

**TRƯỜNG ĐẠI HỌC NHA TRANG**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**THIẾT KẾ ĐỒNG HỒ THỜI GIAN THỰC SỬ DỤNG MODULE DS1307**

**GVHD : Ths. ĐOÀN VŨ THỊNH**

**SVTH : Nguyễn Minh Hoàng**

**MSSV : 59130802**

**Lớp : 59CNTT-2**

Khánh Hòa, tháng 01 năm 2020

## MỤC LỤC

<b>DANH MỤC HÌNH .....</b>	<b>ii</b>
<b>DANH MỤC BẢNG.....</b>	<b>iii</b>
<b>TÓM TẮT .....</b>	<b>1</b>
<b>1. GIỚI THIỆU .....</b>	<b>2</b>
1.1. LED 7 đoạn.....	2
1.2. 74LS47 .....	3
1.3. 74LS154 .....	5
1.4. IC 7404 .....	7
1.5. Module thời gian thực DS1307 .....	8
1.6. Board mạch Arduino UNO R3 .....	11
1.7. IC lập trình được Atmega16u2 .....	13
1.8. Phần mềm Proteus version 7.10SP2 .....	14
1.9. Phần mềm Sprint Layout Viewer .....	15
1.10. Phần mềm Arduino IDE .....	16
<b>2. PHƯƠNG PHÁP NGHIÊN CỨU .....</b>	<b>17</b>
2.1. Thiết kế, thi công mạch in .....	17
2.2. Mô phỏng nguyên lý hoạt động bằng phần mềm Proteus.....	17
2.3. Lập trình .....	19
2.3.1. Giao tiếp RTC với Arduino .....	20
2.3.2. Hiển thị thời gian lên LED 7 đoạn .....	20
<b>3. KẾT QUẢ.....</b>	<b>21</b>
<b>THẢO LUẬN.....</b>	<b>23</b>
<b>PHỤ LỤC .....</b>	<b>24</b>

## DANH MỤC HÌNH

Hình 1.1. Đồng hồ vạn niên .....	2
Hình 1.2. LED 7 đoạn với Cathode chung (bên trái) và Anode chung (bên phải) .....	3
Hình 1.3. IC 74HC47 .....	5
Hình 1.4. IC 74HC154 .....	6
Hình 1.5. IC 7404 .....	7
Hình 1.6. Module thời gian thực DS1307 .....	8
Hình 1.7. Module Arduino UNO R3 .....	11
Hình 1.8. Sơ đồ chân IC Atmega16u2 .....	13
Hình 1.9. Sơ đồ nguyên lý của board mạch Arduino UNO R3 .....	14
Hình 1.10. Phần mềm Proteus ISIS 7 Profesional .....	15
Hình 1.11. Phần mềm sprint Layout 6.0 .....	16
Hình 1.12. Giao diện phần mềm Arduino IDE .....	16
Hình 2.1. Các bước thi công mạch in .....	17
Hình 2.2. Sơ đồ nguyên lý và mô phỏng cho đồng hồ thời gian thực .....	18
Hình 2.3. Cửa sổ nạp chương trình cho board Arduino .....	19
Hình 2.4. Sơ đồ khối chương trình đồng hồ thời gian thực .....	19
Hình 2.5. Trình tự các bước khởi tạo RTC .....	20
Hình 2.6. Các bước hiển thị thời gian lên LED 7 đoạn với giá trị của đơn vị giây .....	21
Hình 3.1. Linh kiện IC và LED 7 đoạn được cố định trên mạch in .....	21
Hình 3.2. LED 7 đoạn, thành phần liên kết của sản phẩm .....	22
Hình 3.3. Sản phẩm hoàn thiện với thời gian được hiển thị trên LED 7 đoạn .....	22

## DANH MỤC BẢNG

Bảng 1.1. Bảng trạng thái điều khiển các vạch của LED 7 đoạn loại Cathode chung....	3
Bảng 1.2. Bảng trạng thái điều khiển IC 7447.....	4
Bảng 1.3. Bảng trạng thái điều khiển IC 74LS154 .....	6
Bảng 1.4. Bảng trạng thái của IC 74LS04 .....	7
Bảng 1.5. Bảng địa chỉ thanh ghi giá trị thời gian của DS1307.....	9
Bảng 1.6. Bảng tầng số trạng thái chân RS1 và RS0 .....	11
Bảng 1.7. Thông số của Arduino UNO R3 .....	12

## TÓM TẮT

Lịch vạn niên hay đồng hồ số 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. Yêu cầu của bài toán là thiết kế, thi công, lắp đặt sản phẩm đồng hồ thời gian thực hiện thị các thông số bao gồm: Ngày, tháng, năm giờ, phút, giây. Thời gian tự động cập nhật ngay cả khi không có nguồn điện cung cấp. Để thực hiện các công việc này cần sử dụng đến các thành phần, linh kiện điện tử như: module lập trình Arduino UNO R3, module thời gian thực DS1307, LED 7 đoạn loại Cathode chung, IC giải mã BCD sang 7 đoạn 74LS47, IC hợp lệnh để giải mã địa chỉ cho các IC 74LS47 phục vụ giải thuật quét LED IC74LS154, các thành phần khác bổ trợ khác như LED đơn, điện trở, nguồn cung cấp.

Quá trình thiết kế mạch in bắt đầu với phần mềm Proteus ver7.10SP2 để xây dựng sơ đồ nguyên lý và tiến hành mô phỏng trạng thái hoạt động của hệ thống. Sau đó, phần mềm chuyên dụng để thiết kế mạch in Sprint Layout ver6.0. Sau đó, sẽ trải qua quá trình ủ, rửa mạch in và tiến hành lắp đặt linh kiện lên bản mạch. Thuật toán được lập trình trên phần mềm Arduino IDE ver1.8.10.

Kết quả thực hiện cho thấy sản phẩm có thể hiển thị thời gian với các thông số như yêu cầu đặt ra, thời gian được cập nhật sau mỗi giây. Toàn bộ sản phẩm được đóng gói, mã nguồn được trình bày trong phụ lục và được upload theo địa chỉ: [https://github.com/thinhdoanvu/ThucTapTHCS\\_2019/tree/master/NguyenMinhHoang](https://github.com/thinhdoanvu/ThucTapTHCS_2019/tree/master/NguyenMinhHoang)

## 1. GIỚI THIỆU

Lịch vạn niên hay đồng hồ số 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 lịch vạn niên được bán trên thị trường. Thành phần cơ bản bao gồm bộ hiển thị LED 7 đoạn cho biết thời gian như: ngày, tháng, năm, giờ, phút, giây và còn hiển thị thứ trong tuần. Để có thể tạo ra một đồng hồ số như trên 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](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 các thành phần LED.

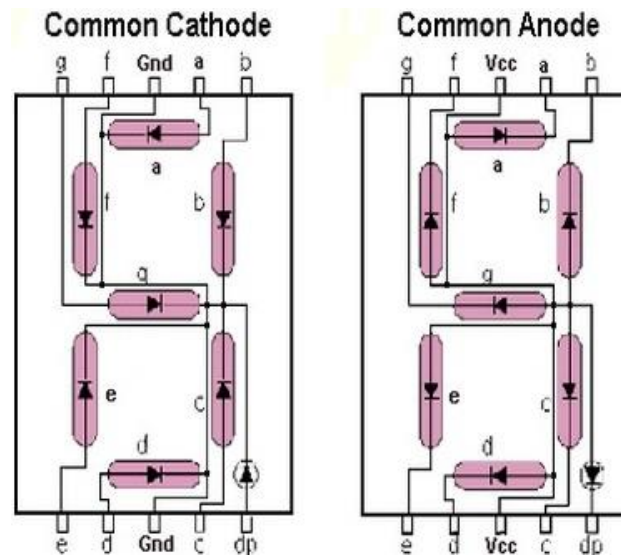


Hình 1.1. Đồng hồ vạn niên (Nguồn: bangdientu.com.vn)

### 1.1. LED 7 đoạn

LED 7 đoạn là thiết bị điện tử để hiển thị số thập phân (0-9). LED 7 đoạn được sử dụng rộng rãi các bộ chỉ thị như các bộ đếm, thiết bị đo, đồng hồ. LED 7 đoạn có 2 loại: Anode chung và Cathode chung. Trong đề tài này, Cathode chung được sử dụng để hiển thị các giá trị của thời gian.

**Cấu tạo:** LED 7 đoạn bao gồm 8 LED được kết nối song song để có thể hiển thị các số 0, 1, 2, 3, 4, 5, 7, 8, 9, A, b, C, d, E, F. Mỗi đoạn của LED được đánh dấu bởi ký tự từ A tới G. Đoạn thứ tám gọi là “chấm thập phân” (*Decimal Point*) ký hiệu là DP được sử dụng khi hiển thị số không phải là số nguyên. LED 7 đoạn có 10 chân, trong đó 8 chân dùng để hiển thị và 2 chân lại được nối nguồn điện.



Hình 1.2. LED 7 đoạn với Cathode chung (bên trái) và Anode chung (bên phải)

(Nguồn: <http://www.datasheetcafe.com/fj5101ah-datasheet-pdf/>)

Đối với loại Anode chung để hiển thị từng đoạn cụ thể cần cung cấp nguồn 5V (VCC) cho chân số 3 và 8, các chân còn lại được nối với đất (GND). Điều ngược lại được áp dụng cho Cathode chung.

**Bảng 1.1. Bảng trạng thái điều khiển các vạch của LED 7 đoạn loại Cathode chung**

a	b	c	d	e	f	g	dp	Hiển thị
H	H	H	H	H	H	L	H	0
L	H	H	L	L	L	L	H	1
H	H	L	H	H	L	H	H	2
H	H	H	H	L	L	H	H	3
L	H	H	L	L	H	H	H	4
H	L	H	H	L	H	H	H	5
H	L	H	H	H	H	H	H	6
H	H	H	L	L	L	L	L	7
H	H	H	H	H	H	H	H	8
H	H	H	H	L	H	H	H	9

(Trong đó: H = HIGH, L = LOW)

## 1.2. 74LS47

Như trình bày ở phần LED 7 đoạn, để hiển thị các giá trị 0-9 hay A-F cần điều khiển 8 chân đầu vào của Cathode hay Anode chung, việc này đòi hỏi phải cung cấp 8 đường điều khiển từ bộ vi điều khiển. Điều này là hết sức tốn kém chân điều khiển, dẫn

đến không đủ chân điều khiển cho các LED khác hay công việc khác. 74LS47 được sử dụng để giảm bớt số lượng chân điều khiển cho LED 7 đoạn (4 chân thay vì 8 chân).

**Cấu tạo:** IC74LS47 gồm 16 chân (hình 1.2), trong đó 4 chân A, B, C, D (chân thứ 7, 1, 2, 6) là chân tín hiệu vào, được dùng để giải mã cho 16 giá trị hiển thị, 8 chân a, b, c, d, e, f, g (chân 13, 12, 11, 10, 10, 9, 15, 14) là 8 đầu ra được nối với LED 7 đoạn dùng để hiển thị các giá trị trên LED, chân VCC (chân số 16) được nối với nguồn +5V, chân GND (chân 8) được nối với GND, chân Lamp Test – LT (chân 3) dùng để kiểm tra các tín hiệu của đèn LED, chân BI/RBO (chân 4) phối hợp với chân RBI (chân 5) để điều khiển LED. Chi tiết xem ở bảng trạng thái (Bảng 1.2).

**Bảng 1.2. Bảng trạng thái điều khiển IC 7447**

Giá trị	Input (vào)							Output (ra)						
	LT	RBI	A3	A2	A1	A0	BI/ RBO	a	b	c	d	e	f	g
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L
5	H	X	L	H	L	H	H	L	H	L	L	H	H	L
6	H	X	L	H	H	L	H	L	H	L	L	H	H	L
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L
9	H	X	H	L	L	H	H	L	L	L	L	H	L	L
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L
13	H	X	H	H	L	H	H	L	H	H	H	H	L	L
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H

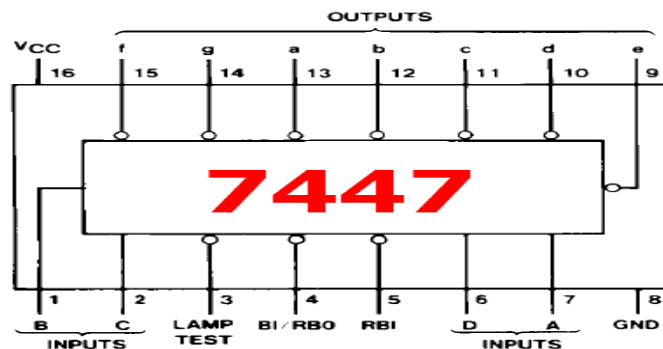


RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L

(Trong đó: H = HIGH, L = LOW, X = HIGH hoặc LOW)

**Nhận xét:** Khi LT=H đồng nghĩa với việc khi 74LS47 được điều khiển thì các chân đầu ra cũng sẽ hiển thị theo giá trong bảng trạng thái, khi LT=L thì tất cả các chân output (ra) đều ở mức cao dẫn đến LED 7 đoạn sẽ không được hiển thị.

Khi một trong 2 chân RBI hoặc BI/RBO=L khi đó tất cả các chân ra của 74LS47 sẽ về trạng thái HIGH khi đó LED sẽ không được hiển thị.



Hình 1.3. IC 74HC47

(Nguồn: <http://www.datasheetcafe.com/7447-datasheet-fairchild/>)

### 1.3. 74LS154

Để điều khiển giá trị của LED 7 đoạn thông qua 7447, cần kích tín hiệu BI/RBO ở các mức tích cực cao và thấp và do đó sẽ mất nhiều đường tín hiệu ở chân Arduino (12 LED 7 đoạn tương ứng 12 đường tín hiệu). Để tiết kiệm số chân Arduino, IC giải mã kênh 74LS154 (demux) được sử dụng. 74LS154 là IC gồm 4 đầu vào được mã hóa nhị phân thành 1 trong 16 đầu ra loại trừ lẫn nhau khi cả hai đầu vào điều khiển G1 và G2 ở mức tích cực thấp (Bảng 1.3).

**Cấu tạo:** Gồm 4 chân đầu vào IN-A, IN-B, IN-C và IN-D ứng với các vị trí chân 23, 22, 21 và 20, 16 đầu ra gồm từ OUT-00 đến OUT-15, VCC (nguồn 5V) tại vị trí chân số 24, chân GND (0V) tại vị trí 12, 2 chân điều khiển IC là G1, G2 tại vị trí chân 18 và 19 tương ứng.

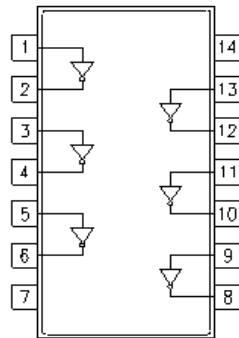
(Nguồn: <http://makeyourownchip.tripod.com/74154.html>)

[illegible]

## 1.4. IC 7404

Khi IC 74LS154 được điều khiển thì chỉ xuất ra 1 chân LOW tất cả các chân còn lại sẽ HIGH nhưng chân RBI và BI/RBO của 74LS47 lại hoạt động khi nó ở mức HIGH, để điều chỉnh cho 1 chân HIGH và tất cả chân còn lại đều LOW và thì cần phải đảo ngược chân LOW thành chân HIGH. Để giúp đảo ngược trạng thái thì ta dùng IC 7404 (IC đảo) hay còn gọi là IC NOT là IC logic khi các cổng vào là các bit 0 và 1 thì các cổng ra sẽ ngược lại ví dụ đầu vào là 0 thì đầu ra sẽ là 1.

**Cấu tạo:** gồm có 6 chân đầu vào, 6 chân đầu ra bị đảo ngược, VCC, GND.



Hình 1.5. IC 7404

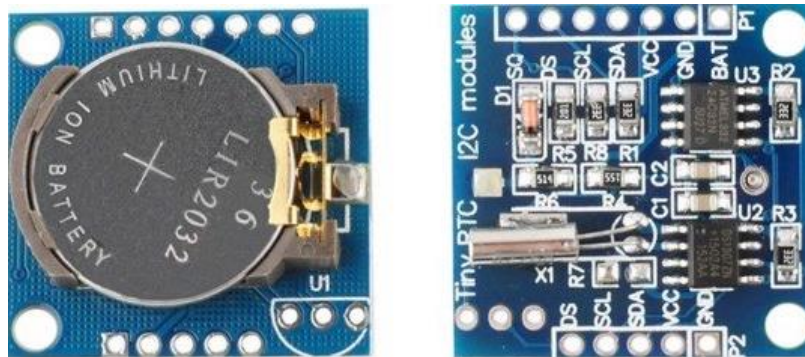
(Nguồn: <https://www.futurlec.com/74/IC7404.shtml>)

**Bảng 1.4. Bảng trạng thái của IC 74LS04**

Đầu vào		Đầu ra	
Chân	1	Chân	2
	H		L
	L		H
Chân	3	Chân	4
	H		L
	L		H
Chân	5	Chân	6
	H		L
	L		H
Chân	9	Chân	8
	H		L
	L		H
Chân	11	Chân	10
	H		L
	L		H
Chân	13	Chân	12
	H		L
	L		H

### 1.5. Module thời gian thực DS1307

Những linh kiện đã đề cập ở trên vẫn chưa thể tạo thành thời gian cho đồng hồ vạn niên được, việc cho các LED sáng theo giờ cần phải cập nhật thời gian cho vi điều khiển. Module thời gian thực DS1307 là chip thời gian thực hay RTC (Read time clock). Đây là một IC tích hợp cho thời gian bởi vì tính chính xác về thời gian: Thứ, ngày, tháng, năm, giờ, phút, giây. DS1307 được sản xuất bởi hãng Dallas, chip này có 7 thanh ghi 8 bit mỗi thanh ghi này chứa: Thứ, ngày, tháng, năm, giờ, phút, giây. Ngoài ra DS1307 còn chứa 1 thanh ghi điều khiển ngõ ra phụ và 56 thanh ghi trống - các thanh ghi này có thể dùng như bộ nhớ RAM. DS1307 sử dụng chuẩn truyền thông I2C nên cấu tạo bên ngoài nó rất đơn giản.



Hình 1.6. Module thời gian thực DS1307

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

#### Cấu tạo:

X1 và X2 là đầu vào dao động cho DS1307 sử dụng thạch anh có tần số dao động là 32.768Khz. Thạch anh là linh kiện điện tử tạo giao động ổn định cho bộ điều khiển, thạch anh được sử dụng trong đa phần thiết bị điện tử vì nó rất ít bị ảnh hưởng bởi nhiệt độ hơn các mạch dao động RC hay RLC.

Vbat là nguồn nuôi cho DS1307, nguồn này có điện áp từ 2V- 3.5V. Nguồn nuôi (Pin CMOS) được dùng để cấp nguồn cho RTC hoạt động trong trường hợp module thời gian thực không được cấp nguồn VCC. Nguồn nuôi này có thể hoạt động đến 10 năm khi module đã tắt nguồn.

VCC là nguồn cung cấp cho giao tiếp I2C, điện áp cung cấp là 5V. Nếu có mất nguồn VCC mà Vbat vẫn có nguồn thì DS1307 vẫn hoạt động bình thường nhưng mà không ghi và đọc được dữ liệu.

GND là nguồn đất (mass) chung cho cả Vcc và Vbat.

SQW/OUT là ngõ ra phụ tạo xung dao động (xung vuông) được sử dụng cho các mục đích tạo xung, bộ đếm cho các thiết bị khác.

SCL và SDA, trong đó SCL (serial clock) và SDA (serial data) là hai bus dữ liệu của DS1307. Thông tin truyền và ghi đều được truyền qua 2 đường truyền này theo chuẩn I2C.

I2C (Inter-Integrated Circuit) là một loại bus nối tiếp được phát triển bởi hãng sản xuất linh kiện điện tử Philips. I2C sử dụng hai đường truyền tín hiệu: một đường xung nhịp đồng hồ (SCL) và một đường dữ liệu (SDA). SCL và SDA luôn được kéo lên nguồn bằng một điện trở treo có giá trị 4,7 K $\Omega$ . Các chế độ hoạt động của I2C bao gồm: (1) Chế độ chuẩn (standard mode) hoạt động ở tốc độ 100 Kbit/s; (2) Chế độ tốc độ thấp (low-speed mode) hoạt động ở tốc độ 10 Kbit/s.

I2C sử dụng 7 bit để định địa chỉ, do đó trên một bus có thể đánh 128 địa chỉ (112 nút (node) và 16 địa chỉ dành cho mục đích riêng). Điểm mạnh của I2C chính là hiệu suất và sự đơn giản của nó: một khối điều khiển trung tâm có thể điều khiển cả một mạng thiết bị mà chỉ cần hai đường điều khiển. Ban đầu, loại bus này chỉ được dùng trong các linh kiện điện tử của Philips. Sau đó, do tính ưu việt và đơn giản của nó, I2C đã được chuẩn hóa và được dùng rộng rãi trong các module truyền thông nối tiếp của vi mạch tích hợp ngày nay. Vì Arduino đã có hỗ trợ sẵn điện trở treo ở đầu vào SDA và SCL nên khi kết nối với DS1307 không cần sử dụng đến trở treo bên ngoài nữa. Để có thể sử dụng giao tiếp I2C, thư viện Wire.h cần được thiết lập trong Arduino IDE.

**Bảng 1.5. Bảng địa chỉ thanh ghi giá trị thời gian của DS1307**

Địa chỉ	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Hàm	Phạm vi
00h	CH	10 Seconds			Seconds				Seconds	00-59
01h	0	10 Minutes			Minutes				Minutes	00-59
02h	0	12	10 Hour	10 Hour	Hour				Hour	1-12 +AM/PM 00-23
		24	PM/AM							
03h	0	0	0	0	DAY				Day	01-07
04h	0	0	10 Date		Date					01-31
05h	0	0	0	10 Month	Month				Month	01-12
06h	10 Year				Year				Year	00-99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	
08h- 3Fh									RAM 56 x 8	

Trong bộ nhớ của DS1307 có tất cả 64 thanh ghi địa chỉ từ 0 đến 63 và được bắt đầu từ 0x00 đến 0x3F nhưng trong đó chỉ có 8 thanh ghi đầu là thanh ghi thời gian thực. Bảng 1.5 là địa chỉ cụ thể cho các thành phần thời gian như sau: Giây, phút, giờ, thứ, ngày, tháng, năm bắt đầu từ địa chỉ 00h và kết thúc là địa chỉ 06h.

Thanh ghi Control dùng để điều khiển ngõ ra của chân SQW/OUT trong trường hợp RTC đóng vai trò là bộ cấp nguồn dao động cho các thiết bị khác.

(1) Thanh ghi giây (0x00): Bit 0 đến bit 3 (Bảng 1.5) dùng để mã hóa bộ đếm BCD hàng đơn vị của giây. Tiếp theo từ bit 4 đến bit 6 dùng để mã hóa BCD hàng chục của giây. Lý do số giây hàng đơn vị cần sử dụng đến 4 bit và số giây hàng chục chỉ cần 3 bit mã hóa là vì hàng chục chỉ hiển thị tối đa là số 5 (101) và số giây hàng đơn vị hiển thị tối đa số 9 (1001). Từ đó bit 7 của thanh ghi này luôn luôn được gán giá trị là 0 (mã lệnh là AND bit với giá trị 0x7F ngay khi khởi động chương trình).

(2) Thanh ghi phút (0x01): 4 bit thấp dùng để mã hóa BCD chữ số hàng đơn vị và 3 bit cao để mã hóa số phút hàng chục (Bảng 1.5). Nhưng thanh ghi này có sự khác biệt với thanh ghi giây là bit 7 mặc định là 0 rồi nên không cần phải thực hiện thao tác như thanh ghi giây đối với bit 7 của thanh ghi này.

(3) Thanh ghi giờ (0x02): Bit 0 đến bit 3 để mã hóa BCD của chữ số hàng đơn vị của giờ và bit 4 đến bit 7 để mã hóa số giờ hàng chục (Bảng 1.5). Tuy nhiên có 2 chế độ để hiển thị số giờ: chế độ 24h và 12h nên được thiết lập bởi bit 6. Nếu bit 6 có giá trị là 0 (chế độ 24h) nên chỉ sử dụng bit 4 và bit 5 để mã hóa BCD chữ số hàng chục của giờ (do giá trị lớn nhất hiển thị số giờ là 23). Nếu bit 6 có giá trị bằng 1 (chế độ 12h) nên chỉ cần dùng duy nhất bit 4 để mã hóa BCD chữ số hàng chục của giờ (do giá trị lớn nhất của số hàng chục của giờ có thể hiển thị là 12). Ngoài ra, ở chế độ 12h, bit 5 được dùng để chỉ buổi sáng/AM (bit 5 có giá trị là 0) hay chiều/PM (bit 5 có giá trị là 1). Trong cả 2 chế độ 12h và 24h bit 7 luôn luôn có giá trị là 0.

(4) Thanh ghi thứ trong tuần (0x03): 3 bit thấp (bit 0 đến bit 2) để mã hóa BCD cho các thứ trong tuần (1 = Chủ nhật và 7 = thứ 7). Do đó, các bit cao từ 3 đến 7 mặc định là 0 (Bảng 1.5).

(5) Thanh ghi ngày (0x04): 4 bit thấp (bit 0 đến bit 3) dùng để mã hóa BCD ra chữ số hàng đơn vị của ngày trong tháng (từ 1 đến 9) và bit 4 và bit 5 mã hóa BCD cho số hàng chục của ngày trong tháng (từ 0 đến 3). Bit 6 và bit 7 mặc định bằng 0 (Bảng 1.5).

(6) Thanh ghi tháng (0x05): 4 bit thấp (từ bit 0 đến bit 3) mã hóa BCD hàng đơn vị của tháng (từ 1 đến 9). Bit thứ 4 để mã hóa BCD số hàng chục của tháng (hiển thị số 0 và 1). Các bit còn lại (bit 5 đến bit 7) mặc định là 0 (Bảng 1.5).

(7) Thanh ghi năm (0x06): Sử dụng 8 bit (bit 0 đến bit 7) để mã hóa BCD của số năm. Do RTC chỉ hiển thị 2 số cuối cùng của năm (từ 00 đến 99). (Bảng 1.5)

(8) Thanh ghi điều khiển (0x07): Đây là thanh ghi điều khiển quá trình ghi đọc của DS1307 với giá trị 0x93.

(8) OUT: Chọn mức logic (1 hoặc 0) xuất ra tại chân SQW/OUT khi chức năng SQW không được kích hoạt (SQW có giá trị là 0).

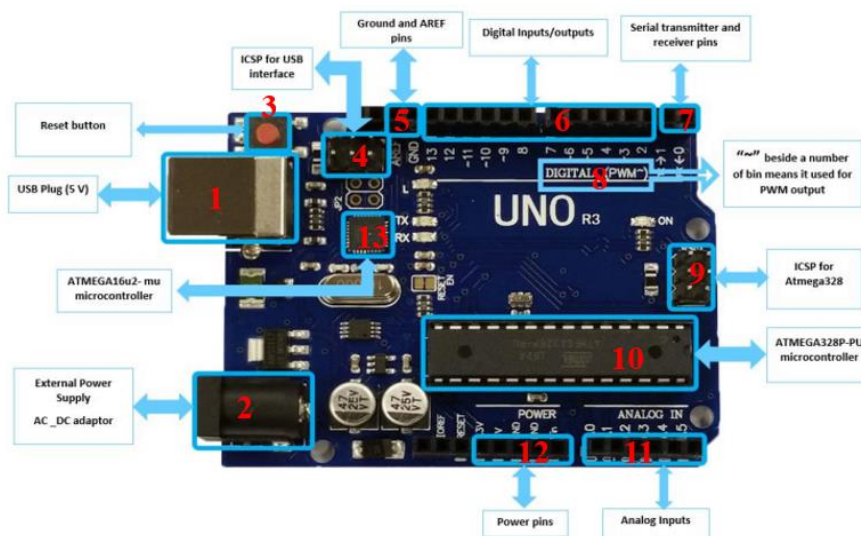
(9) SQW: Tạo xung vuông trên chân SQW/OUT, tần số xung vuông được thiết lập bởi RS0, RS1.

**Bảng 1.6. Bảng tăng số trạng thái chân RS1 và RS0**

RS1	RS0	Tần Số
0	0	1Hz
0	1	4.096KHz
1	0	8.192KHz
1	1	32,768KHz

## 1.6. Board mạch Arduino UNO R3

Để đọc dữ liệu từ module thời gian thực DS1307 và hiển thị lên LED 7 đoạn cần sử dụng Arduino UNO R3 ([http://electronoobs.com/eng\\_arduino\\_tut31\\_sch3.php](http://electronoobs.com/eng_arduino_tut31_sch3.php)). Arduino - một máy tính nhỏ để người dùng có thể lập trình và thực hiện các dự án điện tử mà không cần phải có các công cụ chuyên biệt để phục vụ việc nạp code.



*Hình 1.7. Module Arduino UNO R3*

(Nguồn: [http://www.fecegypt.com/uploads/dataSheet/1522237550\\_arduino%20uno%20r3.pdf](http://www.fecegypt.com/uploads/dataSheet/1522237550_arduino%20uno%20r3.pdf))

**Cấu tạo:**

(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 module 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: 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.

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

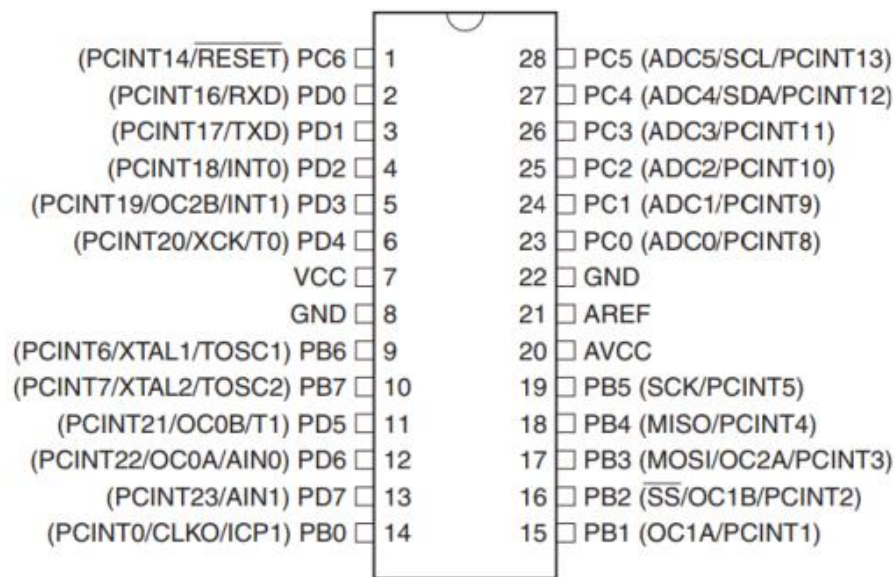
**Bảng 1.7. Thông số của Arduino UNO R3**

<b>Vi điều khiển</b>	<b>ATmega328 họ 8bit</b>
Điện áp hoạt động	5V DC (chỉ được cấp qua cổng USB)
Tần số hoạt động	16 MHz
Dòng tiêu thụ	khoảng 30mA
Điện áp vào khuyến dùng	7-12V DC
Điện áp vào giới hạn	6-20V DC
Số chân Digital I/O	14 (6 chân hardware PWM)
Số chân Analog	6 (độ phân giải 10bit)
Dòng tối đa trên mỗi chân I/O	30 mA
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB, trong đó 0.5KB bootloader
SRAM	2 KB (ATmega328)
EEPROM	1KB (ATmega328)



## 1.7. IC lập trình được Atmega16u2

Arduino R3 không thể hoạt động mà không cần đến bộ não của board mạch này. Atmega16u2 được sản xuất theo kiến trúc RISC (Reduce Instruction Set Computer) tiên tiến bao gồm: 125 tập lệnh đơn lệnh đơn chu kỳ; 32 thanh ghi 8 bit dùng chung; Tối đa 16 tập lệnh MIPS (Microprocessor without Interlocked Pipeline Stages) với tần số cho phép 16 MHz. Một bộ đếm 8 bit (Timer/Counters) và 1 bộ đếm 16 bit (Timer/Counter) dùng cho điều biến độ rộng xung PWM. Giao tiếp USART với chế độ SPI master và chế độ kiểm soát phản cứng (RTS/CTS). Bộ chuyển đổi tín hiệu ADC, công cụ thời gian thực và khả năng hoạt động ở mức điện áp thấp trong chế độ tiết kiệm năng lượng.



Hình 1.8. Sơ đồ chân IC Atmega16u2

(Nguồn: [http://www.fecegypt.com/uploads/dataSheet/1522237550\\_arduino%20uno%20r3.pdf](http://www.fecegypt.com/uploads/dataSheet/1522237550_arduino%20uno%20r3.pdf))

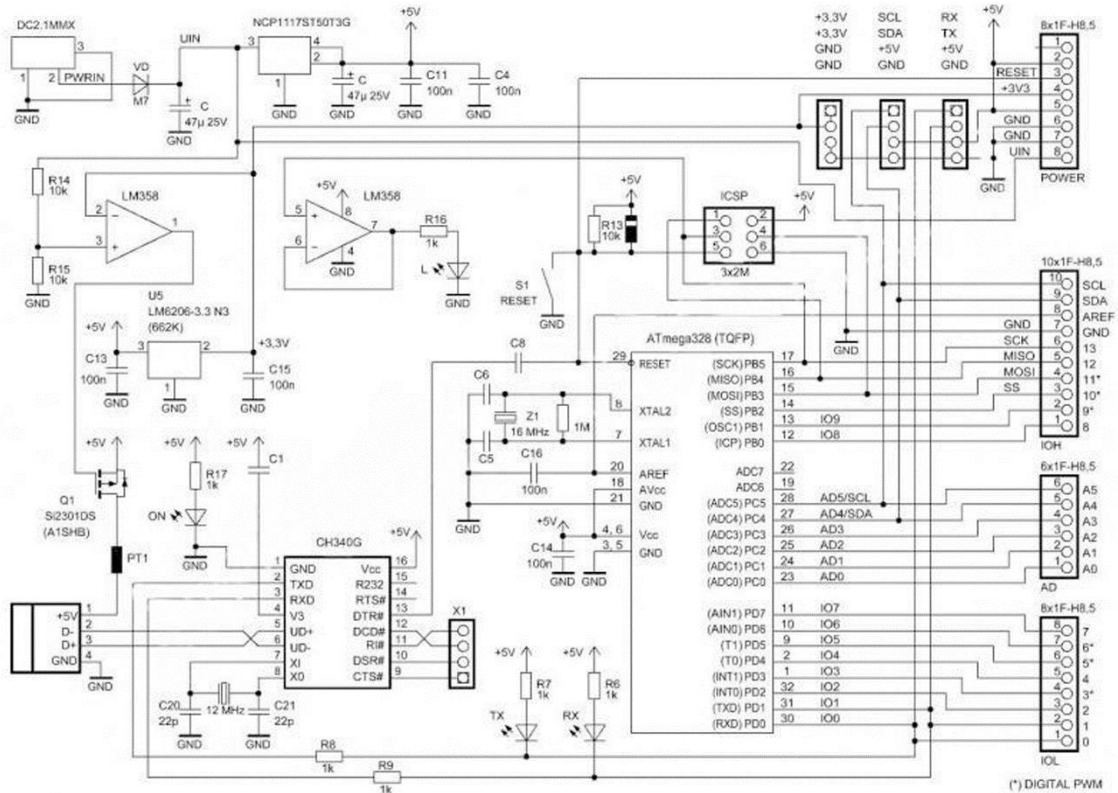
### Cấu tạo:

22 chân đa dụng vừa làm tín hiệu vào vừa làm tín hiệu ra tín hiệu số (PB0 - PB5 ~ chân 14 – chân 19, PC0 – PC6 ~ chân 23 – chân 28 và chân 1, PD0 – PD7 ~ chân 2 – chân 6 và chân 11 – chân 13).

2 chân nối với bộ dao động thạch anh (chân 8 và chân 9), tần số dao động 16MHz.

3 chân nguồn VCC và GND (chân 7, 8 và chân 22) cấp nguồn 5V và Mass.

6 chân ADC (Analog Digital Converter): bộ chuyển đổi ADC 10 bit (chân 23 – chân 28) phối hợp với chân AREF (chân 21) hiệu chỉnh giá trị điện áp đầu vào cho ADC.



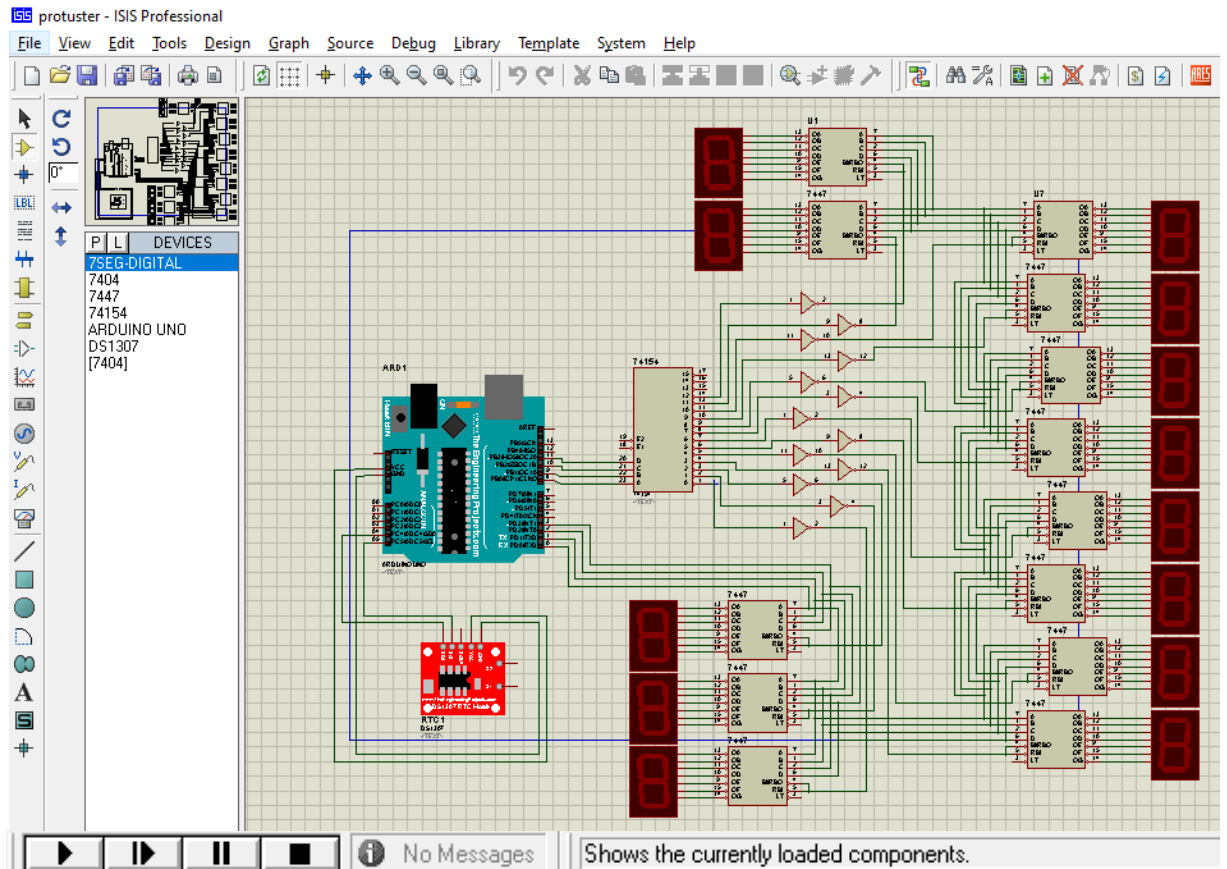
Hình 1.9. Sơ đồ nguyên lý của boad mạch Arduino UNO R3

(Nguồn: [http://electronoobs.com/eng\\_arduino\\_tut31\\_sch3.php](http://electronoobs.com/eng_arduino_tut31_sch3.php))

## 1.8. Phần mềm Proteus version 7.10SP2

Trước khi thi công mạch sản phẩm thực tế cần trải qua quá trình mô phỏng mạch để kiểm chứng thuật toán cũng như hoạt động logic của linh kiện điện tử. Proteus là bộ công cụ chuyên về mô phỏng mạch điện tử chạy trên môi trường Windows, là sản phẩm của công ty Labcenter Electronics - một công ty sản xuất phần mềm CAD của Anh. Sử dụng ISIS Schematic Capture để thiết kế nguyên lý và mô phỏng nguyên lý hoạt động của đề tài (Hình 1.10).

Quan trọng nhất là thanh thư viện nơi chứa các linh kiện và vùng làm việc dùng để vẽ mạch nguyên lý (Hình 1.10). Ngoài ra, các nút mô phỏng được sử dụng cho việc mô phỏng nguyên lý hoạt động (Hình 1.10 bên dưới cùng).



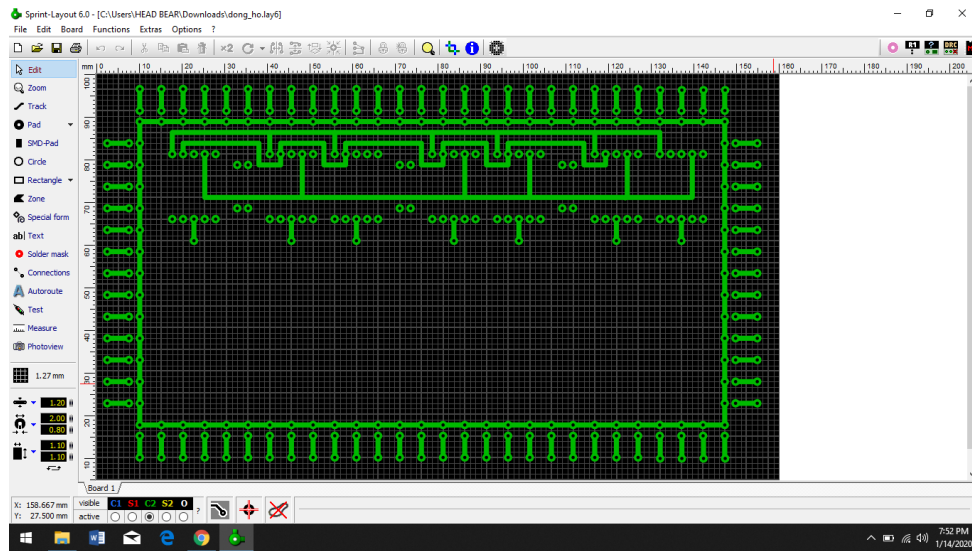
Hình 1.10. Phần mềm Proteus ISIS 7 Professional

## 1.9. Phần mềm Sprint Layout Viewer

Khi hoàn tất việc mô phỏng nguyên lý hoạt động của mạch sản phẩm, tiếp theo sẽ là thiết kế sơ đồ mạch in – bìa đục lỗ để gắn linh kiện và nối dây đến các linh kiện bằng phần mềm Sprint-Layout-Viewer version 6.0.

Sprint-Layout-Viewer là ứng dụng cho phép xem và in các bản thiết kế PCB được tạo ra từ chương trình Sprint-Layout (có định dạng LAY), và kiểm tra các lớp và các thành phần được nhúng vào bản thiết kế này.

Hình 1.12, sơ đồ mạch in của module hiển thị LED 7 đoạn. Trong đó, các đường màu xanh là các tín hiệu được nối với các linh kiện. Mỗi linh kiện được thể hiện là các chấm tròn (vị trí của chân linh kiện). Khung ngoài cùng bên trái chứa các công cụ để vẽ các đường, các điểm và các hình dạng (shapes), và điều chỉnh độ rộng các đường mạch in cho phù hợp.

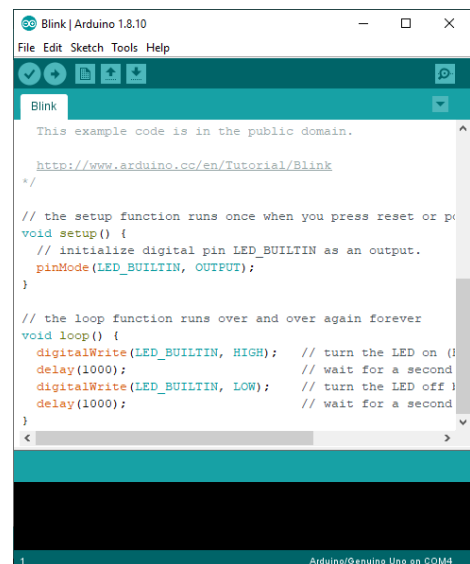


Hình 1.11. Phần mềm sprint Layout 6.0

(Nguồn: <https://www.pcbway.com/blog/4.html>)

### 1.10. Phần mềm Arduino IDE

Arduino IDE (Arduino Integrated Development Environment) là một chương trình phần mềm mã nguồn mở cho phép người dùng viết và tải lên mã lên vi điều khiển. Ứng dụng này hoạt động được cả 3 môi trường Windows, Linux và MacOS, mã nguồn của Arduino IDE được viết bởi C/C++ và tương thích hầu hết các board Arduino. Chương trình viết trên IDE có thể là C hoặc C++, sau khi biên dịch được nạp trực tiếp lên board mạch lập trình thông qua cổng USB.



Hình 1.12. Giao diện phần mềm Arduino IDE

Hình 1.13 là ví dụ về giao diện của Arduino IDE được sử dụng để bật tắt LED tại chân số 13 sau mỗi chu kỳ 1 giây (hàm delay 1000). Trong đó có 2 hàm chính là hàm setup dùng để khai báo và mở cổng hoạt động, hàm loop là hàm chạy chính thao tác các thuật toán và được chạy lặp lại liên tục cho đến khi mạch tắt nguồn.

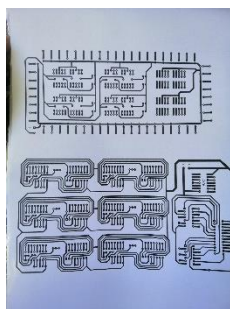
## 2. PHƯƠNG PHÁP NGHIÊN CỨU

Như đã đề cập, đồng hồ thời gian thực bắt đầu với linh kiện hiển thị, IC số, kit Arduino, module cung cấp dữ liệu thời gian. Trong đó module thời gian thực DS1307 có nhiệm vụ cung cấp ngày giờ cho kit Arduino. IC 74LS47 hoạt động như bộ giải mã tín hiệu nhị phân ở đầu ra của Arduino để hiển thị lên LED 7 đoạn. IC 74LS154 được sử dụng để làm bộ giả mã kênh điều khiển các chân BI/RBO của 7447 thực thi việc quét LED. Các linh kiện và thiết bị khác cùng nhau phối hợp để tạo nên sản phẩm hoàn chỉnh.

### 2.1. Thiết kế, thi công mạch in

Mạch sản phẩm được thiết kế nguyên lý bởi phần mềm SprintLayout ver6.0 (Hình 2.1). Sau đó, mạch in được gia công trên board đồng theo trình tự các bước sau:

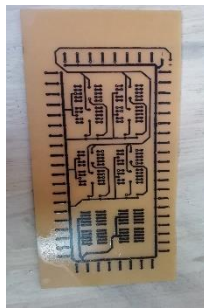
- (1) In mạch: mạch in được in trên giấy decal
- (2) Ủi mạch: giấy decal được phủ lên mặt đồng và gia nhiệt trong khoảng thời gian nhất định (thường mất 5 – 10 phút).
- (3) Ăn mòn: lớp đồng sau khi được phủ bởi mực máy in từ giấy decal sẽ được bảo vệ chống lại sự ăn mòn của dung dịch  $\text{Fe}_2\text{O}_3$  được hòa tan trong nước. Trong quá trình ăn mòn để rút ngắn thời gian này, chúng ta cần gia nhiệt cũng như tốc độ chuyển động của board mạch trong dung dịch này.
- (4) Khoan mạch: vị trí các linh kiện được cố định bởi các kích thước của mũi khoan (thường chọn 1mm).
- (5) Lắp linh kiện: các linh kiện được cố định trên từng vị trí tương ứng bởi thiếc hàn. Trong quá trình này vừa lắp kết hợp với kiểm tra sự thông mạch.



(1)



(2)



(3)



(4)



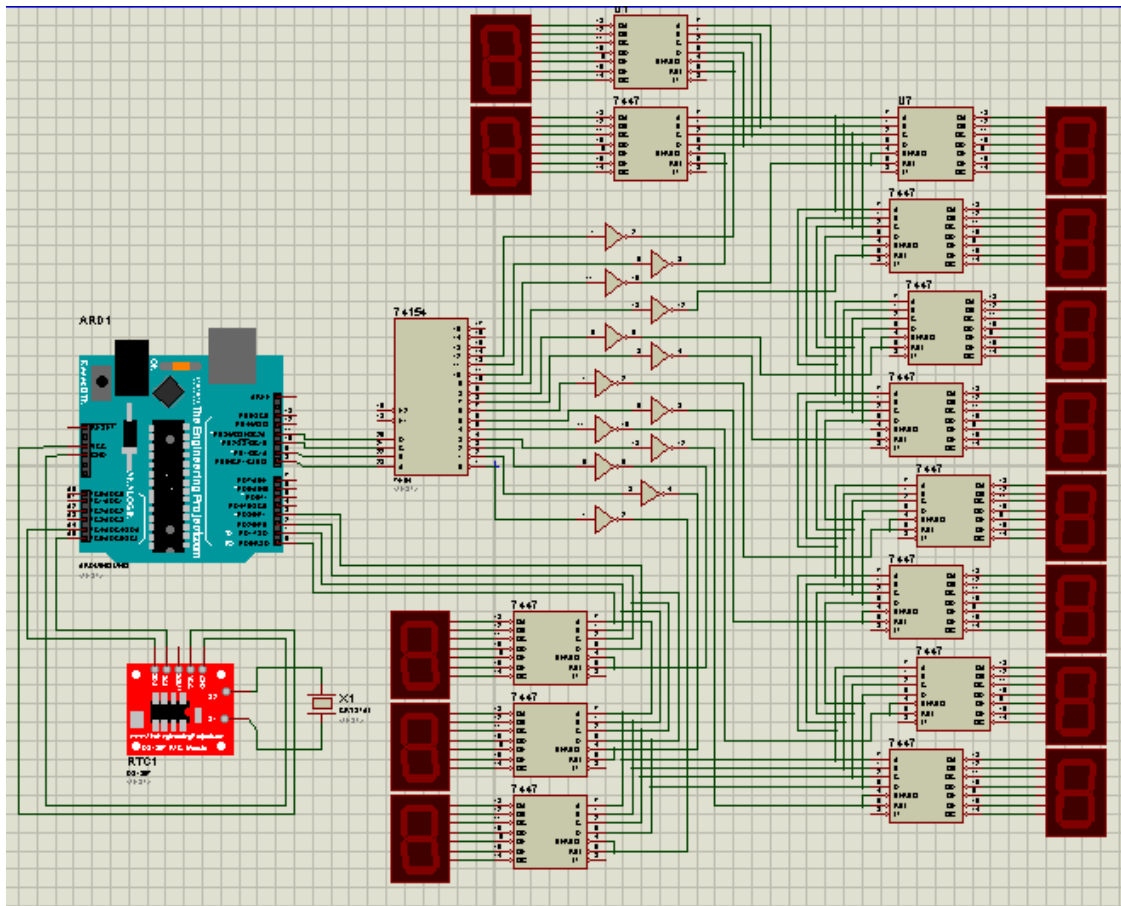
(5)

Hình 2.1. Các bước thi công mạch in

### 2.2. Mô phỏng nguyên lý hoạt động bằng phần mềm Proteus

Việc mô phỏng nguyên lý của mạch có ý nghĩa hết sức quan trọng, thông qua việc mô phỏng các giải thuật được kiểm tra chi tiết nhằm tránh những sai sót không đáng có.







Hình 2.2. Sơ đồ nguyên lý và mô phỏng cho đồng hồ thời gian thực

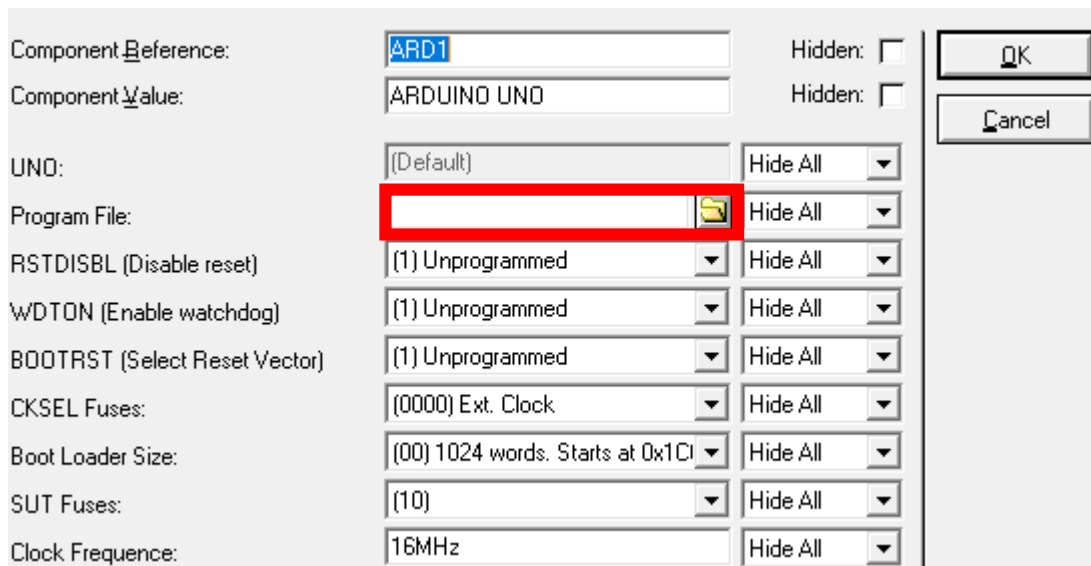
Trong hình 2.2. 13 LED 7 đoạn loại Cathode chung được dùng để hiển thị các thành phần của thời gian như: Thứ, Ngày, Tháng, Năm, Giờ, Phút, Giây. Thuật toán quét LED được áp dụng để Bật/Tắt các LED theo tuần tự với chu kỳ 1s cho mỗi lần hiển thị. Các chân BI/RBO được dùng cho giải thuật này. 12 IC 74LS47 (kết nối trực tiếp với LED 7 đoạn) được sử dụng để giải mã BCD sang LED 7 đoạn. Vì mỗi tín hiệu đầu ra của 74LS47 cần được làm mới sau mỗi chu kỳ nên cần sử dụng đến IC 74LS154 (IC ở vị trí gần với Arduino) để lần lượt quét qua các địa chỉ cho IC 74LS47. Do đầu ra của bộ giải mã địa chỉ luôn có mức tích cực thấp, IC 74LS04 (cổng logic NOT) được sử dụng để đảo ngược các tín hiệu này (2 IC góc trên cùng ở giữa). Module thời gian thực (khung màu đỏ) được dùng cho mô phỏng thời gian của đồng hồ. Board Arduino (màu xanh) là module dùng để lập trình mô phỏng. Các bước tiến hành mô phỏng như sau:

**Bước 1:** Hoàn thiện việc lấy linh kiện và nối dây liên kết

**Bước 2:** Lập trình cho kit Arduino bằng phần mềm Arduino IDE

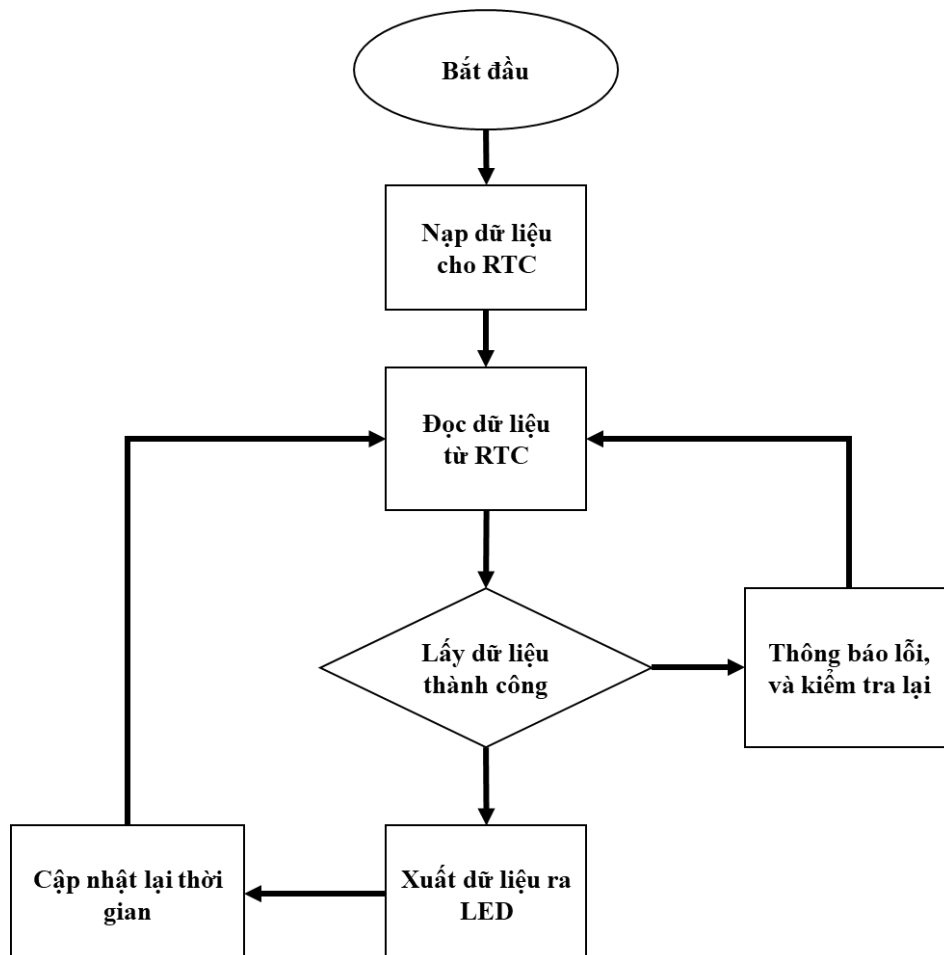
**Bước 3.** Nạp code cho Arduino bằng cách D\_Click lên biểu tượng board lập trình và lựa chọn tập tin có phần mở rộng .HEX để thực thi.

**Bước 4:** Nhấn nút  để bắt đầu mô phỏng, nhấn nút  để tạm dừng.



Hình 2.3. Cửa sổ nạp chương trình cho board Arduino

### 2.3. Lập trình



Hình 2.4. Sơ đồ khối chương trình đồng hồ thời gian thực

### 2.3.1. Giao tiếp RTC với Arduino

Để cho RTC giao tiếp được với Arduino thì cần phải sử dụng đến thư viện “*RTCLib.cpp*” và “*RTCLib.h*” công dụng của thư viện này sẽ giúp cho việc lấy dữ liệu từ bộ nhớ của RTC ra xử lý và nạp vào vi điều khiển.

```
// khai báo thư viện
#include <Wire.h>
#include "RTCLib.h"
//Khai báo thời gian thực.
RTC_DS1307 rtc;
void setup ()
{
    //tốc độ truyền thông nối tiếp 9600 baud
    Serial.begin(9600);
    //Thiết lập các chân 0-13 là các đầu ra
    for (int i = 0; i <= 13; i++)
    {
        pinMode(i, OUTPUT);
    }
    //Khởi tạo RTC
    if (! rtc.begin())
    {
        Serial.print("Couldn't find RTC");
        while (1);
    }
    if (! rtc.isrunning())
    {
        lcd.print("RTC is NOT running!");
        Serial.print("RTC is NOT running!");
    }
    //cập nhật thời gian đầu tiên cho DS1307 từ PC
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
```

Hình 2.5. Trình tự các bước khởi tạo RTC

**Bước 1:** Khởi tạo RTC: *rtc.begin()*;

**Bước 2:** Kiểm tra trạng thái hoạt động của RTC: *!rtc.isrunning()*, kết quả trả về giá trị TRUE khi RTC chưa hoạt động, ngược lại là FALSE.

**Bước 3:** Cập nhật thời gian cho RTC: *rtc.adjust(DateTime(F(\_\_DATE\_\_),F(\_\_TIME\_\_)))*;

### 2.3.2. Hiển thị thời gian lên LED 7 đoạn

Để hiển thị thời gian lên LED 7 đoạn cần sử dụng đến kỹ thuật quét LED, với tất cả các chân DATA của IC 74LS47 được nối chung với nhau đồng nghĩa với việc tất cả LED 7 đoạn đều sáng khi dữ liệu được gửi lên 4 bit data. Tuy nhiên khi chân BI/RBO và chân RBI ở mức thấp thì LED sẽ không hiển thị, như vậy kỹ thuật quét LED sẽ là hiển thị từng LED tương ứng với BI/RBO và RBI của IC 74LS47 tương ứng với giá trị là 1, sau đó tắt LED này đi với giá trị BI/RBO và RI tương ứng với giá trị bằng 0. Tiếp tục thực hiện cho các LED còn lại.



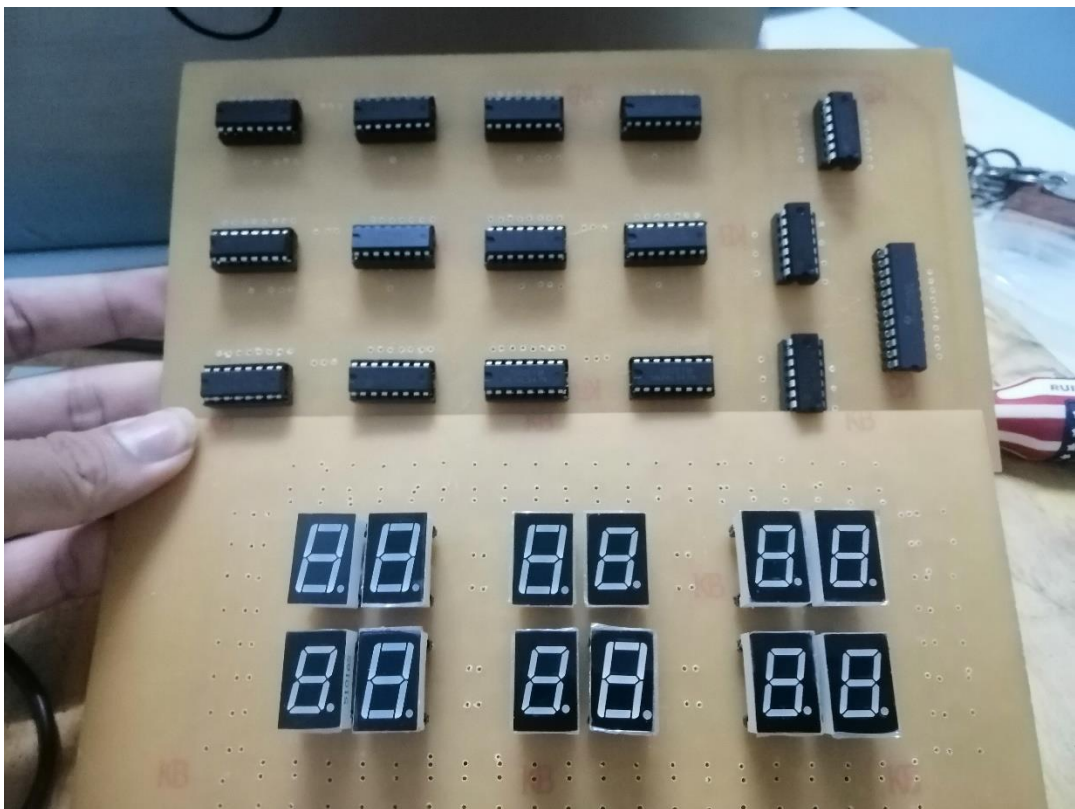
Hình 2.6 là các bước hiển thị số thời gian giây hàng đơn vị và hàng chục với giải thuật quét LED. Trong đó, 4 bit dữ liệu A0 – A3 là địa chỉ của BI/RBO và RI tương ứng với việc hiển thị LED giây hàng đơn vị và LED giây hàng chục. Mỗi LED được thấp sáng trong khoảng thời gian 1 chu kỳ máy (hàm delay (1)), tương tự như các thời gian năm, tháng, ngày, giờ, phút (xem mã nguồn ở bảng phụ lục).

```
sodongigay(giaydonvi);
digitalWrite(A0, 1);
digitalWrite(A1, 1);
digitalWrite(A2, 0);
digitalWrite(A3, 1);
delay(1);
//hiển thị số giây hàng chục
sodongigay(giaychuc);
digitalWrite(A0, 0);
digitalWrite(A1, 1);
digitalWrite(A2, 0);
digitalWrite(A3, 1);
delay(1);
```

Hình 2.6. Các bước hiển thị thời gian lên LED 7 đoạn với giá trị của đơn vị giây

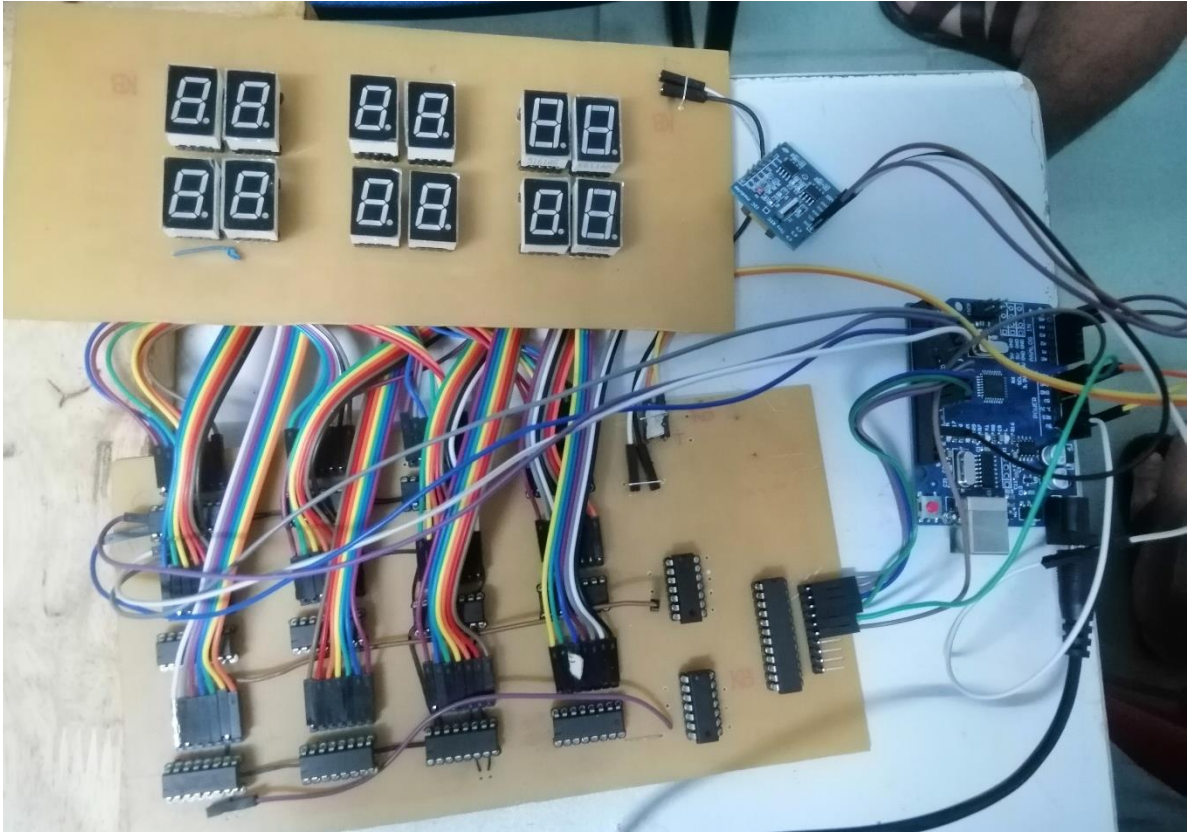
### 3. KẾT QUẢ

Kết quả thiết kế gia công mạch in hoàn thành sẽ được mạch điện với các LED 7 đoạn và các IC được hàn cố định (Hình 3.1). Trong đó, các IC số cho việc điều khiển hiển thị và quét LED (Hình 3.1 trên) và phần LED 7 đoạn hiển thị (Hình 3.1 dưới).



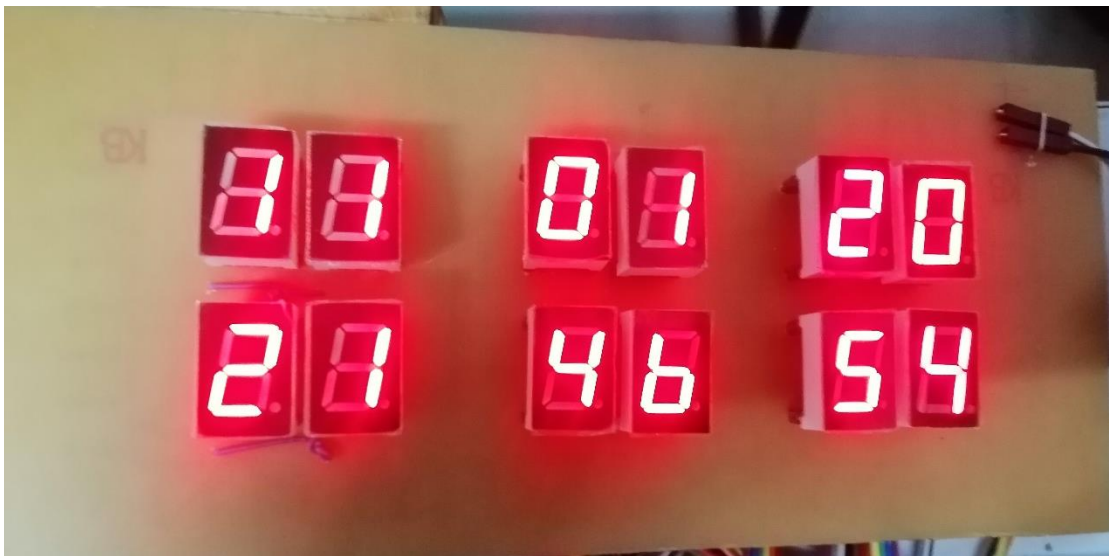
Hình 3.1. Linh kiện IC và LED 7 đoạn được cố định trên mạch in

Sau khi tiến hành nối dây cho các thành phần, kết quả như hình 3.2.



*Hình 3.2. LED 7 đoạn, thành phần liên kết của sản phẩm*

Chương trình được nạp vào board mạch Arduino, thời gian được hiển thị lên LED 7 đoạn ở theo từ trái sáng phải từ trên xuống dưới tương ứng với ngày, tháng năm, giờ, phút, giây. Hình 3.3 đồng hồ thời gian thực tương ứng với ngày 11 tháng 01 năm 2020, 21 giờ 46 phút 54 giây.



*Hình 3.3. Sản phẩm hoàn thiện với thời gian được hiển thị trên LED 7 đoạn*

## **THẢO LUẬN**

Sản phẩm hoàn thiện với các chức năng đã được trình bày ở phần phương pháp. Toàn bộ quy trình thiết kế, thi công sản phẩm được hoàn thiện trong thời gian 5 tuần của đợt thực tập cơ sở. Sản phẩm về cơ bản đáp ứng được các yêu cầu đặt ra trong yêu cầu của Giáo viên hướng dẫn. Tuy nhiên, sản phẩm vẫn chưa hoàn thiện về mặt hình thức và vẫn còn thiếu LED hiển thị phần thứ trong tuần do lúc thiết kế mạch in đã bỏ sót LED hiển thị nội dung này.

**PHỤ LỤC**

```

//khai báo chân Input của IC 74LS47
#define A 4
#define B 5
#define C 6
#define D 7
//Khai báo chân Input của IC 74LS154
#define A0 8
#define A1 9
#define A2 10
#define A3 11
// khai báo thư viện
#include <Wire.h>
#include "RTCLib.h"
//Khai báo thời gian thực.
RTC_DS1307 rtc;
void setup ()
{
  //tốc độ truyền thông nối tiếp 9600 baud
  Serial.begin(9600);
  //Thiết lập các chân 0-13 là các đầu ra
  for (int i = 0; i <= 13; i++)
  {
    pinMode(i, OUTPUT);
  }
  //Khởi tạo RTC
  if (! rtc.begin())
  {
    Serial.print("Couldn't find RTC");
    while (1);
  }
  if (! rtc.isrunning())
  {
    Serial.print("RTC is NOT running!");
  }
  //cập nhật thời gian đầu tiên cho DS1307 từ PC
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); //auto update from computer
  time
}
// case số chuyển từ IC 74LS47 sang LED 7 đoạn
void sodonvigiay(int x)
{
  switch (x)
  {
    case 0: // hiển thị số 0 trên LED 7 đoạn
      digitalWrite(A, 0);
      digitalWrite(B, 0);

```

```

digitalWrite(C, 0);
digitalWrite(D, 0);
break;
case 1: // hiển thị số 1 trên LED 7 đoạn
digitalWrite(A, 1);
digitalWrite(B, 0);
digitalWrite(C, 0);
digitalWrite(D, 0);
break;
case 2: // hiển thị số 2 trên LED 7 đoạn
digitalWrite(A, 0);
digitalWrite(B, 1);
digitalWrite(C, 0);
digitalWrite(D, 0);
break;
case 3: // hiển thị số 3 trên LED 7 đoạn
digitalWrite(A, 1);
digitalWrite(B, 1);
digitalWrite(C, 0);
digitalWrite(D, 0);
break;
case 4: // hiển thị số 4 trên LED 7 đoạn
digitalWrite(A, 0);
digitalWrite(B, 0);
digitalWrite(C, 1);
digitalWrite(D, 0);
break;
case 5: // hiển thị số 5 trên LED 7 đoạn
digitalWrite(A, 1);
digitalWrite(B, 0);
digitalWrite(C, 1);
digitalWrite(D, 0);
break;
case 6: // hiển thị số 6 trên LED 7 đoạn
digitalWrite(A, 0);
digitalWrite(B, 1);
digitalWrite(C, 1);
digitalWrite(D, 0);
break;
case 7: // hiển thị số 7 trên LED 7 đoạn
digitalWrite(A, 1);
digitalWrite(B, 1);
digitalWrite(C, 1);
digitalWrite(D, 0);
break;
case 8: // hiển thị số 8 trên LED 7 đoạn
digitalWrite(A, 0);
digitalWrite(B, 0);

```

```

    digitalWrite(C, 0);
    digitalWrite(D, 1);
    break;
case 9: // hiển thị số 9 trên LED 7 đoạn
    digitalWrite(A, 1);
    digitalWrite(B, 0);
    digitalWrite(C, 0);
    digitalWrite(D, 1);
    break;
default: // hiển thị số mặc định trên LED 7 đoạn
    digitalWrite(A, 1);
    digitalWrite(B, 0);
    digitalWrite(C, 0);
    digitalWrite(D, 1);
    break;
}
}
void loop ()
{
    //thiết lập thời gian hiện tại
    DateTime now = rtc.now();
    int nam = now.year(); //thiết lập thời gian năm
    int namdonvi = nam%10; //thiết lập thời gian năm hàng đơn vị
    int namchucvd = nam/10 ; //thiết lập thời gian năm
    int namchuc = namchucvd%10; //thiết lập thời gian năm hàng chục
    int thang = now.month(); //thiết lập thời gian tháng
    int thangdonvi = thang%10; //thiết lập thời gian tháng hàng đơn vị
    int thangchuc = thang/10; //thiết lập thời gian tháng hàng chục
    int ngay = now.day(); //thiết lập thời gian ngày
    int ngaydonvi = ngay%10; //thiết lập thời gian ngày hàng đơn vị
    int ngaychuc = ngay/10; //thiết lập thời gian ngày hàng chục
    int gio = now.hour(); //thiết lập thời gian giờ
    int giodonvi = gio % 10; //thiết lập thời gian giờ hàng đơn vị
    int giochuc = gio / 10; //thiết lập thời gian giờ hàng chục
    int phut = now.minute(); //thiết lập thời gian phút
    int phutdonvi = phut % 10; //thiết lập thời gian phút hàng đơn vị
    int phutchuc = phut / 10; //thiết lập thời gian phút hàng chục
    int giay = now.second(); //thiết lập thời gian giây
    int giaydonvi = giay % 10; //thiết lập thời gian giây hàng đơn vị
    int giaychuc = giay / 10; //thiết lập thời gian giây hàng chục
    // đọc các giá trị và gửi lên phần mềm
    Serial.print(phutchuc);
    Serial.print(phutdonvi);
    Serial.print(giaychuc);
    Serial.print(giaydonvi);
    //hiển thị số giây hàng đơn vị
    sodonvi = giaydonvi;
    digitalWrite(A0, 1);

```

```

digitalWrite(A1, 1);
digitalWrite(A2, 0);
digitalWrite(A3, 1);
delay(1);
//hiển thị số giây hàng chục
sodongiyay(giaychuc);
digitalWrite(A0, 0);
digitalWrite(A1, 1);
digitalWrite(A2, 0);
digitalWrite(A3, 1);
delay(1);
//hiển thị số phút hàng đơn vị
sodongiyay(phutdonvi);
digitalWrite(A0, 1);
digitalWrite(A1, 0);
digitalWrite(A2, 1);
digitalWrite(A3, 0);
delay(1);
//hiển thị số phút hàng chục
sodongiyay(phutchuc);
digitalWrite(A0, 1);
digitalWrite(A1, 1);
digitalWrite(A2, 0);
digitalWrite(A3, 0);
delay(1);
//hiển thị số giờ hàng đơn vị
sodongiyay(giodonvi);
digitalWrite(A0, 1);
digitalWrite(A1, 0);
digitalWrite(A2, 0);
digitalWrite(A3, 0);
delay(1);
//hiển thị số giờ hàng chục
sodongiyay(giochuc);
digitalWrite(A0, 0);
digitalWrite(A1, 1);
digitalWrite(A2, 1);
digitalWrite(A3, 0);
delay(1);
//hiển thị số ngày hàng đơn vị
sodongiyay(ngaydonvi);
digitalWrite(A0, 0);
digitalWrite(A1, 0);
digitalWrite(A2, 0);
digitalWrite(A3, 1);
delay(1);
//hiển thị số hàng hàng chục
sodongiyay(ngaychuc);

```

```

digitalWrite(A0, 0);
digitalWrite(A1, 0);
digitalWrite(A2, 0);
digitalWrite(A3, 0);
delay(1);
//hiển thị số tháng hàng đơn vị
sodongiyay(thangchuc);
digitalWrite(A0, 0);
digitalWrite(A1, 1);
digitalWrite(A2, 0);
digitalWrite(A3, 0);
delay(1);
//hiển thị số tháng hàng chục
sodongiyay(thangdonvi);
digitalWrite(A0, 1);
digitalWrite(A1, 1);
digitalWrite(A2, 1);
digitalWrite(A3, 0);
delay(1);
//hiển thị số năm hàng đơn vị
sodongiyay(namdonvi);
digitalWrite(A0, 0);
digitalWrite(A1, 0);
digitalWrite(A2, 1);
digitalWrite(A3, 0);
delay(1);
//hiển thị số năm hàng chục
sodongiyay(namchuc);
digitalWrite(A0, 1);
digitalWrite(A1, 0);
digitalWrite(A2, 0);
digitalWrite(A3, 1);
delay(1);
}

```