TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA ĐIỆN

====o0o====

BÁO CÁO ĐỒ ÁN 1

ĐỀ TÀI:

THIẾT KẾ MẠCH ĐO NHIỆT ĐỘ SỬ DỤNG BOARD ARDUINO, HIỂN THỊ TRÊN 4 LED 7 THANH VÀ TRUYỀN PHÁT KHÔNG DÂY SỬ DỤNG MODULE nRF24L01

Giáo viên hướng dẫn : TS. …

Sinh viên thực hiện : …

…

Lớp : TĐH2-K56

Hà nội, 11-2013

.

MỤC LỤC

DANH MỤC HÌNH VẼ

Chương 1: TỔNG QUAN

Chương 2: THIẾT KẾ, LẬP TRÌNH VÀ LẮP ĐẶT MẠCH ĐO NHIỆT ĐỘ VÀ TRUYỀN PHÁT KHÔNG DÂY

DANH MỤC BẢNG SỐ LIỆU

Chương 1:

Chương 2:

LỜI NÓI ĐẦU

Ngày nay khoa học công nghệ ngày càng phát triển, vi điều khiển AVR và vi điều khiển PIC ngày càng thông dụng và hoàn thiện hơn , nhưng có thể nói sự xuất hiện của Arduino vào năm 2005 tại Italia đã mở ra một hướng đi mới cho vi điều khiển. Sự xuất hiện của Arduino đã hỗ trợ cho con người rất nhiều trong lập trình và thiết kế, nhất là đối với những người bắt đầu tìm tòi về vi điều khiển mà không có quá nhiều kiến thức, hiểu biết sâu sắc về vật lý và điện tử . Phần cứng của thiết bị đã được tích hợp nhiều chức năng cơ bản và là mã nguồn mở. Ngôn ngữ lập trình trên nền Java lại vô cùng dễ sử dụng tương thích với ngôn ngữ C và hệ thư viện rất phong phú và được chia sẻ miễn phí. Chính vì những lý do như vậy nên Arduino hiện đang dần phổ biến và được phát triển ngày càng mạnh mẽ trên toàn thế giới.

Trên cơ sở kiến thức đã học trong môn học : Tin học đại cương , Điện tử tương tự và số… cùng với những hiểu biết về các thiết bị điện tử, chúng em đã quyết định thực hiện đề tài : **Thiết kế mạch đo nhiệt độ sử dụng board Arduino, hiển thị trên 4 led 7 thanh và truyền phát không dây sử dụng module nRF24L01** với mục đích để tìm hiểu thêm về Arduino, làm quen với các thiết bị điện tử và nâng cao hiểu biết cho bản thân. Do kiến thức còn hạn hẹp, thêm vào đó đây là lần đầu chúng em thực hiện đồ án nên chắc chắn không tránh khỏi những thiếu sót , hạn chế vì thế chúng em rất mong có được sự góp ý và nhắc nhở từ thầy giáo để có thể hoàn thiện đề tài của mình.

Chúng em xin chân thành cảm ơn thầy giáo TS.Nguyễn Hoàng Nam đã giúp đỡ chúng em rất nhiều trong quá trình tìm hiểu ,thiết kế và hoàn thành đề tài đồ án 1 này.

Hà Nội, ngày 29 tháng 11 năm 2013

Sinh viên thực hiện

Chương 1

TỔNG QUAN

1.1. Giới thiệu chung về Arduino

Arduino thực sự đã gây sóng gió trên thị trường người dùng DIY (là những người tự chế ra sản phẩm của mình) trên toàn thế giới trong vài năm gần đây, gần giống với những gì Apple đã làm được trên thị trường thiết bị di động. Số lượng người dùng cực lớn và đa dạng với trình độ trải rộng từ bậc phổ thông lên đến đại học đã làm cho ngay cả những người tạo ra chúng phải ngạc nhiên về mức độ phổ biến.

Hình 1.1: Những thành viên khởi xướng Arduino.

Arduino là gì mà có thể khiến ngay cả những sinh viên và nhà nghiên cứu tại các trường đại học danh tiếng như MIT, Stanford, Carnegie Mellon phải sử dụng; hoặc ngay cả Google cũng muốn hỗ trợ khi cho ra đời bộ kit Arduino Mega ADK dùng để phát triển các ứng dụng Android tương tác với cảm biến và các thiết bị khác?

Arduino thật ra là một bo mạch vi xử lý được dùng để lập trình tương tác với các thiết bị phần cứng như cảm biến, động cơ, đèn hoặc các thiết bị khác. Đặc điểm nổi bật của Arduino là môi trường phát triển ứng dụng cực kỳ dễ sử dụng, với một ngôn ngữ lập

trình có thể học một cách nhanh chóng ngay cả với người ít am hiểu về điện tử và lập trình. Và điều làm nên hiện tượng Arduino chính là mức giá rất thấp và tính chất nguồn mở từ phần cứng tới phần mềm. Chỉ với khoảng $30, người dùng đã có thể sở hữu một board Arduino có 20 ngõ I/O có thể tương tác và điều khiển chừng ấy thiết bị.

Arduino ra đời tại thị trấn Ivrea thuộc nước Ý và được đặt theo tên một vị vua vào thế kỷ thứ 9 là King Arduin. Arduino chính thức được đưa ra giới thiệu vào năm 2005 như là một công cụ khiêm tốn dành cho các sinh viên của giáo sư Massimo Banzi, là một trong những người phát triển Arduino, tại trường Interaction Design Institute Ivrea (IDII). Mặc dù hầu như không được tiếp thị gì cả, tin tức về Arduino vẫn lan truyền với tốc độ chóng mặt nhờ những lời truyền miệng tốt đẹp của những người dùng đầu tiên. Hiện nay Arduino nổi tiếng tới nỗi có người tìm đến thị trấn Ivrea chỉ để tham quan nơi đã sản sinh ra Arduino.

1.2. Giới thiệu về board Arduino Uno

Arduino Uno là 1 bo mạch thiết kế với bộ xử lý trung tâm là vi điều khiển AVR Atmega328. Cấu tạo chính của Arduino Uno bao gồm các phần sau:

Hình 1.2. Board Arduino Uno.

- *Cổng USB*: đây là loại cổng giao tiếp để ta upload code từ PC lên vi điều khiển. Đồng thời nó cũng là giao tiếp serial để truyền dữ liệu giữa vi điều khiển và máy tính.

- *Jack nguồn*: để chạy Arduino thỉ có thể lấy nguồn từ cổng USB ở trên, nhưng không phải lúc nào cũng có thể cắm với máy tính được . Lúc đó ta cần một nguồn từ 9V đến 12V.

- Có 14 chân vào/ra số đánh số thứ tự từ 0 đến 13, ngoài ra có một chân nối đất (GND) và một chân điện áp tham chiếu (AREF).

- *Vi điều khiển AVR*: đây là bộ xử lí trung tâm của toàn bo mạch. Với mỗi mẫu Arduino khác nhau thì con chip là khác nhau. Ở con Arduino Uno này thì sử dụng ATMega328.

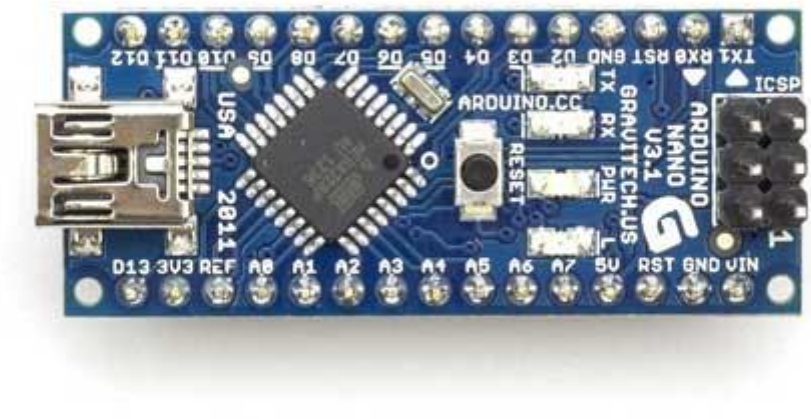
- Các thông số chi tiết của Arduino Uno:

Vi xử lý:	Atmega328
Điện áp hoạt động:	5V
Điện áp đầu vào:	7-12V
Điện áp đầu vào (Giới hạn):	6-20V
Chân vào/ra (I/O) số:	14 (6 chân có thể cho đầu ra PWM)
Chân vào tương tự:	6
Dòng điện trong mỗi chân I/O:	40mA
Dòng điện chân nguồn 3.3V:	50mA
Bộ nhớ trong:	32 KB (ATmega328)
SRAM:	2 KB (ATmega328)
EEPROM:	1 KB (ATmega328)
Xung nhịp:	16MHz

1.3. Giới thiệu về board Arduino Nano

Board Arduino Nano có cấu tạo, số lượng chân vào ra là tương tự như board Arduino Uno tuy nhiên đã được tối giản về kích thước cho tiện sử dụng hơn. Do được tối

giản rất nhiều về kích thước nên Arduino Nano chỉ được nạp code và cung cấp điện bằng duy nhất 1 cổng mini USB.



Hình 1.3. Board Arduino Nano.

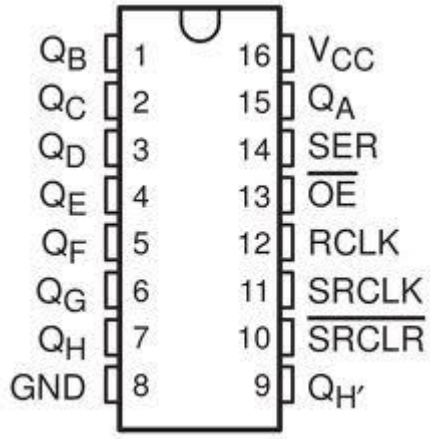
Thông số kĩ thuật chi tiết:

+ Vi xử lý	ATmega328 (phiên bản v3.0)
+ Điện áp hoạt động	5 V
+ Điện áp đầu vào (khuyến nghị)	7-12 V
+ Điện áp đầu vào (giới hạn)	6-20 V
+ Chân vào/ra số	14 (6 chân có khả năng xuất ra tín hiệu PWM)
+ Chân vào tương tự	8
+ Dòng điện mỗi chân vào/ra	40 mA
+ Bộ nhớ	16 KB (ATmega168), 32 KB (ATmega328) trong đó 2 KB dùng để nạp bootloader
+ SRAM	1 KB (ATmega168) hoặc 2 KB (ATmega328)

+ EEPROM	512 bytes (ATmega168) hoặc 1 KB (ATmega328)
+ Xung nhịp	16 MHz
+ Kích thước	0.73" x 1.70"

1.4. Giới thiệu về IC 74HC595

IC 74HC595 là thanh ghi dịch 8bit kết hợp chốt dữ liệu, đầu vào nối tiếp đầu ra song song. Chức năng: Thường dùng trong các mạch quét led 7, led matrix…để tiết kiệm số chân vi điều khiển tối đa (3 chân) . Có thể mở rộng số chân vi điều khiển bao nhiêu tùy thích mà không IC nào làm được bằng việc nối tiếp đầu vào dữ liệu các IC với nhau.



Hình 1.4. Cấu tạo IC 74HC595.

Giải thích ý nghĩa hoạt động của một số chân quan trọng:

+ *Chân 14 (Data pin):* đầu vào dữ liệu nối tiếp. Tại 1 thời điểm xung clock chỉ đưa vào được 1 bit.

+ *Các chân nối ra led (QA=>QH):* 15, 1, 2, 3, 4, 5, 6, 7 tương ứng với các 8 thanh led: a, b, c, d, e, f, g, dp.

+ *Chân 13*: chân cho phép tích cực ở mức thấp. Khi ở mức cao, tất cả các đầu ra của IC 74HC595 trở về trạng thái cao trở, không có đầu ra nào được cho phép.

+ ***Chân 12 (Latch pin):*** xung clock chốt dữ liệu. Khi có 1 xung clock tích cực ở sườn dương thì cho phép xuất dữ liệu trên các chân output. Lưu ý có thể xuất dữ liệu bất cứ lúc nào.

+ ***Chân 11 (Shift clock pin):*** chân vào xung clock. Khi có 1 xung clock tích cực ở sườn dương(từ 0 lên 1) thì 1 bit được dịch vào IC.

+ ***Chân 10:*** khi chân này ở mức thấp(mức 0) thì dữ liệu bị xóa trên chip.

+ ***Chân 9 (QH'):*** chân dữ liệu nối tiếp. Nếu dùng nhiều IC 74HC595 mắc nối tiếp nhau thì chân này đưa vào đầu vào của con tiếp theo khi đã dịch đủ 8 bit.

+ ***Chân 8:*** chân nối đất GND.

+ ***Chân 16:*** nối nguồn VCC.

1.5. Giới thiệu về cảm biến nhiệt độ LM35

Cảm biến LM35 là bộ cảm biến nhiệt mạch tích hợp chính xác cao mà điện áp đầu ra của nó tỉ lệ tuyến tính với nhiệt độ thang Celsius. Chúng không yêu cầu cân chỉnh ngoài vì vốn chúng đã được cân chỉnh.

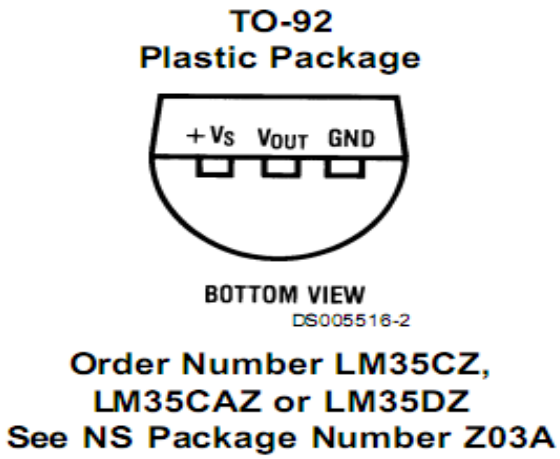
Cảm biến LM35 có 3 chân:

+ Chân nguồn VCC

+ Chân đầu ra Vout (chân tương tự)

+ Chân nối đất GND.

Hình 1.5. Cảm biến LM35.



Hình 1.6. Sơ đồ chân cảm biến LM35.

Đặc điểm chính của cảm biến LM35 :

- Điện áp đầu vào từ 4V đến 30V
- Độ phân giải điện áp đầu ra là 10mV/˚C
- Độ chính xác cao ở 25 ˚C là 0.5˚C

- Trở kháng đầu ra thấp 0.1 cho 1mA tải

Dải nhiệt độ đo được của LM35 là từ -55˚C đến 150˚C với các mức điện áp ra khác nhau. Xét một số mức điện áp sau :

- Nhiệt độ 55˚C điện áp đầu ra -550mV
- Nhiệt độ 25˚C điện áp đầu ra 250mV
- Nhiệt độ 150˚C điện áp đầu ra 1500mV.

Tùy theo cách mắc của LM35 để ta đo các giải nhiệt độ phù hợp. Đối với hệ thống này thì đo từ 0˚C đến 150˚C.

1.6. Giới thiệu về module truyền phát nRF24L01

1.6.1. Thông số kỹ thuật:

Hình 1.7. Module nRF24L01.

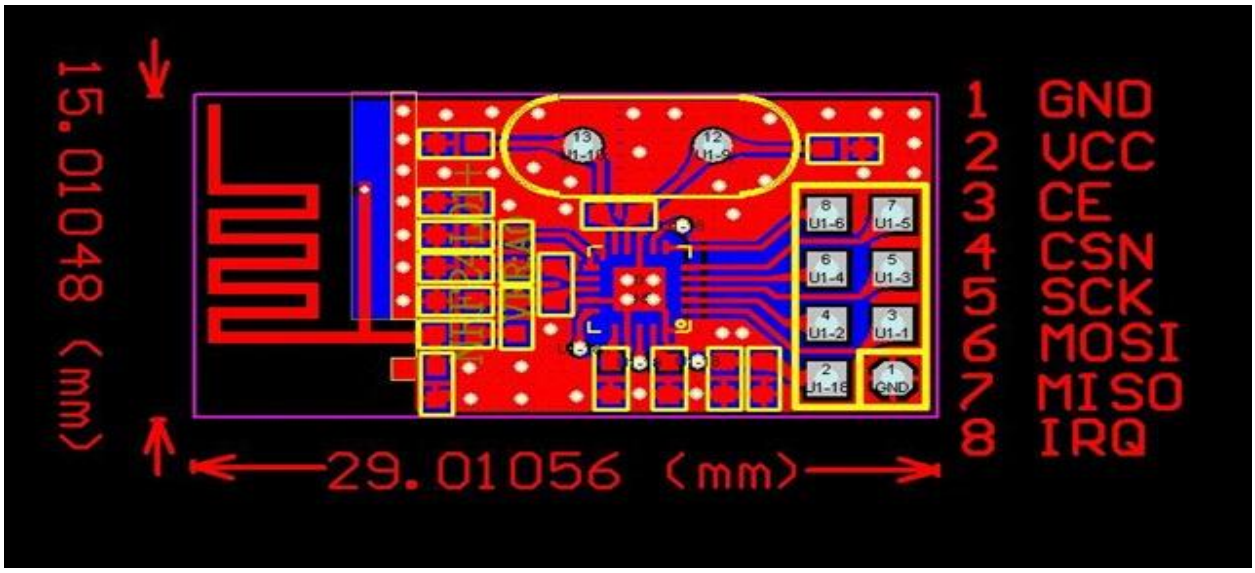
- *Radio:*
    +Hoạt động ở giải tần 2.4G
    + Có 126 kênh
    + Truyền và nhận dữ liệu
    + Truyền tốc độ cao 1Mbps hoặc 2Mbps.
- *Công suất phát:*
    + Có thể cài đặt được 4 công suất nguồn phát: 0,-6,-12,-18dBm.
- *Thu:*
    + Có bộ lọc nhiễu tại đầu thu
    + Khuếch đại bị ảnh hưởng bởi nhiễu thấp (LNA)
- *Nguồn cấp:*
    + Hoạt động từ 1.9-3.6V.
    + Các chân IO chạy được cả 3.3 lẫn 5V.
- *Giao tiếp:*
    + 4 chân giao tiếp theo giao thức SPI
    + Tốc độ tối đa 8Mbps
    + 3-32 bytes trên 1 khung truyền nhận.

1.6.2. Phân tích

+ Module nRF24L01 hoạt động ở tần số sóng ngắn 2.4G nên Modul này khả năng truyền dữ liệu tốc độ cao và truyền nhận dữ liệu trong điều kiện môi trường có vật cản
+ Module nRF24L01 có 126 kênh truyền. Điều này giúp ta có thể truyền nhận dữ liệu trên nhiều kênh khác nhau.
+ Module khả năng thay đổi công suất phát bằng chương trình, điều này giúp nó có thể hoạt động trong chế độ tiết kiệm năng lượng.
+ Chú ý: Điện áp cung cấp cho là 1.9à3.6V. Điện áp thường cung cấp là 3.3V. Nhưng các chân IO tương thích với chuẩn 5V. Điều này giúp nó giao tiếp rộng dãi với các dòng vi điều khiển.

1.6.3. Sơ đồ phần cứng và kết nối với Arduino

+ Sơ đồ chân nRF24L01:



Hình 1.8. Sơ đồ chân module nRF24L01.

+ Sơ đồ kết nối với Arduino:

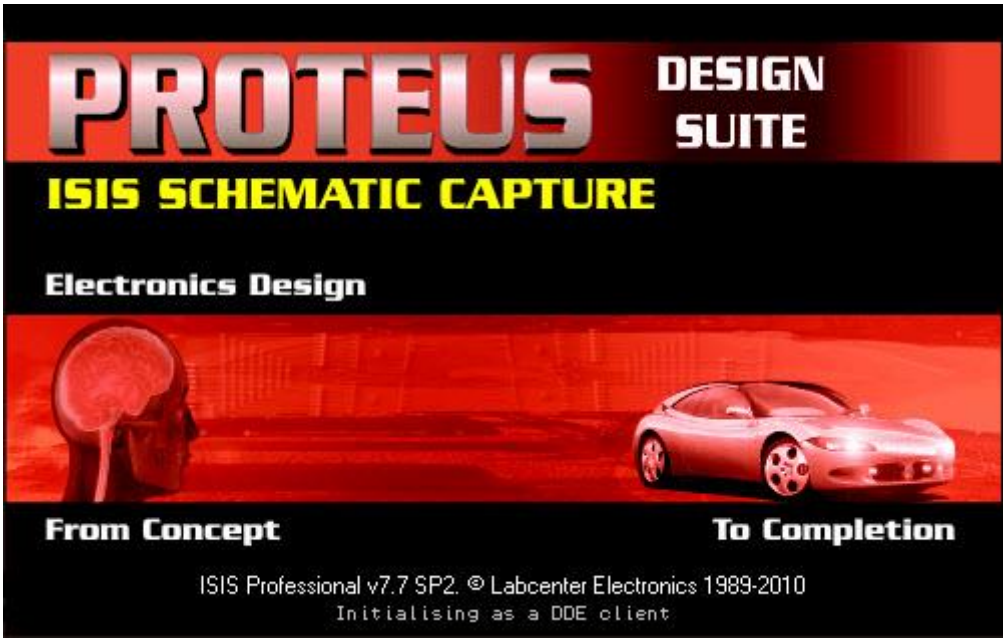
Bảng 1.1. Sơ đồ kết nối chân Arduino với module nRF24L01.

Tên chân	Số thứ tự chân	Chân kết nối tương ứng trên Arduino
GND	**1**	**GND**
VCC	**2**	**3.3V**
CE	**3**	**8**
CSN	**4**	**7**
SCK	**5**	**13**

1.7. Giới thiệu chung về phần mềm mô phỏng Proteus

Phần mềm Proteus là phần mềm cho phép mô phỏng hoạt động của mạch điện tử bao gồm phần thiết kế mạch và viết chương trìn điều khiển cho các họ vi điều khiển như MCS-51, PIC, AVR, … Proteus là phần mềm mô phỏng mạch điện tử của Lancenter Electronics, mô phỏng cho hầu hết các linh kiên điện tử thông dụng, đặn biệt hỗ trợ cho các MCU như PIC, 8051, AVR, Motorola.

Phần mềm bao gồm 2 chương trình: **ISIS** cho phép mô phỏng mạch và **ARES** dùng để vẽ mạch in. Proteus là công cụ mô phỏng cho các loại vi điều khiển khá tốt, nó hỗ trợ các dòng vi điều khiển PIC, 8051, PIC, dsPIC, AVR, HC11,…các giao tiếp I2C, SPI, CAN, USB, Ethenet…ngoài ra còn mô phỏng các mạch số, mạch tương tự một cách hiệu quả.



Hình 1.9. Giao diện khởi động phần mềm Proteus.

1.8. Thư viện Arduino trong Proteus

Thư viện Arduino là một bổ sung rất hay cho phần mềm mô phỏng Proteus nó giúp cho việc mô phỏng Arduino được thuận tiện và dễ dàng hơn thay vì chỉ mô phỏng được chip ATmega328(nhân của Arduino), thư viện này được phát triển bởi các kĩ sư Cesar Osaka, Daniel Cezar, Roberto Bauer và được đăng tải trên blog tiếng Bồ Đào Nha: http://blogembarcado.blogspot.de/

Thư viện bao gồm các linh kiện sau:

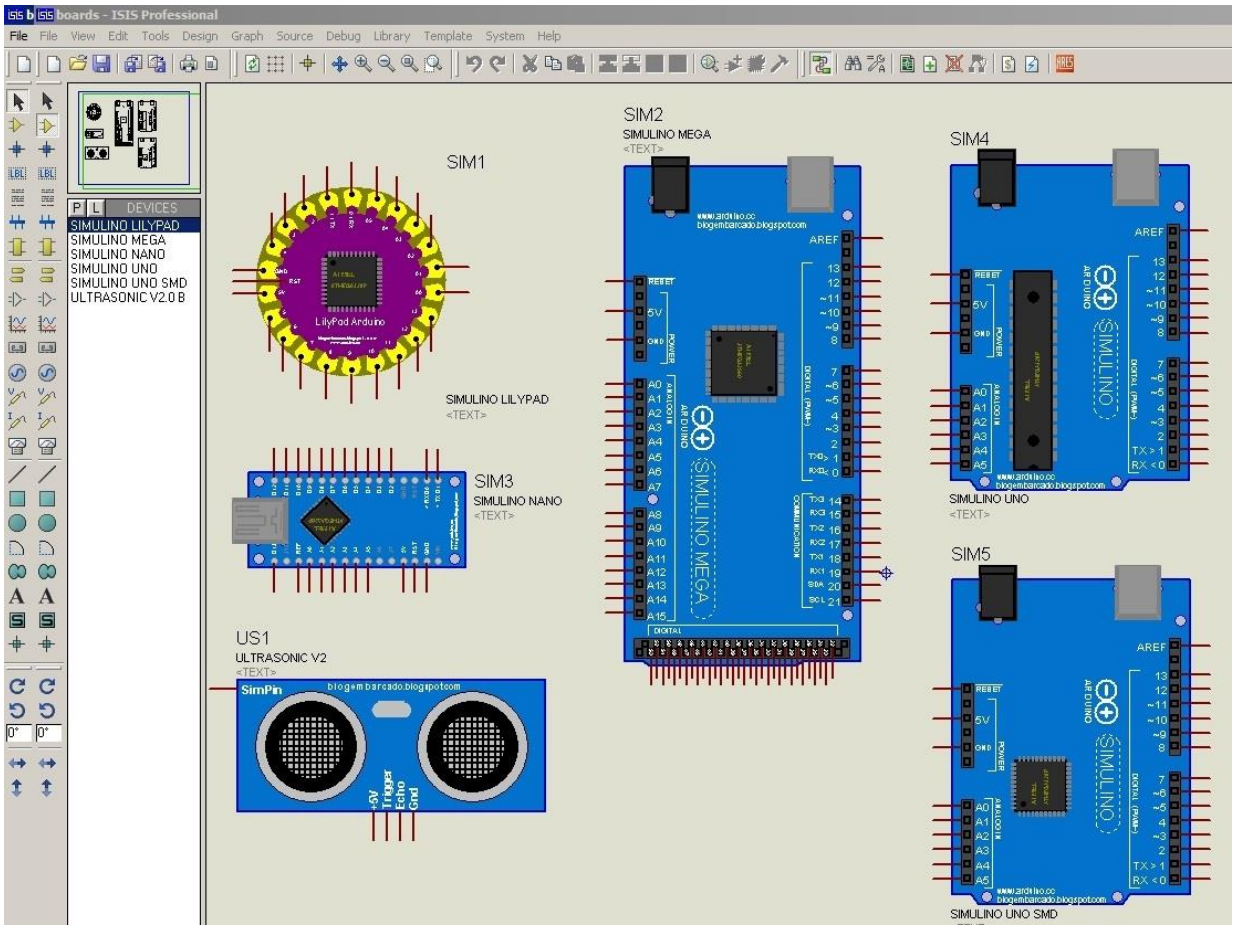
- Arduino Uno (Phiên bản chip ATmega328 chân DIP)
- Arduino Uno (Phiên bản chip ATmega328 chân SMD)

- Arduino Mega

- Arduino Lilypad

- Arduino Nano

- Cảm biến siêu âm Ultrasonic V2

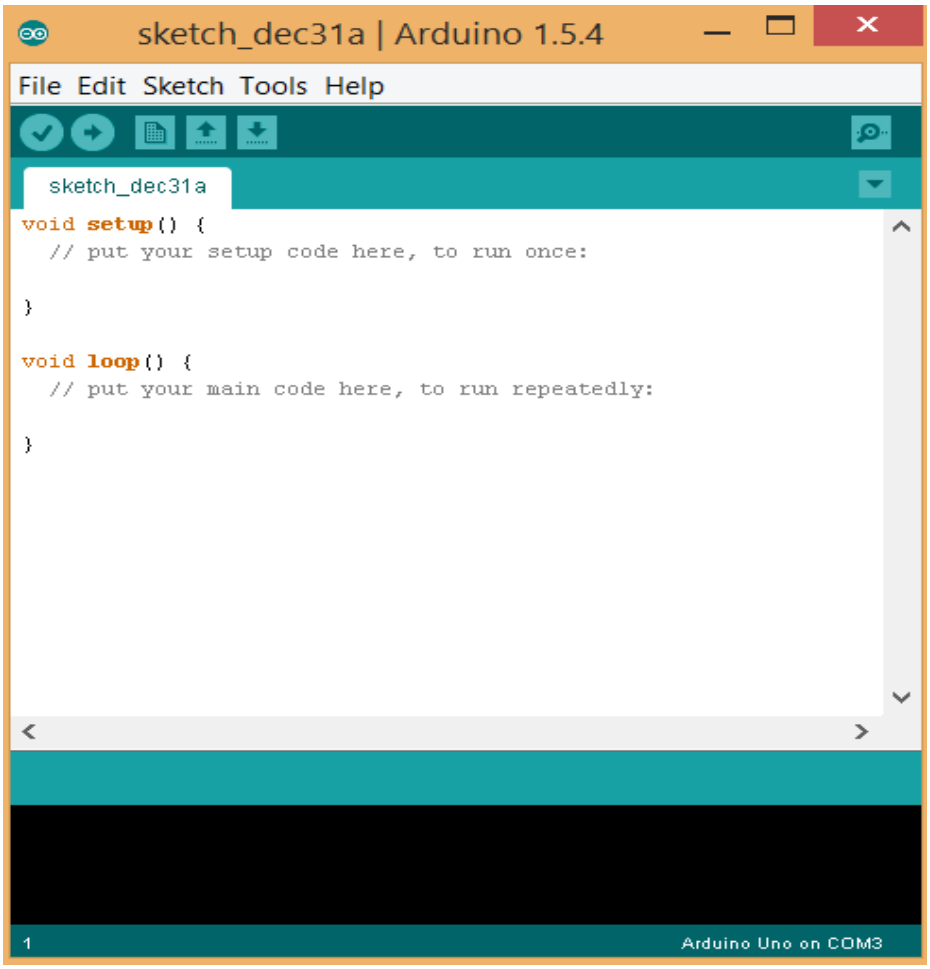


Hình 1.10. Các linh kiện trong thư viện Arduino cho Proteus.

1.9. Giới thiệu về Arduino IDE và ngôn ngữ lập trình cho Arduino

Thiết kế bo mạch nhỏ gọn, trang bị nhiều tính năng thông dụng mang lại nhiều lợi thế cho Arduino, tuy nhiên sức mạnh thực sự của Arduino nằm ở phần mềm. Môi trường lập trình đơn giản dễ sử dụng, ngôn ngữ lập trình Wiring dễ hiểu và dựa trên nền tảng

C/C++ rất quen thuộc với người làm kỹ thuật. Và quan trọng là số lượng thư viện code được viết sẵn và chia sẻ bởi cộng đồng nguồn mở là cực kỳ lớn.



Hình 1.11. Giao diện phần mềm Arduino IDE.

Arduino IDE là phần mềm dùng để lập trình cho Arduino. Môi trường lập trình Arduino IDE có thể chạy trên ba nền tảng phổ biến nhất hiện nay là Windows, Macintosh OSX và Linux. Do có tính chất nguồn mở nên môi trường lập trình này hoàn toàn miễn phí và có thể mở rộng thêm bởi người dùng có kinh nghiệm.

Ngôn ngữ lập trình có thể được mở rộng thông qua các thư viện C++. Và do ngôn ngữ lập trình này dựa trên nền tảng ngôn ngữ C của AVR nên người dùng hoàn toàn có thể nhúng thêm code viết bằng AVR vào chương trình nếu muốn.Hiện tại, Arduino IDE có thể download từ trang chủ http://arduino.cc/ bao gồm các phiên bản sau:

- Arduino 1.0.5

- Arduino1.5.5 BETA (Hỗ trợ cho 2 board Arduino mới nhất là: Arduino Yun và Arduino Due)

- Arduino IDE cho Intel Galileo

Chương 2

THIẾT KẾ, LẬP TRÌNH VÀ LẮP ĐẶT MẠCH ĐO NHIỆT ĐỘ VÀ TRUYỀN PHÁT KHÔNG DÂY.

2.1. Thiết kế mạch trên Proteus

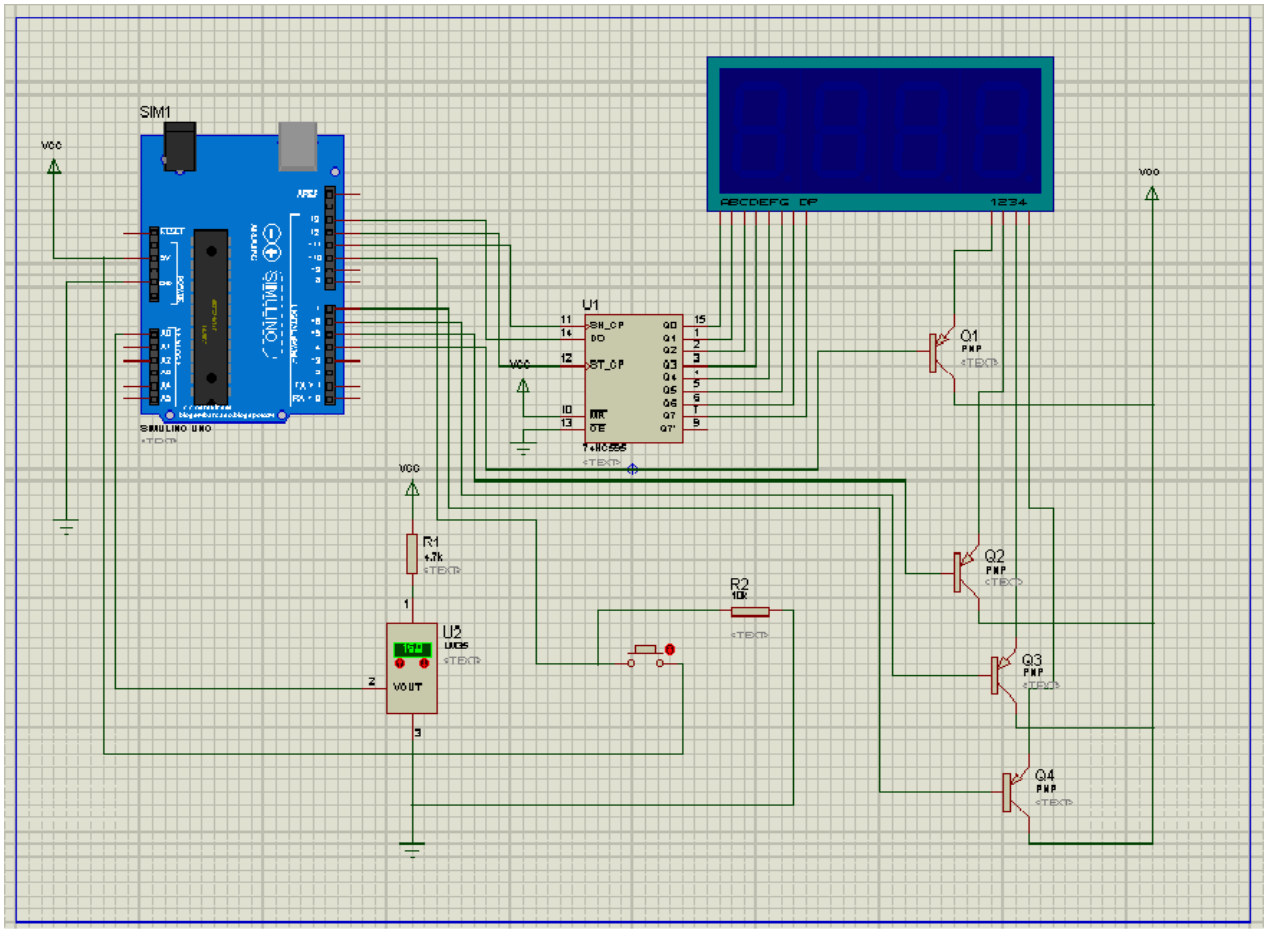
2.1.1. Thiết kế mạch đo nhiệt độ không truyền phát

Ban đầu, nhóm đã đồng ý với phương án thiết kế mạch đo nhiệt độ hiển thị ra 4 led 7 thanh bao gồm các chức năng hiển thị độ C và có thể hiển thị độ F bằng cách nhấn giữ công tắc.

Để thiết kế được mạch này, sau khi thảo luận và tham khảo ý kiến của thầy giáo hướng dẫn, nhóm đã thống nhất sử dụng phương pháp quét led dùng transistor để hiển thị nhiệt độ trên 4 led 7 thanh. Phương pháp này được áp dụng dựa trên hiện tượng lưu ảnh trên võng mạc của mắt. Ở phương pháp này, tín hiệu được truyền từ cảm biến LM35 vào Arduino và được tính toán ra nhiệt độ, rồi tín hiệu này lại được truyền từ Arduino ra led, tín hiệu truyền ra không liên tục mà theo từng xung nhịp một. Mỗi xung nhịp sẽ cách nhau 1ms. Do hiện tượng lưu ảnh trên võng mạc của mắt, hình ảnh được lưu lại trên võng mạc mắt trong khoảng thời gian là 40ms nên ta vẫn nhìn thấy nhiệt độ hiển thị trên led là các số rõ ràng chứ không thấy được tín hiệu bị ngắt quãng.

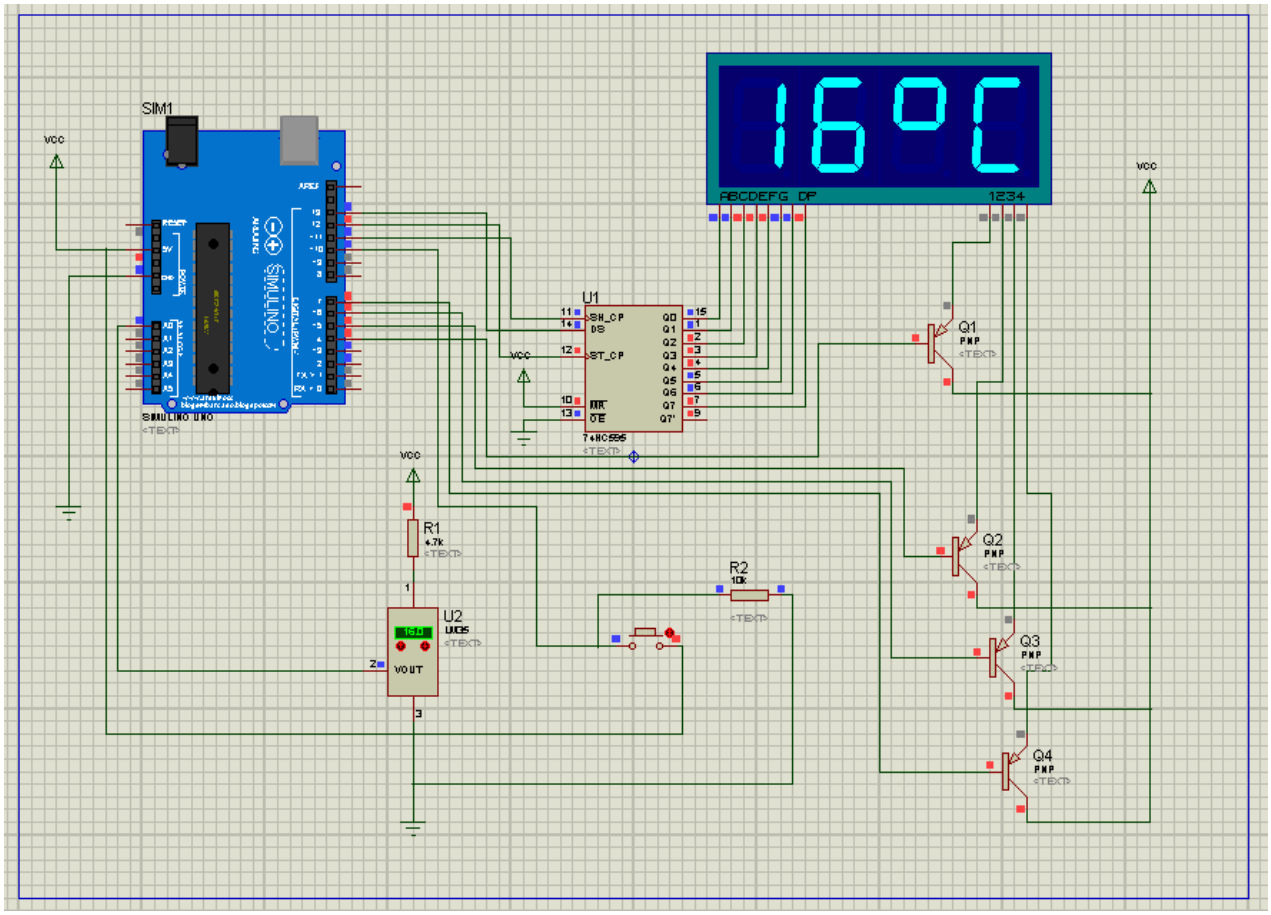
Để truyền dữ liệu từ Arduino ra led 7 thanh, nhóm đã sử dụng thêm 1 IC ghi dịch HC74595 với chức năng là giảm thiểu số chân phải cắm vào Arduino so với trường hợp cắm trực tiếp 4 led 7 thanh vào Arduino. Phương pháp quét led cũng sử dụng đến Transistor A1015 (PNP) để đưa được tín hiệu ra led.

Mạch được thiết kế trên phần mềm Proteus:



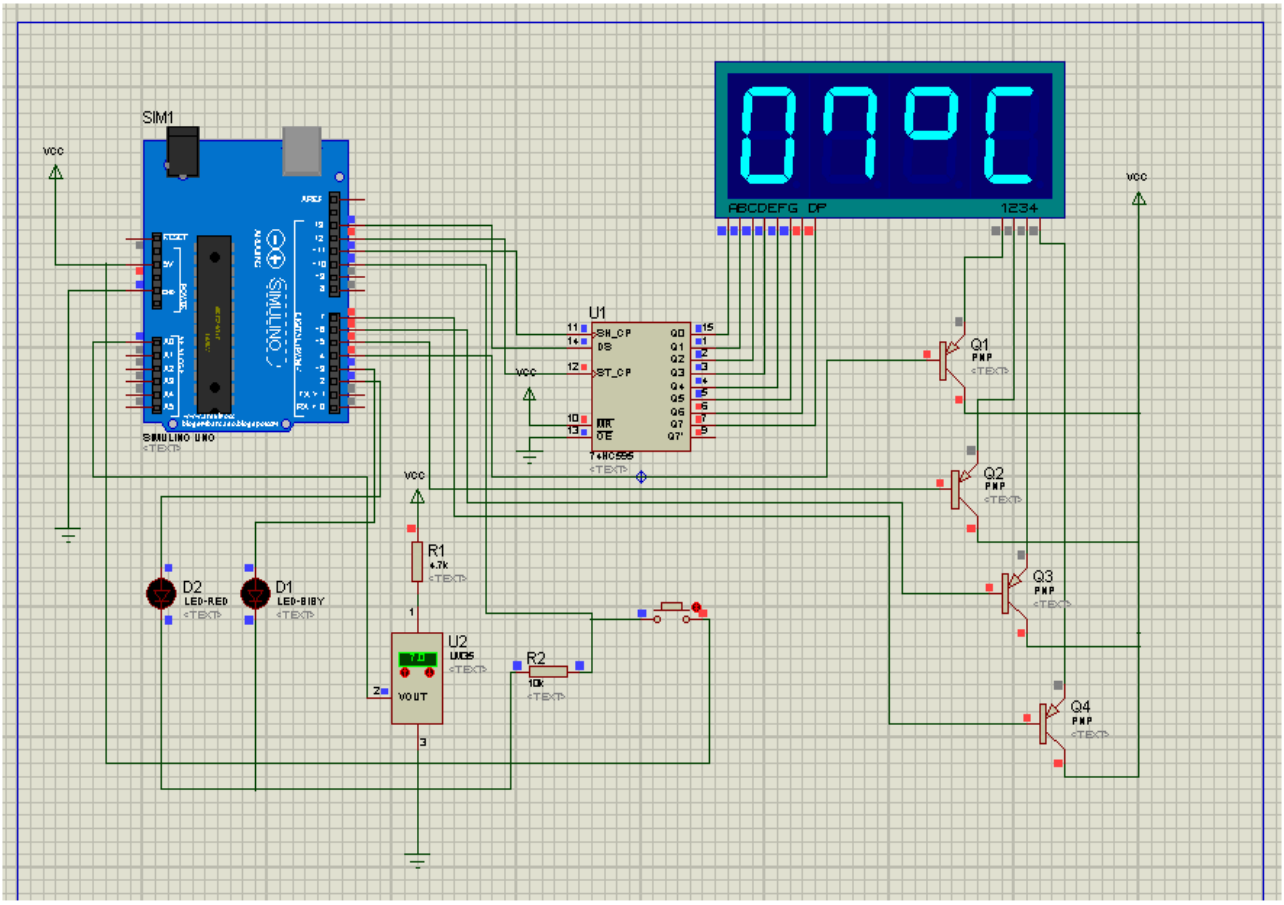
Hình 2.1. Sơ đồ nguyên lý mạch đo nhiệt độ không truyền phát thiết kế trên Proteus.

Sau khi viết được code và nạp code thành công, mạch mô phỏng đã hoạt động đúng theo các chức năng mong muốn.



Hình 2.2. Mô phỏng hiển thị nhiệt độ trên Proteus.

Nhóm đã tiếp tục cải tiến thêm chức năng cảnh báo ngưỡng trên và ngưỡng dưới. Tuy nhiên do cấu tạo của Arduino (không đủ chân ra, chân Interrupt) nên đã không thể làm được mạch cảnh báo có hiển thị và điều chỉnh nhiệt độ cảnh báo ngưỡng trên và ngưỡng dưới mà thay vào đó là mạch đo nhiệt độ có đèn led cảnh báo ngưỡng trên và ngưỡng dưới với nhiệt độ cảnh báo không hiển thị trên led mà được thay đổi trong code lập trình. Mạch đã mô phỏng thành công trên Proteus.



Hình 2.3. Mô phỏng mạch đo nhiệt độ không truyền phát có thêm chức năng cảnh báo giới hạn nhiệt độ bằng đèn led.

2.1.2. Thiết kế mạch đo nhiệt độ truyền phát không dây với module nRF24L01

Do linh kiện module truyền phát không dây module nRF24L01 không có trong bất cứ 1 thư viện nào của Proteus nên nhóm đã không thể thiết kế mạch mô phỏng linh kiện này mà thay vào đó, nhóm đã phải trực tiếp lắp mạch thật trên board test, sau đó viết và nạp code để test trực tiếp trên mô hình thật.

Mỗi board Arduino Uno và Arduino Uno đều chỉ có 14 chân xuất tín hiệu ra số (digital). Số lương chân ra số rất hạn chế nên việc kết nối với các thành phần chính phục vụ đo nhiệt độ (IC ghi dịch, cảm biến, led 7 thanh…) và module truyền phát đã sử dụng hết 12 chân ra số từ chân 2 đến chân 13 của board Arduino. Ta chỉ có thể sử dụng 2 chân ra số 0 và 1 cho các chức năng khác, tuy nhiên 2 chân số 0 (RX) và chân 1(TX) còn

được sử dụng để nhận và truyền dữ liệu nối tiếp TTL (TTL serial data), những chân này được kết nối với các chân tương ứng trên chip nối tiếp Atmega8U2 USB-to-TTL. Do vậy, việc sử dụng các chân này trong mạch với chức năng như chân ra cho công tắc chuyển hiển thị thang C hay thang F hay chân ra led cảnh báo là không khả thi. Thực tế khi nạp code và sử dụng những chân này, mạch đo gặp lỗi hiển thị không chuẩn xác và đèn led không thể điều khiển được nhấp nháy đúng nhiệt độ cảnh báo.

Vì vậy nhóm đã quyết định cắt bỏ chức năng dùng công tắc để chuyển kiểu hiện thị nhiệt độ 2 thang đo C, F và chức năng cảnh báo nhiệt độ giới hạn trên và dưới bằng 2 led cảnh báo. Thay vào đó nhóm đã lập trình để mạch cảnh báo nhiệt độ giới hạn 1 ngưỡng bằng nhấp nháy trực tiếp trên màn hình 4 chữ số led 7 thanh. Mạch truyền và mạch nhận nhiệt độ cũng có khả năng đo và hiển thị riêng biệt tình trạng nhiệt độ đo được, nhiệt độ nhận được lên màn hình máy tính riêng biệt qua chức năng Serial Monitor của phần mềm Arduino IDE. Việc hiển thị này sẽ giúp cho người sử dụng dễ dàng biết được nhiệt độ trực tiếp tại điểm đo và còn giúp người thiết kế đánh giá được độ trễ trong việc truyền, nhận tí hiệu.

Thư viện nRF24L01p cũng đã mặc định sẵn cách kết nối chân giữa Arduino và module nRF24L01, cho nên, để sử dụng được một cách đơn giản nhất và không phải can thiệp quá nhiều vào thư viện, ta sẽ điều chỉnh chân kết nối IC74HC595 và transistor với Arduino. Cụ thể thay đổi thể hiện trong bảng sau:

Bảng 2.1. Sơ đồ kết nối chân linh kiện IC 74HC595 và Transistor trong mạch đo nhiệt độ có truyền phát.

Linh kiện	Chân linh kiện	Chân kết nối tương ứng trên Arduino
IC 74HC595	**11 (Shift Clock)**	**6**
	12 (Latch Clock)	**9**
	14 (Data pin)	**10**
Transistor 1	**Base**	**2**

Transistor 2	**Base**	**3**
Transistor 3	**Base**	**4**
Transistor 4	**Base**	**5**

Kết nối module nRF24L01 với Arduino theo sơ đồ đã giới thiệu ở phần ở trên. Mạch thu và hiển thị nhiệt độ chỉ thực hiện chức năng hiển thị nhiệt độ nên sẽ không được lắp đặt cảm biến LM35, các linh kiện còn lại mạch phát và thu được kết nối giống nhau.

2.2. Lập trình cho mạch đo nhiệt độ

2.2.1. Lập trình cho mạch đo nhiệt độ không truyền phát có cảnh báo giới hạn trên và dưới

Code lập trình cho mạch đo nhiệt độ được viết bằng phần mềm Arduino IDE. Như chương trước đã giới thiệu, cảm biến LM35 đo nhiệt độ và thể hiện nhiệt độ đó dưới dạng điện áp (cứ 10mV là 1 độ, tối đa điện áp cung cấp là 5000mV – 10bit), do vậy để cho ra được giá trị nhiệt độ chính xác, trước hết, ta sử dụng hàm *analogRead()* để đọc giá trị điện áp từ cảm biến LM35 dưới dạng nhiệt độ. Giá trị điện áp này sẽ được tính toán để ra giá trị nhiệt độ theo công thức:

*tempC=(val/(1024*10))*5000 = val* 0.48828125*

(trong đó: *val* là giá trị đọc vào từ cảm biến, *tempC* là giá trị nhiệt độ thang Celsius)

Chuyển đổi sang nhiệt độ thang Farenheit (^oF) ta dùng công thức:

*tempF=(tempC*9)/5+32 = tempC*1.8+32*

Mạch đo được lập trình để giá trị nhiệt độ đo được hiển thị trên led 7 thanh bao gồm 4 kí tự: chữ số hàng chục, hàng đơn vị của nhiệt độ, kí hiệu độ và thang đo (XX^oC hoặc XX^oF), do màn hiển thị chỉ có 4 led 7 thanh nên khi nhiệt độ lớn hơn 100^o thì hiển thị: chữ số hàng trăm, hàng chục, hàng đơn vị của giá trị nhiệt độ, kí hiệu nhiệt độ (XXX^o). Muốn hiển thị được như vậy, ta cần dùng biến *conv* chuyển đổi kiểu dữ liệu của giá trị nhiệt độ đo được từ kiểu thực (float) sang kiểu số nguyên (int) rồi dùng thuật toán tách

riêng các hàng trăm, chục, đơn vị của giá trị nhiệt độ để hiển thị trên từng led 7 thanh. Ta dùng đoạn code sau:

```
conv=tempC;              //Dùng cho thang Celsius
 ht = conv/100;
 h[0] = (conv%100)/10;
 h[1] = ((conv%100)%10)/1;
 h[2] = 10;
 h[3]=  11;
  if(ht>0)
   {
    h[2]=h[1];
    h[1]=h[0];
    h[0]=ht;
   }
```

Để đưa ra cảnh báo giới hạn trên và dưới, ta sử dụng 2 biến kiểu integer: *uptemp* và *downtemp*. Nếu nhiệt độ đo được lớn hơn giới hạn trên hoặc nhỏ hơn giới hạn dưới cho trước thì đèn led tương ứng sẽ nhấp nháy và giá trị hiển thị nhiệt độ trên led sẽ nhấp nháy theo cũng do hiệu ứng từ phương pháp quét led:

```
if(tempC>=uptemp)                  //Cảnh báo giới hạn trên
  {
   digitalWrite(ledpin1, HIGH);
   delay(200);
   digitalWrite(ledpin1, LOW);
   delay(200);
  }
```

```
if(tempC<=downtemp)          //Cảnh báo giới hạn dưới
{
 digitalWrite(ledpin2, HIGH);
 delay(200);
 digitalWrite(ledpin2, LOW);
 delay(200);
}
```

Để đưa giá trị nhiệt độ đã xử lý từ tín hiệu cảm biến ra 4 led 7 thanh, ta dùng đoạn chương trình sau. Hàm *shiftOut()* dùng để đẩy dữ liệu từ Arduino qua IC ghi dịch 74HC595 tới hiển thị trên 4 led 7 thanh:

```
 for(vt=0;vt<4;vt++)
{
  digitalWrite(latchPin,LOW);
  shiftOut(dataPin,clockPin,LSBFIRST,digit[h[vt]]);
  digitalWrite(latchPin,HIGH);
  digitalWrite(digitPins[vt],LOW);
  delay(1);
  digitalWrite(digitPins[vt],HIGH);
}
```

Do cảm biến tiến hành đo giá trị nhiệt độ từ điểm cần đo liên tục nên để cho giá trị nhiệt độ hiển thị lên 4 led 7 thanh không bị thay đổi quá nhanh dẫn đến số hiển thị bị nháy, ta sẽ để cả đoạn chương trình từ đọc giá trị từ cảm biến trong 1 vòng if() và sử dụng 1 biến đếm *k*. Khi cảm biến đo giá trị nhiệt độ *k=125* lần thì mới hiển thị lên led 1 lần. Thời gian từ khi cảm biến đọc giá trị đến khi hiển thị lên led cách nhau khoảng 2s.

2.2.2. Lập trình cho mạch đo nhiệt độ có truyền phát không dây

a. Các thư viện sử dụng:

Thư viện *nRF24L01p.h* gồm các hàm giúp điều khiển được module thu phát được download từ đường link Mediafire được chia sẻ trên kênh Youtube của Jorge Arturo Prado Aparcana. Trên kênh youtube cùng tên, Jorge Arturo Prado Aparcana cũng đã thực hiện rất nhiều video hướng dẫn sử dụng thư viện này. Đây là thư viện mới ở dạng beta và đặc biệt rất dễ sử dụng với người mới bắt đầu làm quen với module thu phát nRF24L01.

Module thu phát không dây nRF24L01 hoạt động theo giao thức SPI, SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hang Motorola đề xuất. Đây là kiểu truyền thông Master-Slave, trong đó có 1 chip Master điều phối quá trình truyền thông và các chip Slaves được điều khiển bởi Master vì thế truyền thông chỉ xảy ra giữa Master và Slave. SPI là một cách truyền song công (full duplex) nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời. SPI đôi khi được gọi là chuẩn truyền thông "4 dây" vì có 4 đường giao tiếp trong chuẩn này đó là SCK (Serial Clock), MISO (Master Input Slave Output), MOSI (Master Ouput Slave Input) và SS (Slave Select). Vì vậy ta cần khai báo để sử dụng thêm thư viện *SPI.h*. Thư viện được cung cấp sẵn trong gói phần mềm Arduino IDE hoặc có thể download dễ dàng từ trang http://arduino.cc/ .

b. Vấn đề lập trình truyền phát không dây với nRF24L01

Phần truyền phát không dây sử dụng module nRF24L01 được coi như một chức năng mở rộng của mạch đo nhiệt độ hiển thị trên led 7 thanh đã giới thiệu ở phần trước. Đối với phần này, ta chỉ cần sửa và bổ sung thêm 1 số phần code vào bài code cho mạch đo nhiệt độ ở trên.

+ Lập trình cho bộ đo và truyền tín hiệu nhiệt độ:

Sau phần khai báo thêm thư viện hỗ trợ, các chân giao tiếp theo chuẩn SPI và các biến, ta thiết đặt các thông số mở đầu cho module nRF24L01 và thiết đặt giao tiếp với máy tính qua cổng COM. Module hoạt động có 126 kênh truyền phát khác nhau, ở trong phần lập trình này, nhóm sử dụng kênh 90:

delay(150);

Serial.begin(115200);

SPI.begin();

SPI.setBitOrder(MSBFIRST);

transmitter.channel(90);

transmitter.TXaddress("Artur");

transmitter.init();

Serial.println("Truyen nhiet do su dung module nRF24L01p:");

Serial.println("----------------");

Để truyền giá trị nhiệt độ cho mạch nhận và hiển thị, ta dùng hàm *transmitter.txPL()* và *transmitter.send()* (transmitter là biến do người lập trình đặt dùng để gọi hàm):

transmitter.txPL(temp);

transmitter.txPL(tempF)

transmitter.send(SLOW);

Các phần tách các chữ số của nhiệt độ, hiển thị ra led 7 thanh,… tương tự như phần code mạch đo nhiệt độ không truyền phát đã giới thiệu ở trên.

+ Lập trình cho bộ nhận và hiển thị tín hiệu nhiệt độ:

Phần khai báo và thiết lập thông số ban đầu cho module nhận tín hiệu được làm tương tự như mạch phát. Ta cũng chọn kênh truyền phát giống như bộ phát là kênh 90. Ở đây, biến dùng để gọi hàm ta khai báo sẽ là *receiver*. Sau đó là phần đặt hiển thị ban đầu trên màn hình giao tiếp với máy tính:

delay(150);

Serial.begin(115200);

SPI.begin();

SPI.setBitOrder(MSBFIRST);

```
receiver.channel(90);
receiver.RXaddress("Artur");
receiver.init();
Serial.print("Led nhap nhay canh bao khi T* > ");
Serial.print(Wtemp);
Serial.println(" oc");
Serial.println("---------------");
Serial.println("Dang nhan du lieu...");
Serial.println("---------------");
```

Để nhận giá trị và hiển thị giá trị nhiệt độ nhận được lên màn hình máy tính, ta dùng các hàm *receiver.available(), receiver.read(), receiver.rxPL()* và *Serial.print()*:

```
if(receiver.available())                    //nếu sẵn sàng nhận giá trị thì khởi chạy hàm nhận
{
 {
   receiver.read();                          // đọc giá trị nhận
   receiver.rxPL(temp);                      //nhận giá trị
   receiver.rxPL(tempF);
   Serial.println("Nhiet do da nhan:  ");   //hiển thị lên Serial Monitor
   Serial.print(temp);
   Serial.println(" oC");
   Serial.print(tempF);
   Serial.println(" oF");
   Serial.println("--------");
 }
```

Mạch được lập trình để màn led 7 thanh hiển thị nhấp nháy cảnh báo nhiệt độ ở một ngưỡng bằng cách thay đổi thời gian trễ của quá trình quét led. Nhiệt độ ngưỡng cảnh báo được gán cho biến *Wtemp* và thời gian trễ được gán vào biến *time*. Hàm *delay()* dùng để tạo trễ được đặt ở trong phần hiển thị giá trị ra led 7 thanh.

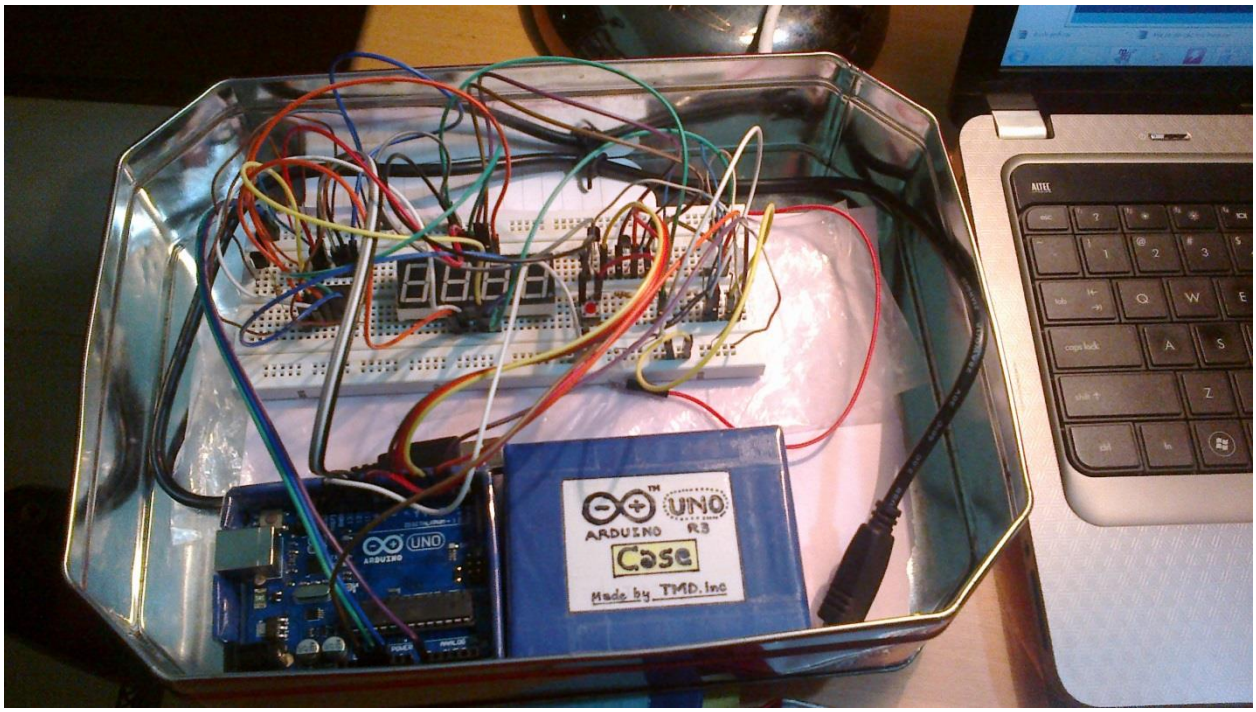
```
//Canh bao gioi han:
    time=1;
    if(temp>Wtemp)
    {
      time=250;
    }
 //Hiển thị độ ra led:
     for(vt=0;vt<4;vt++)
       {
         digitalWrite(latchPin,LOW);
         shiftOut(dataPin,clockPin,LSBFIRST,digit[h[vt]]);
         digitalWrite(latchPin,HIGH);
         digitalWrite(digitPins[vt],LOW;
         delay(time);
         digitalWrite(digitPins[vt],HIGH);
       }
```

2.3. Lắp đặt mạch đo nhiệt độ và thử nghiệm trên test board

2.3.1. Lắp đặt và thử nghiệm mạch đo nhiệt độ không truyền phát

Khi thực hiện lắp đặt linh kiện trên test board cần chú ý đi dây và sắp xếp vị trí các thiết bị sao cho hợp lí tránh tình trạng vướng víu, khi thực hiện nối chân các linh kiện cần

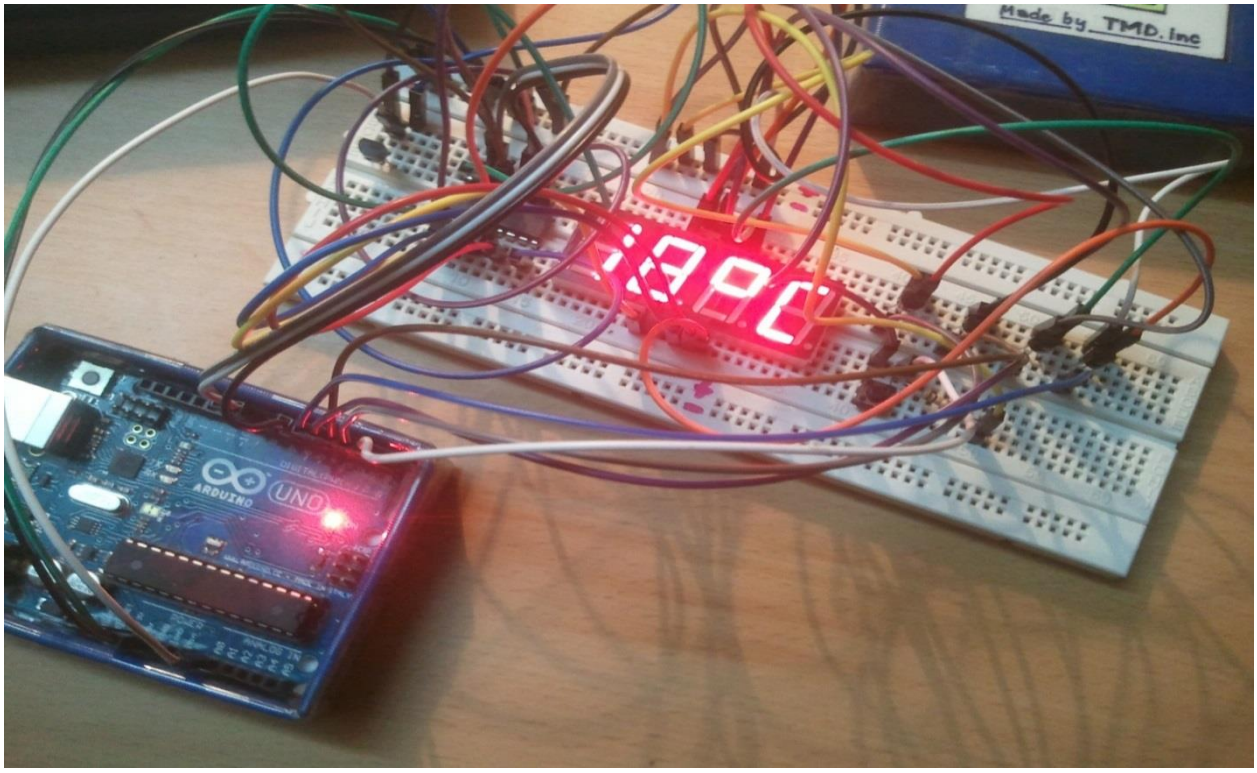
tham khảo từ datasheet chuẩn của mỗi linh kiện tránh nhầm lẫn khiến mạch không chạy được hoặc gây ngắn mạch làm hỏng linh kiện.



Hình 2.4. Mạch đo nhiệt độ không truyền phát lắp đặt trên test board.

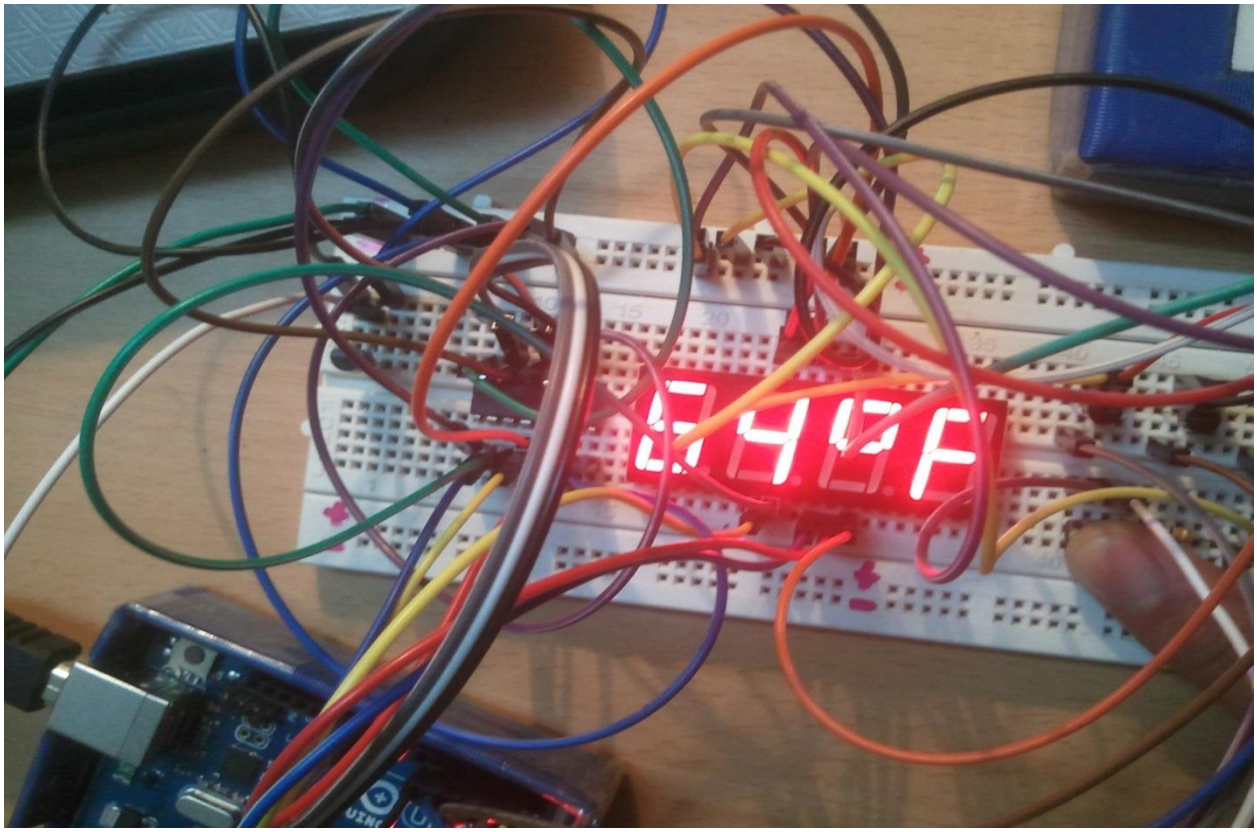
Trong quá trình lắp đặt mạch, nhóm cũng đã gặp nhiều những trục trặc nhỏ vì chất lượng linh kiện hay lỗi khi đi dây…, tuy nhiên những vấn đề đó đã được khắc phục và mạch đã hoạt động được, các chức năng hiển thị và cảnh báo hoạt động tốt. Mạch được lắp đặt chính xác theo sơ đồ nguyên lý đã đề cập ở đầu chương 2.

+ Chế độ hiển thị nhiệt độ thang Celsius (^oC):



Hình 2.5. Chế độ hiển thị nhiệt độ thang Celsius (^oC) trên mạch đo nhiệt độ.

+ Chức năng hiển thị nhiệt độ thang Fahrenheit (^oF):



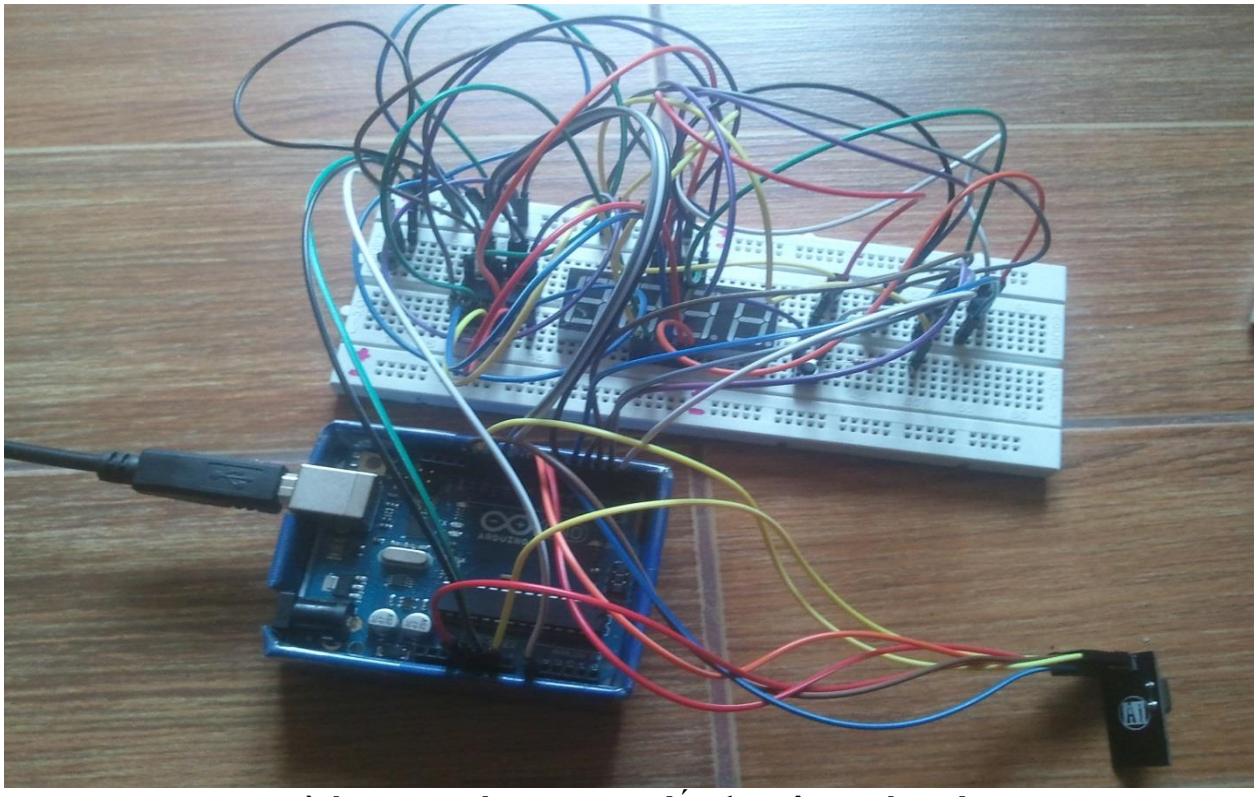
Hình 2.6. Hiển thị nhiệt độ thang Fahrenheit (^{o}F) trên mạch đo nhiệt độ.

2.3.2. Lắp đặt và thử nghiệm mạch đo nhiệt độ có truyền phát với nRF24L01

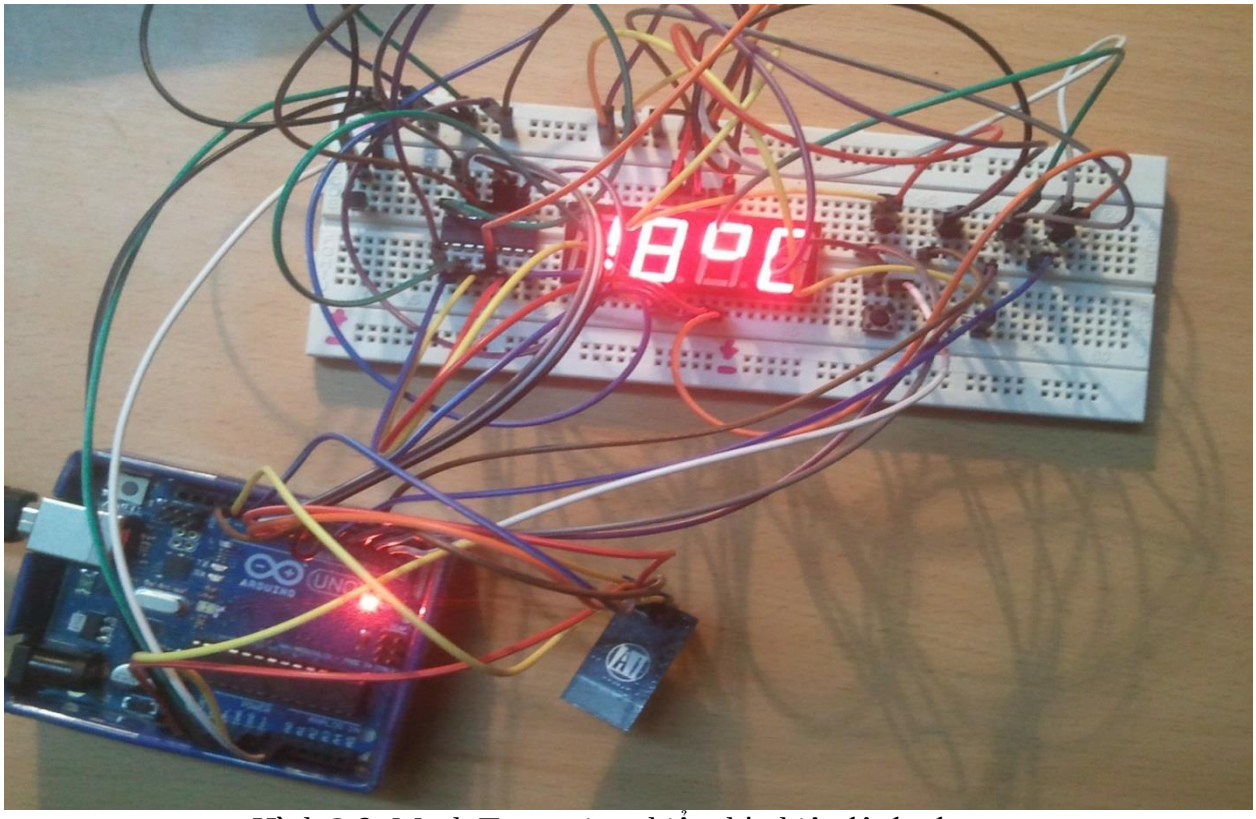
a. Lắp đặt mạch truyền (Transmitter) và mạch nhận (Receiver)

+ Lắp đặt mạch đo và truyền nhiệt độ (Transmitter) :

Mạch Transmitter được lắp đặt sử dụng Arduino Uno. Sơ đồ kết nối tương tự như mạch đo nhiệt độ không truyền phát với một số lưu ý về thay đổi chân cắm của 1 số linh kiện cho phù hợp với điều kiện của Arduino Uno khi kết nối thêm với module nRF24L01. Các thay đổi về chân cắm được thể hiện trong *Bảng 2.1* và sơ đồ kết nối chân Arduino và module nRF24L01 được thể hiện trong *Bảng 1.1.*



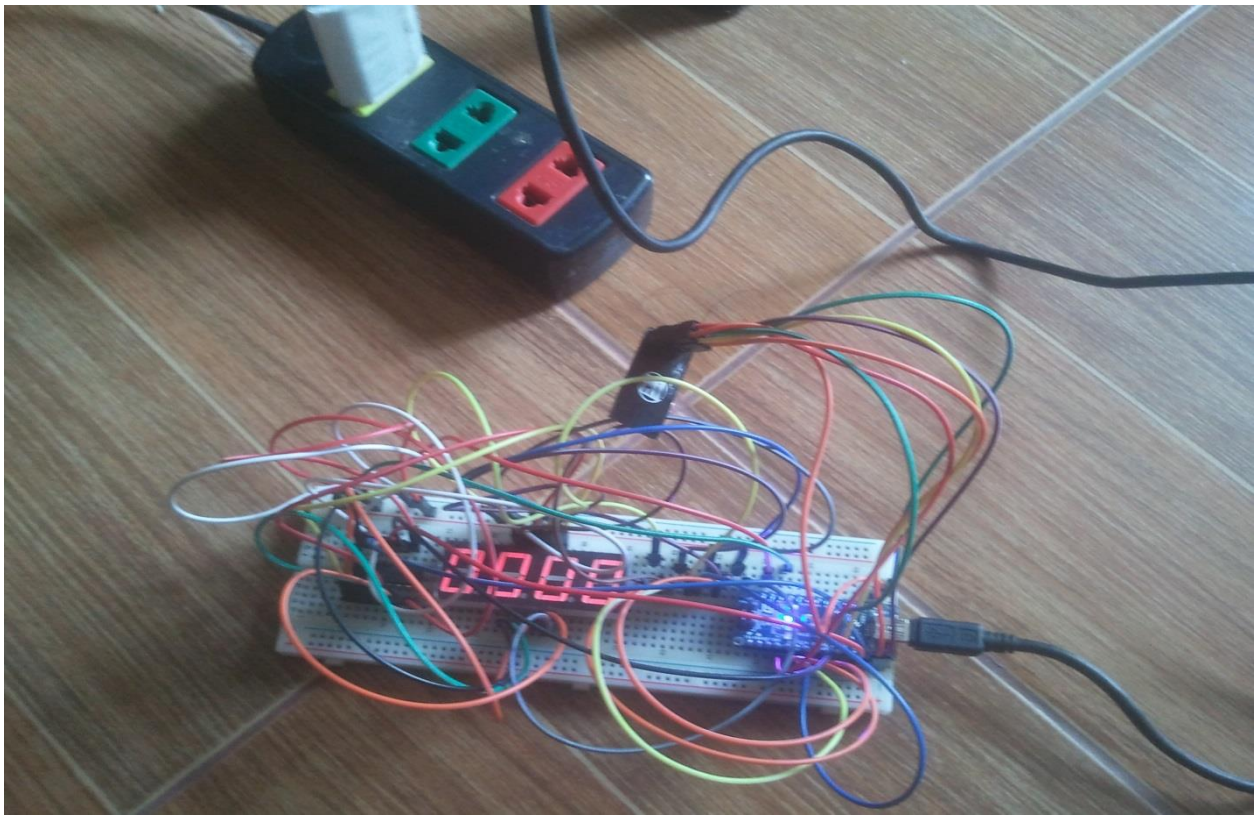
Hình 2.7. Mạch transmitter lắp đặt trên test board.



Hình 2.8. Mạch Transmitter hiển thị nhiệt độ đo được.

Mạch Transmitter khi hoạt động sẽ đo nhiệt độ bằng cảm biến LM35 sau đó gửi tín hiệu nhiệt độ sang mạch nhận Receiver, đồng thời hiển thị giá trị nhiệt độ vừa đo lên màn hình 4 led 7 thanh và cũng sẽ hiển thị giá trị nhiệt độ này lên màn hình máy tính thông qua chế độ Serial Monitor của phần mềm Arduino IDE.

+ Lắp đặt mạch nhận và hiển thị nhiệt độ (Receiver):



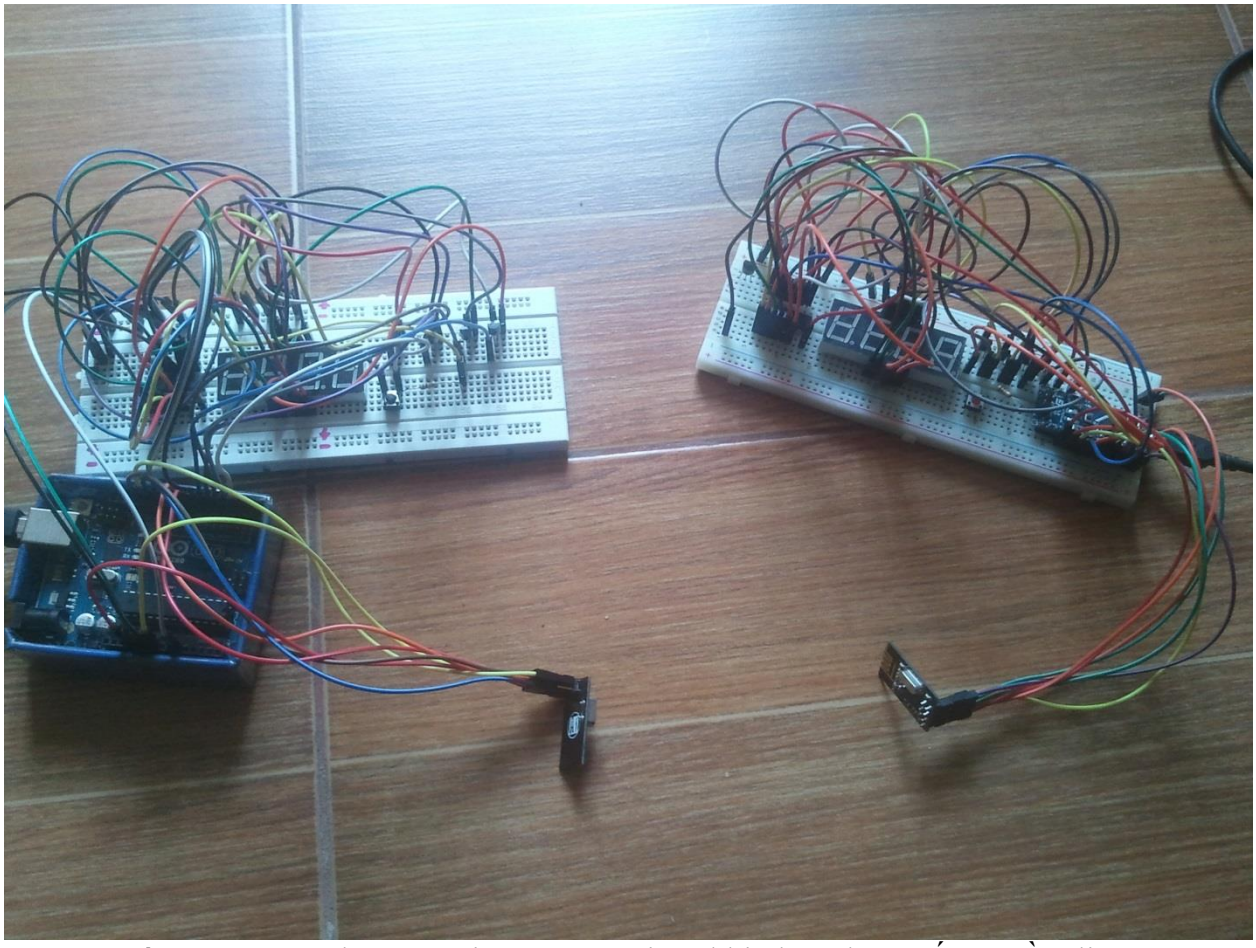
Hình 2.9. Mạch Receiver lắp đặt trên test board sau khi được cấp nguồn điện.

Khi chưa cấp nguồn điện cho mạch Transmitter, mạch Receiver nếu được cấp nguồn sẽ hiển thị 4 chữ số 0 trên màn 4 led 7 thanh.

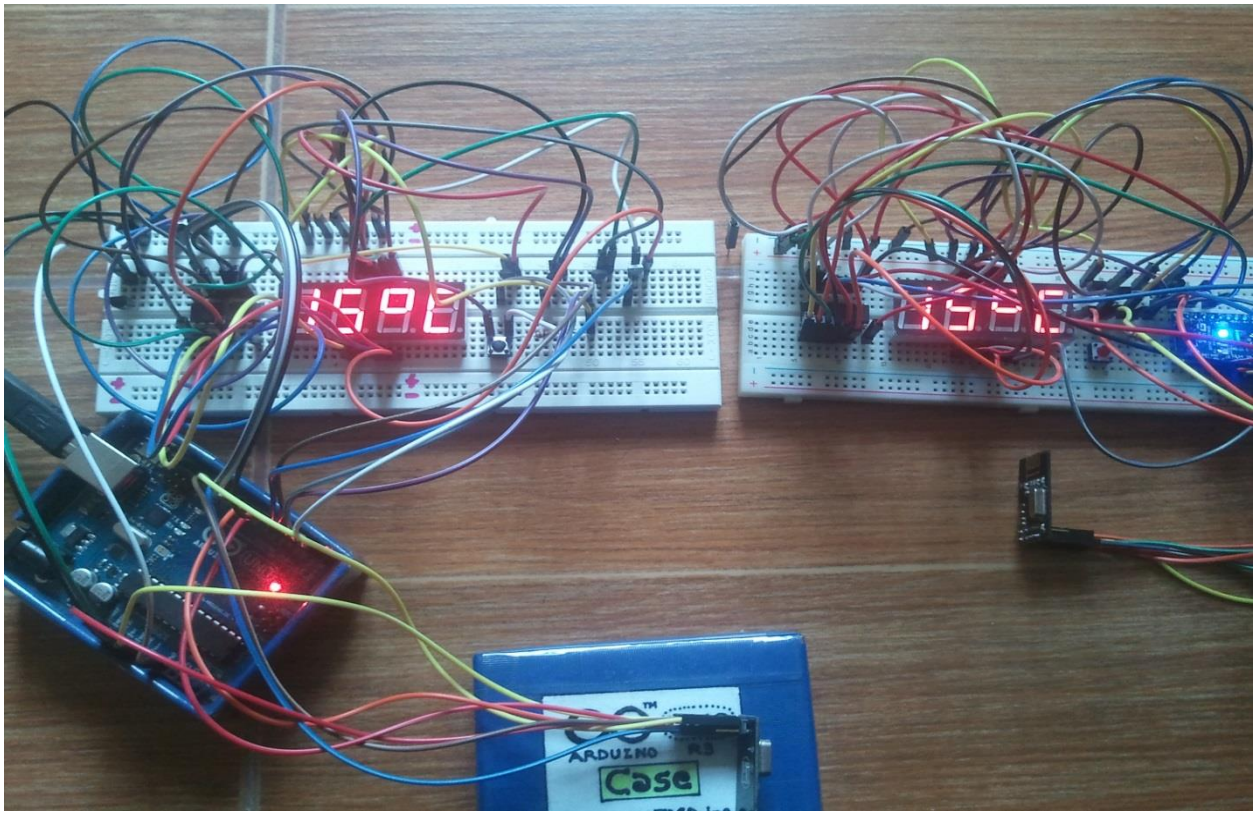
b. Quá trình thử nghiệm

Sau khi kết nối đầy đủ nguồn điện cho mạch phát và mạch thu, giá trị nhiệt độ đã được đo, truyền và hiển thị tốt trên cả 2 mạch. Độ trễ truyền tín hiệu nhiệt độ rất nhỏ, nhiệt độ thay đổi hiển thị trên 4 led 7 thanh mạch Receiver gần như ngay lập tức sau khi

giá trị nhiệt độ đo được và hiển thị trên mạch Transmitter thay đổi. Các chức năng hiển thị và cảnh báo đều đã hoạt động đúng theo mong muốn.

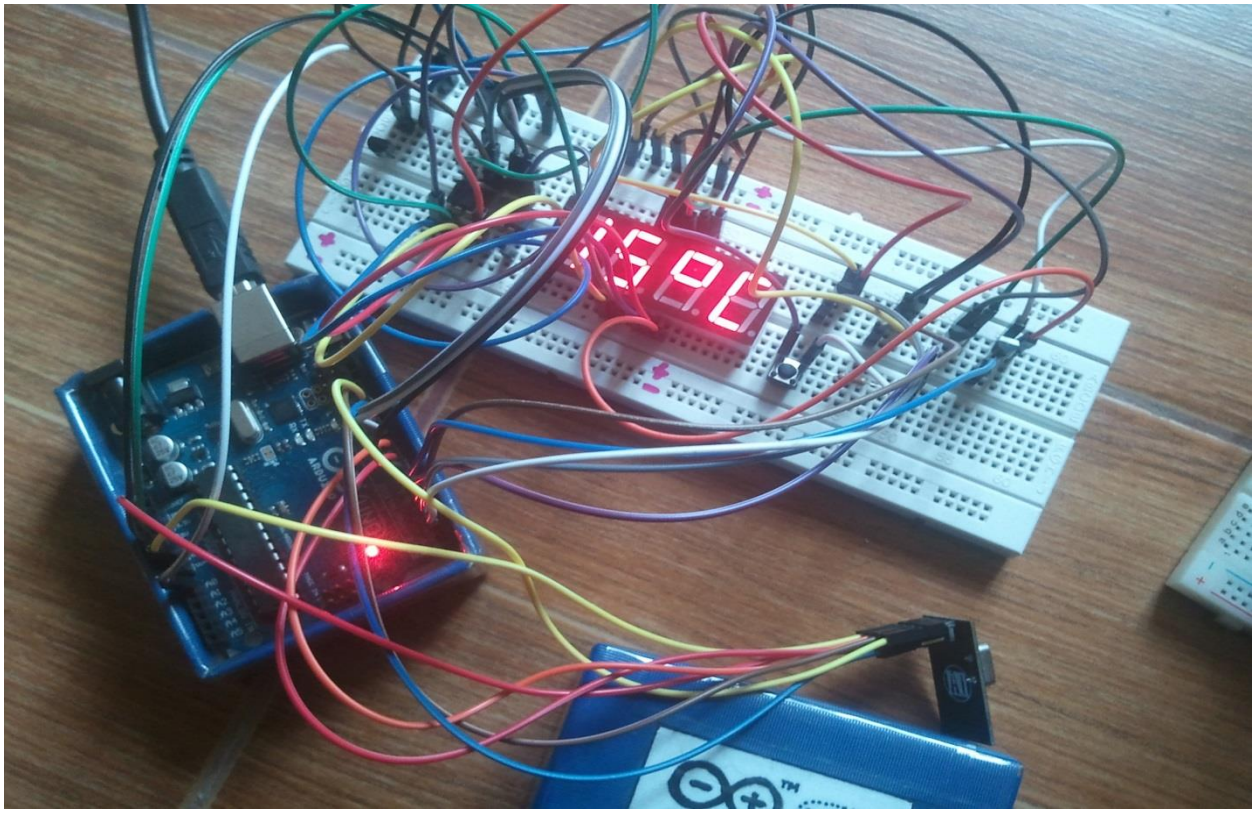


Hình 2.10. Mạch Transmitter và Receiver khi chưa được cấp nguồn điện.

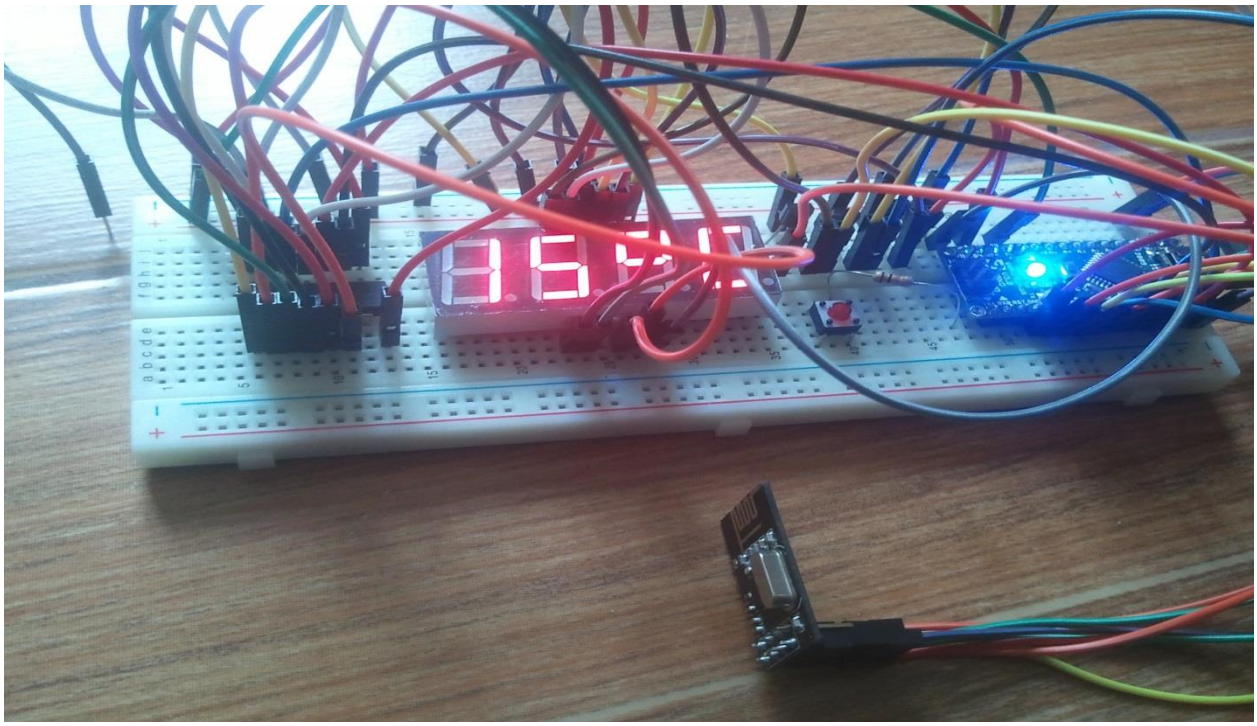


Hình 2.11. Hoạt động của 2 mạch Transmitter và Receiver trong quá trình thử nghiệm.

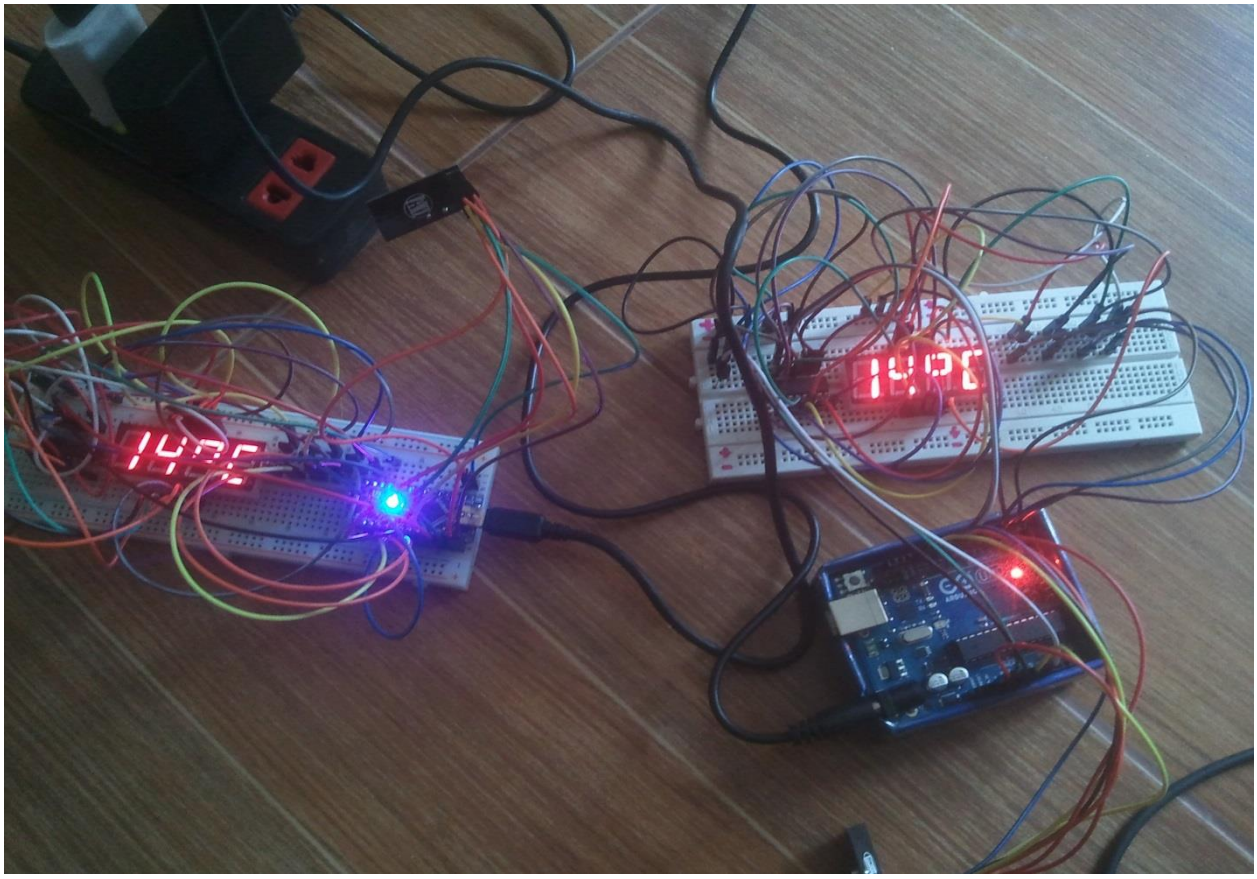
Tại thời điểm thử nghiệm, nhiệt độ trong phòng đo được trên nhiệt kế là 15^{o}C, mạch đo nhiệt độ đã đo được đúng giá trị nhiệt độ trong phòng đồng thời mạch đã truyền được giá trị nhiệt độ này chính xác sang mạch Receiver để hiển thị giá trị 15^{o}C.



Hình 2.12. Hoạt động của mạch Transmitter.



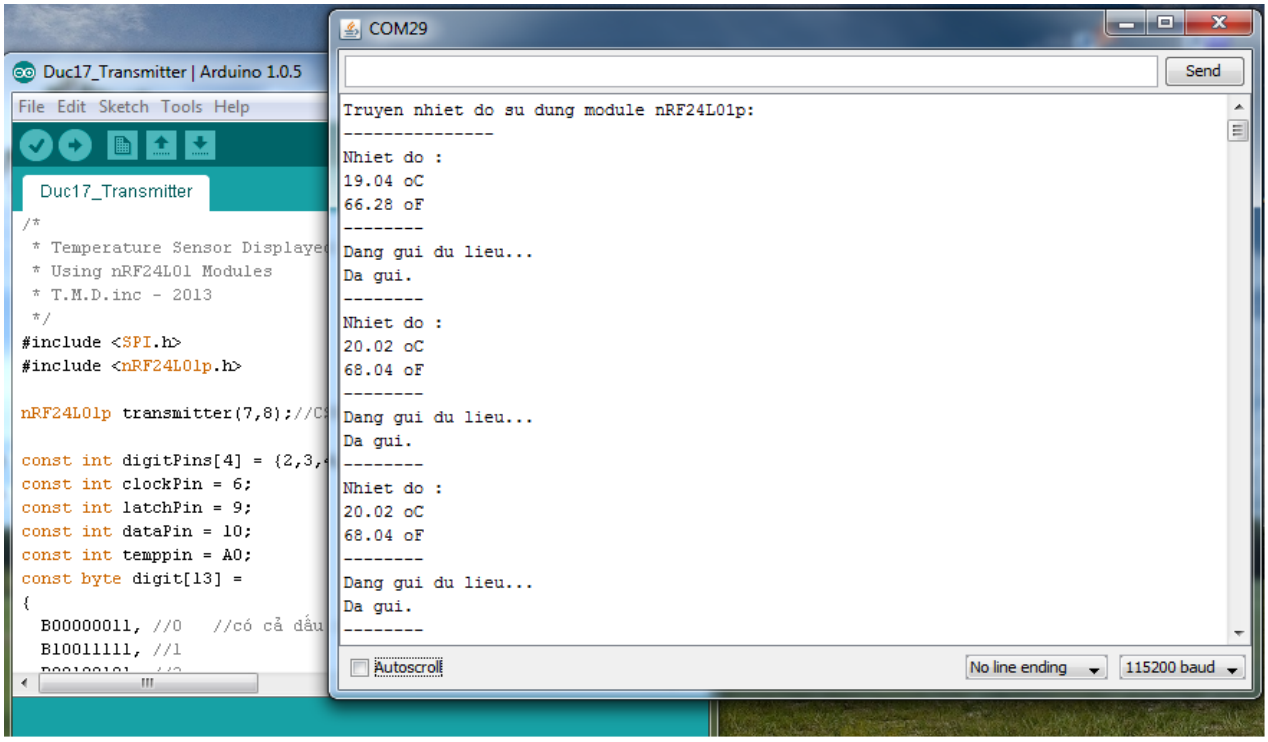
Hình 2.13. Hoạt động của mạch Receiver.



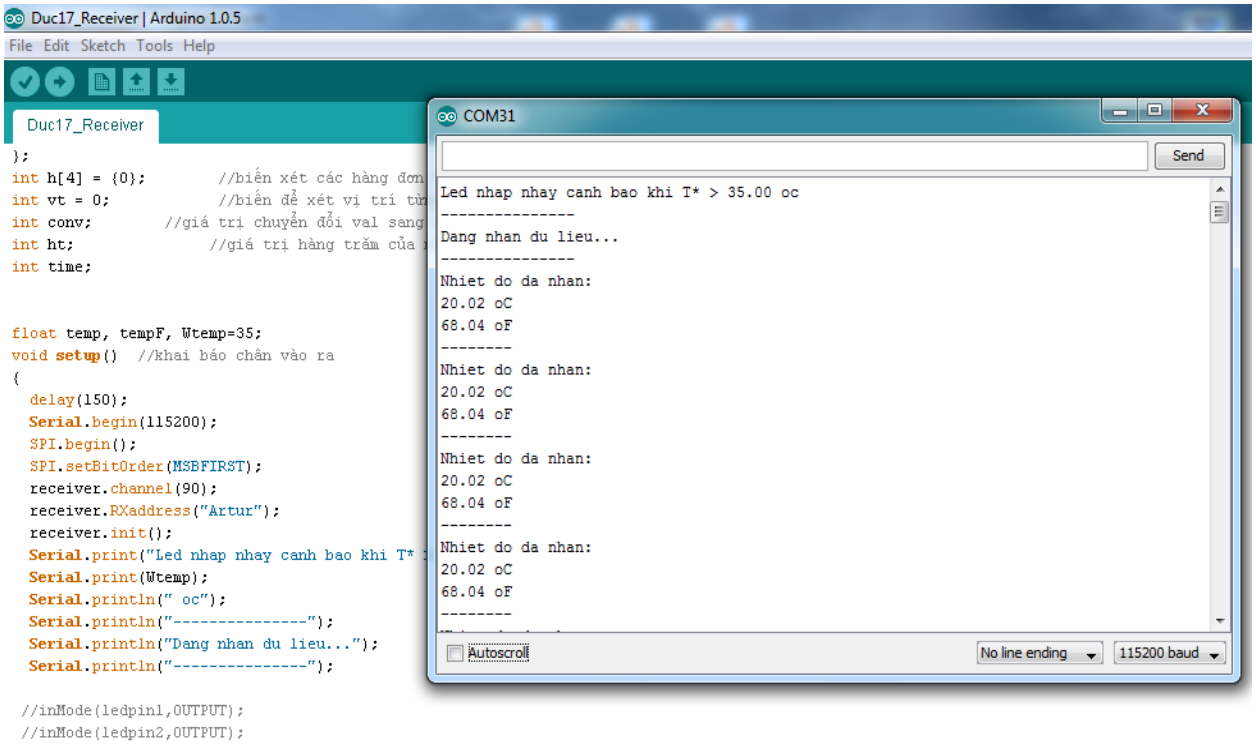
Hình 2.14. Toàn cảnh quá trình đo, truyền - phát, hiển thị nhiệt độ
của mạch Transmitter và Receiver.

+ Chức năng hiển thị trên màn hình máy tính:

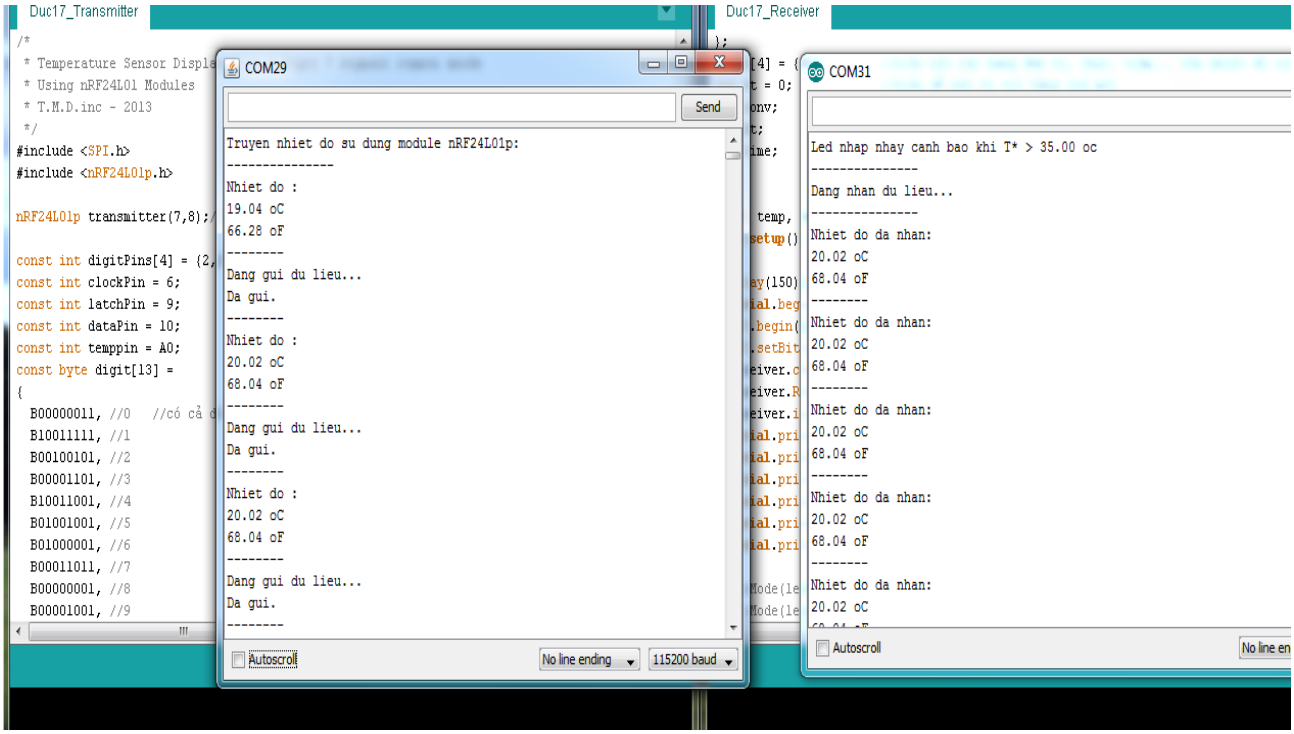
Arduino Uno, Arduino Nano được kết nối với máy tính qua cổng USB và Arduino Uno được máy tính xác định giao tiếp qua cổng COM29, Arduino Nano qua cổng COM31. Giao diện hiển thị giao tiếp của Arduino Uno và Arduino Nano với máy tính qua Serial Monitor sẽ hiển thị đủ giá trị độ thang ^{o}C và thang ^{o}F cùng với thông báo đã gửi thành công… hay thông báo đã nhận được dữ liệu.



Hình 2.15. Giao diện hiển thị của mạch Transmitter qua chức năng Serial Monitor của Aruino IDE.



Hình 2.16. Giao diện hiển thị của mạch Receiver
qua chức năng Serial Monitor của Aruino IDE.



Hình 2.17. Giao diện hiển thị trên máy tính của cả mạch Transmitter và mạch Receiver.

Nhóm đã thử nghiệm sử dụng mạch đo và truyền phát nhiệt độ trong giới hạn nhà 3 tầng: mạch đo-phát nhiệt độ đặt ở một phòng trên tầng ba, mạch thu-hiển thị nhiệt độ đặt ở tầng một. Kết quả, nhiệt độ vẫn được truyền và hiển thị trong điều kiện không gian rộng vừa phải, có tường gạch, bê tông cản trở.

2.4. Chi phí thực hiện đề tài

Bảng 2.2. Chi phí thực hiện đề tài đồ án 1.

STT	Tên linh kiện	Số lượng	Đơn giá (VNĐ)	Thành tiền (VNĐ)
1	Arduino Uno	1	280 000	280 000
2	Arduino Nano	1	240 000	240 000
3	Module nRF24L01	2	65 000	130 000
4	Cảm biến LM35	1	20 000	20 000
5	Board test	2	35 000	70 000
6	IC 74HC595	2	3000	6000
7	Led 7 thanh 4 số (0.56mm)	2	9000	18 000
8	Led cảnh báo	2	350	700
9	Công tắc 2 cực	2	400	800
10	Điện trở 10K (Gói 100 con)	1	2000	2000
12	Transistor A1015 (PNP)	8	400	3200
13	Dây cắm (jack đực - đực) Gói 40 dây	2	35 000	70 000
14	Dây cắm (jack đực - cái) Gói 40 dây	1	35 000	35 000
Tổng chi phí (VNĐ)				889 200

Chương 3

TỔNG KẾT

Như vậy, với đề tài đồ án 1: **Thiết kế mạch đo nhiệt độ sử dụng board Arduino hiển thị trên 4 led 7 thanh và truyền phát không dây sử dụng module nRF24L01,** nhóm thực hiện đã thiết kế được 1 mạch đo nhiệt độ có các chức năng:

- Đo nhiệt độ thang Celsius (^oC).
- Có công tắc hiển thị tham khảo nhiệt độ thang Fahrenheit (^oF) đối với mạch đo nhiệt độ không truyền phát.
- Có đèn led cảnh báo nhiệt độ giới hạn ngưỡng trên và dưới đối với đối với mạch đo nhiệt độ không truyền phát.
- Cảnh báo giới hạn nhiệt độ một ngưỡng bằng nhấp nháy led 7 thanh đối với mạch đo nhiệt độ có truyền phát.

Tuy nhiên do giá trị nhiệt độ hiển thị ra chỉ hiển thị trên 4 led 7 thanh cùng với board Arduino Uno và Nano chỉ có 2 chân ngắt mỗi board nên rất khó khăn trong việc hiển thị và điều chỉnh giá trị nhiệt độ giới hạn ngưỡng trên và dưới nên nhóm đã chấp nhận phương án điều chỉnh giá trị nhiệt độ giới hạn trong code lập trình và không hiển thị nhiệt độ giới hạn đó ra led.

Đối với chức năng thu phát và hiển thị tín hiệu nhiệt độ, mạch có khả năng đo và truyền tín hiệu nhiệt độ từ 1 mạch đo nhiệt độ sử dụng Arduino Uno sang 1 mạch hiển thị nhiệt độ sử dụng Arduino Nano. Do hạn chế về số chân của Arduino và ảnh hưởng của 1 lỗi phát sinh chưa tìm được nguyên nhân của chân ra số D0(RX) và D1(TX) của Arduino Nano nên nhóm chưa thể phát triển được chế độ cảnh báo nhiệt độ trên mạch hiển thị sử dụng Arduino Nano.

Với sự lỗ lực trong việc tìm hiểu nghiên cứu các linh kiện, thiết bị cần thiết cho đề tài, cùng với vận dụng các kiến thức đã học vào công việc thiết kế, lắp đặt, và sự giúp đỡ, hướng dẫn nhiệt tình của thầy giáo TS. Nguyễn Hoàng Nam, nhóm 1 đã hoàn thành được

đề tài đồ án 1: **Thiết kế mạch đo nhiệt độ sử dụng board Arduino, hiển thị trên 4 led 7 thanh và truyền phát không dây sử dụng module nRF24L01**. Trong quá trình thực hiện, lập trình cho mạch đo nhiệt độ, nhóm gặp phải nhiều khó khăn khác nhau như: do phải nghiên cứu nhiều tài liệu nước ngoài, datasheets,… dẫn đến nhiều chỗ dịch sai, dịch nhầm dẫn đến áp dụng các hàm, câu lệnh bị sai ý nghĩa, cấu trúc…, trong quá trình viết code gặp phải nhiều lỗi phát sinh mà không tìm ngay ra nguyên nhân cần đầu tư thời gian để giải quyết, nhiều linh kiện rất khó để tìm được thư viện chuẩn để lập trình… Quá trình lắp mạch cũng gặp phải những khó khăn nhất định tuy nhiên nhóm đã cố gắng giải quyết được vấn đề phát sinh để hoàn thành được đề tài. Nhóm đã hoàn thành thiết kế, lập trình và lắp đặt mạch đo nhiệt độ sử dụng Arduino không truyền phát trong vòng 1 tuần kể từ khi nhận đề tài và sau đó nghiên cứu trong 3 tuần tiếp theo để thực hiện được chức năng truyền phát tín hiệu nhiệt độ giữa 2 board Arduino.

Do đây mới là lần đầu tiên những thành viên trong nhóm làm một đề tài đồ án, cộng với kiến thức còn nhiều hạn chế, chúng em tự thấy đề tài của mình thực hiện được vẫn còn rất nhiều sai sót, khiếm khuyết. Chúng em rất mong được sự ủng hộ và giúp đỡ của thầy giáo để đề tài chúng em thực hiện được hoàn thiện hơn và có thêm nhiều cải tiến đáng kể và ứng dụng tốt hơn vào thực tiễn.

Hà Nội, ngày 29 tháng 11 năm 2013

Sinh viên thực hiện

TÀI LIỆU THAM KHẢO

[1] Massimo Banzi, *Getting Started with Arduino*, O'Reilly Media, Inc, 2009.

[2] Michael Margollis and Nicholas Weldin, *Arduino Cookbook*, O'Reilly Media, Inc, 2011.

[3] http://arduino.cc/, truy nhập cuối cùng ngày 27/11/2013.

[4] http://arduino4projects.com/, truy nhập cuối cùng ngày 6/10/2013.

[5] http://randomnerdtutorials.com/, truy nhập cuối cùng ngày 6/10/2013.

[6] http://techshowvn.com/, truy nhập cuối cùng ngày 6/10/2013.

[7] http://www.airspayce.com/mikem/arduino/RF22/, truy nhập cuối cùng ngày 27/11/2013

[8] http://groups.google.com/group/rf22-arduino/, truy nhập cuối cùng ngày 26/11/2013

[9] http://electrodragon.com/, truy nhập cuối cùng ngày 19/11//2013.

[10] http://blogembarcado.blogspot.de/, truy nhập cuối cùng ngày 29/11/2013.

[11] http://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo/, truy nhập cuối cùng ngày 20/12/2013.

[12] http://www.youtube.com/channel/UCGSloFkUnaUknE-Z21gmmvw?feature=watch/, truy nhập cuối cùng ngày 20/12/2013.

[13] http://www.mediafire.com/download/v6bn0a7g3ep3y7o/nRF24L01p.rar/, truy nhập cuối cùng ngày 20/12/2013.

PHỤ LỤC

1. Code cho mạch đo nhiệt độ không truyền phát có cảnh báo nhiệt độ giới hạn ngưỡng trên và dưới.

```
const int digitPins[4] = {4,5,6,7};
const int clockPin = 11;
const int latchPin = 12;
const int dataPin = 13;
const int tempPin = A0;
const int ledpin1 = 3;
const int ledpin2 = 2;
const int buttonpin = 10;
const byte digit[13] =
{
  B00000011, //0
  B10011111, //1
  B00100101, //2
  B00001101, //3
  B10011001, //4
  B01001001, //5
  B01000001, //6
  B00011011, //7
  B00000001, //8
  B00001001, //9
  B00111001, //o
```

```
  B01100011, //C
  B01110001, //F
};
int h[4] = {0};
int h2[4] = {0};
int vt = 0;
int conv,conv1;
float val;
int ht;
int state =0;
int kt=125;

float tempC, tempF, uptemp= 50, downtemp=17;

void setup()
{
  pinMode(ledpin1,OUTPUT);
  pinMode(ledpin2,OUTPUT);
  pinMode(10,INPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(tempPin, INPUT);
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
```

```
  pinMode(dataPin, OUTPUT);
}



 void loop()
 {
   for(vt=0;vt<4;vt++)
   {
   digitalWrite(digitPins[vt],HIGH);
   }
   if(kt==125)                    //Bien toi uu, cu 125 lan do thi moi hien thi 1 lan
    {
      val=analogRead(tempPin);         //Doc gia tri tu cam bien
      tempC = (val*0.48828125);
      tempF = (tempC*1.8+32);
      kt=0;


      state = digitalRead(buttonpin);   //Xu ly gia tri oC
      if(state == LOW)
       {
         conv=tempC;                      //Tach tung chu so gia tri de dua ra led
      ht = conv/100;
      h[0] = (conv%100)/10;
      h[1] = ((conv%100)%10)/1;
      h[2] = 10;
      h[3]=  11;
```

```
  if(ht>0)                    //Hien thi so hang tram
    {
      h[2]=h[1];
      h[1]=h[0];
      h[0]=ht;
    }
 }

else
if(state == HIGH)                    // Xu ly gia tri oF
{
conv=tempF;
ht = conv/100;
h[0] = (conv%100)/10;
h[1] = ((conv%100)%10)/1;
h[2] = 10;
h[3]=  12;

if(ht>0)
{
  h[2]=h[1];
  h[1]=h[0];
  h[0]=ht;
}
}
```

```
if(tempC>=uptemp)               // Canh bao gioi han tren
{
  digitalWrite(ledpin1, HIGH);
  delay(200);
  digitalWrite(ledpin1, LOW);
  delay(200);
}

if(tempC<=downtemp)             //Canh bao gioi han duoi
{
  digitalWrite(ledpin2, HIGH);
  delay(200);
  digitalWrite(ledpin2, LOW);
  delay(200);
}
}
kt++;

for(vt=0;vt<4;vt++)             //Dua gia tri ra led
    {
      digitalWrite(latchPin,LOW);
      shiftOut(dataPin,clockPin,LSBFIRST,digit[h[vt]]);
      digitalWrite(latchPin,HIGH);
      digitalWrite(digitPins[vt],LOW);
      delay(1);
```

```
        digitalWrite(digitPins[vt],HIGH);
      }
 }
```

2. Code cho mạch đo nhiệt độ truyền phát sử dụng module nRF24L01

a. Code cho mạch đo và truyền tín hiệu nhiệt độ (Transmitter):

```
#include <SPI.h>
#include <nRF24L01p.h>

nRF24L01p transmitter(7,8);    // Khai bao chan CSN, CE

const int digitPins[4] = {2,3,4,5};
const int clockPin = 6;
const int latchPin = 9;
const int dataPin = 10;
const int temppin = A0;
const byte digit[13] =
{
  B00000011, //0
  B10011111, //1
  B00100101, //2
  B00001101, //3
  B10011001, //4
  B01001001, //5
  B01000001, //6
  B00011011, //7
```

```
  B00000001, //8
  B00001001, //9
  B00111001, //o
  B01100011, //C
  B01110001, //F
};

int h[4] = {0};
int vt = 0;
int conv;
int ht;
int kt=125;
float val, temp, tempF;

void setup()
{
  delay(150);
  Serial.begin(115200);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  transmitter.channel(90);
  transmitter.TXaddress("Artur");
  transmitter.init();
  Serial.println("Truyen nhiet do su dung module nRF24L01p:");
  Serial.println("----------------");
```

```
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(temppin, INPUT);
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

 void loop()
 {
   for(vt=0;vt<4;vt++)
   digitalWrite(digitPins[vt],HIGH);

   if(kt==125)
    {
     {
      val=analogRead(temppin);            // Doc gia tri tu cam bien
      temp = (val*0.48828125);
      tempF = (temp*1.8+32);
      kt=0;
     }
      Serial.println("Nhiet do:  ");
      Serial.print(temp);
      Serial.println(" oC");
```

```
  Serial.print(tempF);
  Serial.println(" oF");
  Serial.println("--------");
  Serial.println("Dang gui du lieu...");
{
transmitter.txPL(temp);              //Gui du lieu
transmitter.txPL(tempF);
transmitter.send(SLOW);
Serial.println("Da gui.");
Serial.println("--------");
}
  conv=temp;
  ht = conv/100;
  h[0] = (conv%100)/10;
  h[1] = ((conv%100)%10)/1;
  h[2] = 10;
  h[3]=  11;

    if(ht>0)
      {
        h[2]=h[1];
        h[1]=h[0];
        h[0]=ht;
      }
}
kt++;
```

```
   for(vt=0;vt<4;vt++)                // Xuat ra led
       {
         digitalWrite(latchPin,LOW);
         shiftOut(dataPin,clockPin,LSBFIRST,digit[h[vt]]);
         digitalWrite(latchPin,HIGH);
         digitalWrite(digitPins[vt],LOW);
         delay(1);
         digitalWrite(digitPins[vt],HIGH);
       }
 }
```

b. Code cho mạch nhận và hiển thị giá trị nhiệt độ (Receiver):

```
#include <SPI.h>
#include <nRF24L01p.h>

nRF24L01p receiver(7,8);           //Khai bao chan CSN, CE

const int digitPins[4] = {2,3,4,5};
const int clockPin = 6;
const int latchPin = 9;
const int dataPin = 10;
const byte digit[13] =
{
  B00000011, //0
  B10011111, //1
```

```
  B00100101, //2
  B00001101, //3
  B10011001, //4
  B01001001, //5
  B01000001, //6
  B00011011, //7
  B00000001, //8
  B00001001, //9
  B00111001, //o
  B01100011, //C
  B01110001, //F
};
int h[4] = {0};
int vt = 0, conv,  ht, time;
float temp, tempF, Wtemp=35;
void setup()
{
  delay(150);
  Serial.begin(115200);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  receiver.channel(90);
  receiver.RXaddress("Artur");
  receiver.init();
  Serial.print("Led nhap nhay canh bao khi T* > ");
  Serial.print(Wtemp);
```

```
  Serial.println(" oc");
  Serial.println("---------------");
  Serial.println("Dang nhan du lieu...");
  Serial.println("---------------");

  pinMode(10,INPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

 void loop()
 {
   for(vt=0;vt<4;vt++)
   digitalWrite(digitPins[vt],HIGH);

   if(receiver.available())
{
   {
     receiver.read();                    // Nhan du lieu
     receiver.rxPL(temp);
     receiver.rxPL(tempF);
```

```
  Serial.println("Nhiet do da nhan:  ");
  Serial.print(temp);
  Serial.println(" oC");
  Serial.print(tempF);
  Serial.println(" oF");
  Serial.println("--------");
}

    conv=temp;
    ht = conv/100;
    h[0] = (conv%100)/10;
    h[1] = ((conv%100)%10)/1;
    h[2] = 10;
    h[3]=  11;

   if(ht>0)
     {
      h[2]=h[1];
      h[1]=h[0];
      h[0]=ht;
     }

  //Canh bao gioi han:
  time=1;
  if(temp>Wtemp)
  {
```

```
      time=250;
    }
  }

    for(vt=0;vt<4;vt++)                    // Xuat ra led
      {
        digitalWrite(latchPin,LOW);
        shiftOut(dataPin,clockPin,LSBFIRST,digit[h[vt]]);
        digitalWrite(latchPin,HIGH);
        digitalWrite(digitPins[vt],LOW);
        delay(time);
        digitalWrite(digitPins[vt],HIGH);
      }
}
```