BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**THIẾT KẾ ĐỒNG HỒ THỜI GIAN THỰC SỬ DỤNG KIT ARDUINO VÀ IC DS1307**

**Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh**

**Sinh viên thực hiện: Nguyễn Duy Tín - Nguyễn Võ Tiền**

**Mã số sinh viên: 61131272 - 60137144**

KHÁNH HÒA-2021

TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**THIẾT KẾ ĐỒNG HỒ THỜI GIAN THỰC SỬ DỤNG KIT ARDUINO VÀ IC DS1307**

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Nguyễn Duy Tín **-** Nguyễn Võ Tiền

Mã số sinh viên: 61131272- 60137144

Khánh Hòa, tháng 01/2021

**LỜI CẢM ƠN**

**LỜI MỞ ĐẦU**

MỤC LỤC

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc61963139)

[1. LCD20x4 2](#_Toc61963140)

[2. Mô đun thời gian thực IC DS1307 3](#_Toc61963141)

[3. Arduino UNO R3 6](#_Toc61963142)

[4. Arduino IDE 8](#_Toc61963143)

[5. Proteus version 8.9 SP0 10](#_Toc61963144)

[CHƯƠNG 2: PHƯƠNG PHÁP NGHIÊN CỨU 12](#_Toc61963145)

[1. Sơ đồ nguyên lý mô phỏng bằng Proteus 8.9 SP0 12](#_Toc61963146)

[1.1. Giao tiếp giữa Arduino và IC DS1307 12](#_Toc61963147)

[1.2. Giao tiếp giữa Arduino với LCD20x4 13](#_Toc61963148)

[1.3. Giao tiếp giữa Arduino-Loa speaker 13](#_Toc61963149)

[1.4. Giao tiếp giữa Arduino – 3 nút nhấn 13](#_Toc61963150)

[2. Thuật toán 14](#_Toc61963151)

[2.1. Chương trình chính 14](#_Toc61963152)

[2.2. Khởi tạo RTC và LCD 15](#_Toc61963153)

[2.3. Hiển thị thời gian lên LCD 17](#_Toc61963154)

[2.4. Các hàm phục vụ Báo thức 18](#_Toc61963155)

[2.4.1 Âm nhạc 18](#_Toc61963156)

[2.4.2 Hiển thị giờ phút cho báo thức 23](#_Toc61963157)

[2.4.3 Đặt giờ báo thức 24](#_Toc61963158)

[2.4.4 Kiểm tra giờ báo thức 30](#_Toc61963159)

[CHƯƠNG 3: KẾT QUẢ THỰC HIỆN 31](#_Toc61963160)

[1. Kết quả mô phỏng trên Proteus 8.9 SP0 31](#_Toc61963161)

[1.1. Hiển thị ngày giờ 31](#_Toc61963162)

[1.2. Đặt báo thức 31](#_Toc61963163)

[CHƯƠNG 4: PHỤ LỤC 33](#_Toc61963164)

**DANH MỤC HÌNH**

[Hình 1‑1 Đồng hồ điện tử 1](#_Toc61963189)

[Hình 1‑2 LCD 20x4 2](#_Toc61963190)

[Hình 1‑3 Các chân của LCD 20x4 2](#_Toc61963191)

[Hình 1‑4 Đồng hồ thời gian thực IC DS1307 3](#_Toc61963192)

[Hình 1‑5 Sơ đồ chân IC DS1307 4](#_Toc61963193)

[Hình 1‑6: Sơ đồ các chân Arduino UNO R3 7](#_Toc61963194)

[Hình 1‑10: Giao diện làm việc Arduino IDE version 1.8.5 9](#_Toc61963195)

[Hình 1‑11: Giao diện làm việc Proteus 8.9 SP0 10](#_Toc61963196)

[Hình 2‑1: Sơ đồ nguyên lý đồng hồ trên Proteus version 8.9 12](#_Toc61963197)

[Hình 2‑2 Sơ đồ thuật toán chương trình chính 14](#_Toc61963198)

[Hình 2‑3 Sơ đồ trình tự khởi tạo RTC và LCD 15](#_Toc61963199)

[Hình 2‑4 Sơ đồ đặt báo thức 24](#_Toc61963200)

[Hình 2‑5 Sơ đồ thuật toán chọn chế độ báo thức 25](#_Toc61963201)

[Hình 2‑6 Sơ đồ thuật toán cho đặt phút báo thức 28](#_Toc61963202)

[Hình 3‑1 Hiển thị ngày giờ mô phỏng trên Proteus 8.9 SP0 31](#_Toc61963203)

[Hình 3‑2 Hiển thị chế độ báo thức mô phỏng trên Proteus 8.9 SP0 31](#_Toc61963204)

[Hình 3‑3 Đặt giờ báo thức mô phỏng trên Proteus 8.9 SP0 32](#_Toc61963205)

[Hình 3‑4 Báo thức khi đúng giờ mô phỏng trên Proteus 8.9 SP0 32](#_Toc61963206)

# TỔNG QUAN

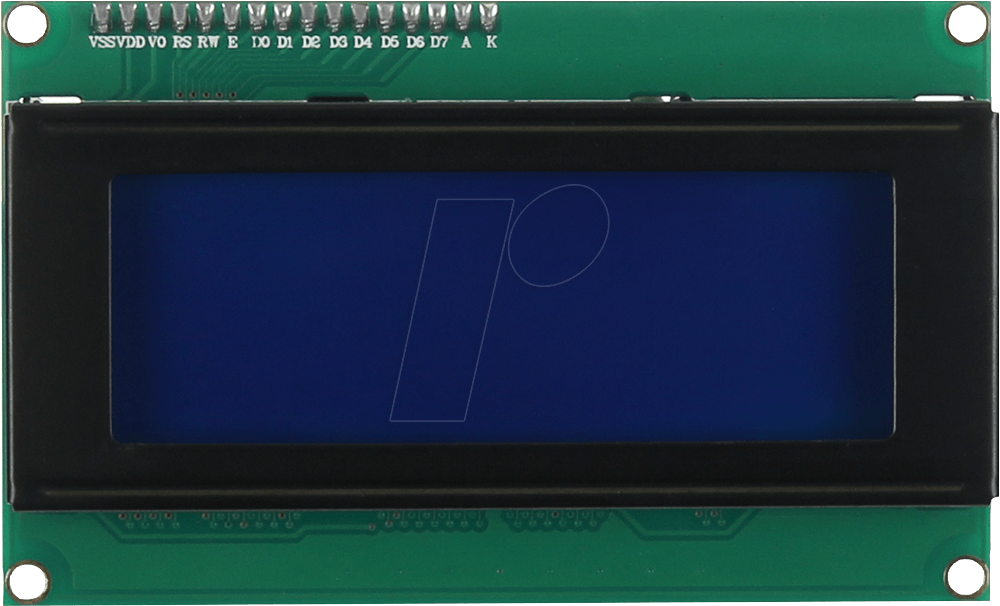
Đồng hồ báo thức điện tử hiện đang kinh doanh trên thị trường có thể theo dõi được ngày, tháng, năm dương lịch, âm lịch và giờ giấc giúp chúng ta sắp xếp được thời gian biểu của mình một cách hợp lý. Hình 1.1 là một ví dụ về Đồng hồ báo thức điện tử được bán trên thị trường. Thành phần cơ bản bao gồm bộ hiển thị LCD cho biết thời gian như: ngày, tháng, năm, giờ, phút, giây và còn hiển thị thứ trong tuần, ngoài ra còn có thể đặt báo thức. Để có thể tạo ra một đồng hồ điện tử như hình 1.1 cần vận dụng các kiến thức về lập trình thiết bị nhúng, cụ thể là Kit Arduino (http://electronoobs.com/eng\_arduino\_tut31\_sch3.php), kiến thức về điện tử số và điện tử tương tự để tính toán các thành phần điện tử cơ bản và thực hiện các thao tác ghép nối các thành phần điện tử với nhau. Ngoài ra, một thành phần không thể thiếu chính là module thời gian thực DS1307 (https://www.electroschematics.com/ds1307-datasheet/) được dùng để đọc thời gian hiện hành và hiển thị lên màn hình LCD



Hình 1‑1 Đồng hồ điện tử

(Nguồn : https://www.officeworks.com.au/shop/officeworks/p/phillips-aj3400-lcd-alarm-clock-fm-black-pmaj3400)

## LCD20x4



Hình 1‑2 LCD 20x4

(Nguồn: https://www.reichelt.com/de/en/developer-boards-display-20-x-4-characters-blue-debo-lcd-20x4-bl-p192144.html)

LCD (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của vi điều khiển. LCD có rất nhiều ưu điểm so với các dạng hiển thị khác. Nó có khả năng hiển thị kí tự đa dạng, trực quan (chữ, số và kí tự đồ họa), dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tốn rất ít tài nguyên hệ thống và giá thành rẻ.

**Thông số kỹ thuật:**

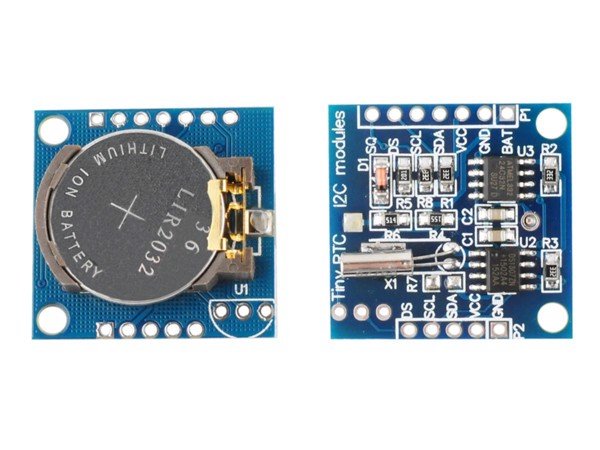
* Điện áp hoạt động: 5V
* Hiển thị tối đa 20 tự trên 4 dòng
* Chữ trắng nền xanh dương
* Kích thước: 98 x 60 x 13.5 mm
* Khoảng cách giữa hai chân kết nối là 0.1 inch tiện dụng khi kết nối với Breadboard.
* driver HD44780

****

Hình 1‑3 Các chân của LCD 20x4

* Chân (1): (Vss) Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
* Chân (2): VDD Là chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với VCC = 5V của mạch điều khiển
* Chân (3): V0 là chân điều chỉnh độ tương phản của LCD.
* Chân (4): RS Là chân chọn thanh ghi (Register select). Nối chân RS với logic “0” (GND) hoặc logic “1” (VCC) để chọn thanh ghi.
* 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.
* Chân (5): R/W là 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.
* Chân (6): E Là 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 (high-to-low transition) của tín hiệu chân E.
* Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
* Chân (7 - 14): D0 - D7 - 8 đường của bus dữ liệu dùng để trao đổi thông tin với vi điều khiển. Có 2 chế độ sử dụng 8 đường bus này: (1) Chế độ 8 bit: Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. (2) Chế độ 4 bit: Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7
* Chân (15): Nguồn dương cho đèn nền
* Chân (16): GND cho đèn nền

## Mô đun thời gian thực IC DS1307



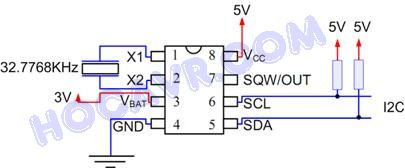
Hình 1‑4 Đồng hồ thời gian thực IC DS1307

(Nguồn https://hshop.vn/products/mach-thoi-gian-thuc-rtc-ds1307)

Đồng hồ thời gian thực nối tiếp DS1307 (RTC) là đồng hồ / lịch thập phân được mã hóa nhị phân (BCD) công suất thấp, đầy đủ cộng với 56 byte NV SRAM. Địa chỉ và dữ liệu là được chuyển nối tiếp qua chuẩn I2C, bus hai chiều. Đồng hồ / lịch cung cấp giây, phút, giờ, thông tin ngày, tháng, và năm. Sự kết thúc của ngày tháng được tự động điều chỉnh cho các tháng với ít hơn 31 ngày, bao gồm các chỉnh sửa cho bước nhảy vọt năm. Đồng hồ hoạt động 24 giờ hoặc 12 giờ định dạng giờ với chỉ báo AM / PM. DS1307 có một mạch cảm biến nguồn tích hợp phát hiện sự cố mất điện và tự động chuyển sang nguồn dự phòng. Hoạt động thời gian vẫn tiếp tục trong khi phần hoạt động từ nguồn cung cấp dự phòng

**Thông số kỹ thuật:**

* IC chính: RTC DS1307 + EEPROM AT24C32
* Nguồn cung cấp: 5VDC.
* Giao tiếp: I2C
* Lưu trữ và cung cấp các thông tin thời gian thực: ngày, tháng, năm, giờ, phút, giây,...
* Có pin backup duy trì thời gian trong trường hợp không cấp nguồn.
* Có ngõ ra tần số 1Hz.
* Kích thước: 27 x 28 x 8.4mm



Hình 1‑5 Sơ đồ chân IC DS1307

(Nguồn: http://www.hocavr.com/2018/06/ong-ho-thoi-gian-thuc-ds1307.html)

* Chân (1,2) X1-X2: Là 2 ngõ kết nối với 1 thạch anh 32.768KHz làm nguồn tạo dao động cho chip. Mạch dao động bên trong là được thiết kế để hoạt động với tinh thể có điện dung tải xác định (CL) là 12,5pF. X1 là đầu vào cho bộ dao động và có thể được kết nối tùy chọn với 32,768kHz bên ngoài bộ dao động. Đầu ra của bộ dao động bên trong, X2, được thả nổi nếu bộ dao động bên ngoài là kết nối với X1.
* Chân (3) VBAT:  cực dương của một nguồn pin 3V nuôi chip. Đầu vào nguồn cung cấp dự phòng cho mọi tế bào Lithium 3V tiêu chuẩn hoặc nguồn năng lượng khác. Ắc quy điện áp phải được giữ giữa giới hạn tối thiểu và tối đa để hoạt động thích hợp. Điốt mắc nối tiếp giữa pin và chân VBAT có thể ngăn hoạt động bình thường. Nếu một nguồn cung cấp dự phòng là không cần thiết, VBAT phải được nối đất. Điểm chuyến đi mất điện danh định (VPF) điện áp mà tại đó quyền truy cập vào RTC và RAM của người dùng bị từ chối được thiết lập bởi nội bộ mạch như 1.25 x VBAT danh nghĩa. Pin lithium có 48mAh trở lên sẽ sao lưu DS1307 trong hơn 10 năm khi không có điện ở + 25 ° C
* Chân (4) GROUND: chân mass chung cho cả pin 3V và Vcc
* Chân (5) SDA: đường giao xung nhịp và dữ liệu của giao diện I2C .Đầu vào / đầu ra dữ liệu nối tiếp. SDA là đầu vào / đầu ra dữ liệu cho chuẩn giao tiếp I2C. Các Chân SDA bị hở cống và cần có điện trở pullup bên ngoài. Điện áp kéo lên có thể là lên đến 5.5V bất kể điện áp trên VCC.
* Chân (6) SCL: đường giao xung nhịp và dữ liệu của giao diện I2C . Đầu vào đồng hồ nối tiếp. SCL là đầu vào xung nhịp cho chuẩn giao tiếp I2C và được sử dụng để đồng bộ hóa chuyển động dữ liệu trên giao diện nối tiếp. Điện áp kéo lên có thể lên đến 5,5V bất kể điện áp trên VCC.
* Chân (7) SQW/OUT: một ngõ phụ tạo xung vuông . Khi được bật, bit SQWE được đặt thành 1, chân SQW / OUT xuất ra một trong bốn tần số sóng vuông (1Hz, 4kHz, 8kHz, 32kHz). SQW / OUT chân là cống mở và yêu cầu một điện trở pullup bên ngoài. SQW / OUT hoạt động với Áp dụng VCC hoặc VBAT. Điện áp kéo lên có thể lên đến 5,5V bất kể điện áp bật VCC. Nếu không sử dụng, ghim này có thể được để nổi
* Chân (8) VCC: nguồn cho giao diện I2C, thường là 5V và dùng chung với vi điều khiển. Nguồn điện sơ cấp, khi điện áp được đặt trong giới hạn bình thường, thiết bị hoàn toàn có thể truy cập và dữ liệu có thể được ghi và đọc. Khi nguồn cung cấp dự phòng được kết nối với thiết bị và VCC ở dưới VTP, đọc và ghi bị hạn chế. Tuy nhiên, việc chấm công chức năng tiếp tục mà không bị ảnh hưởng bởi điện áp đầu vào thấp hơn.

## Arduino UNO R3

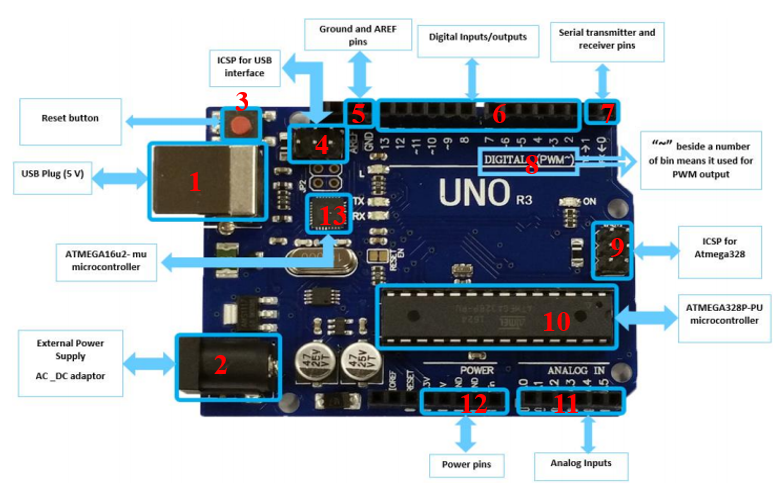
Arduino một nền tảng mã nguồn mở phần cứng và phần mềm. Phần cứng Arduino (các board mạch vi xử lý) được sinh ra tại thị trấn Ivrea ở Ý, nhằm xây dựng các ứng dụng tương tác với nhau hoặc với môi trường được thuận lợi hơn. Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM Atmel 32-bit. Những Model hiện tại được trang bị gồm 1 cổng giao tiếp USB, 6 chân đầu vào analog, 14 chân I/O kỹ thuật số tương thích với nhiều board mở rộng khác nhau.

Được giới thiệu vào năm 2005, những nhà thiết kế của Arduino cố gắng mang đến một phương thức dễ dàng, không tốn kém cho những người yêu thích, sinh viên và giới chuyên nghiệp để tạo ra những thiết bị có khả năng tương tác với môi trường thông qua các cảm biến và các cơ cấu chấp hành. Những ví dụ phổ biến cho những người yêu thích mới bắt đầu bao gồm các robot đơn giản, điều khiển nhiệt độ và phát hiện chuyển động. Đi cùng với nó là một môi trường phát triển tích hợp (IDE) chạy trên các máy tính cá nhân thông thường và cho phép người dùng viết các chương trình cho Aduino bằng ngôn ngữ C hoặc C++.

Phần cứng Arduino gốc được sản xuất bởi công ty Italy tên là Smart Projects. Một vài board dẫn xuất từ Arduino cũng được thiết kế bởi công ty của Mỹ tên là SparkFun Electronics. Một số phiên bản Arduino phổ biến như: Arduino Diecimila, Arduino Duemilanove, Arduino UNO, Arduino Leonardo, Arduino Mega, Arduino MEGA 2560 R3, Arduino Nano, Arduino Due, LilyPad Arduino

**Đặc tính kỹ thuật:**

* Vi điều khiển ATMEGA328P-PU
* Nguồn Cấp : 7-12V
* Dòng Max chân 5V : 500mA
* Dòng Max 3.3V : 50mA
* Dòng Max Chân I/O : 30mA
* 14 Chân Digital I/O  (6 chân PWM)
* 6 Chân Analog Inputs
* 32k Flash Memory
* 16Mhz Clock Speed
* SRAM 2 KB
* EEPROM 1 KB



Hình 1‑6: Sơ đồ các chân Arduino UNO R3

(Nguồn: <http://www.fecegypt.com/uploads/dataSheet/1522237550_arduino%20uno%20r3.pdf>)

(1) USB: Cổng giao tiếp USB có 2 chức năng: cấp nguồn cho board mạch và truyền thông nối tiếp với máy tính trong việc nạp chương trình hay giao tiếp nối tiếp.

(2) Jack Power: Nguồn cấp cho board mạch Arduino. Có 2 loại nguồn có thể sử dụng được là nguồn xoay chiều (AC) tối đa 6V và nguồn một chiều (DC) tối đa 5V.

(3) RESET: Đặt lại trạng thái ngay khi nạp chương trình

(4) ISCP: Chân giao tiếp với USB, tín hiệu giao tiếp có thể giám sát từ đây.

(5) AREF và GND: GND hay chân Mass hay chân đất dùng để cấp nguồn 0V cho các modue khác có kết nối với Arduino. Trong khi chân AREF được dùng để phối hợp với các chân Analog Input (11) để điều chỉnh dải điện áp đầu vào cho ADC 10 bit.

(6) DIGITAL Input/Output: 12 chân tín hiệu số được dùng làm đầu vào hoặc đầu ra tại mỗi thời điểm.

(7) Serial Transmition: 2 chân RXD và TXD được dùng trong truyền thông nối tiếp. Khi 2 chân này cũng có thể được cấu hình làm đầu vào hoặc đầu ra tín hiệu số.

(8) PWM (Pulse Width Modulation): Chân băm xung có ký hiệu là ~ dùng để tạo ra chuỗi xung vuông trong điều khiển tốc độ động cơ hay hiển thị LED sáng dần hay tối dần.

(9) ICSP: Giao tiếp với Atmega328.

(10) IC lập trình ATMEGA16u2

(11) ANALOG Input: Tín hiệu chuyển đổi ADC được đưa vào các chân này. Trong trường hợp khác các chân này cũng có thể sử dụng để làm đầu vào/ra tín hiệu số. Đặc biệt, Arduino UNO có 2 chân **A4 (SDA)** và **A5 (SCL)** hỗ trợ giao tiếp **I2C/TWI** với các thiết bị khác

(12) POWER: Cung cấp nguồn 5V, GND cho các thành phần mở rộng

## Arduino IDE

Môi trường phát triển tích hợp ([IDE](https://vi.wikipedia.org/wiki/IDE)) của Arduino là một ứng dụng [cross-platform](https://vi.wikipedia.org/wiki/Cross-platform) (đa nền tảng) được viết bằng [Java](https://vi.wikipedia.org/wiki/Java_(programming_language)), và từ IDE này sẽ được sử dụng cho [Ngôn ngữ lập trình xử lý](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh_x%E1%BB%AD_l%C3%BD&action=edit&redlink=1) (Processing programming language) và project [Wiring](https://vi.wikipedia.org/w/index.php?title=Wiring_(development_platform)&action=edit&redlink=1). Nó được thiết kế để dành cho những người mới tập làm quen với lĩnh vực phát triển phần mềm. Nó bao gồm một chương trình code editor với các chức năng như đánh dấu cú pháp, tự động [brace matching](https://vi.wikipedia.org/w/index.php?title=Brace_matching&action=edit&redlink=1), và tự động canh lề, cũng như compile(biên dịch) và upload chương trình lên board chỉ với 1 cú nhấp chuột. Một chương trình hoặc code viết cho Arduino được gọi là một sketch.

Các chương trình Arduino được viết bằng [C](https://vi.wikipedia.org/wiki/C_(programming_language)) hoặc [C++](https://vi.wikipedia.org/wiki/C%2B%2B). Arduino IDE đi kèm với một [thư viện phần mềm](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) được gọi là "[Wiring](https://vi.wikipedia.org/w/index.php?title=Wiring_(development_platform)&action=edit&redlink=1)", từ project Wiring gốc, có thể giúp các thao tác input/output được dễ dàng hơn. Người dùng chỉ cần định nghĩa 2 hàm để tạo ra một chương trình [vòng thực thi](https://vi.wikipedia.org/w/index.php?title=V%C3%B2ng_th%E1%BB%B1c_thi&action=edit&redlink=1) (cyclic executive) có thể chạy được:

* setup(): hàm này chạy mỗi khi khởi động một chương trình, dùng để thiết lập các cài đặt
* loop(): hàm này được gọi lặp lại cho đến khi tắt nguồn board mạch

Chương trình Arduino có thể được chia làm 3 phần: cấu trúc (structure), biến số (variable) và hằng số (constant), hàm và thủ tục (function). Một số hàm được liệt kê trong bảng ở **phụ lục**



Hình 1‑10: Giao diện làm việc Arduino IDE version 1.8.5

**Nút kiểm tra chương trình**: Dùng để kiểm tra xem chương trình được viết có lỗi không. Nếu chương trình bị lỗi thì phần mềm Arduino IDE sẽ hiển thị thông tin lỗi ở vùng thông báo thông tin.

**Nút nạp chương trình xuống bo Arduino**: Dùng để nạp chương trình được viết xuống mạch Arduino. Trong quá trình nạp, chương trình sẽ được kiểm tra lỗi trước sau đó mới thực hiện nạp xuống mạch Arduino.

**Hiển thị màn hình giao tiếp với máy tính:** Khi nhấp vào biểu tượng cái kính lúp thì phần giao tiếp với máy tính sẽ được mở ra. Phần này sẽ hiển thị các thông số mà người dùng muốn đưa lên màn hình. Muốn đưa lên màn hình phải có lệnh Serial.print() mới có thể đưa thông số cần hiển thị lên màn hình

**Vùng lập trình:** vùng này để người lập trình thực hiện việc lập trình cho chương trình của mình.

**Vùng thông báo thông tin:** Có chức năng thông báo các thông tin lỗi của chương trình hoặc các vấn đề liên quan đến chương trình được lập.

Có vài menu trong phần mềm IDE, tuy nhiên thông dụng nhất vẫn là menu File, ngoài những tính năng như mở một file mới hay lưu một file, phần menu này có một mục đáng chú ý là Example. Phần Example (ví dụ) đưa ra các ví dụ sẵn để người lập trình có thể tham khảo, giảm bớt thời gian lập trình. Hình bên dưới thể hiện việc chọn một ví dụ cho led chớp tắt (blink) để nạp cho mạch Arduino. Ví dụ về led chớp tắt này thường được dùng để kiểm tra bo khi mới mua về

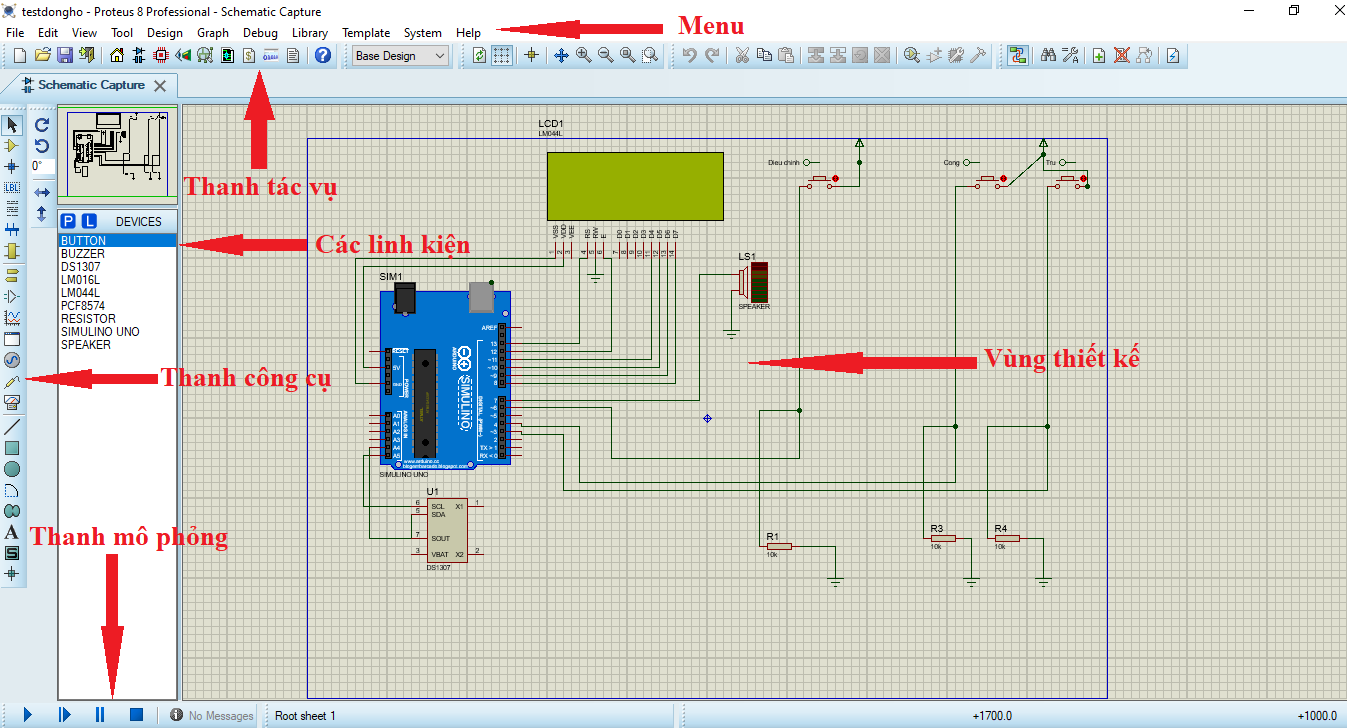
Khi mới kết nối bo Arduino với máy tính ta click vào **Tools->board** để chọn loại board sử dụng. Phần mềm chọn sẵn kiểu bo là bo **Arduino Uno**, nếu người dùng dùng kiểu bo khác thì chọn kiểu bo đang dùng.

Bên cạnh việc chọn bo thì một phần quan trọng nữa là chọn cổng COM. Hình bên dưới minh họa cho việc chọn cổng COM. Khi lần đầu gắn mạch Arduino vào máy tính, người sử dụng cần nhấn chọn cổng COM bằng cách vào **Tools -> Serial Port** (một số phiên bản dùng từ Port) sau đó nhấn chọn cổng COM, ví dụ như COM5. Những lần sau khi đưa chính board Arduino đó vào máy tính thì không cần chọn cổng COM, nếu đưa bo Arduino khác vào máy thì cần phải chọn lại cổng COM, quy trình thực hiện cũng tương tự.

Ngoài ra có thể xuất file .HEX từ menu Sketch , phím tắt Ctrl + Alt + S (đường dẫn file .HEX nằm chung thư mục với chương trình)

## Proteus version 8.9 SP0

Proteus Design Suite là bộ công cụ phần mềm độc quyền được sử dụng chủ yếu cho tự động hóa thiết kế điện tử. Phần mềm được sử dụng chủ yếu bởi các kỹ sư thiết kế điện tử và kỹ thuật viên để tạo sơ đồ và bản in điện tử để sản xuất bảng mạch in.



Hình 1‑11: Giao diện làm việc Proteus 8.9 SP0

**Thanh Menu:** chứa các chức năng chính của chương trình

**Thanh công cụ:** chứa các công cụ để làm việc như thêm thiết bị, lựa chọn, thêm nhãn v.v.v. Một chức năng quan trọng là chọn thiết bị sử dụng. Trên thanh công cụ, chọn Component mode, Pick Device, tìm kiếm thiết bị trong các thư viện.

**Các linh kiện:** danh sách các linh kiện được thêm

**Vùng thiết kế:** nơi thiết kế, mô phỏng các linh kiện, sơ đồ mạch

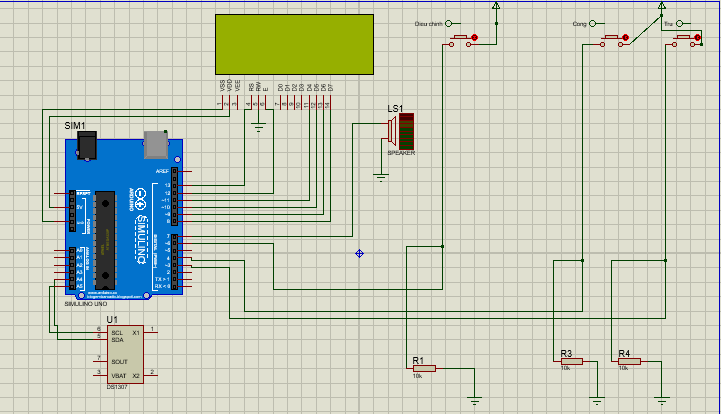
**Thanh mô phỏng:** mô phỏng hoạt động của các linh kiện trong vùng thiết kế

Để nạp chương trình vào mạch điều khiển mô phỏng ,nhấp chuột vào vi xử lý mô phỏng trên mạch, mở thư mục và tìm đến file .HEX mong muốn.

Để sử dụng các thiết bị không có sẵn trong thư viện Proteus, tải các thư viện thiết bị và copy file .IDX và .LIB vào thư mục LIBRARY của thư mục Proteus

# PHƯƠNG PHÁP NGHIÊN CỨU

## Sơ đồ nguyên lý mô phỏng bằng Proteus 8.9 SP0



Hình 2‑1: Sơ đồ nguyên lý đồng hồ trên Proteus version 8.9

Sử dụng phần mềm mô phỏng Proteus version 8.9 SP0, cùng các thiết bị chính:

* LCD20x4 (LM044) thư viện DISPLAY
* DS1307 thư viện MAXIM
* Arduino UNO thư viện ARDUINO
* Button thư viện ACTIVE
* Speaker thư viện ACTIVE

### Giao tiếp giữa Arduino và IC DS1307

Giao tiếp với IC DS1307 theo chuẩn giao tiếp I2C. I2C là viết tắt của "Inter-Integrated Circuit", một chuẩn giao tiếp được phát minh bởi Philips’ semiconductor division (giờ là NXP) nhằm đơn giản hóa việc trao đổi dữ liệu giữa các ICs. Đôi khi nó cũng được gọi là Two Wire Interface (TWI) vì chỉ sử dụng 2 kết nối để truyền tải dữ liệu, 2 kết nối của giao tiếp I2C gồm: SDA (Serial Data Line) và SCL (Serial Clock Line). Trên board Arduino UNO, SDA là chân analog 4, SCL là chân analog 5.

* Chân A5 - SCL
* Chân A4 - SDA

### Giao tiếp giữa Arduino với LCD20x4

Sau khi đọc dữ liệu từ IC DS1307, Arduino sẽ xuất thông tin thời gian lên LCD. LCD 20x4 có thể được sử dụng ở chế độ 4 bit hoặc chế độ 8 bit tùy thuộc vào yêu cầu của ứng dụng

* Trong chế độ 4 bit, dữ liệu / lệnh được gửi ở định dạng 4 bit (nibble).
* Để làm điều này 1 gửi cao hơn 4-bit và sau đó gửi thấp hơn 4-bit của dữ liệu / lệnh.
* Chỉ có 4 chân dữ liệu (D4 - D7) của LCD 20x4 được kết nối với vi điều khiển và các chốt điều khiển khác RS (Đăng ký chọn), RW (Đọc / ghi), E (Bật) được kết nối với các Chân GPIO khác của bộ điều khiển.
* Vì vậy, do các kết nối như vậy, có thể tiết kiệm bốn chân GPIO có thể được sử dụng cho một ứng dụng khác.
* Cấp nguồn cho LCD:
* GND – VSS
* 5V – VDD
* Thiết lập chế độ LCD:
* 13 – RS
* 12 – E
* RW nối mass
* 4 chân truyền dữ liệu:
* 11 – D4
* 10 – D5
* 9 – D6
* 8 – D7

(cần thêm dữ liệu về 4 bit lcd)

### Giao tiếp giữa Arduino-Loa speaker

Arduino sẽ điều khiển Loa phát âm thanh khi báo thức

* 7 – 1 chân loa
* Chân còn lại của Loa nối mass

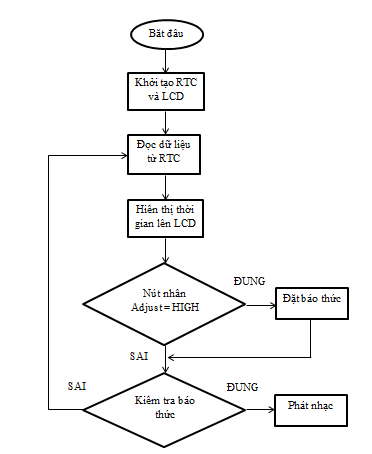
### Giao tiếp giữa Arduino – 3 nút nhấn

Dùng để điều khiển các chức năng của chương trình

* 6 – Điều chỉnh (Enter)
* 4 – Tiến (tăng)
* 3 – Lùi (giảm)
* Các đầu cực còn lại của 4 nút nhấn nối với nguồn điện
* Nối các điện trở theo sơ đồ (liên quan đến dòng điện trong vật lý)

## Thuật toán

### Chương trình chính



Hình 2‑2 Sơ đồ thuật toán chương trình chính

void loop() {

DateTime now = rtc.now();

showtime(now);

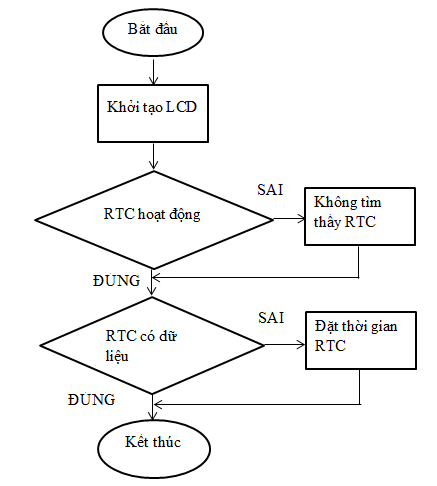
if(digitalRead(Adjust)==HIGH)

alarm(now);

checkalarm(now);

}

### Khởi tạo RTC và LCD



Hình 2‑3 Sơ đồ trình tự khởi tạo RTC và LCD

* Để làm việc với RTC và LCD ta cần 3 thư viện là Wire.h, RTClib.h, LiquidCrystal.h.

#include <Wire.h>

#include "RTClib.h"

#include <LiquidCrystal.h>

* Khai báo LCD bằng hàm **LiquidCrystal lcd()** , và đối tượng rtc thuộc **RTC\_DS1307**

//khai bao man hinh lcd

const int pin\_RS = 13;

const int pin\_EN = 12;

const int pin\_d4 = 11;

const int pin\_d5 = 10;

const int pin\_d6 = 9;

const int pin\_d7 = 8;

// khai báo lcd

LiquidCrystal lcd(pin\_RS,pin\_EN,pin\_d4,pin\_d5,pin\_d6,pin\_d7);

// doi tuong dong ho

RTC\_DS1307 rtc;

* Trong hàm setup(). Khởi tạo LCD bằng hàm **lcd.begin().** Kiểm tra RTC có hoạt động không bằng hàm **rtc.begin(),** kiểm tra RTC có dữ liệu không bằng hàm **rtc.isrunning().** Đặt thời gian cho RTC là ngày giờ hiện tại của hệ thống bằng hàm **rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)))**

void setup() {

//khoi tao cac nut bam

pinMode(Adjust,INPUT);

pinMode(Tien,INPUT);

pinMode(Lui,INPUT);

// khoi tao lcd

lcd.begin(20, 4);

// khoi tao dong ho

if (! rtc.begin()) {

lcd.print("Khong tim thay RTC");

while (1);

}

if (! rtc.isrunning()) {

lcd.clear();

lcd.print("RTC chua dat thoi gian");

// Dat ngay gio dong ho theo hien tai cua may tinh

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

// ta cung co the dat theo y muon theo vd ben duoi

// January 21, 2014 at 3am

// rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));

}

}

### Hiển thị thời gian lên LCD

Cùng với thư viện RTClib.h, Wire.h , các hàm có sẵn hỗ trợ lấy dữ liệu thời gian của RTC mà không cần chỉ địa chỉ RTC:

* <biến DateTime>.day(): lấy ngày
* <biến DateTime>.month(): lấy tháng
* <biến DateTime>.year(): lấy năm
* <biến DateTime>.dayOfTheWeek(): trả về ngày trong tuần
* <biến DateTime>.hour(): lấy giờ
* <biến DateTime>.minute(): lấy phút
* <biến DateTime>.second(): lấy giây

Với thư viện LiquidCrystal.h , một số hàm để sử dụng LCD trong chương trình:

* lcd.print(“<dữ liệu>”): xuất dữ liệu lên LCD theo vị trí hiện hành của con trỏ
* lcd.setCursor(“<cột>,<hàng>”): đặt vị trí con trỏ (lưu ý: chỉ số hàng cột tính   
  từ 0)

Để hiển thị thời gian trong chương trình, các số có ít hơn 2 chữ số sẽ hiển thị thêm số 0 phía trước, mã lệnh dưới đây hiển thị giờ, tương tự cho phút, giây, ngày, tháng , năm

// hien thi gio

lcd.setCursor(0, 1);

if(now.hour()<=9)

{

lcd.print("0");

lcd.print(now.hour());

}

else {

lcd.print(now.hour());

}

### Các hàm phục vụ Báo thức

#### Âm nhạc

Một nốt nhạc có tần số và thời gian phát, hàm **tone**(<chân loa>,<tần số>,<thời gian>). Một bản nhạc bao gồm nhiều nốt, có 7 nốt nhạc cơ bản Đồ Rê Mi Fa Sol La Si, và có nhiều quãng 7 nốt như vậy nhưng khác tần số, để lưu trữ các nốt nhạc trong chương trình ta cần một mảng lưu trữ các nốt và một mảng lưu trữ thời gian phát của nốt.

Khai báo tên các nốt Đồ Rê Mi Fa Sol La Si có trong bài nhạc cùng cao độ của chúng.

#define NOTE\_C4 262

#define NOTE\_D4 294

#define NOTE\_E4 330

#define NOTE\_F4 349

#define NOTE\_G4 392

#define NOTE\_A4 440

#define NOTE\_B4 494

#define NOTE\_C5 523

#define NOTE\_D5 587

#define NOTE\_E5 659

#define NOTE\_F5 698

#define NOTE\_G5 784

#define NOTE\_A5 880

#define NOTE\_B5 988

**Mảng lữu trữ các nốt nhạc có trong bài nhạc:**

int notes[] = {

NOTE\_E4, NOTE\_G4, NOTE\_A4, NOTE\_A4, 0,

NOTE\_A4, NOTE\_B4, NOTE\_C5, NOTE\_C5, 0,

NOTE\_C5, NOTE\_D5, NOTE\_B4, NOTE\_B4, 0,

NOTE\_A4, NOTE\_G4, NOTE\_A4, 0,

NOTE\_E4, NOTE\_G4, NOTE\_A4, NOTE\_A4, 0,

NOTE\_A4, NOTE\_B4, NOTE\_C5, NOTE\_C5, 0,

NOTE\_C5, NOTE\_D5, NOTE\_B4, NOTE\_B4, 0,

NOTE\_A4, NOTE\_G4, NOTE\_A4, 0,

NOTE\_E4, NOTE\_G4, NOTE\_A4, NOTE\_A4, 0,

NOTE\_A4, NOTE\_C5, NOTE\_D5, NOTE\_D5, 0,

NOTE\_D5, NOTE\_E5, NOTE\_F5, NOTE\_F5, 0,

NOTE\_E5, NOTE\_D5, NOTE\_E5, NOTE\_A4, 0,

NOTE\_A4, NOTE\_B4, NOTE\_C5, NOTE\_C5, 0,

NOTE\_D5, NOTE\_E5, NOTE\_A4, 0,

NOTE\_A4, NOTE\_C5, NOTE\_B4, NOTE\_B4, 0,

NOTE\_C5, NOTE\_A4, NOTE\_B4, 0,

NOTE\_A4, NOTE\_A4,

//Repeat of first part

NOTE\_A4, NOTE\_B4, NOTE\_C5, NOTE\_C5, 0,

NOTE\_C5, NOTE\_D5, NOTE\_B4, NOTE\_B4, 0,

NOTE\_A4, NOTE\_G4, NOTE\_A4, 0,

NOTE\_E4, NOTE\_G4, NOTE\_A4, NOTE\_A4, 0,

NOTE\_A4, NOTE\_B4, NOTE\_C5, NOTE\_C5, 0,

NOTE\_C5, NOTE\_D5, NOTE\_B4, NOTE\_B4, 0,

NOTE\_A4, NOTE\_G4, NOTE\_A4, 0,

NOTE\_E4, NOTE\_G4, NOTE\_A4, NOTE\_A4, 0,

NOTE\_A4, NOTE\_C5, NOTE\_D5, NOTE\_D5, 0,

NOTE\_D5, NOTE\_E5, NOTE\_F5, NOTE\_F5, 0,

NOTE\_E5, NOTE\_D5, NOTE\_E5, NOTE\_A4, 0,

NOTE\_A4, NOTE\_B4, NOTE\_C5, NOTE\_C5, 0,

NOTE\_D5, NOTE\_E5, NOTE\_A4, 0,

NOTE\_A4, NOTE\_C5, NOTE\_B4, NOTE\_B4, 0,

NOTE\_C5, NOTE\_A4, NOTE\_B4, 0,

//End of Repeat

NOTE\_E5, 0, 0, NOTE\_F5, 0, 0,

NOTE\_E5, NOTE\_E5, 0, NOTE\_G5, 0, NOTE\_E5, NOTE\_D5, 0, 0,

NOTE\_D5, 0, 0, NOTE\_C5, 0, 0,

NOTE\_B4, NOTE\_C5, 0, NOTE\_B4, 0, NOTE\_A4,

NOTE\_E5, 0, 0, NOTE\_F5, 0, 0,

NOTE\_E5, NOTE\_E5, 0, NOTE\_G5, 0, NOTE\_E5, NOTE\_D5, 0, 0,

NOTE\_D5, 0, 0, NOTE\_C5, 0, 0,

NOTE\_B4, NOTE\_C5, 0, NOTE\_B4, 0, NOTE\_A4

};

**Mảng lưu trữ thời gian phát các nốt nhạc tương ứng**:

int duration[] = {

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 375, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 375, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 125, 250, 125,

125, 125, 250, 125, 125,

250, 125, 250, 125,

125, 125, 250, 125, 125,

125, 125, 375, 375,

250, 125,

//Rpeat of First Part

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 375, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 375, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 250, 125, 125,

125, 125, 125, 250, 125,

125, 125, 250, 125, 125,

250, 125, 250, 125,

125, 125, 250, 125, 125,

125, 125, 375, 375,

//End of Repeat

250, 125, 375, 250, 125, 375,

125, 125, 125, 125, 125, 125, 125, 125, 375,

250, 125, 375, 250, 125, 375,

125, 125, 125, 125, 125, 500,

250, 125, 375, 250, 125, 375,

125, 125, 125, 125, 125, 125, 125, 125, 375,

250, 125, 375, 250, 125, 375,

125, 125, 125, 125, 125, 500

};

Tiết tấu bài nhạc dựa vào thời gian phát một nốt nhạc và thời gian giữa các nốt nhạc. Kiểm soát tốc độ bài nhạc bằng việc nhân mảng lưu trữ thời gian phát và khoảng thời gian giữa các nốt với một số x, tốc độ bài nhạc sẽ tăng lên x lần.

Có tổng cộng 203 nốt nhạc trong bài nhạc chương trình, ta cần 1 vòng lặp for duyệt lần lượt và phát các nốt nhạc đó.

void music(){

for (int i = 0; i < 203; i++) {

int wait = duration[i] \* songspeed;

tone(buzzer, notes[i], wait); //tone(pin,frequency,duration)

delay(wait);

}

}

#### Hiển thị giờ phút cho báo thức

Hàm này hiển thị giờ của báo thức, tương tự như hiển thị giờ đồng hồ hiện tại, một ví dụ đoạn chương trình hiển thị giờ báo thức

lcd.setCursor(0, 1);

if(gio<=9)

{

lcd.print("0");

lcd.print(gio);

}

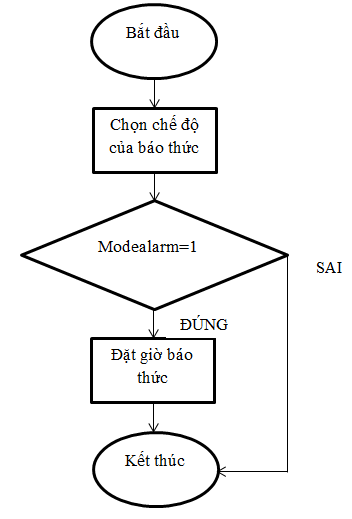
else {

lcd.print(gio);

}

lcd.print(':');

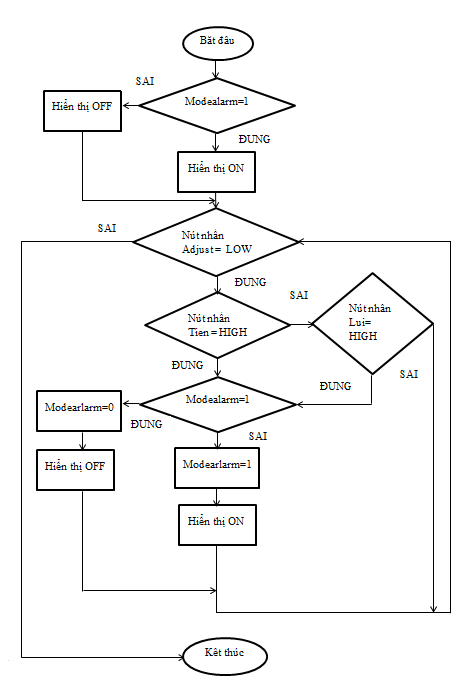
#### Đặt giờ báo thức



Hình 2‑4 Sơ đồ đặt báo thức

Trong hàm này, thực hiện 2 công việc chính, chọn chế độ báo thức Bật/Tắt và đặt giờ báo thức

**Chọn chế độ báo thức:** một biến toàn cục (Modealarm) lưu trữ chế độ của báo thức 1/0 tương ứng Bật/Tắt và hiển thị lên LCD chế độ hiện tại, tiếp theo đọc dữ liệu nút nhấn Tien hoặc Lui để chọn thay đổi giá trị Modealarm 1/0 và hiển thị lên LCD, kết thúc khi dữ liệu đầu vào nút nhấn Adjust là HIGH (khi nút Adjust được nhấn)



Hình 2‑5 Sơ đồ thuật toán chọn chế độ báo thức

lcd.setCursor(0,0);

lcd.print("Alarm: ON/OFF ??");

if(modealarm==0)

{

lcd.setCursor(0,1);

lcd.print("Mode: OFF");

}else {lcd.setCursor(0,1); lcd.print("Mode: ON");}

while(digitalRead(Adjust)==LOW){

if(digitalRead(Tien)==HIGH)

{

delay(350);

if(modealarm==0)

{

modealarm=1;

lcd.setCursor(0,1);

lcd.print("Mode: ON ");

}else if(modealarm==1) {

modealarm=0;

lcd.setCursor(0,1);

lcd.print("Mode: OFF");

}

}

else if(digitalRead(Lui)==HIGH)

{

delay(350);

if(modealarm==0)

{

modealarm=1;

lcd.setCursor(0,1);

lcd.print("Mode: ON ");

}else if(modealarm==1) {

modealarm=0;

lcd.setCursor(0,1);

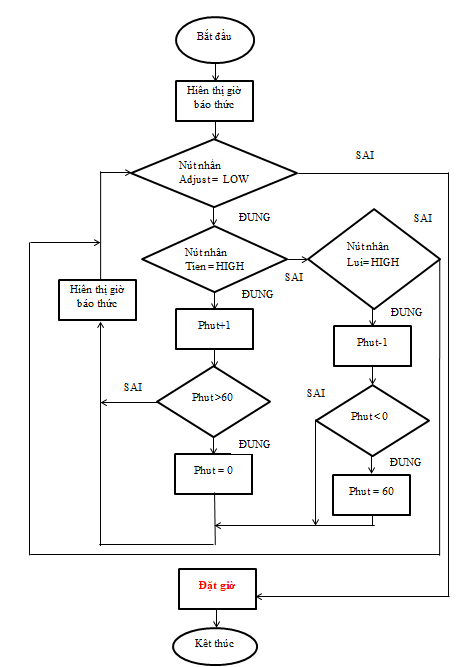
lcd.print("Mode: OFF");

}

}

}

**Đặt giờ báo thức:** hai biến toàn cục lưu trữ giờ (gio) phút (phut) của báo thức. Đầu tiên là đặt phút, hiển thị giờ phút báo thức hiện tại.. Tiếp theo đọc dữ liệu từ 2 nút nhấn Tien và Lui để tăng/giảm biến phut 1 đơn vị, vì giới hạn của phút là [0,60], khi phút đến giá trị 60/0 nếu tiếp sẽ tăng/giảm quay ngược giá trị lại 61=0, -1=60 . Đặt phút báo thức kết thúc khi dữ liệu nút Adjust=HIGH (nút nhấn Adjust được nhấn), và chuyển đến **đặt giờ**, tương tự như đặt phút.



Hình 2‑6 Sơ đồ thuật toán cho đặt phút báo thức

lcd.setCursor(0,0);

lcd.print("Set time alarm:");

//hien thi gio bao thuc

timehour(gio,phut);

//hien thi con tro

lcd.cursor();

lcd.setCursor(4,1);

//dat thoi gian bao thuc

//dat phut

delay(350);

while(digitalRead(Adjust)==LOW){

if(digitalRead(Tien)==HIGH)

{

delay(350);

phut=phut+1;

if(phut>60) phut=0;

timehour(gio,phut);

lcd.setCursor(4,1);

}

else if(digitalRead(Lui)==HIGH)

{

delay(350);

phut=phut-1;

if(phut<0) phut=60;

timehour(gio,phut);

lcd.setCursor(4,1);}

}

Để giúp nhận biết đang đặt giờ phút báo thức tới đâu, hàm **lcd.Cursor()** và **lcd.noCursor()** bật/tắt nhấp nháy vị trí con trỏ hiện tại.

#### Kiểm tra giờ báo thức

Để kiểm tra đúng giờ báo thức, hàm kiểm tra giờ báo thức và giờ hệ thống cùng với chế độ của báo thức, nếu thỏa mãn sẽ thông báo lên màn hình và gọi hàm music() để phát nhạc

void checkalarm(DateTime now){

if(gio==now.hour()&& phut==now.minute()&& modealarm==1)

{

lcd.clear();

lcd.print("\*\*\*\*\*\*\*Alarm\*\*\*\*\*\*\*\*");

timehour(gio,phut);

music();

lcd.clear();

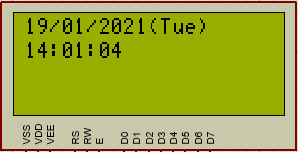
}

}

# KẾT QUẢ THỰC HIỆN

## Kết quả mô phỏng trên Proteus 8.9 SP0

### Hiển thị ngày giờ



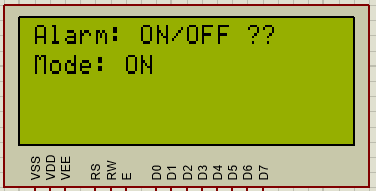
Hình 3‑1 Hiển thị ngày giờ mô phỏng trên Proteus 8.9 SP0

Mặc định của hiển thị, hoặc khi không chuyển sang các chế độ khác. Theo định dạng ngày/tháng /năm/thứ, giờ/phút/giây. Định dạng 24 giờ

### Đặt báo thức

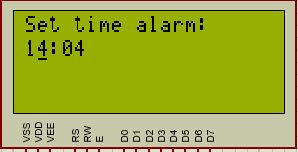
**Bước 1**: nhấn Dieu chinh (Adjust) để chuyển sang chế độ đặt báo thức

**Bước 2:** bật/ tắt báo thức ON/OFF sử dụng Tien/Lui để chọn ON/OFF nhấn Dieu chinh (Adjust) để xác nhận



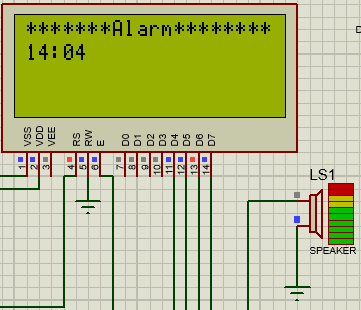
Hình 3‑2 Hiển thị chế độ báo thức mô phỏng trên Proteus 8.9 SP0

**Bước 3**: nếu chế độ ON, đặt giờ báo thức, con trỏ bắt đầu từ điều chỉnh phút, sử dụng Tien/Lui để tăng hoặc giảm số phút, nhấn Dieu chinh (Adjust) để xác nhận và chuyển sang điều chỉnh tương tự cho giờ



Hình 3‑3 Đặt giờ báo thức mô phỏng trên Proteus 8.9 SP0

Khi đúng giờ Báo thức sẽ phát tiếng nhạc trong cả phút và hiển thị giờ đã đặt báo thức lên màn hình



Hình 3‑4 Báo thức khi đúng giờ mô phỏng trên Proteus 8.9 SP0

# PHỤ LỤC

**Các thành phần cơ bản trong chương trình Arduino**

|  |  |  |
| --- | --- | --- |
| **Cấu trúc** | **Giá trị** | **Hàm và thủ tục** |
| * setup() * loop()   **Cấu trúc điều khiển**   * if * if...else * switch / case * for * while * break * continue * return * goto   **Cú pháp mở rộng**   * ; (dấu chấm phẩy) * {} (dấu ngoặc nhọn) * // (single line comment) * /\* \*/ (multi-line comment) * #define * #include   **Toán tử số học**   * = (phép gán) * + (phép cộng) * - (phép trừ) * \* (phép nhân) * / (phép chia) * % (phép chia lấy dư)   **Toán tử so sánh**   * == (so sánh bằng) * != (khác bằng) * > (lớn hơn) * < (bé hơn) * >= (lớn hơn hoặc bằng) * <= (bé hơn hoặc bằng)   **Toán tử logic**   * && (và) * || (hoặc) * ! (phủ định) * ^ (loại trừ)   **Phép toán hợp nhất**   * ++ (cộng thêm 1 đơn vị) * --(trừ đi 1 đơn vị) * += (phép rút gọn của phép cộng) * -= (phép rút gọn của phép trừ) * \*= (phép rút gọn của phép nhân) * /= (phép rút gọn của phép chia) | **Hằng số**   * HIGH | LOW * INPUT | INPUT\_INPUT\_PULLUP | OUTPUT * LED\_BUILTIN * true | false * hằng số nguyên (interger constans) * hằng số thực (floating point constans)   **Kiểu dữ liệu**   * void * boolean * char * unsigned char * byte * int * unsigned int * word * long * unsigned long * short * float * double * array * string (chuỗi kí tự biểu diễn bằng array) * String (object)   **Chuyển đổi kiểu dữ liệu**   * char() * byte() * int() * word() * long() * float()   **Phạm vi của biến và phân loại**   * Phạm vi hiệu lực của biến * static – biến tĩnh * const – biến hằng * volatile   **Hàm hỗ trợ**   * sizeof() | **Nhập xuất Digital (Digital I/O)**   * pinMode() * digitalWrite() * digitalRead()   **Nhập xuất Analog (Analog I/O)**   * analogReference() * analogRead() * analogWrite() * PWM –PPM   **Hàm thời gian**   * millis() * micros() * delay() * delayMicroseconds()   **Hàm toán học**   * min() * max() * abs() * map() * pow() * sqrt() * sq() * isnan() * constrain() * exp(x) * frexp(x, int \*exp) * ldexp(x,int exp) * log(x) * log10(x) * modf(x, \*i) * ceil(x) * floor(x) * atoi(a[])   **Hàm lượng giác**   * cos() * sin() * tan() * asin(x) * acos(x) * atan(x) * atan2(x,y) * cosh(x) * sinh(x) * tanh(x)   **Sinh số ngẫu nhiên**   * randomSeed() * random()   **Nhập xuất nâng cao (Advanced I/O)**   * tone() * noTone() * shiftOut() * shiftIn() * pulseIn()   **Xử lý chuỗi**   * isAscii() * isWhitespace() * isAlpha() * isAlphaNumeric() * isControl() * isDigit() * isGraph() * isLowerCase() * isPrintable() * isPunct() * isSpace() * isUpperCase() * isHexadecimalDigit() * tolower() * toupper()   **Bits và Bytes**   * lowByte() * highByte() * bitRead() * bitWrite() * bitSet() * bitClear() * bit()   **Ngắt (interrupt)**   * attachInterrupt() * detachInterrupt() * interrupts() * noInterrupts()   **Giao tiếp**   * Serial |