩm khi Mode=3



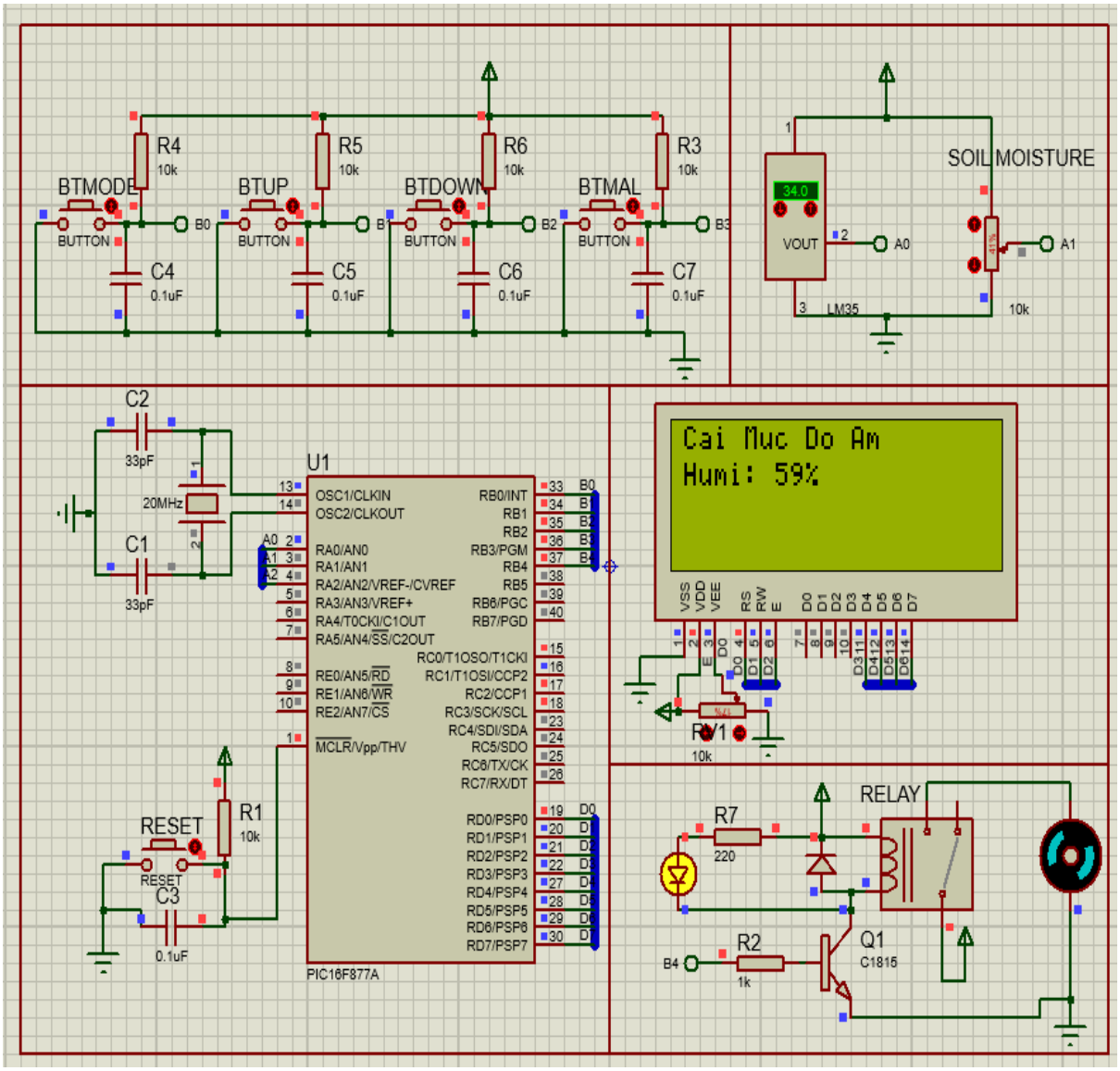
Hình 5. 10 Hoạt động ở chế độ cài đặt độ ẩm

+Nhấn nút Up để tăng độ ẩm lên 1:

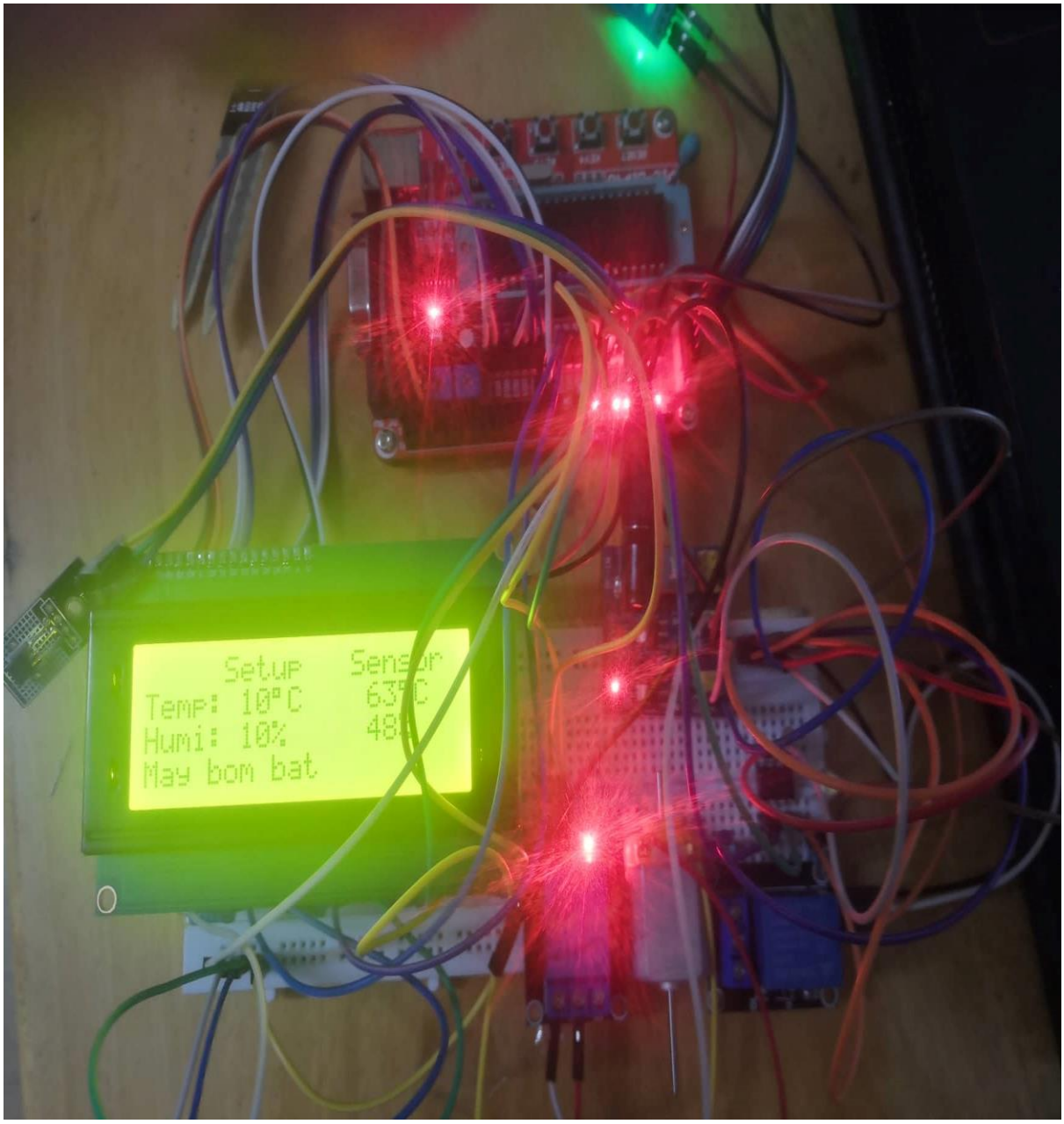


Hình 5. 11 Nhấp UP để tăng độ ẩm cài đặt lên 1

+Nhấn núp Down để giảm độ ẩm xuống 1



Hình 5. 12 Nhập DOWN để giảm độ ẩm cài đặt xuống 1



Hình 5. 13 . Mạch mô hình

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Sau khi hoàn thành được đồ án này đã giúp em nắm rõ hơn về lập trình vi điều khiển, giao tiếp vi điều khiển với các ngoại vi, các loại cảm biến cà tuy duy để phát triển 1 hệ thống. Ngoài ra biết cách sư dụng các công cụ hỗ trợ lập trình và mô phỏng như: Pic-C Compiler, Protues,….

Những ưu điểm của đồ án này:

-Dễ thực đơn, đơn giản

-Chính xác cao, hiệu quả

Những khuyết điểm của đồ án này:

-Chưa hoàn thiện tối ưu để làm ra sản phẩm thực tế

-Chưa biết cách ứng dụng vào truyền thông IOT.

6.2 Hướng phát triển

Có thể sử dụng các họ vi điều khiển khác, mở rộng các chức năng như quạt, đèn và có thể cập nhật thời gian thực (đề cử sử dụng DS1307). Thêm nhiều chế độ tưới như tưới ít, nhiều, phục sương tùy theo điều kiện nhiệt đô và độ ẩm khác/

Điều khiển đồng thời nhiều máy bơm, chứ không chỉ dừng lại ở việc dừng 1 máy bơm.

Ứng dụng mạng truyền thông IOT vào, có thể sử dụng ESP8266 để điều khiển hệ thống qua điện thoại.

Tóm lại, tất cả các tính năng phát triển ở trên ghóp ở để hoàn chỉnh và đưa ra bộ tưới cây tự động thực tế hoàn chỉnh nhất .

7. TÀI LIỆU THAM KHẢO

[1] Lê Đức Hạnh, “Lập trình vi điều khiển họ Pic và ứng dụng”, Nhà xuất bản đại học quốc gia thành phố hồ chí minh, 2019.

[2] Cảm biến nhiệt độ LM35, https://dientuviet.com/cam-bien-nhiet-do-lm35/

[3] Cảm biến độ ẩm đất Soil Moisture Sensor, https://hshop.vn/products/cam-bien-do-am-dat-2

[4] LCD 20x4, https://suachualaptop24h.com/linh-kien-laptop/thong-so-ki-thuat-cua-lcd-20x4-n5214.html

[5] Pic 16F877A, https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

[6] Relay 5V, https://www.dosangtao.vn/module-relay-5v-1-kenh

8. PHỤ LỤC

Mã nguồn chương trình: #include <16F877A.h>

```
#device *=16 adc=10

#fuses HS,NOWDT,NOPROTECT,NOLVP

#use delay(clock=4M)

#use I2C(MASTER, I2C1, SLOW = 100000, STREAM = DS1307_STREAM)

#include <lcd.h>

#include <DS1307.c>  // include DS1307 driver source file

////////////////////////////////

#define BTMode input(PIN_B0)

#define BTUp   input(PIN_B1)

#define BTDown input(PIN_B2)

#define BTMal  input(PIN_B3)

#define Relay1  PIN_B4

#define Relay2  PIN_B5

////////////////////////////////

RTC_Time *mytime;

int8 Mode;

unsigned int8 TempLv, HumiLv;//Nhiet Do, Do Am cai dat

unsigned int8 TempS, HumiS;//Nhiet Do, Do Am tu cam bien

int1 Status = 0;

////////////////////////////////

#int_ext
```

```
_NgatRB()
{
  Mode++;
 lcd_putc('\f');
 delay_ms (200);
  if (Mode==5)
     Mode = 0;
  return Mode;
}
////////////////////////////////
void _ConRelay(unsigned int8 TempLv, unsigned int8 HumiLv, unsigned int8 TempS,
unsigned int8 HumiS);
void _BtMode(int8 Mode);
int16 _ReadADC(unsigned int8 Pin);
int8 _ReadTemp(unsigned int16 GiaTriADC0);
int8 _ReadHumi(unsigned int16 GiaTriADC1);
////////////////////////////////.
void main()
{
   unsigned int16 GiaTriADC0,GiaTriADC1;
   SET_TRIS_A(0xff);
   SET_TRIS_D(0x00);
   SET_TRIS_B(0xff);
```

```
//
enable_interrupts(int_ext);
ext_int_edge(h_to_l);
enable_interrupts(global);
Mode = 0;
//
TempLv = read_eeprom(0x02);
HumiLv = read_eeprom(0x08);
//
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
//
lcd_init();
lcd_putc('\f');
lcd_gotoxy(1,1);
printf(lcd_putc,"  HE THONG TUOI CAY ");
lcd_gotoxy(1,2);
printf(lcd_putc,"      TU DONG");
delay_ms(2000);
lcd_putc('\f');

while(TRUE)
{
```

```
GiaTriADC0 = _ReadADC(0);
GiaTriADC1 = _ReadADC(1);
//Nhiet do, Do am tu cam bien
TempS = _ReadTemp(GiaTriADC0);
HumiS = _ReadHumi(GiaTriADC1);

switch(Mode)
{
  case 0:
    lcd_gotoxy(1,1);
    printf(lcd_putc,"    Setup   Sensor");
    lcd_gotoxy(1,2);
    printf(lcd_putc,"Temp: ");
    lcd_gotoxy(1,3);
    printf(lcd_putc,"Humi: ");
    lcd_gotoxy(7,2);
    printf(lcd_putc,"%02d",TempLv);
    lcd_putc(223);
    lcd_putc("C    ");
    lcd_gotoxy(7,3);
    printf(lcd_putc,"%02d",HumiLv);
    lcd_putc("%     ");
```

```
    lcd_gotoxy(15,2);
    printf(lcd_putc,"%02d",TempS); lcd_putc(223); lcd_putc(67);
    lcd_gotoxy(15,3);
    printf(lcd_putc,"%02d",HumiS); lcd_putc("% ");

    _ConRelay(TempLv, HumiLv, TempS, HumiS);
    break;

  case 1:
    lcd_gotoxy(1,1);
    printf(lcd_putc,"                    ");
    lcd_gotoxy(1,2);
    printf(lcd_putc,"Nhan nut MAL      ");
    lcd_gotoxy(1,3);
    printf(lcd_putc,"de bat/tat may bom");
    if(BTMal == 0)
    {
      delay_ms(20);
      while(BTMal == 0);
      if(Status == 1)
      {
        Status = 0;
```

```
            output_high(Relay1);
            lcd_gotoxy(1,4);
            printf(lcd_putc,"May bom bat");
          }
          else
          {
            Status = 1;
            output_low(Relay1);
            lcd_gotoxy(1,4);
            printf(lcd_putc,"May bom tat");
          }
        }
        break;
      case 2:
        if(BTUp == 0)
        {
          delay_ms(20);
          while(BTUp == 0);
          TempLv = TempLv + 1;
          if(TempLv >= 100)
          TempLv = 100;
        }
        else if(BTDown == 0)
```

```
{
  delay_ms(20);
  while(BTDown == 0);
  TempLv = TempLv - 1;
  if(TempLv <= 0)
  TempLv = 0;
}

write_eeprom(0x02, TempLv);

lcd_gotoxy(1,1);
printf(lcd_putc,"Cai Muc Nhiet Do  ");
lcd_gotoxy(1,2);
printf(lcd_putc,"Temp: %02d",TempLv);
lcd_putc(223);
lcd_putc("C   ");
lcd_gotoxy(14,2);
printf(lcd_putc,"      ");
lcd_gotoxy(1,3);
printf(lcd_putc,"                ");
lcd_gotoxy(1,4);
printf(lcd_putc,"                ");
break;
```

```
case 3:
  if(BTUp == 0)
  {
    delay_ms(20);
    while(BTUp == 0);
    HumiLv = HumiLv + 1;
    if(HumiLv >= 100)
    HumiLv = 100;
  }
  else if(BTDown == 0)
  {
    delay_ms(20);
    while(BTDown == 0);
    HumiLv = HumiLv - 1;
    if(HumiLv <= 0)
    HumiLv = 0;
  }

  write_eeprom(0x08, HumiLv);

  lcd_gotoxy(1,1);
  printf(lcd_putc,"Cai Muc Do Am      ");
  lcd_gotoxy(1,2);
```

```
            printf(lcd_putc,"Humi: %02d",HumiLv);
            lcd_putc("% ");
            break;
          case 4:


 delay_ms(500);
  mytime->hours   = 8;
  mytime->minutes = 30;
  mytime->seconds = 50;
  mytime->day   = 1;
  mytime->month = 6;
  mytime->year  = 2022;
           lcd_init();
            lcd_putc('\f');
    // print them
      lcd_gotoxy(1, 1);
      printf(lcd_putc, "TIME: %02u:%02u:%02u", mytime->hours, mytime->minutes, mytime->seconds);
      lcd_gotoxy(1, 2);
      printf(lcd_putc, "DATE: %02u/%02u/20%02u", mytime->day, mytime->month, mytime->year);
      lcd_gotoxy(1,3);
      printf(lcd_putc,"Mal de tat/mo den");
```

```
      delay_ms(500);
      break;


      }
   }
}


void _ConRelay(unsigned int8 TempLv, unsigned int8 HumiLv, unsigned int8 TempS,
unsigned int8 HumiS)
{
   if (TempS >= TempLv || HumiS <= HumiLv)
   {
      output_high(Relay1);
      lcd_gotoxy(1,4);
      printf(lcd_putc,"May bom bat");
   }
   else
   {
      output_low(Relay1);
      lcd_gotoxy(1,4);
      printf(lcd_putc,"May bom tat");
   }
}
```

```
int16 _ReadADC(unsigned int8 Pin)
{
  set_adc_channel(Pin);
  unsigned int16 Value = read_adc();
  return Value;
}

int8 _ReadTemp(unsigned int16 GiaTriADC0)
{
  float DienAp = ((float)GiaTriADC0 * 500)/1023.0f;
  return (int8)DienAp;
  /*LM35
    datasheet:
    10mv              -->     1 C
    (5000*GiaTriADC)/1023  -->     y
    => y = ((5000*GiaTriADC)/1023)/10 = (500*GiaTriADC)/1023*/
}

int8 _ReadHumi(unsigned int16 GiaTriADC1)
{
  unsigned int32 Value;
  Value = ((int32)GiaTriADC1*100)/1023;
```

```
  return (int8)Value;
}
```