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](#_Toc61874575)

[1 Giới thiệu lập trình thiết bị nhúng 1](#_Toc61874576)

[2 Giới thiệu LCD20x4 2](#_Toc61874577)

[2.1 Thông số kỹ thuật: 2](#_Toc61874578)

[2.2 Chức năng các chân của LCD: 3](#_Toc61874579)

[3 Giới thiệu mô đun thời gian thực IC DS1307 4](#_Toc61874580)

[3.1 Ý nghĩa các chân trên IC DS1307: 5](#_Toc61874581)

[4 Giới thiệu Arduino 6](#_Toc61874582)

[4.1 Các board Arduino chính thức 7](#_Toc61874583)

[4.2 Aruino Uno R3 9](#_Toc61874584)

[4.2.1 Thông số kỹ thuật 10](#_Toc61874585)

[4.2.2 Ý nghĩa các chân Arduino UNO R3 10](#_Toc61874586)

[5 Arduino IDE 12](#_Toc61874587)

[5.1 Giao diện Arduino IDE (ver 1.8) 13](#_Toc61874588)

[5.2 Các thành phần trong chương trình Ardruino 16](#_Toc61874589)

[6 Giới thiệu phần mềm mô phỏng Proteus 16](#_Toc61874590)

[6.1 Giao diện làm việc Proteus (version 8) 17](#_Toc61874591)

[6.2 Nạp file .HEX vào Arduino mô phỏng 18](#_Toc61874592)

[CHƯƠNG 2: PHƯƠNG PHÁP NGHIÊN CỨU 19](#_Toc61874593)

[1. Sơ đồ nguyên lý mô phỏng bằng Proteus 19](#_Toc61874594)

[1.1 Giao tiếp giữa Arduino và IC DS1307 20](#_Toc61874595)

[1.2 Giao tiếp giữa Arduino với LCD20x4 21](#_Toc61874596)

[1.3 Giao tiếp giữa Arduino-Loa speaker 22](#_Toc61874597)

[1.4 Thứ tự nối các chân từ Arduino – 3 nút nhấn 23](#_Toc61874598)

[2 Thuật toán 23](#_Toc61874599)

[2.1 Khởi tạo RTC và LCD 25](#_Toc61874600)

[2.2 Hiển thị thời gian lên LCD 26](#_Toc61874601)

[2.3 Các hàm phục vụ Báo thức 28](#_Toc61874602)

[2.3.1 Âm nhạc 28](#_Toc61874603)

[2.3.2 Hiển thị giờ phút cho báo thức 29](#_Toc61874604)

[2.3.3 Kiểm tra giờ báo thức 29](#_Toc61874605)

[2.3.4 Đặt giờ báo thức 30](#_Toc61874606)

[2.4 Chương trình chính 34](#_Toc61874607)

[CHƯƠNG 3: CHƯƠNG 3: KẾT QUẢ THỰC HIỆN (đang làm tới đây) 35](#_Toc61874608)

[1. Kết quả mô phỏng trên Proteus 35](#_Toc61874609)

[1.1 Hiển thị ngày giờ 35](#_Toc61874610)

[1.2 Đặt báo thức 35](#_Toc61874611)

[CHƯƠNG 4: Phục lục 37](#_Toc61874612)

[Hình 1‑1 Đồng hồ điện tử 1](#_Toc61879407)

[Hình 1‑2 LCD 20x4 2](#_Toc61879408)

[Hình 1‑3 Các chân của LCD 20x4 3](#_Toc61879409)

[Hình 1‑4 Đồng hồ thời gian thực IC DS1307 4](#_Toc61879410)

[Hình 1‑5 Sơ đồ chân IC DS1307 5](#_Toc61879411)

[Hình 1‑6 Một mạch Arduino 6](#_Toc61879412)

[Hình 1‑7 Các phiên bản Arduino 8](#_Toc61879413)

[Hình 1‑8 Các phiên bản Ảduino 8](#_Toc61879414)

[Hình 1‑9 Các phiên bản Arduino 9](#_Toc61879415)

[Hình 1‑10 Mạch Arduino UNO R3 9](#_Toc61879416)

[Hình 1‑11: Sơ đồ các chân Arduino UNO R3 10](#_Toc61879417)

[Hình 1‑12: Các chân Digital Arduino UNO R3 11](#_Toc61879418)

[Hình 1‑13: Các chân Analog Arduino UNO R3 12](#_Toc61879419)

[Hình 1‑14 Các chân Power Arduino R3 12](#_Toc61879420)

[Hình 1‑15: Một giao diện Arduino IDE 12](#_Toc61879421)

[Hình 1‑16: Giao diện làm việc Arduino IDE version 1.8.5 13](#_Toc61879422)

[Hình 1‑17: Menu File Arduino IDE 14](#_Toc61879423)

[Hình 1‑18: Tool Menu Arduino IDE 15](#_Toc61879424)

[Hình 1‑19: Chọn cổng giao tiếp với Arduino 16](#_Toc61879425)

[Hình 1‑20: Xuất file .HEX 16](#_Toc61879426)

[Hình 1‑21: Địa chỉ file.HEX 16](#_Toc61879427)

[Hình 1‑22: Một ví dụ giao diện Proteus 8.9 17](#_Toc61879428)

[Hình 1‑23: Giao diện làm việc Proteus version 8.9 17](#_Toc61879429)

[Hình 1‑24: Chọn các thiết bị trong thư viện Proteus 18](#_Toc61879430)

[Hình 1‑25: Nạp file .HEX vào mạch Arduino mô phỏng 18](#_Toc61879431)

[Hình 1‑26: Thư mục chứa file .HEX 19](#_Toc61879432)

[Hình 2‑1: Sơ đồ nguyên lý đồng hồ trên Proteus version 8.9 19](#_Toc61879433)

[Hình 2‑2: Giao tiếp giữa Arduino và IC DS1307 20](#_Toc61879434)

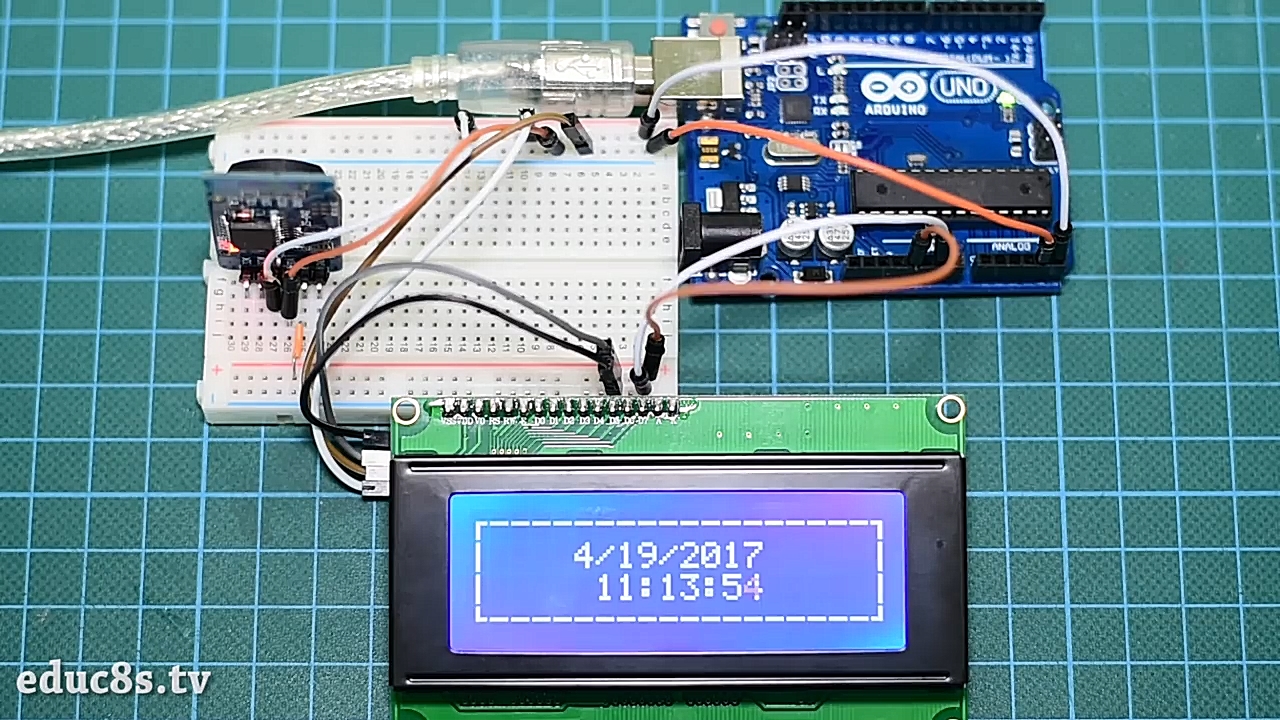
[Hình 2‑3: Giao tiếp giữa Arduino với LCD20x4 21](#_Toc61879435)

[Hình 2‑4: Giao tiếp giữa Arduino với Loa 22](#_Toc61879436)

[Hình 2‑5: Giao tiếp giữa Arduino với các nút nhấn 23](#_Toc61879437)

# TỔNG QUAN

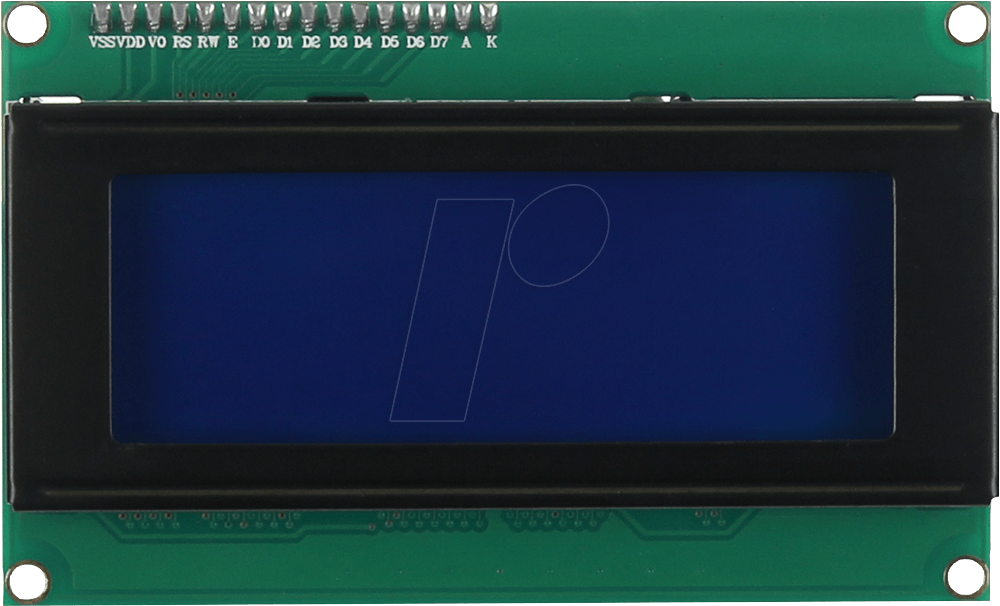
nxfjshdjkgh.



Hình ‑ Đồng hồ điện tử

(Nguồn : https://www.electronics-lab.com/project/real-time-clock-20x4-i2c-lcd-display/)

## LCD20x4



Hình ‑ 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.

* Điện áp hoạt động: 5V
* Hiển thị tối đa 20 tự trên 4 dòng
* Chữ đen nền xanh lá

****

Hình ‑ 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.

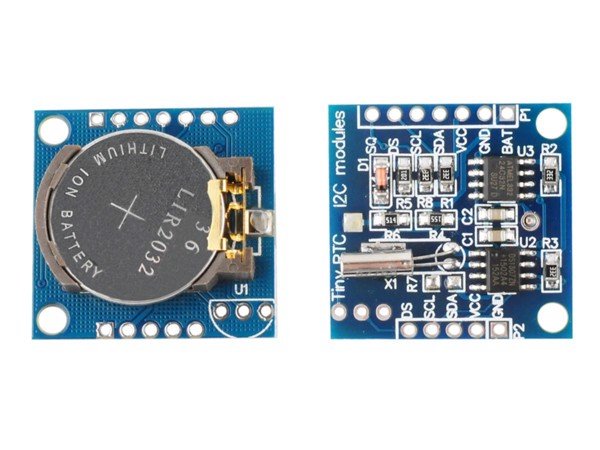
* Chân (5): R/W là chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD ở chế độ đọc.
* Chân (6): E Là chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của 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

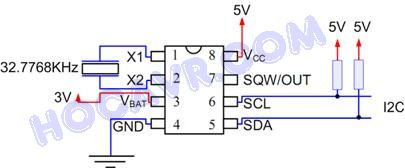
## Mô đun thời gian thực IC DS1307



Hình ‑ 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



Hình ‑ 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

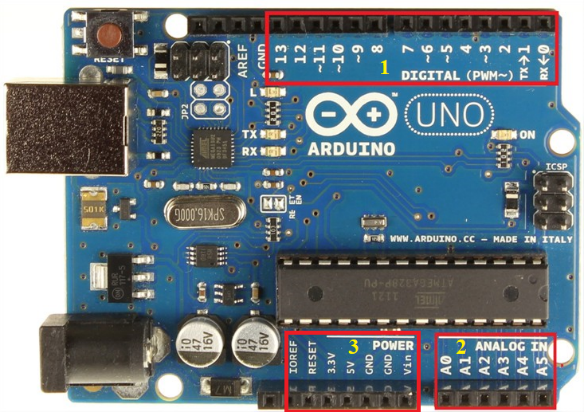
Arduino một nền tảng mã nguồn mở phần cứng và phần mềm. Phần cứng Arduino (các board mạch vi xử lý) được sinh ra tại thị trấn Ivrea ở Ý, nhằm xây dựng các ứng dụng tương tác với nhau hoặc với môi trường được thuận lợi hơn. Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý 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. Sáu phiên bản phần cứng của Arduino cũng đã được sản xuất thương mại tính đến thời điểm hiện tại kể ra....

**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 ‑: Sơ đồ các chân Arduino UNO R3

(Nguồn <http://arduino.vn/>)

Arduino UNO có 14 chân digital dùng để đọc hoặc xuất tín hiệu. Chúng chỉ có 2 mức điện áp là 0V và 5V với dòng vào/ra tối đa trên mỗi chân là 40mA. Ở mỗi chân đều có các điện trở pull-up từ được cài đặt ngay trong vi điều khiển ATmega328 (mặc định thì các điện trở này không được kết nối)

Một số chân digital có các chức năng đặc biệt như sau:

* **2 chân Serial: 0 (RX) và 1 (TX):** dùng để gửi (transmit – TX) và nhận (receive – RX) dữ liệu TTL Serial. Arduino Uno có thể giao tiếp với thiết bị khác thông qua 2 chân này. Kết nối bluetooth thường thấy nói nôm na chính là kết nối Serial không dây. Nếu không cần giao tiếp Serial, bạn không nên sử dụng 2 chân này nếu không cần thiết
* **Chân PWM (~): 3, 5, 6, 9, 10, và 11:** cho phép xuất ra xung PWM với độ phân giải 8bit (giá trị từ 0 → 28-1 tương ứng với 0V → 5V) bằng hàm analogWrite(). Nói một cách đơn giản, bạn có thể điều chỉnh được điện áp ra ở chân này từ mức 0V đến 5V thay vì chỉ cố định ở mức 0V và 5V như những chân khác.
* **Chân giao tiếp SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).**  Ngoài các chức năng thông thường, 4 chân này còn dùng để truyền phát dữ liệu bằng giao thức SPI với các thiết bị khác.
* **LED 13:** trên Arduino UNO có 1 đèn led màu cam (kí hiệu chữ L). Khi bấm nút Reset, bạn sẽ thấy đèn này nhấp nháy để báo hiệu. Nó được nối với chân số 13. Khi chân này được người dùng sử dụng, LED sẽ sáng.
* **Chân 2 và 3 External Interrupts (Ngắt ngoài)** : được sử dụng để cung cấp các ngắt ngoài.



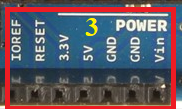
Hình ‑: Các chân Digital Arduino UNO R3

Arduino UNO có 6 chân analog (A0 → A5) cung cấp độ phân giải tín hiệu 10bit (0 → 210-1) để đọc giá trị điện áp trong khoảng 0V → 5V. Với chân AREF trên board, bạn có thể để đưa vào điện áp tham chiếu khi sử dụng các chân analog. Tức là nếu bạn cấp điện áp 2.5V vào chân này thì bạn có thể dùng các chân analog để đo điện áp trong khoảng từ 0V  → 2.5V với độ phân giải vẫn là 10bit. Đặc biệt, Arduino UNO có 2 chân **A4 (SDA)** và **A5 (SCL)** hỗ trợ giao tiếp **I2C/TWI** với các thiết bị khác.



Hình ‑: Các chân Analog Arduino UNO R3

* Vin :  Đây là điện áp đầu vào được cung cấp cho board mạch Arduino. Khác với 5V được cung cấp qua cổng USB. Pin này được sử dụng để cung cấp điện áp toàn mạch thông qua jack nguồn, thông thường khoảng 7-12VDC
* 5V, 3.3V : Chân 5V, 3,3V được sử dụng để cung cấp điện áp đầu ra. Arduino được cấp nguồn bằng ba cách đó là USB, chân Vin của bo mạch hoặc giắc nguồn DC.
* GND : Chân mass chung cho toàn mạch Arduino
* IOREF : Chân này rất hữu ích để cung cấp tham chiếu điện áp cho Arduino
* Reset : Chân reset để thiết lập lại về ban đầu



Hình ‑ Các chân Power Arduino R3

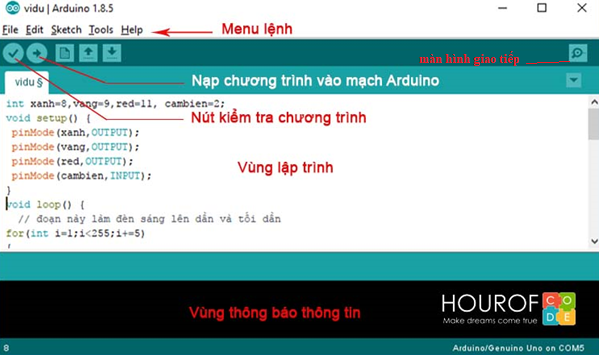
## Arduino IDE

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.[[5]](https://vi.wikipedia.org/wiki/Arduino#cite_note-5)

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

### Giao diện Arduino IDE (ver 1.8)



Hình ‑: Giao diện làm việc Arduino IDE version 1.8.5

(Nguồn: https://hourofcode.vn/lap-trinh-arduino-gioi-thieu-giao-dien-arduino-ide/)

**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)

### Các thành phần trong chương trình Ardruino

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**

## Proteus

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 ‑: Một ví dụ giao diện Proteus 8.9

### Giao diện làm việc Proteus (version 8)

Hình ‑: Giao diện làm việc Proteus version 8.9

Để chọn các chọn các thiết bị, trên thanh tác vụ, chọn Component mode, Pick Device, chọn thư viện và thiết bị.

Hình ‑: Chọn các thiết bị trong thư viện Proteus

### Nạp file .HEX vào Arduino mô phỏng

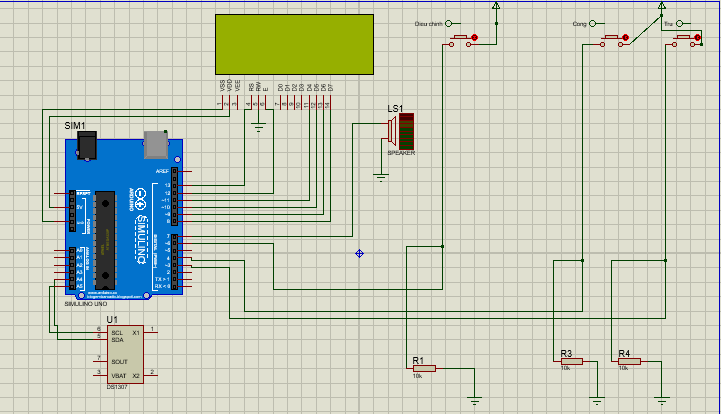
Hình ‑: Nạp file .HEX vào mạch Arduino mô phỏng

Nhấp chuột vào Chip mô phỏng trên mạch Arduino, mở thư mục và tìm đến file .HEX mong muốn

Hình ‑: Thư mục chứa file .HEX

# PHƯƠNG PHÁP NGHIÊN CỨU

## Sơ đồ nguyên lý mô phỏng bằng Proteus



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

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

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

### Giao tiếp giữa Arduino và IC DS1307

Hình ‑: 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

Hình ‑: 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 đất

* 4 chân truyền dữ liệu:

+ 11 – D4

+ 10 – D5

+ 9 – D6

+ 8 – D7

### Giao tiếp giữa Arduino-Loa speaker

Hình ‑: Giao tiếp giữa Arduino với Loa

Arduino sẽ điều khiển Loa phát âm thanh khi báo thức

+ 7 – 1 chân loa

+ Chân còn lại của Loa nối đất

### Thứ tự nối các chân từ Arduino – 3 nút nhấn

Hình ‑: Giao tiếp giữa Arduino với các 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

Phát nhạc

Đặt báo thức

SAI

ĐÚNG

ĐÚNG

SAI

Kiểm tra báo thức

Khởi tạo RTC và LCD

Đọc dữ liệu từ RTC

Hiển thị thời gian lên LCD

Nút nhấn Adjust = HIGH

### 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 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, và đối tượng RTC

// khai báo lcd

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

// doi tuong dong ho

RTC\_DS1307 rtc;

Khởi tạo RTC và LCD trong hàm setup()

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

void showtime(DateTime now){

// hien thi ngay

lcd.setCursor(0, 0);

if(now.day()<=9)

{

lcd.print("0");

lcd.print(now.day());

}

else {

lcd.print(now.day());

}

lcd.print('/');

if(now.month()<=9)

{

lcd.print("0");

lcd.print(now.month());

}

else {

lcd.print(now.month());

}

lcd.print('/');

if(now.year()<=9)

{

lcd.print("0");

lcd.print(now.year());

}

else {

lcd.print(now.year());

}

lcd.print("(");

lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);

lcd.print(") ");

// hien thi gio

lcd.setCursor(0, 1);

if(now.hour()<=9)

{

lcd.print("0");

lcd.print(now.hour());

}

else {

lcd.print(now.hour());

}

lcd.print(':');

if(now.minute()<=9)

{

lcd.print("0");

lcd.print(now.minute());

}

else {

lcd.print(now.minute());

}

lcd.print(':');

if(now.second()<=9)

{

lcd.print("0");

lcd.print(now.second());

}

else {

lcd.print(now.second());

}

}

### Các hàm phục vụ Báo thức

#### Âm nhạc

void music(){

for (int i = 0; i < 203; i++) { //203 is the total number of music notes in the song

int wait = duration[i] \* songspeed;

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

delay(wait);

}

}

#### Hiển thị giờ phút cho báo thức

void timehour(int gio,int phut){

lcd.setCursor(0, 1);

if(gio<=9)

{

lcd.print("0");

lcd.print(gio);

}

else {

lcd.print(gio);

}

lcd.print(':');

if(phut<=9)

{

lcd.print("0");

lcd.print(phut);

}

else {

lcd.print(phut);

}

}

#### Kiểm tra giờ báo thức

void checkalarm(DateTime now){

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

{

lcd.clear();

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

lcd.setCursor(0, 1);

if(gio<=9)

{

lcd.print("0");

lcd.print(gio);

}

else {

lcd.print(gio);

}

lcd.print(':');

if(phut<=9)

{

lcd.print("0");

lcd.print(phut);

}

else {

lcd.print(phut);

}

music();

lcd.clear();

}

}

#### Đặt giờ báo thức

void alarm(DateTime now){

lcd.clear();

//triet tieu dong dien ton dong

delay(400);

// bat hay tat bao thuc ??

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");

}

}

}

if(modealarm==1){

lcd.clear();

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);}

}

//dat gio

lcd.setCursor(1,1);

delay(350);

while(digitalRead(Adjust)==LOW){

if(digitalRead(Tien)==HIGH)

{

delay(350);

gio=gio+1;

if(gio>=24) gio=0;

timehour(gio,phut);

lcd.setCursor(1,1);

}

else if(digitalRead(Lui)==HIGH)

{

delay(350);

gio=gio-1;

if(gio<=0) gio=24;

timehour(gio,phut);

lcd.setCursor(1,1);}

}

}

lcd.noCursor();

//triet tieu dong dien ton dong

delay(400);

}

### Chương trình chính

void loop() {

DateTime now = rtc.now();

showtime(now);

if(digitalRead(Adjust)==HIGH)

alarm(now);

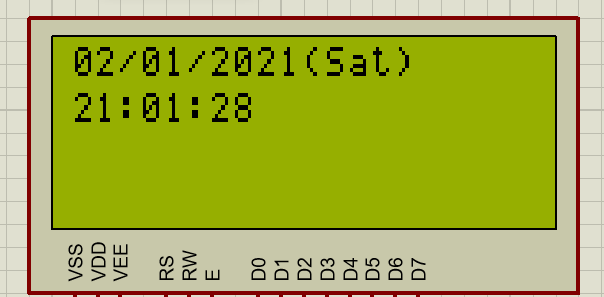
checkalarm(now);

}

# KẾT QUẢ THỰC HIỆN (đang làm tới đây)

## Kết quả mô phỏng trên Proteus

### Hiển thị ngày giờ

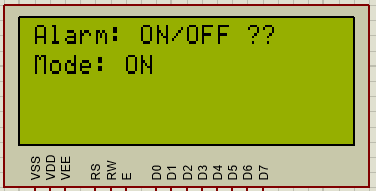


Mặc định của hiển thị, hoặc khi không chuyển sang các chế độ khác. Theo định dạng ngày/tháng /năm/thứ, giờ/phút/giây. Định dạng 24 giờ

### 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



**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



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

# PHỤ LỤC

**Các thành phần 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 |