ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÀI TẬP LỚN

THIẾT KẾ HỆ THỐNG NHÚNG

ĐỀ TÀI

ĐỒNG HỒ KỸ THUẬT SỐ HIỂN THỊ LÊN LCD DÙNG DS1307

GVHD: BÙI QUỐC BẢO

Nhóm: 17 – Lớp: L01

SINH VIÊN THỰC HIỆN	MSSV
Đào Nguyễn Bắc Tuyên	2213818
Huỳnh Ngọc Trí	2213634
Bùi Quốc Triệu	2213616

Thành phố Hồ Chí Minh, ngày 8 tháng 5 năm 2025

MỤC LỤC

1. GIỚI THIỆU

1.1. Tổng quan

Đề tài "Đồng hồ báo thức sử dụng vi điều khiển ATmega32" nhằm thiết kế và xây dựng một hệ thống có khả năng hiển thị thời gian thực và thực hiện chức năng báo thức. Hệ thống sử dụng mô đun DS1307 để lấy dữ liệu thời gian, hiển thị lên LCD, đồng thời so sánh với thời gian báo thức đã được cài đặt để kích hoạt chuông khi đến giờ. Ngoài ra, hệ thống cho phép người dùng tùy chỉnh thời gian và cài đặt báo thức linh hoạt thông qua các nút nhấn. Để hoàn thành đề tài, sinh viên cần nắm vững kiến thức về vi điều khiển, lập trình nhúng, thiết kế mạch điện và tích hợp các module ngoại vi. Qua đó, sinh viên không chỉ củng cố kỹ năng chuyên môn mà còn có cơ hội tiếp cận thực tế với việc xây dựng hệ thống nhúng hoàn chỉnh và có tính ứng dụng cao.

1.2. Nhiệm vụ đề tài

Nhiệm vụ chính của hệ thống là nhằm đảm bảo sự nắm bắt về thời gian và báo thức nhằm nhắc nhở mốc thời gian quan trọng tránh để lãng quên hoặc trễ giờ. Đảm bảo được sự chủ động của việc sử dụng lượng thời gian trong ngày. Do đó cần đảm bảo độ trễ là tối thiểu giữa hệ thống và thời gian thực tế. Thậm trí là một vật dụng tốt để trang trí trên bàn học hoặc đầu giường.

1.3. Phân chia công việc

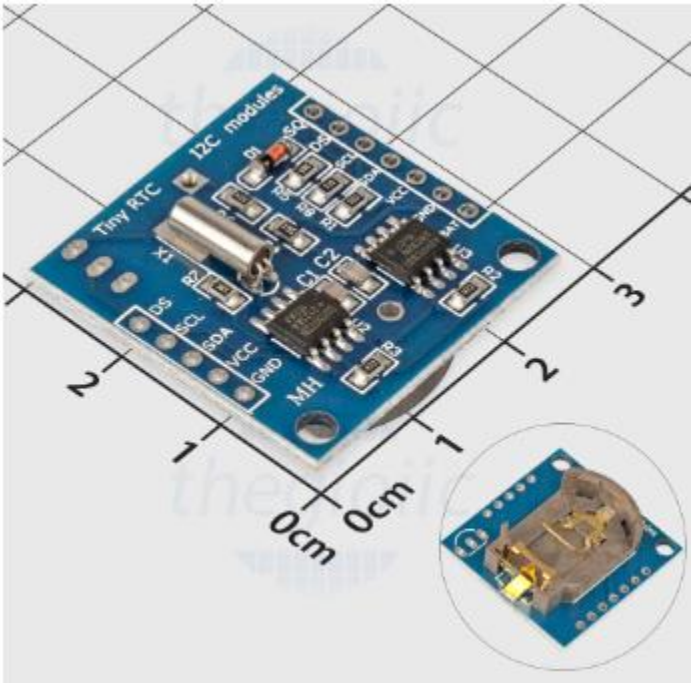
Sinh viên thực hiện	Công việc	Đánh giá
Đào Nguyễn Bắc Tuyên	Thiết kế phần mềm, Thiết kế phần cứng, vẽ PCB	40%
Huỳnh Ngọc Trí	Hỗ trợ phần mềm, viết báo cáo	30%
Bùi Quốc Triệu	Hỗ trợ phần cứng, viết báo cáo	30%

Trong quá trình thực hiện đề tài, các thành viên cùng nhau thực hiện và hỗ trợ lẫn nhau, tất cả đều hoàn thành công việc đúng thời gian để chuẩn bị báo cáo.

Thời gian thực hiện đề tài: 2 tháng

2. TỔNG QUAN LÝ THUYẾT VÀ LINH KIỆN

2.1. MODULE HW-111



Mạch thời gian thực sử dụng IC DS1307 có chức năng cung cấp thông tin thời gian như ngày, tháng, năm, giờ, phút, giây,... cho vi điều khiển thông qua giao tiếp I2C. Bên cạnh đó, mạch còn được tích hợp thêm IC EEPROM AT24C32 nhằm lưu trữ dữ liệu khi cần thiết, phù hợp cho các ứng dụng yêu cầu đồng bộ và lưu trữ thông tin thời gian thực (RTC). Hệ thống cũng được trang bị một nguồn dự phòng từ pin CR2032, giúp DS1307 duy trì hoạt động ngay cả khi mất nguồn chính. Khi chỉ sử dụng pin, DS1307 có thể hoạt động liên tục lên đến một năm, đảm bảo hệ thống giữ thời gian chính xác trong suốt vòng đời sử dụng của thiết bị.

Thông số kỹ thuật:

IC DS1307 giao tiếp I2C real-time clock chip (RTC)

Bộ nhớ 24C32 EEPROM I2C 32K

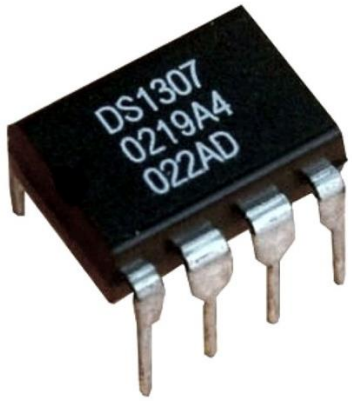
Sử dụng pin 2032 Lithium

IC DS1307 không thể đọc, viết và lưu data khi không có nguồn.

Kích thước: 27x28x8.4mm

Điện áp làm việc: 3.3V đến 5V

2.2. DS1307



Chức năng thời gian thực đầy đủ: Có khả năng đếm giây, phút, giờ, ngày, tháng và năm, đồng thời hỗ trợ bù năm nhuận chính xác đến năm 2100.

Bộ nhớ RAM không mất dữ liệu: Tích hợp 56 byte RAM tĩnh, không bị mất dữ liệu khi mất nguồn, dùng để lưu trữ thông tin người dùng.

Giao tiếp I2C chuẩn 2 dây: Sử dụng giao thức truyền thông I2C với hai chân SDA (dữ liệu) và SCL (xung nhịp), giúp dễ dàng kết nối với các vi điều khiển.

Tiêu thụ năng lượng thấp: Ở chế độ chạy bằng pin dự phòng, dòng tiêu thụ cực thấp, dưới 0,5mA, giúp kéo dài thời gian hoạt động mà không cần nguồn chính.

Dải nhiệt độ hoạt động rộng: Hoạt động ổn định trong khoảng nhiệt độ từ -40°C đến +85°C, phù hợp với nhiều điều kiện môi trường.

Gói linh kiện tiêu chuẩn: DS1307 được sản xuất dưới dạng vỏ 8 chân DIP hoặc SOIC, thuận tiện cho việc hàn tay hoặc hàn SMD trên mạch in.

2.3. AT24C32

Thông số kỹ thuật:

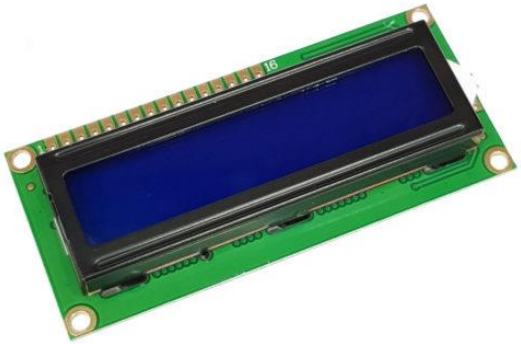
Định dạng bộ nhớ EEPROM

Dung lượng 32Kbit.

Giao tiếp I2C.

Nhiệt độ hoạt động 0°C đến 70°C

2.4. LCD1602



LCD 16x2 là loại màn hình hiển thị ký tự gồm 2 dòng, mỗi dòng hiển thị tối đa 16 ký tự. Màn hình này sử dụng công nghệ STN (Super Twisted Nematic) kết hợp với đèn nền LED màu xanh dương, cho phép hiển thị rõ nét các ký tự màu trắng ngay cả trong điều kiện thiếu sáng. IC điều khiển tích hợp là HD44780 hoặc tương đương, hỗ trợ giao tiếp song song 4-bit hoặc 8-bit với vi điều khiển như ATmega32, Arduino, PIC, v.v.

Thông số kỹ thuật:

Số dòng/ký tự: 2 dòng, 16 ký tự mỗi dòng

Công nghệ hiển thị: STN (Super Twisted Nematic)

Màu sắc: Chữ trắng, nền xanh dương

Ký tự hiển thị: 5x8 điểm ảnh

IC điều khiển: HD44780 hoặc tương đương

Chế độ giao tiếp: 4-bit hoặc 8-bit

Điện áp hoạt động: 5.0VDC

Kích thước module: 80.0 x 36.0 x 13.5 mm

Vùng hiển thị (khung chữ): 66.0 x 16.0 mm

Đèn nền: LED màu xanh dương

2.5. BUTTON

Tact switch là loại công tắc điện tử có kích thước nhỏ gọn, được sử dụng để tạo tín hiệu khi người dùng nhấn vào. Khi nhấn, hai cặp chân của nút sẽ được nối với nhau (đóng mạch), và khi thả ra thì hở mạch trở lại. Loại nút này thường được sử dụng để nhập liệu, cài đặt hoặc điều khiển menu trên các thiết bị điện tử.

Thông số kỹ thuật:

Loại: Nút nhấn tact switch 2 chân (kiểu đứng, xuyên lỗ)

Kích thước thân: 6mm × 6mm

Chiều cao nút nhấn: Khoảng 7mm (tùy mẫu)

Số chân: 2 chân

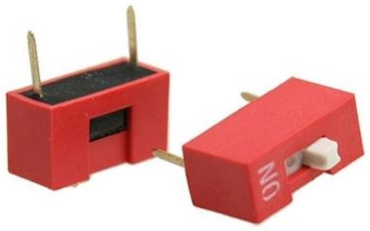
Chất liệu vỏ: Nhựa đen, đế kim loại

Chế độ hoạt động: Nhấn giữ (Momentary type)

Độ bền: Khoảng 100.000 lần nhấn

Cách đấu nối: Hai chân sẽ nối mạch khi nhấn (không phân cực)

2.6. SWITCH



DIP switch (Dual In-line Package switch) là loại công tắc cơ khí dạng gạt, được thiết kế để cắm trực tiếp lên PCB. Với thiết kế nhỏ gọn và thao tác đơn giản, DIP switch được dùng để thiết lập cấu hình phần cứng như chọn chế độ, địa chỉ thiết bị, hoặc chọn tham số đầu vào.

Thông số kỹ thuật:

Loại: DIP switch 1 vị trí (1 pole)

Số chân: 2 chân

Trạng thái: ON/OFF (gạt tay)

Chất liệu vỏ: Nhựa ABS hoặc Polyamide chống cháy

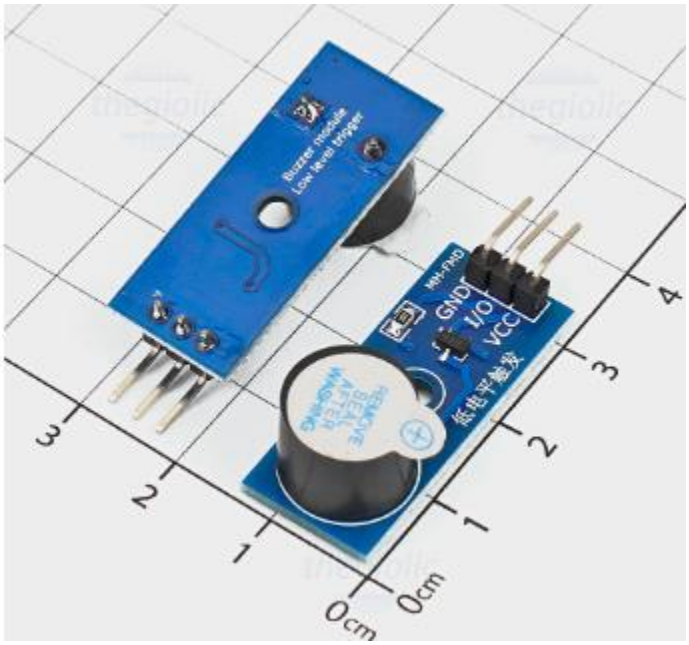
Màu sắc: Thân màu đỏ, cần gạt màu trắng

Khoảng cách chân: 2.54mm (chuẩn hàng rào)

Độ bền: Trên 1.000 lần gạt

Lắp đặt: Cắm xuyên lỗ (DIP – Through Hole)

2.7. BUZZER



Module còi 5V (Buzzer) được sử dụng để phát ra âm thanh khi kích tín hiệu, được ứng dụng trong các hệ thống cảnh báo, báo trộm,... Thông số kỹ thuật Điện áp sử dụng: 3.3~5VDC Tín hiệu kích: TTL mức thấp Low 0VDC.

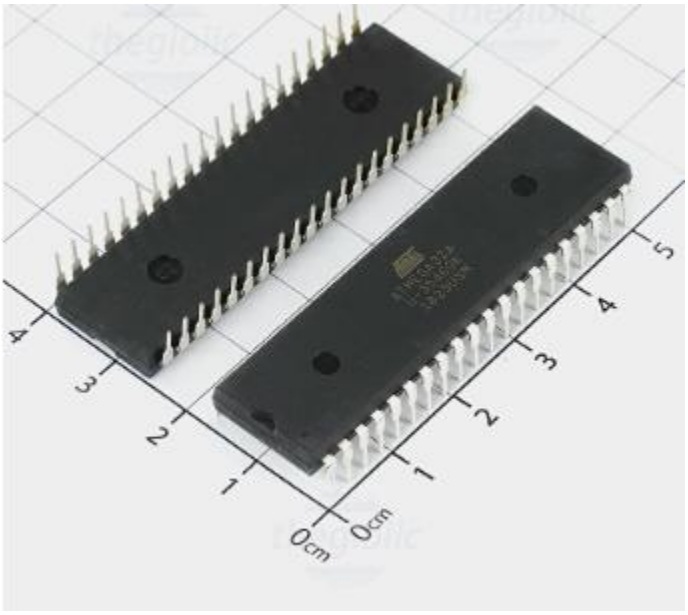
Thông số kỹ thuật:

Điện áp sử dụng: 3.3~5VDC

Tín hiệu kích: TTL mức thấp Low 0VDC

Kích thước: 38 x 13 mm

2.8. ATMEGA32



ATmega32A-PU là vi điều khiển 8-bit thuộc dòng AVR của hãng Atmel (nay là Microchip). Với bộ nhớ lớn, nhiều chân I/O và các ngoại vi tích hợp như ADC, UART, SPI, I2C, vi điều khiển này được sử dụng phổ biến trong các ứng dụng điều khiển nhúng, đào tạo lập trình vi điều khiển và các hệ thống điện tử tự động.

Thông số kỹ thuật:

Loại: Vi điều khiển 8-bit AVR

Bộ nhớ Flash: 32 KB

SRAM: 2 KB

EEPROM: 1 KB

Tốc độ xung nhịp: Tối đa 16 MHz

Số chân: 40 chân (DIP – Dual In-line Package)

Số chân I/O: 32 chân (chia thành 4 cổng: PORTA, PORTB, PORTC, PORTD)

ADC: 8 kênh ADC độ phân giải 10-bit

Timer/Counters: 2 bộ đếm 8-bit, 1 bộ đếm 16-bit

Giao tiếp ngoại vi: UART, SPI, I2C (TWI)

Điện áp hoạt động: 2.7V – 5.5V

Dòng tiêu thụ: Thấp, có hỗ trợ chế độ tiết kiệm năng lượng

Chất liệu vỏ: Nhựa đen epoxy, chân kim loại

Màu sắc: Thân đen, chân bạc

3. YÊU CẦU HỆ THỐNG

Tên sản phẩm:

Đồng hồ kỹ thuật số hiển thị lên LCD dùng DS1307.

Mục đích:

Liên tục cập nhật thời gian từ DS1307 và hiển thị lên LCD nhằm giúp người dùng nắm bắt được thời gian và báo thức khi đến mốc thời gian đã cài đặt sẵn.

Input và output:

- Input:

+ 5 nút nhấn: 1 nút nhấn reset, 1 nút nhấn tăng đơn vị thời gian, 1 nút nhấn giảm đơn vị thời gian, 1 nút chỉnh thời gian, 1 nút tắt báo thức.

+ 2 switch: 1 switch cho phép cài đặt thời gian báo thức, 1 switch cho phép bật/tắt chế độ báo thức.

+ 1 biến trở: điều chỉnh độ phân giải của màn hình LCD.

+ Module RTC DS1307: liên tục gửi giá trị giờ, phút, giây cho MCU

- Output:

+ LCD 16x02: Hiển thị thời gian đọc từ DS1307 và chế độ báo thức ON/OFF.

+ BUZZER: Báo thức khi thời gian chạy tới đúng thời gian đặt đúng báo thức.

Trường hợp sử dụng:

Khi bật nguồn, LCD sẽ hiển thị thời gian đọc từ module DS1307. Nếu thời gian chạy chưa đúng thì nhấn nút Setting Time để chỉnh thời gian cho đúng thời gian thực. Nếu cài báo thức thì bật switch Alarm ON/OFF và cài đặt thời gian theo ý người dùng muốn báo thức bằng switch Setting Alarm. Khi thời gian chạy đúng thời gian cài đặt báo thức thì Buzzer sẽ reo lên. Người dùng có thể dùng nút nhấn Turn Off Alarm để dừng báo thức.

Chức năng:

Hệ thống đọc thời gian từ module RTC DS1307 và hiển thị lên LCD giờ, phút, giây, ngày, tháng, năm. Khi bật chế độ báo thức lên thì nếu thời gian chạy đúng với thời gian báo thức thì Buzzer sẽ reo lên.

Hiệu suất:

Sai lệch thời gian là 1 ngày lệch 10s.

Chi phí sản xuất

STT	TÊN LINH KIỆN	SỐ LƯỢNG	THÀNH TIỀN
1	NPN - 2N2222	1 Cái	500đ
2	Công tắc gạt SS-12F44	2 Cái	1.000đ
3	Nút nhấn 12x12x7mm	4 Cái	2.400đ
4	Điện trở 10K 1/4W 1%	1 Gói	3.000đ
5	ATMEGA32A-PU	1 Cái	82.000đ
6	Tụ gốm 22pF – 50V	1 Gói	1.000đ
7	Thạch anh 32.768KHz	1 Cái	1.400đ
8	Biến trở đứng 1K	1 Cái	2.000đ
9	LCD nền xanh dương chữ trắng	1 Cái	25.000đ
10	DS1307	1 Cái	5.000đ

11	Buzzer 60dB	1 Cái	3.000đ
12	Hộp pin 1x9V	1 Cái	8.500đ
13	L7805ADCD2T – CN	1 Cái	5.000đ
14	Hàng rào cái	2 Cái	4.000đ
Tổng:			143.800đ

Công suất, nguồn điện:

- Nguồn cung cấp: Sử dụng adapter chuyển từ 220VAC thành 5VDC

- Công suất tiêu thụ: ≤10W

Kích thước vật lý, khối lượng:

- Kích thước 150 cm^2

- Khối lượng: < 500g

Lắp đặt: được đặt trên bàn làm việc

4. NHỮNG VẤN ĐỀ THIẾT KẾ

Vấn đề ràng buộc:

- Giá thành < 200 nghìn VNĐ

- Thời gian sử dụng, tuổi thọ của sản phẩm: < 5 năm

- Hệ thống tiêu thụ năng lượng thấp: 5V

- Kích thước nhỏ gọn: < 150 cm^2

- Khối lượng: < 500g

Vấn đề chức năng:

- Điện áp 5V khi cấp cho các linh kiện được mắc song song để ổn áp

- Vật liệu chịu được nhiệt độ tốt không bị chảy vào ngày nắng nóng và phát thải khí có hại.

Vấn đề thời gian thực:

- LCD là bộ hiển thị thời gian thực mềm với độ trễ < 0.25s

- Buzzer có độ trễ < 0.1s

Vấn đề đồng thời:

- Vừa đọc thời gian từ Module DS1307 và vừa so sánh với thời gian báo thức được cài đặt, nếu trùng thì bật báo thức.

Vấn đề phản hồi:

- Tương tác liên tục: Hệ thống liên tục đọc thời gian từ DS1307 và hiển thị lên LCD.

- Phản ứng các sự kiện không có chu kì: Thời gian cài đặt báo thức là theo ý muốn người dùng.

5. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

5.1. Nguyên lý hoạt động

- Sau khi khởi động, màn hình LCD sẽ hiển thị giao diện giới thiệu thiết bị "BTL TKHTN NHOM 17" trong 2 giây.

- Sau đó hệ thống sẽ hiện thời gian ngày tháng năm. Khi mở lần đầu người dùng sẽ phải thiết lập thời gian ngày tháng năm cho hệ thống.

- Thiết bị gồm 4 nút ấn và 2 nút gạt:

SWITCH chọn mode điều chỉnh: khi SWITCH 1 ON thì hệ thống sẽ nhận biết người dung muốn điều chỉnh thời gian ngày tháng năm thứ. Nếu SWITCH 1 OFF thì hệ thống sẽ biết người dùng muốn điều chỉnh thời gian báo thức.

SWITCH bật tắt báo thức: khi SWITCH 2 ON thì báo thức sẽ bật và OFF thì báo thức sẽ tắt. Trạng thái của báo thức sẽ hiển thị trên LCD là ON và OFF tương ứng.

SETTING MODE BUTTON: khi ấn nút này hệ thống sẽ vào màn hình điều chỉnh của từng thông số và sẽ hiện trên màn hình để người dùng biết. Cứ mỗi lần ấn nút này sẽ tự động chuyển đến giá trị tiếp theo và khi đến giá trị điều chỉnh cuối ấn một lần nữa sẽ trở về màn hình chính.

Nếu SWITCH 1 ON: thì các giá trị điều chỉnh lần lượt là hour -> minute -> second -> year -> month -> day -> date.

Nếu SWITCH 1 OFF: thì các giá trị lần lượt là ALARMHOUR -> ALARMMINUTE.

INCREASE BUTTON: Dùng để tăng các giá trị trong mode điều chỉnh khi đạt max sẽ tự động trở về giá trị 0.

DECREASE BUTTON: Dùng để giảm các giá trị trong mode điều chỉnh khi đạt giá trị 0 sẽ tự động chuyển về giá trị max được thiết lập.

TURN OFF ALARM BUTTON: Dùng để dừng tiếng báo thức nhằm cho hệ thống không kêu nữa, khi đã dừng báo thức thì phải thiết lập các giá trị thời gian báo thức trở lại.

5.2. Thành phần phần cứng

5.2.1. MCU

8-bit Microcontroller: ATmega32A thu thập các dữ liệu từ DS1307 đồng thời nhận tín hiệu từ các nút và nút gạt để hiện thông tin lên ngõ ra.

5.2.2. Ngoại vi (Peripherals)

Ngõ vào:

- Nút nhấn
- Nút gạt switch
- Module RTC

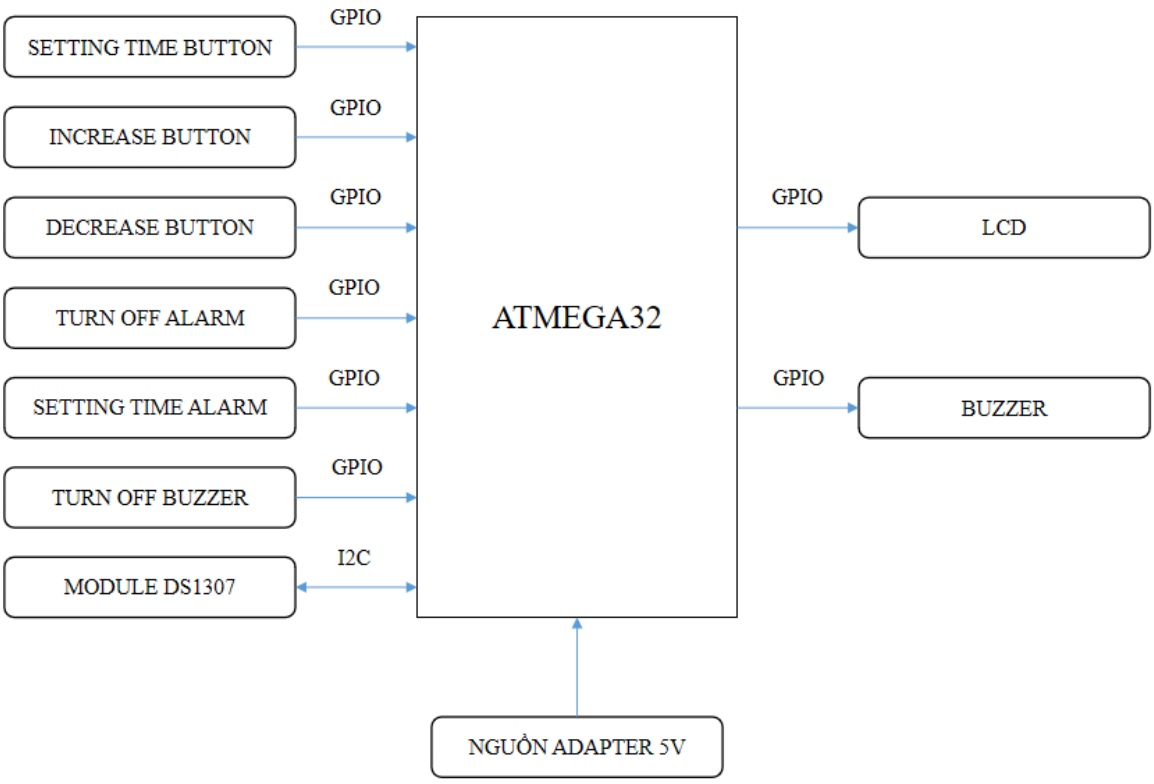
Ngõ ra:

- LCD 1602
- Buzzer

5.2.3. NGUỒN (Power Supply)

Nguồn 5V từ adapter dô kit ATmega và mắc song song ra các linh kiện.

5.3. Sơ đồ khối phần cứng



5.4. Giải thích các khối

a. Khối xử lý trung tâm ATmega32

Đây là bộ xử lý trung tâm của hệ thống, điều khiển các thành phần khác. Nó thực hiện nhiệm vụ đọc dữ liệu từ DS1307 và các chân GPIO để điều chỉnh thời gian, bật tắt báo thức, giao tiếp với LCD và RTC.

b. Khối thời gian thực RTC: DS1307

Giao tiếp I2C: cung cấp thời gian thực.

Các giá trị điện trở, thạch anh, pin có sẵn trên module theo DATASHEET.

c. Các khối giao tiếp nút

- Nút ấn: Vào mode điều chỉnh, tang giảm giá trị, tắt báo thức.

- Nút gạt: Chọn mode điều chỉnh, bật tắt báo thức.

Đặc điểm:

- Điện trở 10Kohm kéo xuống đất, là nút thường tích cực thấp.

- Tụ gốm 22pF để chống rung phần cứng.

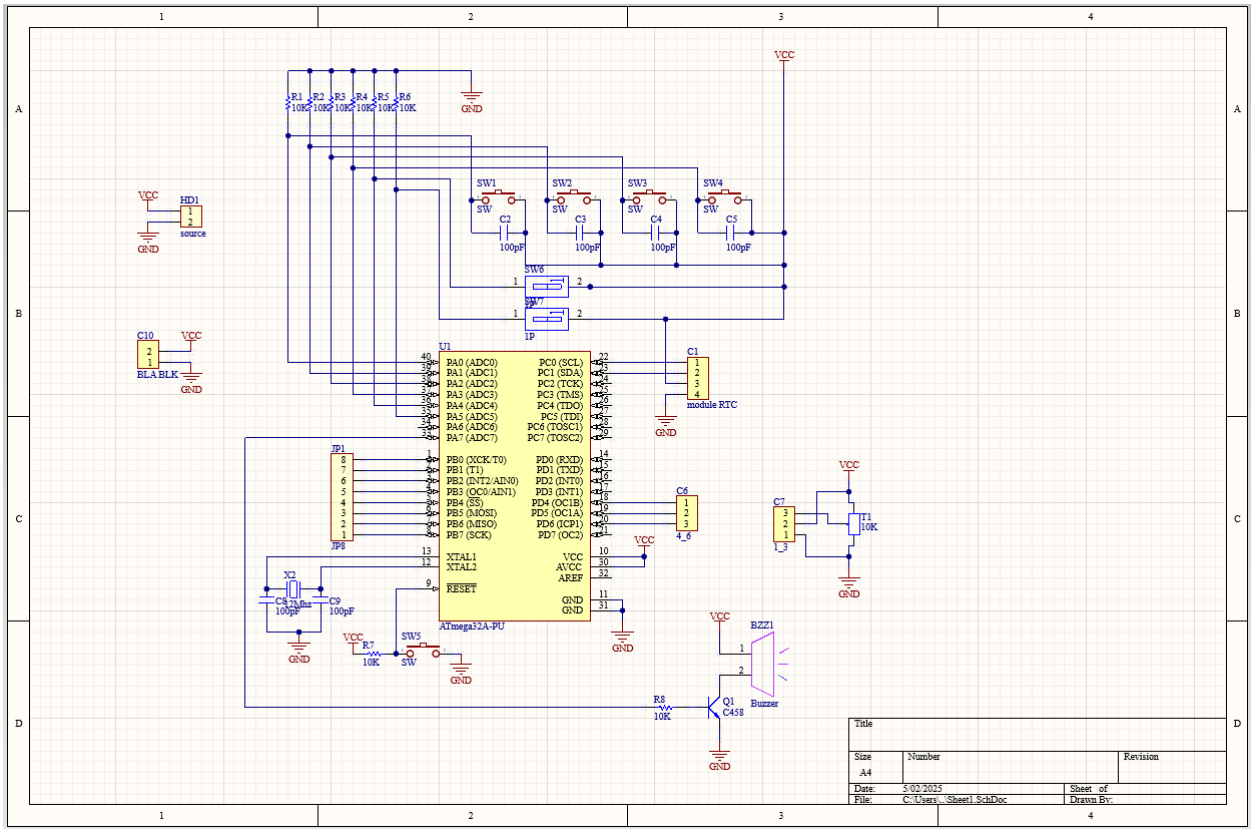
d. Buzzer

- Báo thức, hoạt động trong vùng 3.3 ~ 5VDC.

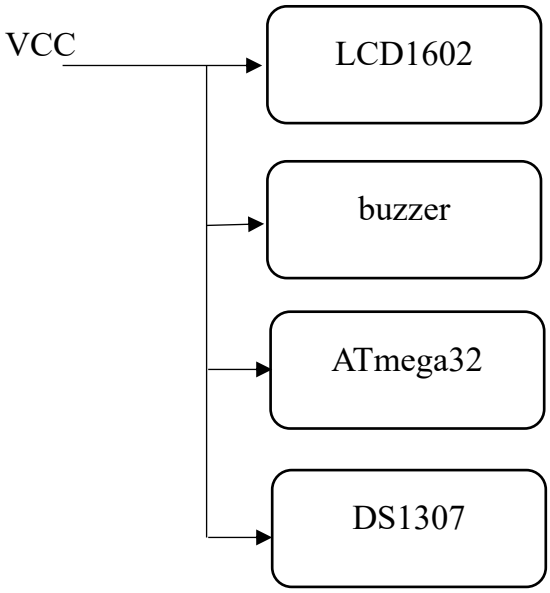
- Dùng một trở 220 Ohm và BJT NPN 2N2222 để đệm dòng áp cho buzzer.

5.5. Sơ đồ mạch thiết kế

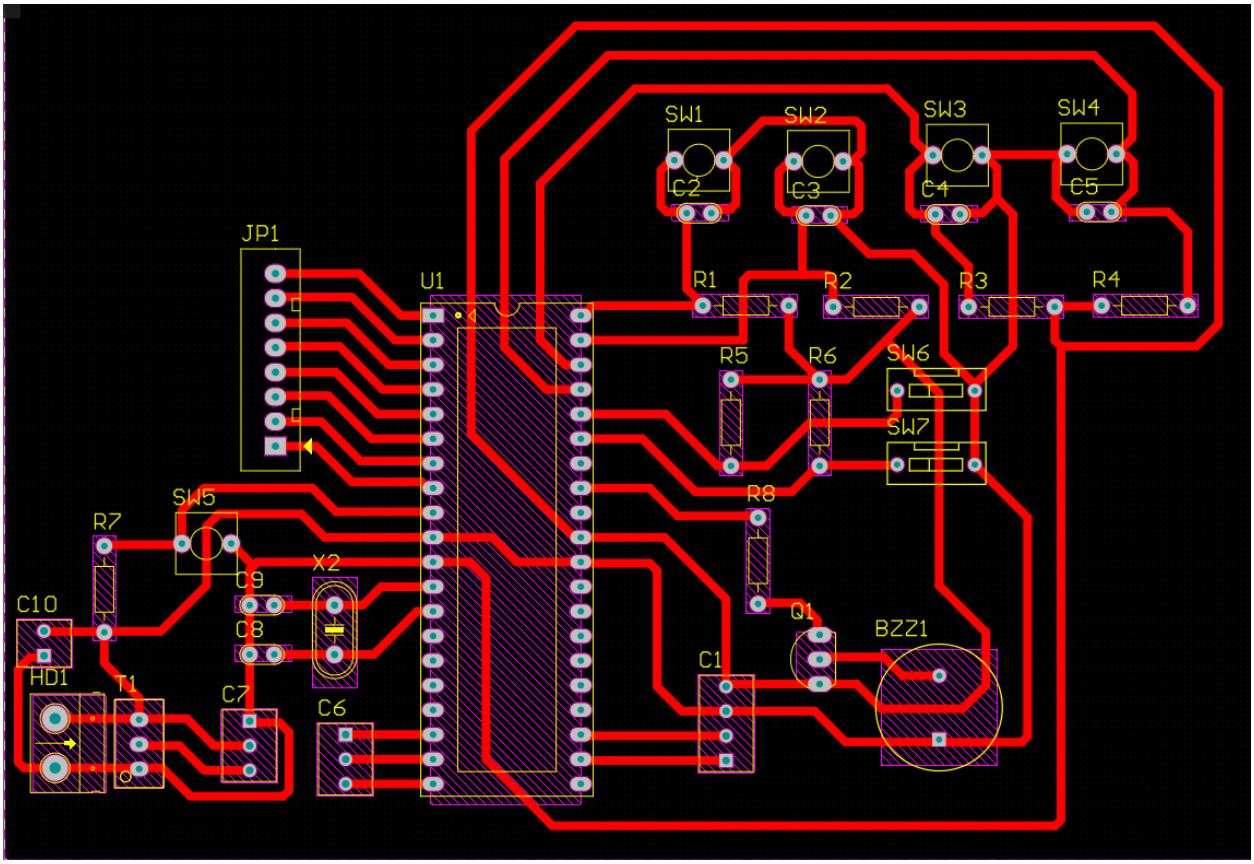
a. Tổng quan sơ đồ

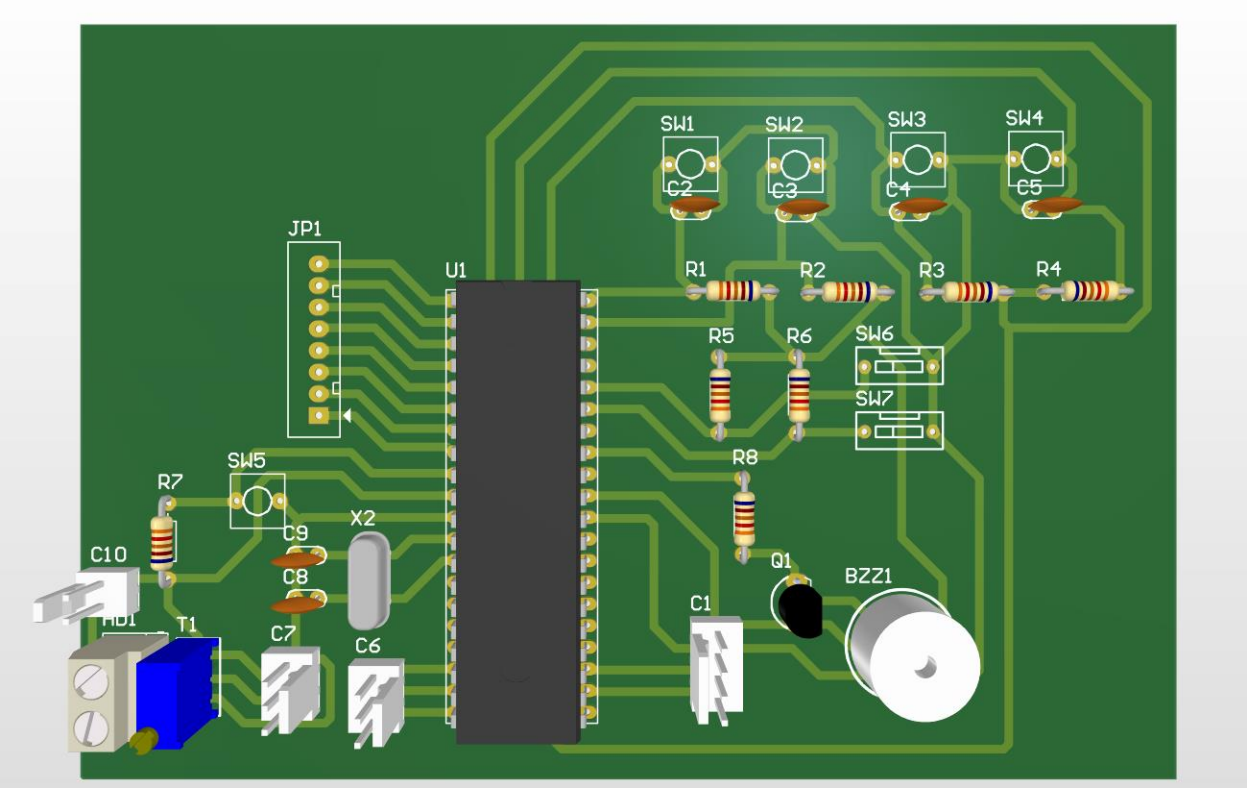


b. Sơ đồ mạch nguồn



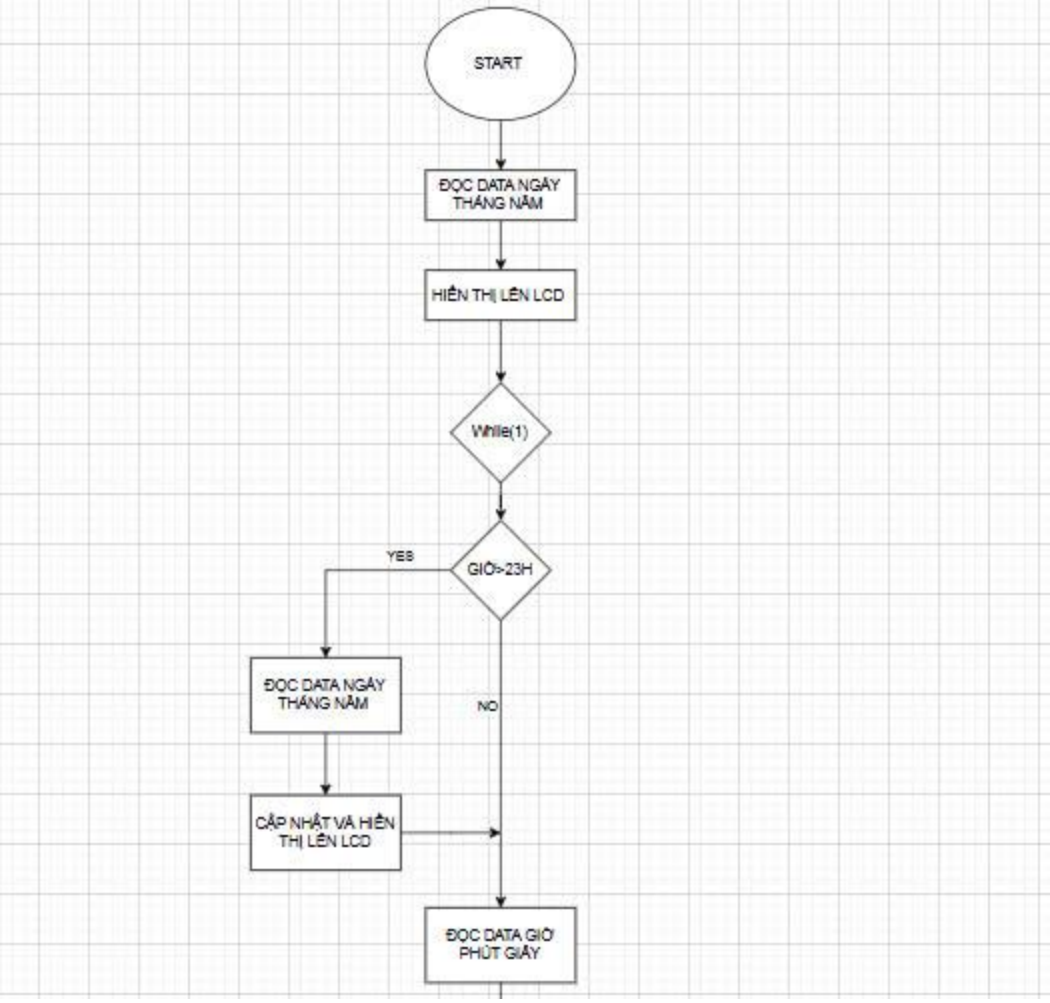
c. Layout PCB trong Altium

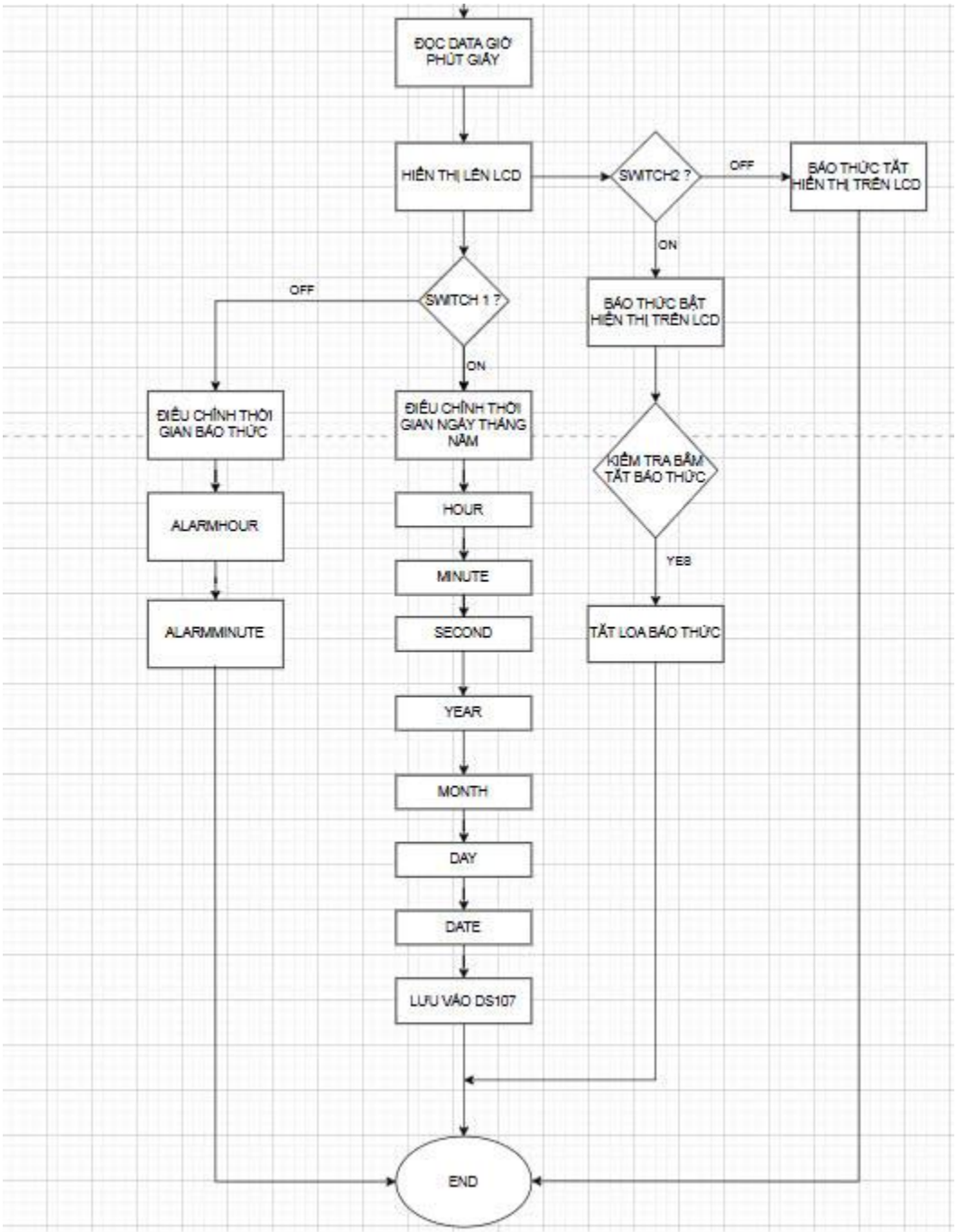



SW1
SW2
SW3
SW4
C2
C3
C4
C5
JP1
U1
R1
R2
R3
R4
R5
R6
SW6
SW7
R8
SW5
R7
X2
C9
C8
C10
HD1
T1
C7
C6
C1
Q1
BZZ1

6. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

6.1. Lưu đồ giải thuật





6.2. Chương trình phần mềm

```
#include <avr/io.h>
#define F_CPU 11059200
#include <util/delay.h>
#include <stdlib.h>
```

```
#include <avr/interrupt.h>
#define F_CPU 8000000UL
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include <LCD.h>
#include <I2C.h>

int second,minute,hour, day, date, month, year;
int seconds,minutes,hours,dates,dayss,months,years;
int ALhour,ALMIN,ALH,ALM;

char tmp[2];
```

- #include <avr/io.h>: Bao gồm các định nghĩa cho việc truy cập vào các thanh ghi và cổng của vi điều khiển AVR.

- #define F_CPU 11059200 và #define F_CPU 8000000UL: Định nghĩa tần số của vi điều khiển, giúp các hàm delay và các thao tác thời gian khác hoạt động chính xác.

- #include <util/delay.h>: Sử dụng các hàm tạo độ trễ.

- #include <stdlib.h>, #include <stdio.h>, #include <string.h>, #include <stdbool.h>, #include <math.h>: Bao gồm các thư viện chuẩn của C cho việc xử lý chuỗi, toán học và các thao tác logic khác.

- #include <LCD.h>: Thư viện để điều khiển màn hình LCD.

- #include <I2C.h>: Thư viện để giao tiếp I2C.

- second, minute, hour, day, date, month, year: Các biến lưu trữ thời gian thực (giây, phút, giờ, ngày, ngày tháng, tháng, năm).

- seconds, minutes, hours, dates, dayss, months, years: Các biến lưu trữ thời gian khác, có thể được sử dụng cho mục đích so sánh hoặc thay đổi tạm thời.

- ALhour, ALMIN, ALH, ALM: Các biến lưu trữ giờ và phút cho báo thức (alarm).

- char tmp[2]: Một mảng ký tự tạm thời, có thể được sử dụng để lưu trữ các giá trị chuỗi nhỏ hoặc các ký tự riêng lẻ.

```
uint8_t BCDToDecimal(uint8_t bcd) {
        return ((bcd >> 4) * 10) + (bcd & 0x0F);
}

// Convert decimal to BCD
uint8_t decimalToBCD(uint8_t decimal) {
        return ((decimal / 10) << 4) | (decimal % 10);
}
```

Hai lệnh BCDToDecimal và decimalToBCD là hai lệnh dùng để chuyển đổi qua lại giữa số thập phân và số BCD. Vì RTC thường lưu trữ thời gian dưới dạng BCD và cần chuyển đổi về dạng thập phân để hiển thị giờ, phút, giây lên màn hình LCD. Khi cần lưu trữ hoặc gửi dữ liệu dưới dạng BCD, ta cần chuyển đổi từ số thập phân sang BCD.

```
void RTC_WRITE_CALENDER (char WRITE_CLOCK_CALENDER, char years, char months, char dates,char dayss)
{
        years  = decimalToBCD(years);
        months = decimalToBCD(months);
        dayss  = decimalToBCD(dayss);
        dates  = decimalToBCD(dates);

        I2C_Start(Device_write_address);
        I2C_Write(WRITE_CLOCK_CALENDER);
```

```
        I2C_Write(dayss);
        I2C_Write(dates);
        I2C_Write(months);
        I2C_Write(years);
        I2C_Stop();

}

void RTC_WRITE_CLOCK (char WRITE_CLOCK_ADDRESS, char hours, char minutes, char seconds)
{

        hours = decimalToBCD(hours);
        minutes = decimalToBCD(minutes);
        seconds = decimalToBCD(seconds);

        I2C_Start(Device_write_address);
        I2C_Write(WRITE_CLOCK_ADDRESS);
        I2C_Write(seconds);
        I2C_Write(minutes);
        I2C_Write(hours);
        I2C_Stop();
}

void RTC_READ_CLOCK (char READ_CLOCK_ADDRESS)
{
        I2C_Start(Device_write_address);
```

```
        I2C_Write(READ_CLOCK_ADDRESS);
        I2C_Repeated_Start(Device_Read_address);

        second = I2C_Read_Ack();
        minute = I2C_Read_Ack();
        hour   = I2C_Read_NACK();
        I2C_Stop();
}

void RTC_READ_CALENDER( char READ_CALENDER_ADDRESS)
{
        I2C_Start(Device_write_address);
        I2C_Write(READ_CALENDER_ADDRESS);
        I2C_Repeated_Start(Device_Read_address);
        day   = I2C_Read_Ack();
        date  = I2C_Read_Ack();
        month = I2C_Read_Ack();
        year  = I2C_Read_NACK();
        I2C_Stop();
}
```

- **Ghi thông tin vào RTC**: Các hàm RTC_WRITE_CALENDER và RTC_WRITE_CLOCK sử dụng giao tiếp I2C để ghi thông tin ngày tháng và giờ vào RTC, sau khi chuyển đổi các giá trị từ thập phân sang BCD.

- **Đọc thông tin từ RTC**: Các hàm RTC_READ_CLOCK và RTC_READ_CALENDER sử dụng giao tiếp I2C để đọc thông tin giờ và ngày tháng từ RTC, và lưu trữ vào các biến tương ứng

.

```
void adjust_value(char *label, int *value, int max_value, unsigned char position) {
	char temp[10];
	send_a_command(0x01);
	_delay_ms(1);
	send_a_command(0x80);
	send_a_string(label);
	_delay_ms(1);

	while (1) {
		send_a_command(position);
		_delay_ms(1);
		sprintf(temp, "%02d", *value);
		send_a_string(temp);
		_delay_ms(1);

		if (bit_is_set(PINA, 1)) {
			if (*value < max_value) {
				(*value)++;
				} else {
				*value = 0;
			}
			_delay_ms(10);
		}

		if (bit_is_set(PINA, 2)) {
			if (*value > 0) {
				(*value)--;
				} else {
```

```
                    *value = max_value;
                }
                _delay_ms(6);
            }

            if (bit_is_set(PINA, 0)) break;
        }
    }

int NHUANKH(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

int MAXDATE(int month, int year) {
    if (month == 2)
    { // Tháng 2
        if (is_leap_year(year)) return 29;

        else return 28;
    }
    else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 ||
month == 10 || month == 12) return 31;
    else return 30;
}
```

- Hàm adjust_value: cho phép điều chỉnh giá trị hiển thị trên màn hình LCD thông qua các nút bấm.

- Hàm NHUANKH: kiểm tra xem năm có phải đang là năm nhuận không.

- Hàm MAXDATE: trả về số ngày tối đa trong một tháng của một năm nhất định.

```
int main(void)


{
        char temp [10];
        char buffer[20];
        char* days[7] = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"};
        I2C_Init();
        DDRA = 0xFF;
        LCD_Data_Dir = 0xFF;
        LCD_Command_Dir |= (1 << RS)| (1 << RW) | (1 << EN);
                                                    // setting compare value equal to
counter clock frequency to get an interrupt every second
        send_a_command(0x01);
        _delay_ms(1);
        send_a_command(0x38);
        _delay_ms(1);
        send_a_command(0x0C);
        _delay_ms(1);


        send_a_string("BTL TKHTN NHOM 17");
        _delay_ms(250);
        send_a_command(0x01);
```

- Khởi tạo các biến và mảng ngày trong tuần.

- Khởi tạo giao tiếp I2C và cấu hình các cổng vào/ra.

- Thiết lập màn hình LCD và hiển thị dòng chữ “BTL TKHTN NHOM 17”.

```
START:
        send_a_command(0x80);
        //format 24h
        RTC_READ_CALENDER(3);
        _delay_ms(1);
        sprintf(buffer,"%02x/%02x/%02x %3s ", date, month, year, days[day-1] );
        send_a_string(buffer);
        _delay_ms(1);
```

- Đọc và hiển thị lịch từ RTC.

```
while (1)
        {
                if(hour>23)
                {
                        send_a_command(0x80);
                        //format 24h
                        RTC_READ_CALENDER(3);
                        _delay_ms(1);
                        sprintf(buffer,"%02x/%02x/%02x %3s ", date, month, year,
days[day-1] );
                        send_a_string(buffer);
                        _delay_ms(1);
                }
                RTC_READ_CLOCK(0);
                _delay_ms(1);
                sprintf(buffer, "%02x:%02x:%02x ",(hour & 0b00111111), minute,
second);
                send_a_command(0xC0);   // dua con tro toi vi tri thu 4
                send_a_string(buffer);                  // gui chuoi ki tu ra LCD
```

```
        _delay_ms(1);
        send_a_command(0xC0 + 0x09);
```

- Đọc giờ từ RTC và hiển thị lên màn hình LCD.

```
    if (bit_is_set(PINA,5))
        {
            send_a_string(" ON ");
            if(bit_is_set(PINA,3)&&(bit_is_clear(PINA,7)))
            {
                ALMIN = 0;
                ALhour = 0;
            }
            if ((decimalToBCD(ALhour) == hour) &&
(decimalToBCD(ALMIN) == minute)) PORTA &= (0 << PINA7);
            else PORTA |= (1 << PINA7);
        }
        else
        {
            send_a_string("OFF");
            PORTA |= (1 << PINA7);
        }
```

- Kiểm tra trạng thái báo thức và điều khiển loa dựa trên trạng thái này.

```
if((bit_is_set(PINA,4))&&(bit_is_set(PINA,0)))
        {
            seconds = BCDToDecimal(second);
            minutes = BCDToDecimal(minute);
            hours   = BCDToDecimal(hour);
            dayss  = BCDToDecimal(day);
```

```
dates   = BCDToDecimal(date);
months  = BCDToDecimal(month);
years   = BCDToDecimal(year);
adjust_value("hour: ", &hours, 23,0x85);
adjust_value("minute: ", &minutes, 59,0x87);
adjust_value("second: ", &seconds, 59,0x87);
adjust_value("year: ", &years, 100,0x85);
adjust_value("month: ", &months, 12,0x86);
adjust_value("date: ", &dates, MAXDATE(years,months),0x85);
send_a_command(0x01);
send_a_command(0x80);
send_a_string("thu: ");
while(1)
{
        send_a_command(0x84);
        _delay_ms(1);
        sprintf(temp,"%02s", days[dayss-1]);
        send_a_string(temp);

        if (bit_is_set(PINA, 1))
        {
                if (dayss < 7) dayss++;
                else  dayss = 0;

        }
        if (bit_is_set(PINA,2))
        {
                if (dayss > 0) dayss--;
```

```
                    else dayss = 6;


                }
                if(bit_is_set(PINA,0)) break;
            }
            send_a_command(0x01);
            RTC_WRITE_CLOCK(0,hours,minutes,seconds);
            RTC_WRITE_CALENDER(3,years,months,dates,dayss);
            goto START;
```

- Cho phép người dùng điều chỉnh thời gian (giờ, phút, giây, ngày, tháng, năm) thông qua các nút nhấn.

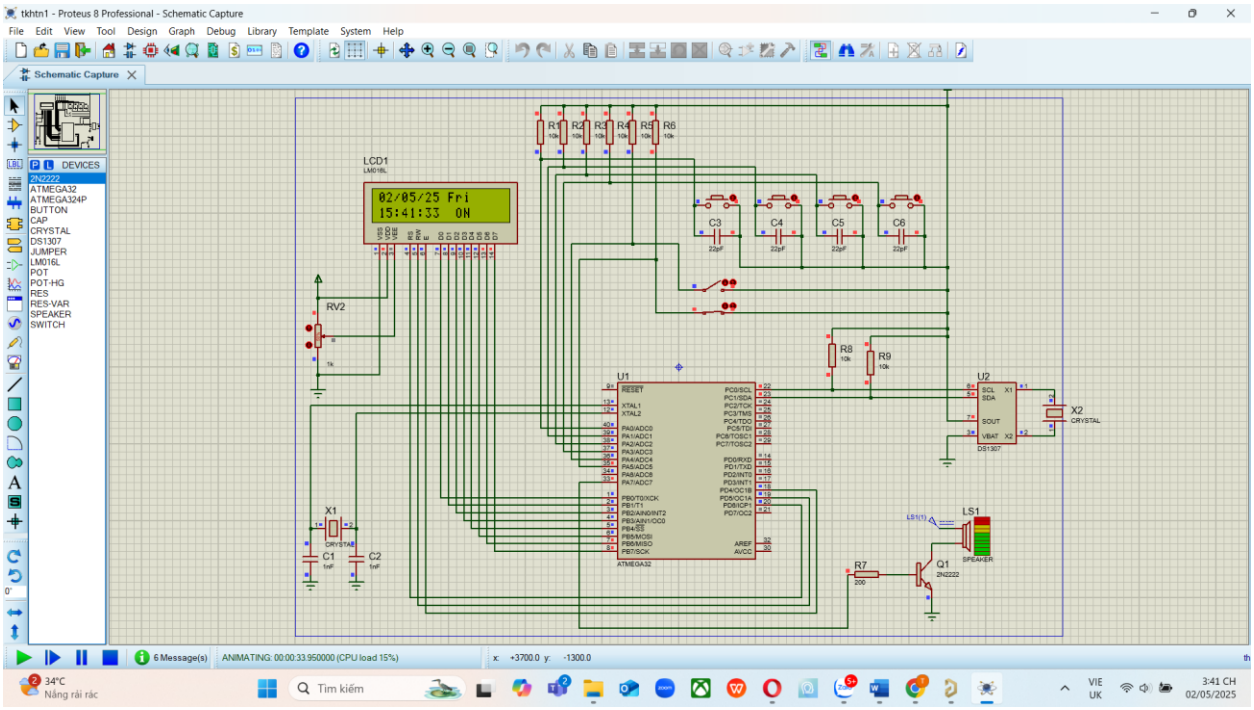
```
if((bit_is_clear(PINA,4))&&(bit_is_set(PINA,0)))
        {
            ALH = BCDToDecimal(hour);
            adjust_value("ALHOUR: ", &ALH, 23,0x87);
            ALhour = ALH;

            ALM = BCDToDecimal(minute);
            adjust_value("ALminute: ",&ALM,59,0x89);
            ALMIN = ALM;
            send_a_command(0x01);
            goto START;
```

- Cho phép người dùng thiết lập báo thức (giờ, phút) thông qua các nút nhấn.

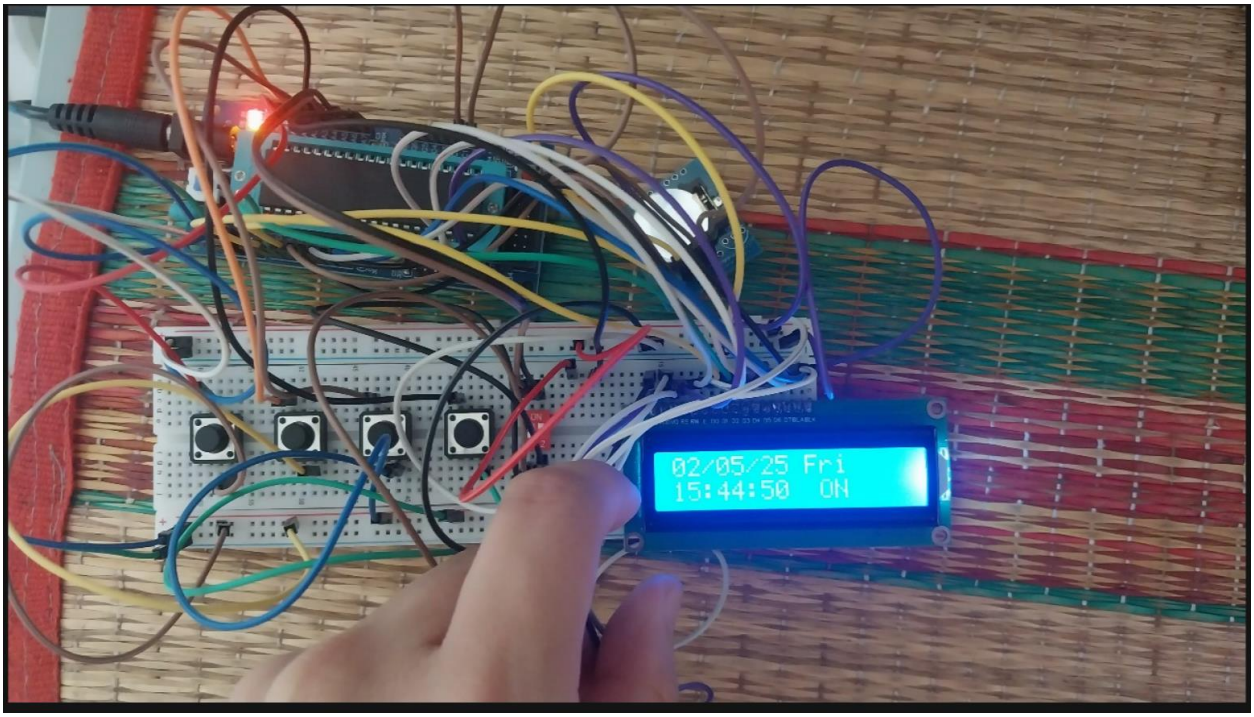
7. KẾT QUẢ THỰC HIỆN

7.1. Mô phỏng mạch



7.2. Sản phẩm thực tế

Mạch breadboard:



7.3. Hạn chế

Sau khi mô phỏng cũng như là thực hiện mạch xong thì nhóm nhận thấy hệ thống có một số hạn chế nhất định sau:

- Độ chính xác thời gian: Với độ lệch là cứ 1 ngày 10s thì nhóm nhận thấy hệ thống không phù hợp với các ứng dụng yêu cầu độ chính xác cao, nếu chỉ dùng để báo thức hoặc nhắc nhở về thời gian thì sẽ không gây hậu quả đáng kể cho người dùng. Bên cạnh đó nếu nhận thấy thời gian hiển thị trên hệ thống sai lệch so với thời gian thực thì người dùng có thể chỉnh giờ thông qua các nút nhấn.

- Cấu hình thời gian: Do DS1307 không có cơ chế tự động cập nhật thời gian từ mạch đồng hồ chính quốc tế (NTP) hoặc GPS, do đó bạn phải cập nhật thời gian bằng tay nếu thiết bị bị mất điện. DS1307 chỉ có thể lưu trữ thời gian và ngày trong định dạng BCD (Binary-Coded Decimal), do đó bạn cần phải chuyển đổi giữa BCD và số nguyên.

- Hệ thống sử dụng giao tiếp 8 bit LCD nên sẽ tốn nhiều chân PORT của vi xử lý hơn so với các giao tiếp khác như 4 bit, I2C,... Vì vậy, nếu thiết lập thêm chức năng thì phải lựa chọn lại chân vi xử lý cũng như là thiết lập lại chân PORT.

- Nguồn điện: Hệ thống sử dụng adapter để chuyển từ 220VAC thành 5VDC, chính vì vậy mà đồng hồ chỉ sử dụng trong phạm vi ngắn, không linh động trong việc di chuyển vd: để trên bàn học, gần nơi có nguồn điện 220VAC. Khi cúp điện lưới thì hệ thống sẽ không hoạt động.

- Đồng hồ chỉ cài đặt được một khung giờ báo thức, không cài đặt được nhiều khung giờ để báo thức. Loa cũng phát được một loại âm thanh “bíp”, không phát nhạc được. Nếu muốn phát nhạc thì phải sử dụng các loại module khác có bộ nhớ lưu trữ tập tin âm thanh.

8. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

8.1. Kết luận

Bài tập lớn Thiết kế hệ thống nhúng với chủ đề Đồng hồ kỹ thuật số hiển thị LCD dùng DS1307 đã cung cấp một cái nhìn sâu rộng về ứng dụng vi điều khiển trong việc quản

lý thời gian thực. Qua quá trình thiết kế và triển khai, nhóm đã thực hiện thành công một hệ thống nhúng có khả năng hiển thị chính xác thời gian và ngày tháng, đồng thời cung cấp chức năng báo thức.

Việc sử dụng DS1307 – một module đồng hồ thời gian thực phổ biến, kết hợp với màn hình LCD đã chứng minh được tính hiệu quả và khả năng ứng dụng cao của hệ thống. Bên cạnh đó, nhóm cũng học được thêm các kỹ thuật lập trình giao tiếp I2C, xử lý ngắt ngoại vi, hiển thị dữ liệu lên LCD và đã được áp dụng các kĩ năng trên một cách hiệu quả, giúp hệ thống hoạt động ổn định và chính xác.

Tuy nhiên, hệ thống cũng tồn tại một số hạn chế như độ chính xác thời gian chưa cao và nguồn điện để duy trì hoạt động của đồng hồ chưa thật sự hiệu quả và vẫn còn nhiều thiếu sót, chính vì vậy, nhóm rất mong nhận được sự góp ý từ thầy.

8.2. Hướng phát triển

Dựa trên các hạn chế đã nêu, nhóm đề xuất một số hướng phát triển nhằm hoàn thiện và nâng cao tính ứng dụng của hệ thống:

Tăng độ chính xác thời gian: Thay thế IC DS1307 bằng DS3231 – một module RTC có độ chính xác cao hơn và ít bị ảnh hưởng bởi nhiệt độ môi trường, giúp giảm thiểu sai số thời gian

Cải thiện giao diện hiển thị: Nâng cấp từ màn hình LCD truyền thống sang màn hình OLED hoặc TFT có độ phân giải cao, cho phép hiển thị đồ họa và thông tin đa dạng hơn.

Tích hợp tính năng thông minh: Bổ sung các giao tiếp không dây như Wi-Fi hoặc Bluetooth để điều khiển từ xa qua ứng dụng di động/web. Ngoài ra, tích hợp thêm các cảm biến môi trường (ánh sáng, nhiệt độ, độ ẩm...) để hiển thị thông tin thời tiết và điều chỉnh báo thức tự động theo điều kiện thực tế.

Ứng dụng trong điều khiển nhà thông minh: Mở rộng hệ thống để điều khiển các thiết bị gia dụng như quạt, đèn, hoặc hệ thống tưới cây tự động theo thời gian cài đặt sẵn.

Nâng cao trải nghiệm người dùng: Thêm tính năng lựa chọn âm báo thức đa dạng hoặc nhạc chuông theo sở thích cá nhân, mang lại trải nghiệm sử dụng thân thiện và linh hoạt hơn.

9. TÀI LIỆU THAM KHẢO

(1) Điện tử tương lai. Truy cập từ: https://dientutuonglai.com/ve-mach-nguyen-ly-bang-altium.html

(2) Nguyễn Lý Thiên Trường. *Slide bài giảng: Chương 6: Giao tiếp ngoại vi*

(3) Nguyễn Lý Thiên Trường. *Slide bài giảng: Chương 8: Cổng nối tiếp*

(4) Nguyễn Lý Thiên Trường. *Slide bài giảng: Chương 10: Ngắt*

(5) Sepehr Naimi (12/12/2017). *C programming in Atmel Studio 7*

(6) Muhammad Ali Mazidi, Sarmad Naimi, Sepehr Naimi (2014). *The AVR Micontroller anh Embedded Systems: Using Assembly and C*