ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN – ĐIỆN TỬ

****

BÁO CÁO
ĐỒ ÁN THIẾT KẾ II

Đề tài:

Thiết kế mạch điều khiển thiết bị từ xa qua Bluetooth/Wifi

Giảng viên hướng dẫn	**:**	**Vũ Sinh Thượng**
Sinh viên thực hiện	**:**	**Vũ Việt Anh**
Mã số sinh viên	**:**	**20223865**
Mã lớp	**:**	**760165**

Hà Nội, -2026

LỜI NÓI ĐẦU

Trong bối cảnh đất nước ta đang từng bước phát triển mạnh mẽ theo xu hướng công nghiệp hóa, hiện đại hóa, cuộc Cách mạng công nghiệp lần thứ tư đã và đang tạo ra những thay đổi sâu sắc trong nhiều lĩnh vực của đời sống xã hội. Đặc biệt, sự phát triển nhanh chóng của khoa học – công nghệ đã mở ra nhiều cơ hội để con người tiếp cận, nghiên cứu và ứng dụng các thành tựu kỹ thuật tiên tiến, trong đó nổi bật là lĩnh vực Điện tử – Viễn thông và tự động hóa.

Trước yêu cầu ngày càng cao của xã hội, thế hệ trẻ cần không ngừng học tập, trau dồi kiến thức chuyên môn cũng như kỹ năng thực hành để bắt kịp sự phát triển của khoa học kỹ thuật. Nhận thức rõ điều đó, Trường Đại học Bách khoa Hà Nội đã xây dựng chương trình đào tạo theo hướng toàn diện, kết hợp chặt chẽ giữa lý thuyết và thực tiễn. Việc triển khai các đồ án học phần, đặc biệt là Đồ án Thiết kế II, nhằm giúp sinh viên củng cố kiến thức đã học, rèn luyện tư duy kỹ thuật và nâng cao khả năng giải quyết các bài toán thực tế.

Cùng với sự phát triển của công nghệ vi điều khiển và truyền thông không dây, các hệ thống điều khiển từ xa ngày càng được ứng dụng rộng rãi trong đời sống và sản xuất. Công nghệ Bluetooth và WiFi cho phép các thiết bị giao tiếp với nhau một cách linh hoạt, tiện lợi, mang lại nhiều ưu điểm như điều khiển từ xa, giảm thiểu dây dẫn, nâng cao tính tự động hóa và tiết kiệm chi phí. Các ứng dụng điều khiển thiết bị từ xa hiện nay đã xuất hiện phổ biến trong các hệ thống nhà thông minh, thiết bị dân dụng và công nghiệp.

Xuất phát từ thực tiễn đó, em đã lựa chọn đề tài “Thiết kế mạch điều khiển thiết bị từ xa qua Bluetooth/WiFi” làm nội dung cho Đồ án Thiết kế II. Thông qua đề tài này, em mong muốn vận dụng các kiến thức đã học về vi điều khiển, mạch điện tử và truyền thông không dây để xây dựng một hệ thống điều khiển có tính ứng dụng thực tế cao.

Trong quá trình thực hiện đồ án, mặc dù đã cố gắng hoàn thành đề tài một cách tốt nhất, song do thời gian và kiến thức còn hạn chế nên không tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp quý báu của thầy để đồ án được hoàn thiện hơn.

Cuối cùng, em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy Vũ Sinh Thượng đã tận tình hướng dẫn, chỉ bảo và tạo điều kiện thuận lợi để em hoàn thành đồ án này.

MỤC LỤC

DANH MỤC VIẾT TẮT

Từ viết tắt	Tên tiếng anh	Tên tiếng việt
ADC	Analog Digital Converter	Bộ chuyển đổi tương tự sang số
IC	Integrated Circuit	Vi mạch
LCD	Liquid Crystal Display	Màn hình LCD
LED	Light Emitting Diode	Đèn led diode
USB	Universal Serial Bus	Chuẩn truyền thông nối tiếp đa năng
AVR	Advanced Virtual RISC	Kiến trúc vi điều khiển AVR
VĐK-MCU	Microcontroller Unit	Vi điều khiển
UART	Universal Asynchronous Receiver/Transmitter	Bộ truyền/nhận nối tiếp không đồng bộ
SPI	Serial Peripheral Interface	Giao tiếp ngoại vi nối tiếp

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN

Đồ án Thiết kế II tập trung vào việc nghiên cứu và tìm hiểu họ vi điều khiển AVR, trong đó vi điều khiển được sử dụng là ATmega16A. Thông qua đồ án, em tiến hành tìm hiểu nguyên lý hoạt động của bộ kit vi điều khiển, các khối chức năng cơ bản cũng như khả năng ứng dụng của bộ kit trong các bài toán kỹ thuật thực tế.

Trong báo cáo này, em trình bày ứng dụng của bộ kit vi điều khiển AVR trong việc xây dựng hệ thống điều khiển thiết bị từ xa thông qua công nghệ Bluetooth, với đề tài "Thiết kế mạch điều khiển thiết bị từ xa qua Bluetooth". Hệ thống sử dụng module HC-05 để thực hiện truyền thông không dây giữa vi điều khiển và thiết bị điều khiển bên ngoài. Nội dung đồ án được chia thành bốn chương chính như sau:

Chương I. Cơ sở lý thuyết chung. Trình bày tổng quan về đề tài, giới thiệu họ vi điều khiển AVR, vi điều khiển ATmega16A và các kiến thức nền tảng liên quan đến lập trình vi điều khiển và truyền thông nối tiếp.

Chương II. Cấu hình và thiết kế mạch kit. Trình bày chi tiết cấu trúc phần cứng của mạch kit được sử dụng trong đồ án, các thông số kỹ thuật chính, cũng như cách kết nối vi điều khiển với các thiết bị ngoại vi như module Bluetooth HC-05, LCD và các phần tử điều khiển.

Chương III. Thực hành lập trình vi điều khiển. Nội dung chương tập trung vào việc xây dựng chương trình điều khiển cho vi điều khiển ATmega16A, cấu hình giao tiếp UART, nạp chương trình và kiểm tra hoạt động của hệ thống, đồng thời đánh giá tính ổn định của mạch và chương trình.

Chương IV. Vận dụng kiến thức vào thực tế. Chương này trình bày quá trình triển khai hệ thống điều khiển thiết bị từ xa thông qua Bluetooth, phân tích nguyên lý hoạt động, kết quả thử nghiệm và đánh giá khả năng ứng dụng của hệ thống trong thực tế.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT CHUNG

1.1 Giới thiệu chung

Với học phần Đồ án II – Vi điều khiển, nội dung môn học được xây dựng nhằm trang bị và nâng cao năng lực chuyên môn cho sinh viên thông qua việc kết hợp các kiến thức đã học về điện tử tương tự, điện tử số, kỹ thuật vi xử lý và truyền thông số. Học phần này giúp sinh viên rèn luyện khả năng phân tích, thiết kế hệ thống và vận dụng kiến thức lý thuyết vào các bài toán kỹ thuật mang tính thực tiễn.

Trong quá trình thực hiện học phần, sinh viên được tiếp cận với các công cụ thiết kế mạch điện, thực hành lập trình phần cứng cho vi điều khiển và xây dựng các ứng dụng cơ bản trên nền tảng các vi mạch có khả năng lập trình. Để hỗ trợ việc triển khai đồ án, nhà trường và viện đã cung cấp một số nền tảng và phương tiện kỹ thuật cơ bản, giúp sinh viên có điều kiện kế thừa và phát triển sản phẩm theo đúng tiến độ học tập.

Sau khi làm quen với vi điều khiển họ AVR thông qua việc xây dựng một số ứng dụng cơ bản, trong báo cáo này em sử dụng vi điều khiển ATmega16A để giải quyết một bài toán cụ thể. Nội dung chính của đồ án là thiết kế mạch điều khiển thiết bị từ xa thông qua công nghệ Bluetooth, sử dụng module HC-05 làm khối truyền thông không dây giữa vi điều khiển và thiết bị điều khiển bên ngoài.

Hệ thống được thiết kế cho phép người dùng gửi các lệnh điều khiển từ xa thông qua kết nối Bluetooth để điều khiển hoạt động của thiết bị, qua đó giúp em hiểu rõ hơn về nguyên lý giao tiếp nối tiếp UART, cách cấu hình và khai thác module Bluetooth HC-05, cũng như khả năng ứng dụng của vi điều khiển AVR trong các hệ thống điều khiển hiện đại.

1.2 Vi điều khiển Atmega16

1.2.1. Tổng quan về VĐK Atmega16

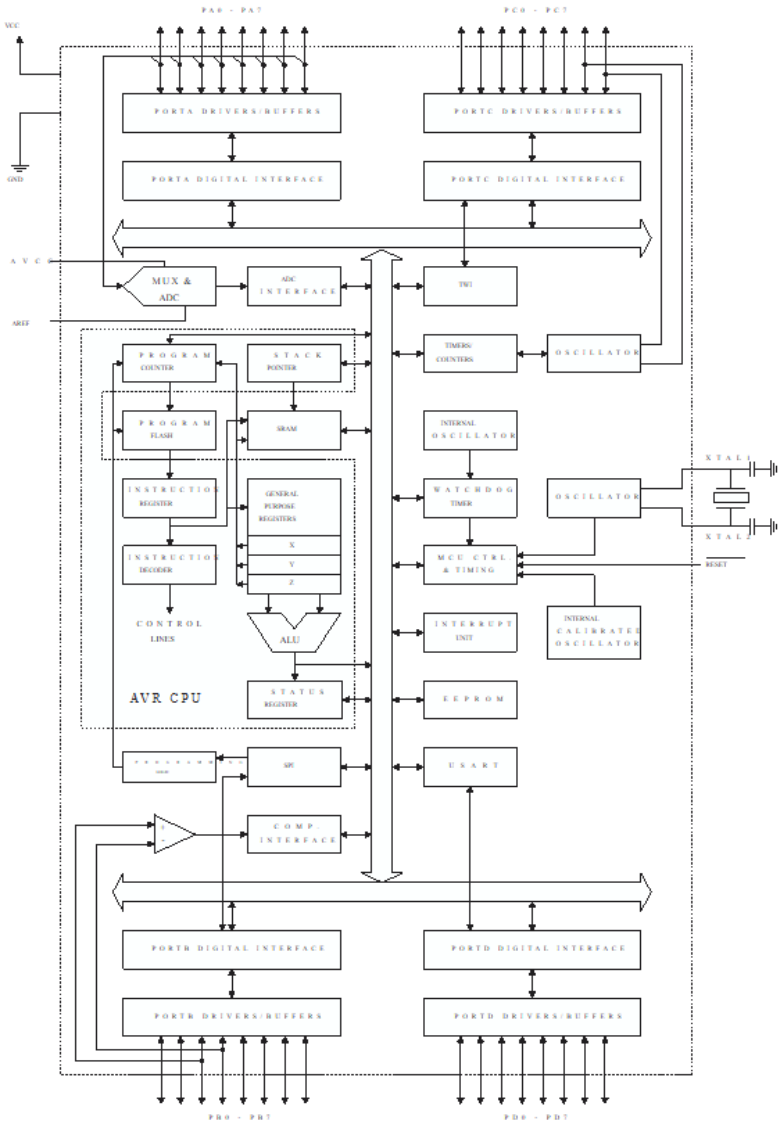
ATmega16 là vi điều khiển 8 bit thuộc họ AVR do hãng Atmel (nay là Microchip Technology) phát triển. Vi điều khiển này được thiết kế dựa trên kiến trúc RISC, cho phép thực thi phần lớn các lệnh trong một chu kỳ xung nhịp, nhờ đó đạt hiệu suất xử lý cao. ATmega16 tích hợp nhiều ngoại vi trên cùng một chip như bộ nhớ chương trình, bộ nhớ dữ liệu, các cổng vào/ra, bộ chuyển đổi ADC, các bộ định thời và các giao tiếp truyền thông nối tiếp, đáp ứng tốt yêu cầu của các hệ thống nhúng hiện đại.

Với ưu điểm dễ lập trình, tài liệu phong phú và độ ổn định cao, ATmega16 được sử dụng rộng rãi trong các ứng dụng điều khiển, đo lường, hiển thị và truyền thông không dây. Đây là vi điều khiển phổ biến trong các mô hình học tập và đồ án sinh viên ngành điện – điện tử.

1.2.2. Cấu trúc Atmega16

ATmega16 là vi điều khiển CMOS 8-bit công suất thấp dựa trên lõi AVR nâng cao.

Kiến trúc RISC. Bằng cách thực hiện các hướng dẫn mạnh mẽ trong một chu kỳ đồng hồ, ATmega16 đạt được thông lượng lên tới gần 1 MIPS trên mỗi MHz cho phép hệ thống thiết kế để tối ưu hóa mức tiêu thụ điện năng so với tốc độ xử lý.



Hình 1.1 Cấu trúc các khối Atmega16

- Khối xử lý trung tâm (CPU). Khối CPU của ATmega16 thực hiện việc xử lý và điều khiển toàn bộ hoạt động của vi điều khiển. CPU bao gồm:
  - Đơn vị logic – số học (ALU), thực hiện các phép toán số học và logic.
  - 32 thanh ghi đa dụng 8 bit, được kết nối trực tiếp với ALU, giúp tăng tốc độ xử lý dữ liệu.
  - Thanh ghi trạng thái (Status Register – SREG), lưu trữ các cờ trạng thái phục vụ điều khiển chương trình.

Nhờ kiến trúc RISC, hầu hết các lệnh của CPU được thực thi trong một chu kỳ xung nhịp, giúp ATmega16 đạt hiệu suất cao.

- Hệ thống bộ nhớ. ATmega16 tích hợp nhiều loại bộ nhớ khác nhau, mỗi loại đảm nhiệm một chức năng riêng:
  - Bộ nhớ Flash: dùng để lưu trữ chương trình điều khiển.
  - Bộ nhớ SRAM: dùng để lưu trữ dữ liệu tạm thời trong quá trình thực thi chương trình.
  - Bộ nhớ EEPROM: dùng để lưu trữ dữ liệu lâu dài, không bị mất khi ngắt nguồn.
  Việc tích hợp các bộ nhớ ngay trên chip giúp giảm kích thước mạch và tăng độ ổn định của hệ thống.

- Hệ thống vào/ra (I/O) ATmega16 cung cấp các cổng vào/ra số khả trình, cho phép kết nối trực tiếp với các thiết bị ngoại vi như LED, LCD, cảm biến, relay,…
  Mỗi chân I/O có thể được cấu hình linh hoạt là ngõ vào hoặc ngõ ra, đồng thời hỗ trợ các chức năng thay thế tùy theo chế độ hoạt động của vi điều khiển.

- Các khối ngoại vi tích hợp ATmega16 được tích hợp nhiều khối ngoại vi phục vụ các ứng dụng điều khiển và đo lường, bao gồm:
  - Các bộ định thời/bộ đếm phục vụ đo thời gian, tạo xung và điều khiển PWM.
  - Bộ chuyển đổi ADC cho phép xử lý tín hiệu tương tự.
  - Các khối giao tiếp nối tiếp hỗ trợ trao đổi dữ liệu với các thiết bị bên ngoài.
  Những khối ngoại vi này giúp giảm số lượng linh kiện ngoài và đơn giản hóa thiết kế hệ thống.

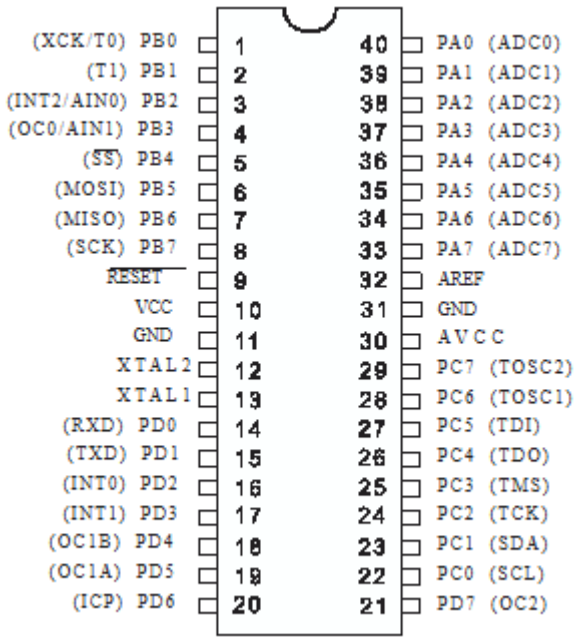
- Hệ thống ngắt. ATmega16 hỗ trợ hệ thống ngắt đa dạng, bao gồm ngắt nội và ngắt ngoại. Hệ thống ngắt cho phép vi điều khiển phản hồi nhanh với các sự kiện, nâng cao tính linh hoạt và hiệu quả trong các ứng dụng thời gian thực.

- Hệ thống xung nhịp và reset. Vi điều khiển ATmega16 có thể sử dụng nguồn xung nhịp ngoài hoặc mạch dao động nội, đáp ứng nhiều yêu cầu ứng dụng khác nhau. Ngoài ra, ATmega16 được trang bị các mạch reset và giám sát nguồn, giúp đảm bảo hệ thống khởi động và hoạt động ổn định.

1.2.3. Các cổng I/O

Vi điều khiển ATmega16 được trang bị 32 chân vào/ra số khả trình, được chia thành 4 cổng chính là PORTA, PORTB, PORTC và PORTD. Các cổng này cho phép ATmega16 giao tiếp trực tiếp với các thiết bị ngoại vi như LED, LCD, cảm biến, nút nhấn, module truyền thông,…

Mỗi cổng I/O gồm 8 chân, có thể cấu hình linh hoạt ở chế độ ngõ vào (Input) hoặc ngõ ra (Output) thông qua các thanh ghi điều khiển tương ứng.



Hình 1.2 Sơ đồ cấu hình chân Atmega16

- Các thanh ghi điều khiển cổng I/O

Mỗi cổng I/O của ATmega16 được quản lý bởi ba thanh ghi chính:

- DDRx (Data Direction Register). Dùng để cấu hình hướng dữ liệu cho từng chân I/O.
  - Bit = 1 → chân được cấu hình là ngõ ra
  - Bit = 0 → chân được cấu hình là ngõ vào
- PORTx (Port Data Register)
  - Khi chân ở chế độ ngõ ra, thanh ghi PORTx dùng để ghi mức logic 0 hoặc 1 ra chân.
  - Khi chân ở chế độ ngõ vào, PORTx dùng để kích hoạt điện trở kéo lên nội (pull-up resistor).

- PINx (Port Input Register)

  Dùng để đọc trạng thái logic của các chân khi được cấu hình là ngõ vào.

Trong đó, ký tự x tương ứng với các cổng A, B, C, D.

- Cổng PORTA gồm 8 chân (PA0 – PA7).
  Ngoài chức năng vào/ra số thông thường, các chân của PORTA còn có thể đảm nhiệm chức năng ngõ vào analog cho bộ chuyển đổi ADC khi vi điều khiển hoạt động ở chế độ tương ứng. PORTA thường được sử dụng trong các ứng dụng đo lường và thu thập dữ liệu từ cảm biến.

- Cổng PORTB gồm 8 chân (PB0 – PB7).
  Ngoài chức năng I/O số, PORTB còn hỗ trợ các chức năng đặc biệt liên quan đến:
  - Giao tiếp SPI
  - Bộ định thời
  - Ngắt ngoài
    PORTB thường được sử dụng để kết nối với các thiết bị ngoại vi yêu cầu tốc độ cao.
- Cổng PORTC gồm 8 chân (PC0 – PC7).
  Các chân của PORTC có thể được sử dụng làm cổng I/O số hoặc đảm nhiệm các chức năng thay thế tùy theo cấu hình.
  PORTC thường được dùng để giao tiếp với LCD 16x2, bàn phím ma trận hoặc các thiết bị hiển thị khác.
- Cổng PORTD gồm 8 chân (PD0 – PD7).
  Ngoài chức năng I/O số, PORTD còn đảm nhiệm các chức năng liên quan đến:
  - Giao tiếp USART (RXD, TXD)
  - Ngắt ngoài
    PORTD thường được sử dụng trong các ứng dụng truyền thông nối tiếp như kết nối với máy tính hoặc module Bluetooth.
- Đặc điểm chung của các cổng I/O
  - Mỗi chân I/O có thể hoạt động độc lập và được cấu hình bằng phần mềm.
  - Các cổng I/O hỗ trợ điện trở kéo lên nội, giúp giảm số lượng linh kiện ngoài.
  - Chức năng của mỗi chân có thể thay đổi tùy theo chế độ hoạt động của vi điều khiển.

- Chân RESET (RST): ngõ vào RST ở chân 9 là ngõ vào Reset dùng để thiết lập trạng thái ban đầu cho vi điều khiển. Hệ thống sẽ được thiết lập lại các giá trị ban đầu nếu ngõ này ở mức 1 tối thiểu 2 chu kì clock.
- Chân XTAL1 và XTAL2: Hai chân này có vị trí chân là 12 và 13 được sử dụng để nhận nguồn xung clock từ bên ngoài để hoạt động, thường được ghép nối với thạch anh và các tụ để tạo nguồn xung clock ổn định.
- Chân AVCC: Nguồn cấp cho cổng A và bộ chuyển đổi ADC, chân này được nối với nguồn VCC bên ngoài, ngay cả khi bộ chuyển đổi ADC không được sử dụng. Nếu bộ chuyển đổi ADC không được sử dụng, chân AVCC nên được nối với nguồn qua bộ lọc.
- Chân AREF: AREF là chân chuẩn analog cho bộ chuyển đổi ADC.

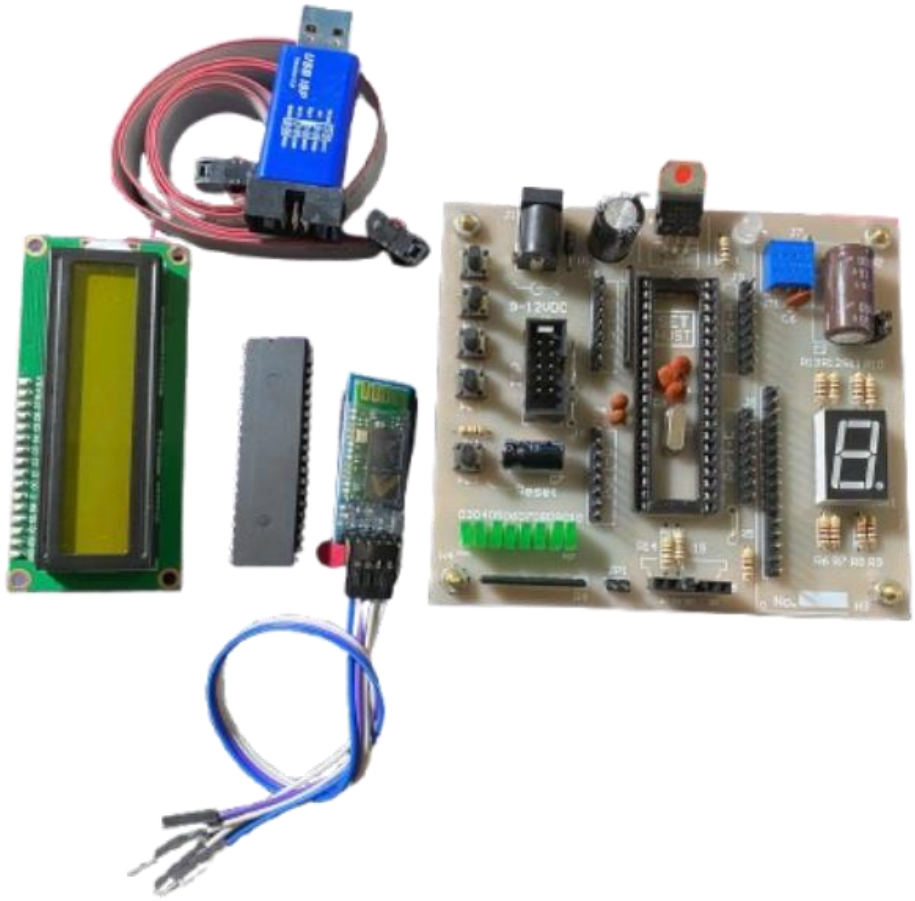
1.3 Mạch Kit cho VĐK họ AVR

AVR là một VĐK 8 bit khá mạnh và thông dụng tại thị trường Việt Nam. Với tốc độ xung nhịp tối đa lên tới 16 Mhz, bộ nhớ chương trình tối đa tới 256 kB, và rất nhiều chức năng ngoại vi tích hợp sẵn, VĐK họ AVR có thể đáp ứng tốt cho nhiều ứng dụng trong thực tế, từ đơn giản đến phức tạp.

Với Kit phát triển sử dụng trong học phần Đồ án II (hình 1.1) được Viện Điện tử - Viễn thông thiết kế riêng để đảm bảo tính hiệu quả trong quá trình đào tạo. Bộ kit có thể được thử nghiệm với các ứng dụng cơ bản sau:

- Điều khiển cổng ra số, với LED đơn và LED 7 thanh
- Đọc trạng thái logic đầu vào số, từ bán phím và giắc cắm mở rộng
- Đo điện áp tương tự với biến trở vi chỉnh và bộ ADC 10-bit
- Điều khiển màn hình tinh thể lỏng, với màn hình LCD dạng text
- Giao tiếp với máy tính qua chuẩn UART ↔ USB
- Thử nghiệm các ngắt ngoài, thử khả năng điều chế độ rộng xung.

❖ Nhiều ứng dụng điều khiển các chức năng tích hợp sẵn trong VĐK như: vận hành các bộ định thời (Timer) và bộ đếm (Counter), đọc ghi EEPROM, lập trình các ngắt chương trình, thiết lập Watchdog, v.v.



Hình 1.3 Mạch Kit phát triển và các phụ kiện

Ngoài ra, bằng việc kết nối giữa các mô-dun mở rộng, mạch Kit hoàn toàn có thểthực hiện các ứng dụng phức tạp hơn như:

- Đo tham số môi trường: nhiệt độ, độ ẩm, độ sáng, v.v.
- Điều khiển tải cơ bản: đèn báo, van điện tử, động cơ DC, động cơ bước, v.v.
- Điều khiển hiển thị cơ bản: LED ma trận, LCD ma trận, màn hình cảm ứng, v.v.
- Giao tiếp I2C và SPI: IC thời gian thực, IC EEPROM, cảm biến gia tốc, v.v.
- Ứng dụng tổng hợp: đo và duy trì sự ổn định các tham số mô trường; số hóa và xử lý tín hiệu âm thanh, điều khiển robot hoặc xe tự hành, v.v.

1.4 Ngôn ngữ lập trình và phần mềm

AVR nói chung chung cũng như Atmega16 nói riêng hỗ trợ 2 ngôn ngữ lập trình thông dụng là Assembly và C. Việc lập trình bằng Assembly giúp chương trình nhỏ gọn nhưng khá phức tạp do gần với ngôn ngữ máy. Lập trình bằng C tuy cho chương trình có dung lượng lớn hơn so với khi lập trình bằng Assembly, nhưng đổi lại dễ dàng hơn trong việc code và debug.

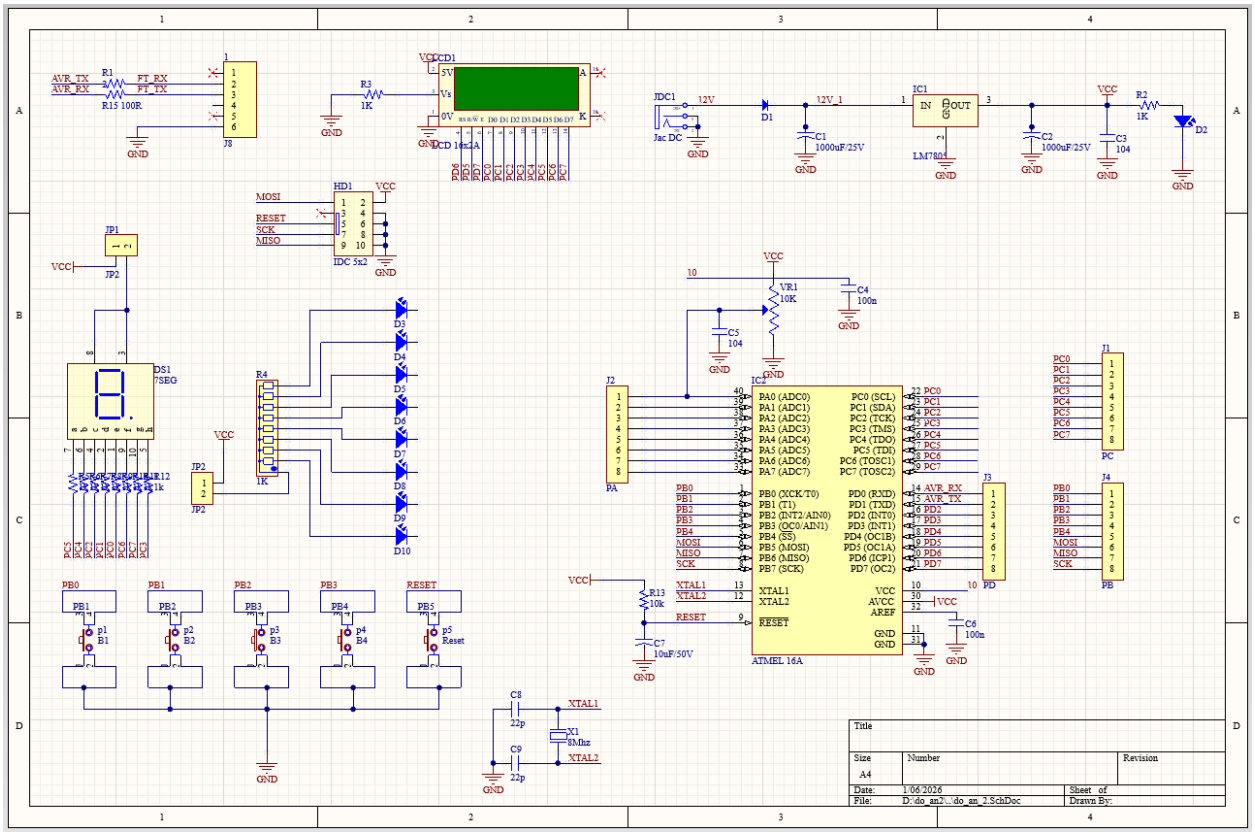
Để lập trình cho AVR, có khá nhiều trình biên dịch, ví dụ như AVR studio, WinAVR, codevisionAVR…

Trong nội dung về học phần Đồ án II, em sẽ sử dụng ngôn ngữ lập trình C để lập trình cho VĐK. Môi trường để soạn thảo và biên dịch là phần mềm AtmelStudio 7. Phần mềm nạp mã máy là PROGISP (phiên bản 1.72). Phần mềm giao tiếp giữa máy tính và VĐK là Terminal (phiên bản 1.9b). Ngoài ra, trong phần vận dụng emcó sử dụng thêm phần mềm Proteus 8.6 để mô phỏng mạch và phần mềm Altium Designer 19 để thiết kế mạch.

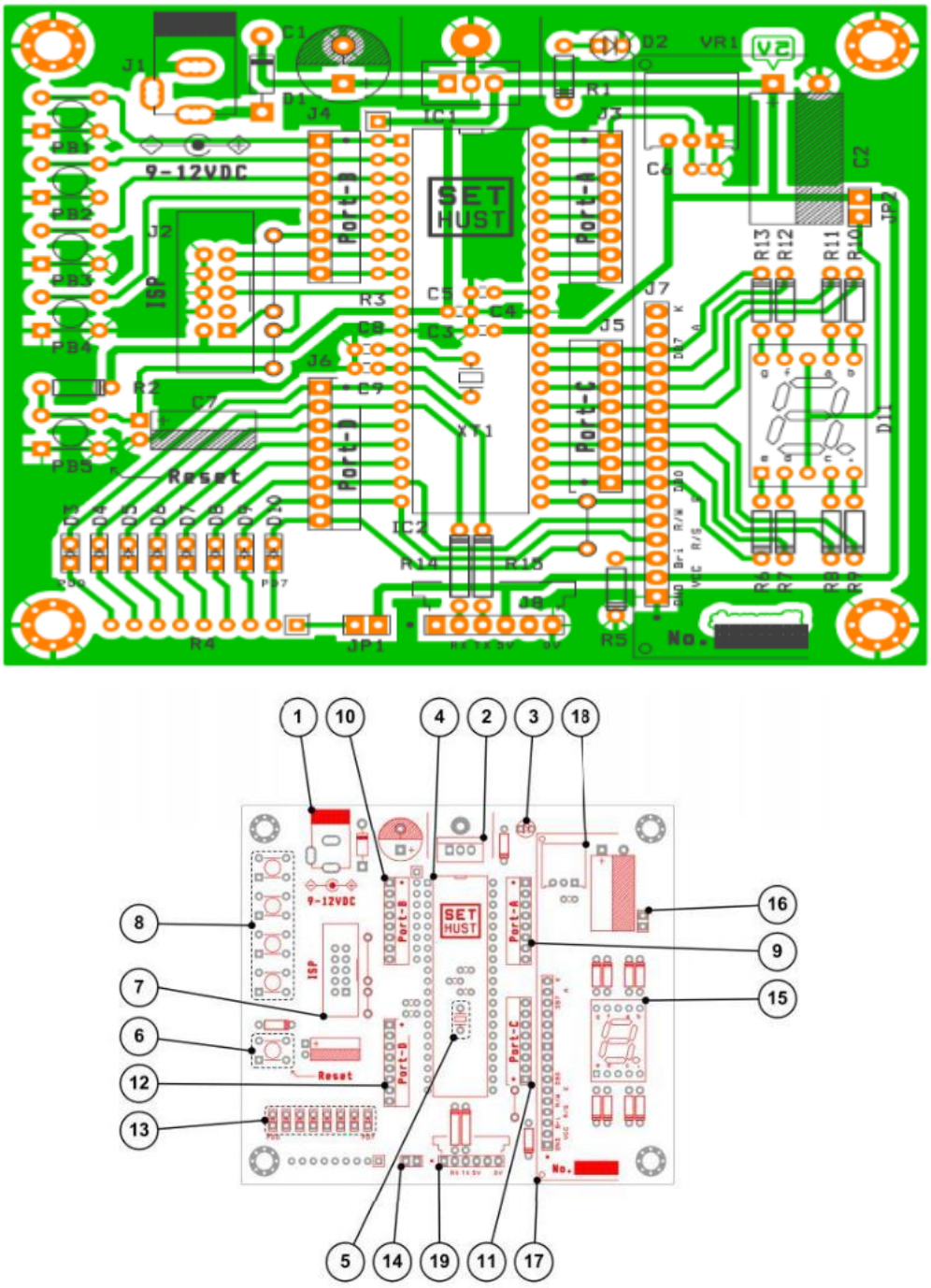
CHƯƠNG 2. CHI TIẾT CẤU HÌNH CỦA MẠCH KIT

2.1 Cấu trúc của mạch Kit

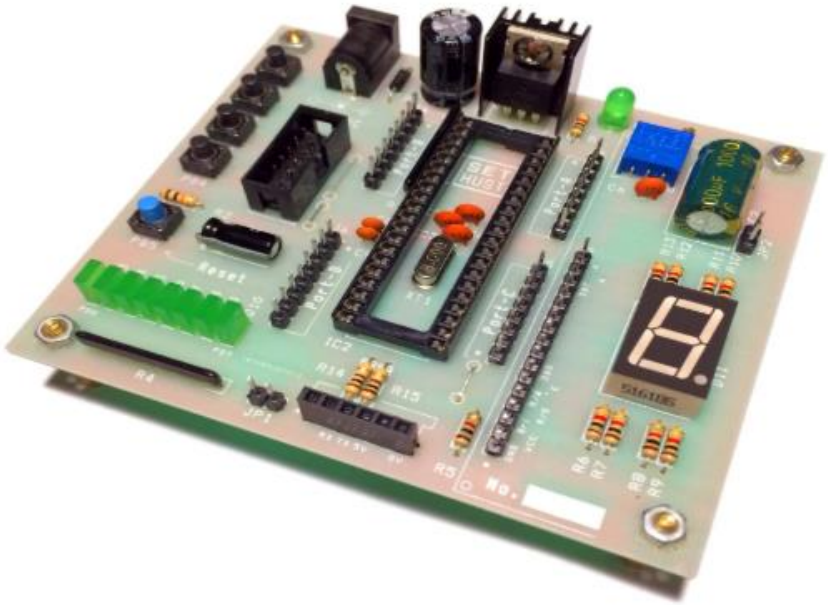
Mạch Kit được cung cấp để đáp ứng các ứng dụng cơ bản có cấu trúc và chức năng của từng linh kiện quan trọng được nêu trong bảng dưới đây. Đầu tiên ta sẽ tìm hiểu về sơ đồ nguyên lý của bộ kit cùng với đó là mạch nguyên lý được vẽ bởi Altium một trong những công cụ vẽ mạch đi dây cũng như để in mạch mạnh và phổ biến nhất hiện nay. Thông qua môn học đồ án II này em có thể thành thạo hơn việc sử dung altium để vẽ và thiết kế tất cả các loại mạch. Tự mình tạo ra một bản mạch mà mình mong muốn.



Hình 2.1 Sơ đồ nguyên lý của bộ kit



Hình 2.2 Sơ đồ PCB của bộ kit



Hình 2.3 Bộ kit hoàn thiện

Bảng 2.1 Linh kiện quan trọng của mạch Kit và chức năng tương ứng

STT	Tên linh kiện	Chức năng
1	Giắc cắm nguồn	Nhận nguồn điện 9-12 VDC cấp cho mạch Kit
2	IC ổn áp 7805	Hạ 9-12 VDC xuống 5 VDC và giữ ổn định mức điện áp này để cấp cho toàn mạch
3	LED báo nguồn	Báo nguồn (sáng: có nguồn 5 VDC, tắt: mất nguồn)
4	VĐK Atmega16	Điều khiển hoạt động của mạch theo mã nguồn đã được nạp
5	Thạch anh	Quyết định tần số xung nhịp cấp cho VĐK
6	Nút ấn Reset	Khởi động lại VĐK
7	Giắc ISP	Kết nối mạch nạp để nạp mã nguồn cho VĐK
8	Nhóm 4 phím ấn	Nhận lệnh điều khiển từ người sử dụng
9	Giắc cắm 8 chân	Nối tới 8 chân vào/ra (Port-A) của VĐK
10	Giắc cắm 8 chân	Nối tới 8 chân vào/ra (Port-B) của VĐK
11	Giắc cắm 8 chân	Nối tới 8 chân vào/ra (Port-C) của VĐK
12	Giắc cắm 8 chân	Nối tới 8 chân vào/ra (Port-D) của VĐK
13	Dãy LED đơn	Báo trạng thái logic của 8 chân ở Port-D của VĐK
14	Jumper dãy LED đơn	Cho phép hoặc vô hiệu hóa dãy LED đơn
15	LED 7 thanh	Hiển thị số 0-9 và một vài kí tự do người dung định nghĩa
16	Jumper LED 7 thanh	Cho phép hoặc vô hiệu hóa dãy LED 7 thanh
17	Giắc cắm LCD	Kết nối tới màn hình LCD1602

18	Biến trở vi chỉnh	Điều chỉnh trơn và liên tục, từ 0 đến 5 VDC, từ mức điện áp tại đầu vào ADC0 của bộ ADC (chân PA0)
19	Giắc UART-USB	Kết nối module chuyển đổi UART-USB (còn gọi là COM-USB)

2.2 Các thông số chính của Kit

Các thông số kĩ thuật của Kit:

- Điện áp nguồn:
  - Tiêu chuẩn: 9-12 VDC
  - Giới hạn: 7-18 VDC
- Dòng điện tiêu thụ:
  - Khi không có module mở rộng, toàn bộ LED chỉ thị I/O tắt: 18mA
  - Khi có LCD và module USB, LED chỉ thị I/O bị vô hiệu hóa: 22mA
  - Khi có LCD và module USB, toàn bộ LED chỉ thị I/O sáng: 80mA
- Mạch có khả năng tự bảo vệ khi bị lắp ngược cực tính nguồn
- Mức logic các cổng I/O: TTL (5V)
- Điện áp tương tự vào các chân ADC: 0 – 5V
- Loại VĐK được hỗ trợ: Atmega16, Atmega32 và tương đương•Cổng I/O mở rộng: 4 giắc cắm (loại 8 chân) ứng với 4 port
- Hỗ trợ màn hình LCD: dạng text, giao tiếp 8 bit hay 4 bit•Hỗ trợ module USB: UART-USB hay COM-USB (mức 5 VDC)
- Xung nhịp tích hợp sẵn: thạch anh 8 Mhz

2.3 Mạch nạp mã nguồn

Mạch nạp ISP (In-System Programming) là một phương pháp cho phép lập trình viên ghi dữ liệu vào bộ nhớ flash hoặc EEPROM của một vi điều khiển AVR mà không cần phải tháo rời nó khỏi mạch điện. Điều này rất hữu ích trong quá trình phát triển và gỡ lỗi phần cứng.

- ·Mạch nạp usb asp chuyên nạp cho dòng Vi điều khiển AVR.
- ·Giao tiếp và cấp nguồn qua cổng USB 5VDC.
- ·Cổng nạp chuẩn ISP 10 pins.

Hình 2.4 Mạch nạp ISP 89S/AVR và các chân ISP

- · MOSI (Master Out Slave In): Chân để truyền dữ liệu từ thiết bị lập trình sang vi điều khiển.
- · MISO (Master In Slave Out): Chân để truyền dữ liệu ngược lại từ vi điều khiển về thiết bị lập trình.
- · SCK (Serial Clock): Cung cấp tín hiệu đồng hồ để đồng bộ hóa việc truyền dữ liệu qua MOSI và MISO.
- · RESET: Chân để đặt lại hoặc khởi động lại vi điều khiển khi bắt đầu quá trình lập trình.
- ·NC: Not-connected .
- · VCC: Chân nguồn điện dương (thường là 5V hoặc 3.3V).
- · GND: Chân nối đất (ground).

2.4 Màn hình LCD và module UART-USB

LCD 16×2 là màn hình tinh thể lỏng dạng ký tự, có khả năng hiển thị 2 dòng, mỗi dòng 16 ký tự, sử dụng IC điều khiển chuẩn HD44780 hoặc tương thích.

Thông số kỹ thuật

- Điện áp hoạt động: **4.5 – 5.5 VDC**
- Dòng tiêu thụ (không đèn nền): **1 – 2 mA**
- Dòng đèn nền LED: **15 – 30 mA**
- Số dòng hiển thị: **2 dòng**
- Số ký tự mỗi dòng: **16 ký tự**
- Chuẩn giao tiếp: **Song song (4-bit / 8-bit)**
- Tần số giao tiếp tối đa: **≈ 270 kHz**

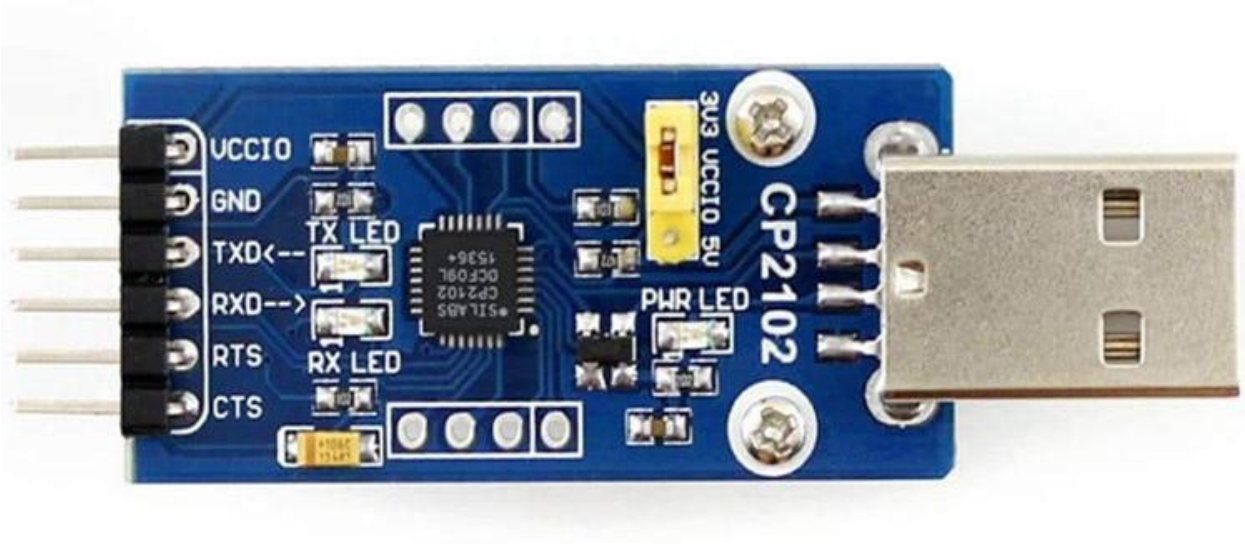
- Bộ nhớ hiển thị (DDRAM): **80 byte**
- Bộ nhớ ký tự tự tạo (CGRAM): **64 byte**
- Thời gian lệnh Clear Display: ≈ **1.64 ms**
- Kích thước module: ≈ **80 × 36 mm**
- Kích thước vùng hiển thị: ≈ **64.5 × 16 mm**

LCD sử dụng giao tiếp song song nên dễ dàng kết nối với các vi điều khiển phổ biến như ATmega16A, Arduino, PIC và STM32. Màn hình hỗ trợ chế độ 4-bit giúp tiết kiệm chân I/O và có thể hiển thị các ký tự ASCII cũng như ký tự tự tạo.

Hình 2.5 Màn hình LCD 1602 sử dụng trong bộ Kit

USBS-UART là mô-đun hoặc mạch cho phép giao tiếp giữa cổng USB (Universal Serial Bus) trên máy tính hoặc thiết bị và giao diện UART (Bộ thu/phát không đồng bộ đa năng), thường được sử dụng để liên lạc nối tiếp trong các hệ thống nhúng và bộ vi điều khiển.

Mục đích chính của mô-đun USBS-UART là hoạt động như một cầu nối hoặc bộ chuyển đổi giữa giao thức USB và giao thức UART. Nó cho phép truyền dữ liệu giữa máy chủ USB (chẳng hạn như máy tính) và các thiết bị hoặc bộ vi điều khiển giao tiếp bằng giao thức UART**.**



Hình 2.6 Module USB-UART

Mô-đun USB-UART thường bao gồm các thành phần sau:

- Giao diện USB: Phần này xử lý giao thức giao tiếp USB, cho phép module kết nối với cổng USB trên máy tính hoặc thiết bị chủ USB khác.
- Giao diện UART: Phần này cung cấp giao diện UART, bao gồm các chân cần thiết (TX, RX và đôi khi là các chân điều khiển bổ sung) để giao tiếp với các thiết bị hoặc bộ vi điều khiển hỗ trợ UART.
- Vi điều khiển hoặc mạch logic: Thành phần này đóng vai trò là cầu nối, dịch dữ liệu giữa giao thức USB và UART theo cả hai hướng. Nó nhận dữ liệu từ máy chủ USB, chuyển đổi sang định dạng UART và truyền dữ liệu đến thiết bị UART. Tương tự, nó nhận dữ liệu từ thiết bị UART, chuyển đổi dữ liệu sang định dạng USB và gửi đến máy chủ USB**.**

2.5 Module Bluetooth HC-05

Module Bluetooth HC-05 là module truyền thông không dây sử dụng chuẩn Bluetooth Classic, cho phép giao tiếp nối tiếp UART giữa vi điều khiển và các thiết bị như điện thoại Android hoặc máy tính.

Thông số kỹ thuật

- Chuẩn Bluetooth: **Bluetooth v2.0 + EDR**
- Tần số hoạt động: **2.402 – 2.480 GHz**
- Điện áp cấp nguồn: **3.6 – 5.0 VDC**
- Dòng tiêu thụ khi hoạt động: **30 – 40 mA**

- Khoảng cách truyền: **≤ 10 m (không vật cản)**
- Tốc độ UART mặc định: **9600 bps**
- Công suất phát: **≈ +4 dBm**
- Độ nhạy thu: **≈ –80 dBm**
- Kiểu giao tiếp: **UART (TTL)**
- Chế độ hoạt động: **Master / Slave**
- Số chân giao tiếp: **6 chân**
- Kích thước module: **≈ 28 × 15 mm**

Hình 2.7 Module Bluetooth HC-05

HC-05 hỗ trợ cấu hình thông qua tập lệnh AT, cho phép thay đổi tên thiết bị, baudrate và chế độ hoạt động. Module thường được sử dụng trong các ứng dụng điều khiển và truyền dữ liệu không dây tầm ngắn trong hệ thống nhúng.

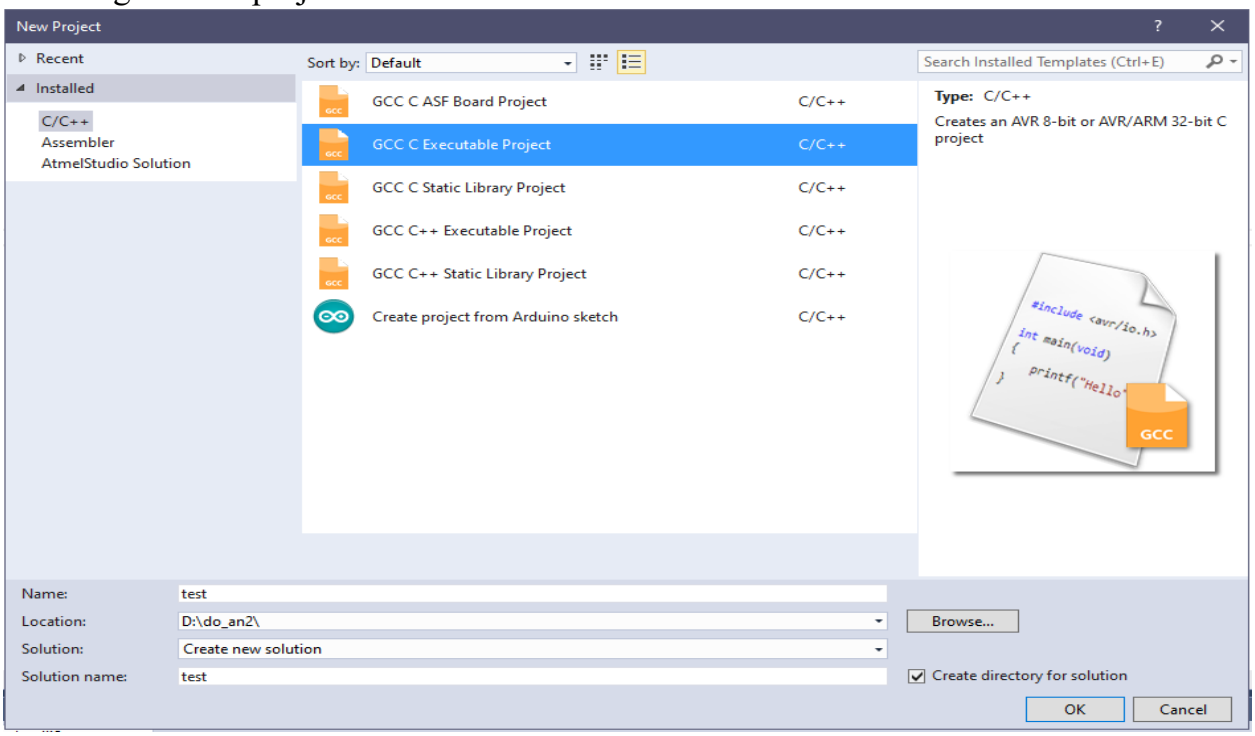
CHƯƠNG 3. THỰC HÀNH LẬP TRÌNH CHO VĐK

3.1 Tạo Project mới với Atmel Studio 7 và nạp thử mã máy cho VĐK

Khi bắt đầu lập trình với VĐK họ AVR có rất nhiều phần mềm hỗ trợ tốt cho dòng VĐK này, tuy nhiên, lí do mà Atmel Studio được sử dụng là mang lại sự hiểu biết về VĐK đang lập trình – vì với Atmel Studio ta phải khai báo cụ thể các chân, v.v. còn một số IDE khác hỗ trợ điều này khi tạo một project mới.

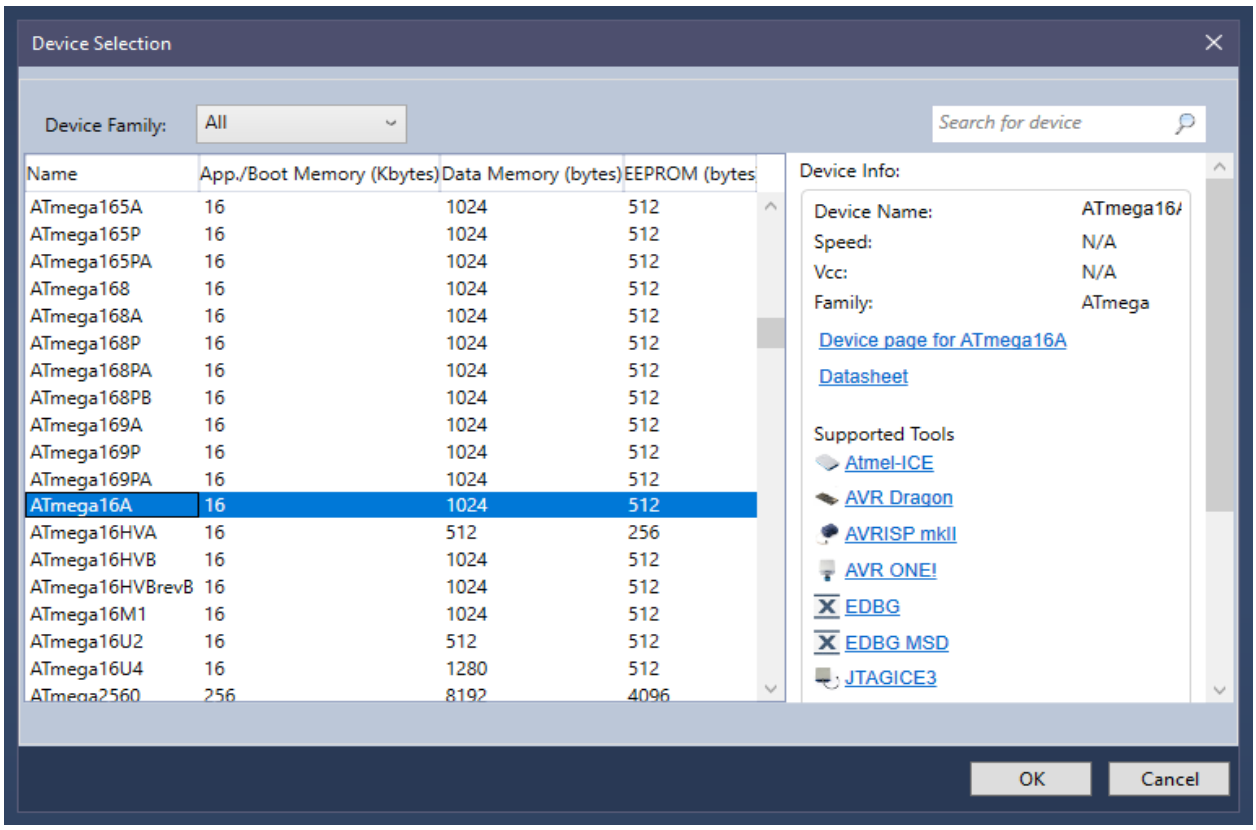
Cụ thể, để tạo một Project mới và nạp thử mã máy cho VĐK ta cần thông qua các bước sau:

Bước 1: Trong Atmel Studio 7, vào File > New > Project . Sau khi thay đổi tên và đường dẫn lưu project thì ta bấm OK.

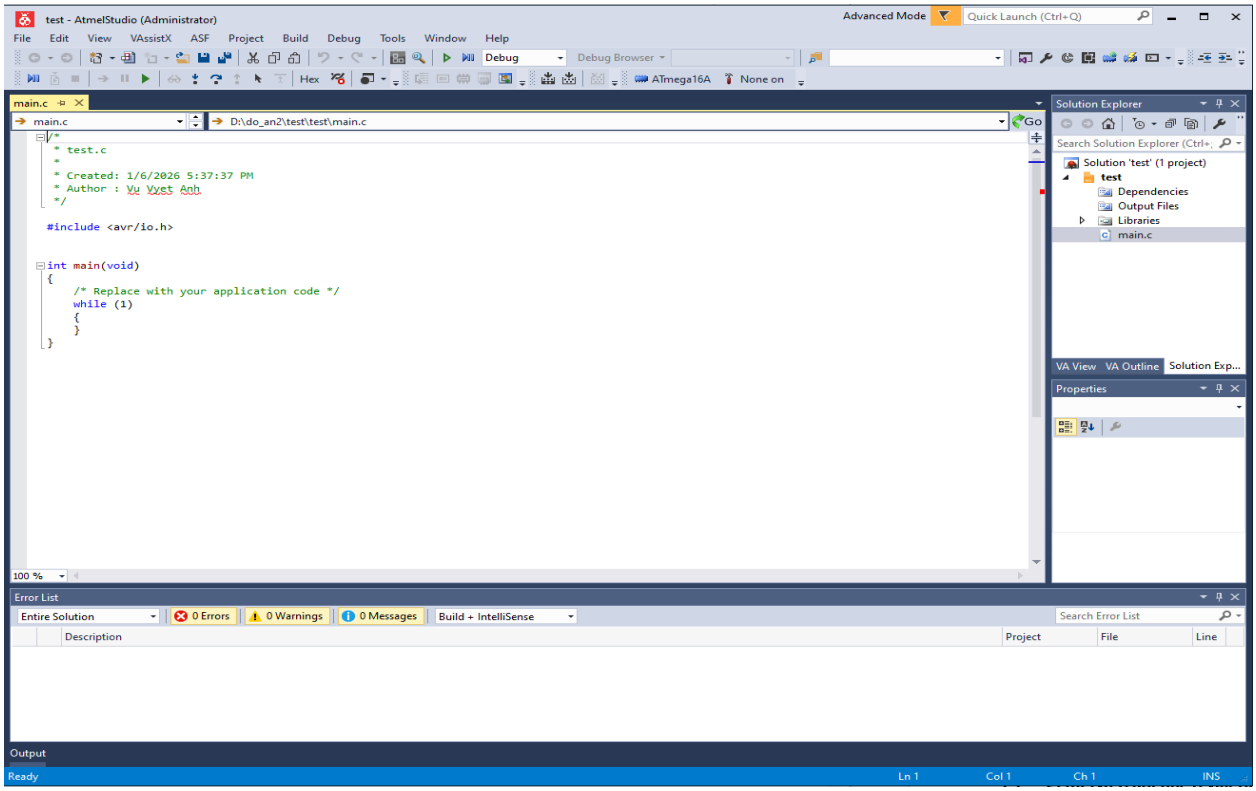


Hình 3.1 Giao diện tạo Project mới trong Atmel Studio 7

Bước 2: Sau khi tạo Project, giao diện chọn loại vi điều khiển sẽ hiện ra, ở đây ta sẽ chọn loại vi điều khiển là Atmega 16A.

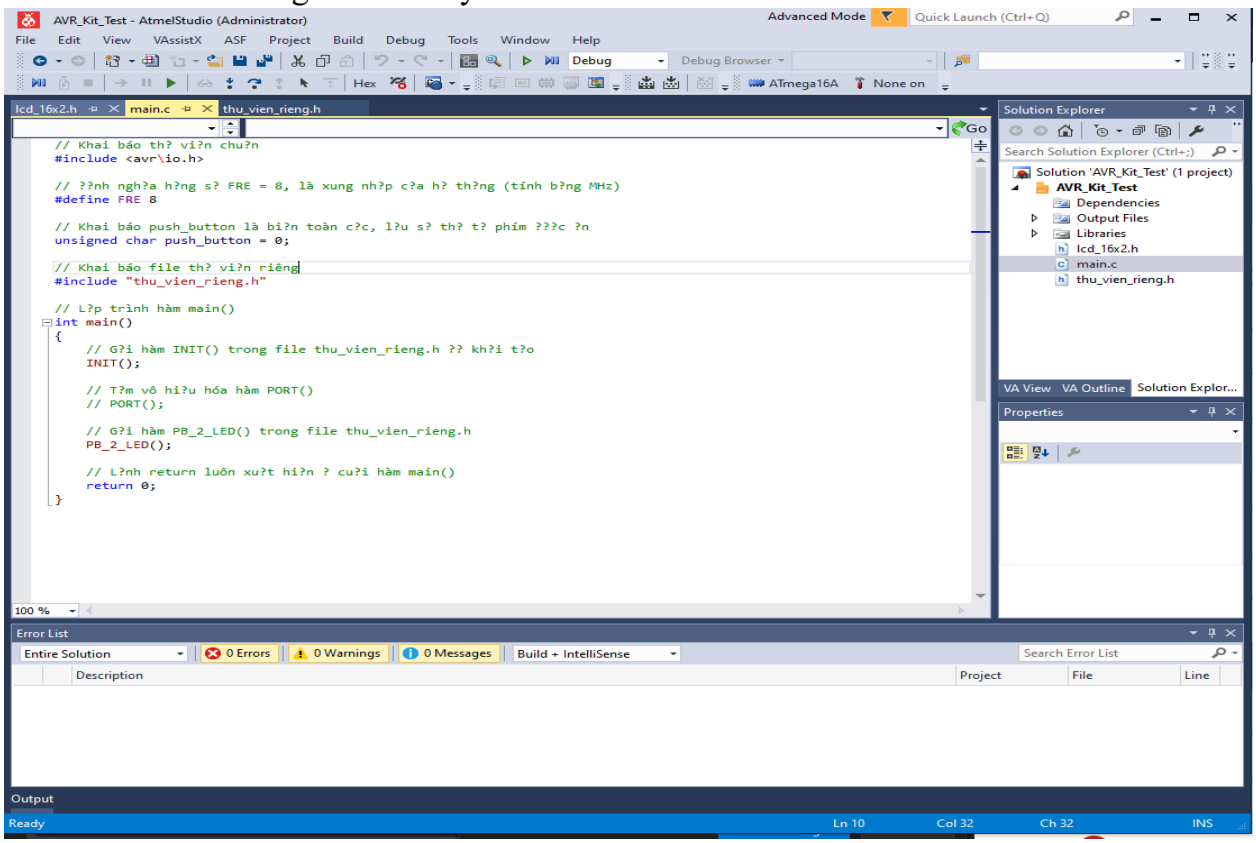


Hình 3.2 Giao diện chọn loại vi điều khiển



Hình 3.3 Giao diện làm việc chính sau khi tạo Project

Bước 3: Cuối cùng thì chương trình sẽ hiện ra file chưa hàm main của project từ đó ta có thể code chương trình theo ý muốn.



Hình 3.4 Hàm main của chương trình

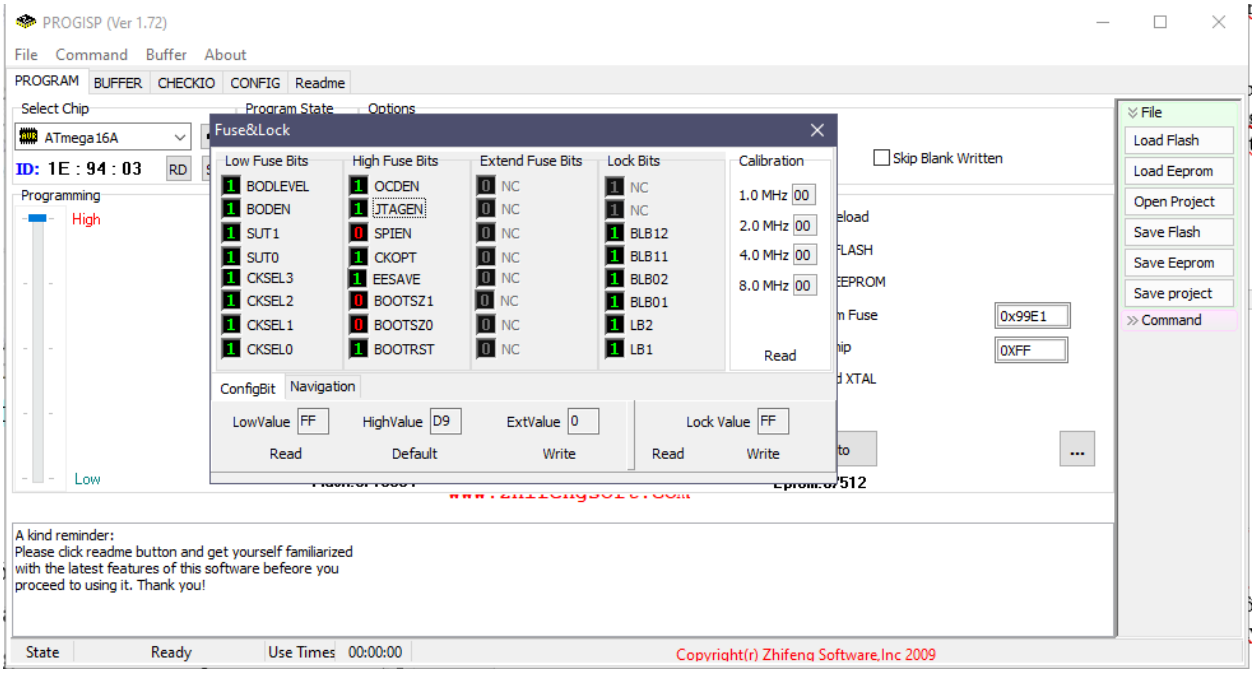
Bước 4: Copy đoạn mã nguồn vào file main.c trong project vừa tạo:

```
#include <avr\io.h>

int main()
{
   DDRD |= 0xFF;
   PORTD |= 0xAA;
   DDRC |= 0xFF;
   PORTC |= 0x00;
   return 0;
}
```

Đoạn code trên có nghĩa đang xét chế độ cho Port D và C với DDRx là mode Inputhay Output của mỗi chân vi điều khiển. 1 tức là chân đó là chân xuất tín hiệu ra vàngược lại.

Bước 5: Dịch đoạn mã nguồn trên sang mã máy bằng cách chọn Project > Build. Nếu không có lỗi gì, file mã máy "tên project".hex sẽ được tạo trong thư mục Exe của Project.

Bước 6: Sử dụng phần mềm PROGISP để chỉnh cấu hình Fuse bit cho VĐK như hình rồi chọn file hex vừa được tạo để nạp xuống VĐK. Nếu không có lỗi gì, dãy 8 LED đơn trên Kit sẽ sáng tắt xen kẽ nhau và LED 7 thanh sẽ sáng toàn bộ.



Hình 3.5 Thiết lập Fuse bit trong PROGISP

Để có thể nạp file hex vào cho vi điều khiển ta cần cắm chân nạp của usbisp vào đê có ghi isp trên bản mạch của mình.

Lưu ý: Hạn chế thay đổi Fuse&Lock bit vì thiết lập sai có thể đưa VĐK vào chế độ đặc biện – không thể khôi phục lại. Khi Fuse&Lock bit cần tích vào ô Program Fuse trong PROGISP, còn nạp file mã máy bình thường không cần tích vào ô này.

3.2 Ví dụ lập trình điều khiển cổng ra số

Mục tiêu

- Với LED đơn: khi bật nguồn, toàn bộ LED tắt. Sau mỗi 0.5 s, có thêm một LED sáng (từ trái sang phải) để tạo thành dải sáng có độ dài tăng dần. Sau khi dải sáng đạt độ dài cực đại, toàn bộ LED tắt và quy trình được lặp lại từ đầu.
- Với LED 7 thanh: khi bật nguồn, LED 7 thanh hiện số 0. Sau mỗi 0.5 s,số đếm trên LED 7 thanh tăng hêm 1 đơn vị và dấu chấm trên LED đảotrạng thái (nhấp nháy). Sau khi tăng đến 9, số đếm quay lại giá trị 0 vàquy trình được lặp lại từ đầu.

Thực hiện

Bước 1: Tạo mới Project AVR_Kit_Test như đã hướng dẫn ở trên. Trong project này ta tạo tiếp 2 file là AVR_Kit_Test.c (chưa hàm main()) và thu_vien_rieng.h

Bước 2: Trong file thu_vien_rieng.h ta tạo 4 chương trình con:

- INIT(): là chương trình con dùng để khởi tạo trạng thái cho các chân I/O của VĐK.
- PORT(): là chương trình còn dùng để bật/tắt các LED thong qua việc điều khiển các PORT của VĐK.

- LED7_OUT(num): là chương trình con dùng để điều khiển LED 7 thanh hiển thị số theo biến num ($0 \leq num \leq 9$).
- DELAY_MS(mili_count): là chương trình con dùng để tạo ra các khoảng thời gian trễ, tính bằng mili giây, giữa các lần bật/tắt LED để dễ dàng quan sát hiệu ứng (*Lưu ý: project phải chọn Optimization là -O0 thì hàm này mới hoạt động đúng!*).

Trong file AVR_Kit_Test.c ta cần khai báo các thư viện sẽ sử dụng (file *.h) – gồm thư viện chuẩn của AVR (avr/io.h) và thu_vien_rieng.h, định nghĩa các hằng số, khai báo biến toàn cục và gọi hàm main(). Trong hàm main() ta sẽ gọi hai chương trình con INIT() và PORT() từ file thu_vien_rieng.h.

Bước 3: Tiến hành biên dịch sang mã mãy (Build - F7) và nạp mã máy xuống IC VĐK như đã hướng dẫn ở trên.

Chi tiết các hàm

❖ File *AVR_Kit_Test.c*

```
// Khai báo thư viện chuẩn
#include <avr\io.h>

// Định nghĩa hằng số FRE = 8, là xung nhịp của hệ thống (tính bằng MHz)
#define FRE 8

// Khai báo file thư viện riêng
#include "thu_vien_rieng.h"

// Lập trình hàm main()
int main()
{
	// Gọi hàm INIT() trong file thu_vien_rieng.h để khởi tạo
	INIT();

	// Gọi hàm PORT() trong file thu_vien_rieng.h để điều khiển LED
	PORT();

	// Lệnh return luôn xuất hiện ở cuối hàm main()
	return 0;
}
```

❖ File *thu_vien_rieng.h*

- Hàm INIT(), để sử dụng các chân I/O của VĐK ta cần khai báo trạng thái output của các chân (1 tương ứng với trạng thái output, 0 tương ứng với trạng thái input). Sau khi đã khai báo trạng thái ta cần khai báo mức logic của các chân đó.

```
void INIT()
{
	// Khởi tạo trạng thái Output cho các chân nối tới các LED đơn
```

```
	DDRD |= 0xFF;

	// Khởi tạo trạng thái logic 1 cho các chân nối tới các LED đơn
	PORTD |= 0xFF;

	// Khởi tạo trạng thái Output cho các chân nối tới LED 7 thanh
	DDRC |= 0xFF;

	// Khởi tạo trạng thái logic 1 cho các chân nối tới LED 7 thanh
	PORTC |= 0xFF;
}
```

- Hàm PORT() đây là hàm không có tham số và không trả về giá trị, hàm điều khiển trạng thái logic của các chân trong các Port của VĐK với mục đích cho các LED sáng tắt theo yêu cầu ở trên.

```
void PORT()
{
	/* Khai báo các biến sẽ dùng tới trong hàm này */

	// Biến led_shift để điều khiển các LED đơn
	// Giá trị đầu là 255 = 0xFF = 0b11111111 -> tất cả các LED đều tắt
	unsigned char led_shift = 255;

	// Biến đếm cho LED 7 thanh, giá trị đầu là 0
	unsigned char led_7_count = 0;

	// Vòng for giúp các LED sáng/tắt theo quy luật lặp đi lặp lại
	for(;;)
	{
		/* Đoạn mã điều khiển các LED đơn */

		// Các LED sáng/tắt theo 8 bit của biến led_shift
		PORTD = led_shift;

		// Thay đổi biến led_shift
		if(led_shift != 0)                  // Nếu led_shift khác 0
			led_shift = led_shift << 1;   // Dịch trái 1 bit
		else
			led_shift = 255;              // Trở lại giá trị 255

		/* Đoạn mã điều khiển LED 7 thanh */

		// Xuất giá trị đếm ra LED 7 thanh
		LED7_OUT(led_7_count);

		// Đảo trạng thái PC3 để nhấp nháy dấu chấm trên LED 7 thanh
		PORTC ^= (1<<PC3);

		// Tăng dần giá trị đếm
		led_7_count = led_7_count + 1;
```

```
            // Khi vượt quá 9, giá trị đếm được reset về 0
            if(led_7_count > 9)
                  led_7_count = 0;

            // Hàm trễ khoảng 0.5 s = 500 ms
            DELAY_MS(500);
      }
}
```

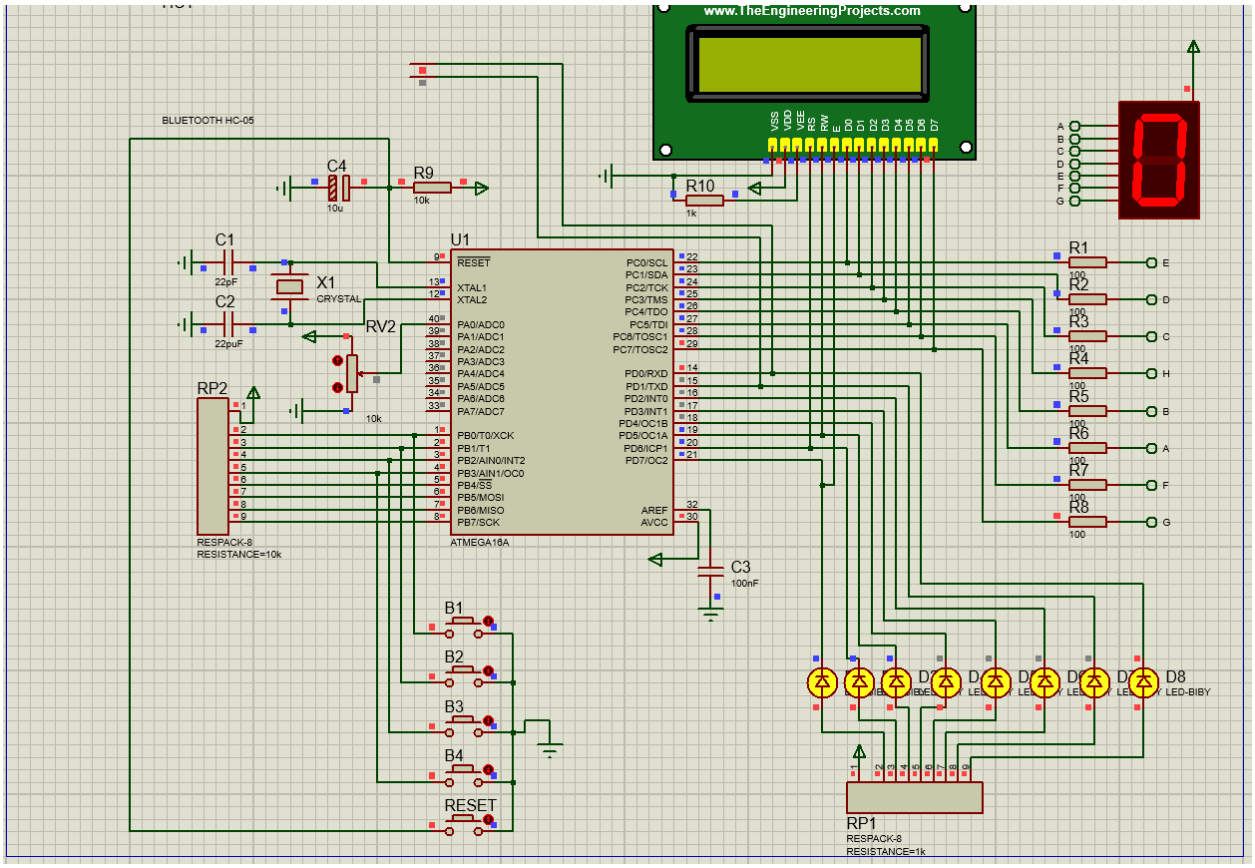
- Hàm LED7_OUT() có tham số là num và không trả về giá trị, hàm điều khiển LED 7 thanh hiển thị num bằng cách sáng tắt các thanh LED một cách phù hợp.

```
void LED7_OUT(unsigned char num)
{
      // Khai báo biến temp lưu trạng thái của PORTC
      unsigned char temp = PORTC;

      // Các chân vi điều khiển ứng với các thanh LED
      // a - PC5                    PC5
      // b - PC4              PC6          PC4
      // c - PC2              PC6          PC4
      // d - PC1                    PC7
      // e - PC0              PC0          PC2
      // f - PC6              PC0          PC2
      // g - PC7                    PC1          PC3
      // dot - PC3

      // Tắt các đoạn LED hiện đang sáng trước khi sáng các đoạn LED mới
      temp &= 0B00001000;

      // Gán mức logic cho 8 bit của biến temp ứng với giá trị của biến num
      switch(num)
      {
            case 0: temp |= 0B10000000; break;
            case 1: temp |= 0B11100011; break;
            case 2: temp |= 0B01000100; break;
            case 3: temp |= 0B01000001; break;
            case 4: temp |= 0B00100011; break;
            case 5: temp |= 0B00010001; break;
            case 6: temp |= 0B00010000; break;
            case 7: temp |= 0B11000011; break;
            case 8: temp |= 0B00000000; break;
            case 9: temp |= 0B00000001; break;
      }

      // Xuất giá trị logic mới ra PORTC để làm sáng LED 7 thanh
      PORTC = temp;
}
```

- Hàm DELAY_MS(), hàm này nhận mili_count làm tham số và không trả về giá trị, được sử dụng để tạo ra khoảng thời gian trễ tính bằng mili giây. Việc trễ được thực hiện bằng các vòng lặp rỗng, tuy không thực hiện gì nhưng vẫn làm CPU tiêu tốn một khoảng thời gian nhất định cho việc khởi tạo và kết thúc.

```
void DELAY_MS(unsigned int mili_count)
{
	// Khai báo hai biến chạy cho hai vòng for
	unsigned int i,j;

	// Xung nhịp của hệ thống càng cao, số vòng lặp càng tăng
	mili_count = mili_count * FRE;

	// Các vòng for gây trễ
	for(i = 0; i < mili_count; i++)
		for(j = 0; j < 53; j++);
}
```

❖ Nạp code và test trên proteus



Hình 3.6 Ảnh test ví dụ 3.2 trên proteus

3.3 Ví dụ lập trình đọc trạng thái logic đầu vào số

Mục tiêu

- Với LED đơn: khi bật nguồn, toàn bộ LED tắt. Nếu phím PB1 được ấn,chỉ hai LED ngoài cùng bên trái sáng. Nếu phim PB2 được ấn, chỉ hai LED tiếp theo sáng,… Nếu phim PB4 được ấn, chỉ hai LED ngoài cùng bên phải sáng.
- Với LED 7 thanh: khi bật nguồn, LED 7 thanh hiện số 0. Nếu phim PBx được ấn, LED 7 thanh hiện giá trị của x.

Thực hiện

Bước 1: Tạo một bản sao lưu của toàn bộ Project AVR_Kit_Test sau đó chỉnh sửa lại project này.

Bước 2: Trong file thu_vien_rieng.h thêm hai chương trình con:

- PB_2_LED(): chương trình con dùng để điều khiển LED theo phím ấn.
- PB_CHECK(): chương trình con dùng để nhận diện phím đang được ấn.

Trong file AVR_Kit_Test.c ta khởi tạo một biến toàn cục unsigned char push_button - 0, xóa hoặc comment hàm PORT() và gọi hàm PB_2_LED() sau INIT().

Bước 3: Tiến hành biên dịch sang mã mãy và nạp mã máy xuống IC VĐK.

Chi tiết các hàm

❖ File *AVR_Kit_Test.c*

Chỉnh sửa mã nguồn và bổ sung lệnh mới:

```
// Khai báo thư viện chuẩn
#include <avr\io.h>

// Định nghĩa hằng số FRE = 8, là xung nhịp của hệ thống (tính bằng MHz)
#define FRE 8

// Khai báo push_button là biến toàn cục, lưu số thứ tự phím được ấn
unsigned char push_button = 0;

// Khai báo file thư viện riêng
#include "thu_vien_rieng.h"

// Lập trình hàm main()
int main()
{
      // Gọi hàm INIT() trong file thu_vien_rieng.h để khởi tạo
      INIT();
```

```
	// Tạm vô hiệu hóa hàm PORT()
	// PORT();

	// Gọi hàm PB_2_LED() trong file thu_vien_rieng.h
	PB_2_LED();

	// Lệnh return luôn xuất hiện ở cuối hàm main()
	return 0;
}
```

❖ File *thu_vien_rieng.h*

Bổ sung hai nguyên mẫu hàm

```
// Khai báo các nguyên mẫu hàm
void INIT();
void PORT();
void LED7_OUT(unsigned char num);
void DELAY_MS(unsigned int mili_count);
void PB_2_LED();
unsigned char PB_CHECK();
```

Bổ sung hai hàm

- Hàm PB_2_LED() là hàm không có tham số và không trả về giá trị, được sửdụng để điều khiển LED theo phím ấn theo quy tắc nhất định đã được mô tả ở trên.

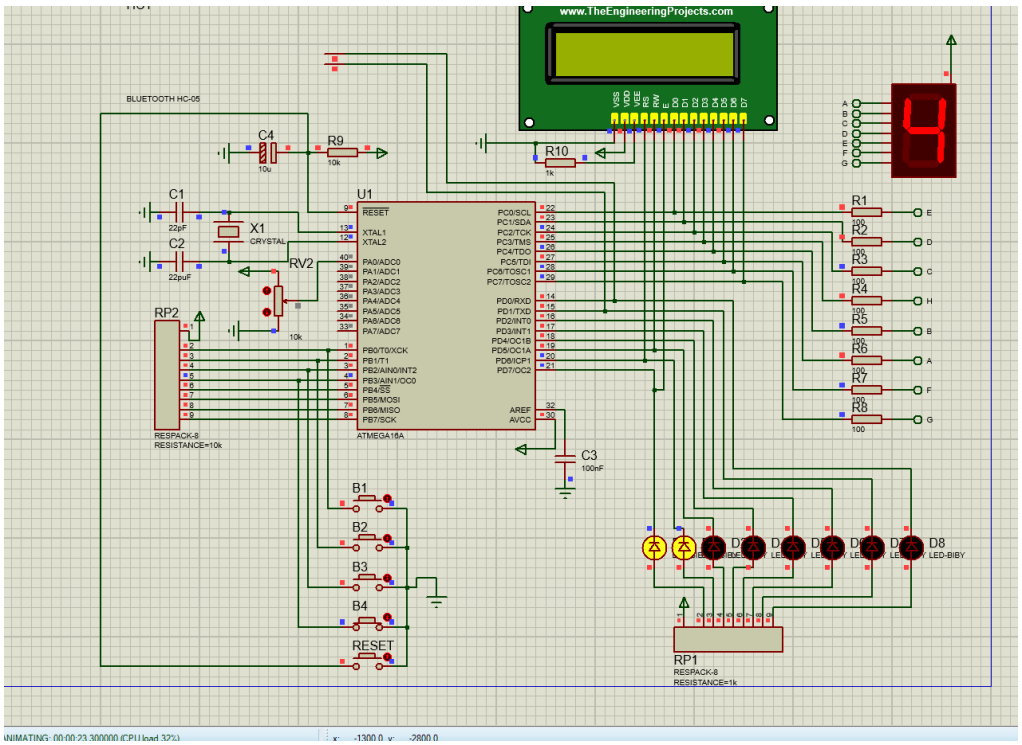
```
void PB_2_LED()
{
	// Vòng for giúp việc quét phím ấn được lặp đi lặp lại
	for(;;)
	{
		// Gọi hàm quét phím, lưu kết quả phím ấn vào biến push_button
		push_button = PB_CHECK();

		// Hiện số thứ tự phím ấn ra LED 7 thanh
		LED7_OUT(push_button);

		// Điều khiển hàng LED đơn
		switch (push_button)
		{
			// Nếu push_button = 1, sáng 2 LED ngoài cùng bên trái
			case 1: PORTD = 0b11111100; break;

			// Nếu push_button = 2, …
			case 2: PORTD = 0b11110011; break;
			case 3: PORTD = 0b11001111; break;
			case 4: PORTD = 0b00111111; break;

			// push_button = 0, tắt tất cả các LED
			default: PORTD = 0xFF;
```

```
            }
        }
}
```

- Hàm PB_CHECK() là hàm không có tham số và có trả về giá trị, được sử dụng để nhận diễn các phím ấn. Giá trị trả về của hàm là thứ tự các phím ấn. Khi được ấn, phím sẽ kết nối chân tương ứng của VĐK với GND (mức logic 0). Khi nhả phím, chân tương ứng của VĐK sẽ được treo lên mức 1 nhờ các điện trở kéo có sẵn. Hàm trả về 0 khi không có phím nào được ấn (khi không có phím nào được ấn thì trạng thái LED sẽreset), trong trường hợp hàm trả về push_button thì trạng thái của các LED sẽ được giữ cho đến khi phím khác được ấn. (push_button là biến toàn cục lưu giá trị trả về của hàm PB_CHECK()).

```
unsigned char PB_CHECK()
{
      // Kiểm tra trạng thái logic của 4 chân PB0-3. Nếu khác 1111
      if((PINB & 0x0F) != 0x0F)
      {
            // Kiểm tra PB0. Nếu là mức logic 0, hàm kết thúc và = 1
            if(!(PINB & (1<<PB0)))
                  return 1;

            // Kiểm tra PB1. Nếu là mức logic 0, hàm kết thúc và = 2
            if(!(PINB & (1<<PB1)))
                  return 2;

            // Kiểm tra PB2. Nếu là mức logic 0, hàm kết thúc và = 3
            if(!(PINB & (1<<PB2)))
                  return 3;

            // Kiểm tra PB3. Nếu là mức logic 0, hàm kết thúc và = 4
            if(!(PINB & (1<<PB3)))
                  return 4;
      }

      // Nếu không có phím nào được ấn, hàm kết thúc và = 0
      return 0;
}
```

❖ Nạp code và test trên proteus



Hình 3.7 Ảnh test ví dụ 3.3 trên proteus

CHƯƠNG 4. VẬN DỤNG CÁC KIẾN THỨC VÀO THỰC TẾ

4.1 Mục tiêu

Sau quá trình làm quen với các công cụ thiết kế mạch điện và thực hành lập trình phần cứng dựa trên các mã nguồn mẫu, em sẽ vận dụng những kiến thức đã học để thiết kế và hoàn thiện một hệ thống đáp ứng yêu cầu đề ra. Nội dung thực hiện tập trung vào việc xây dựng một ứng dụng cụ thể trên nền tảng vi điều khiển, bao gồm:

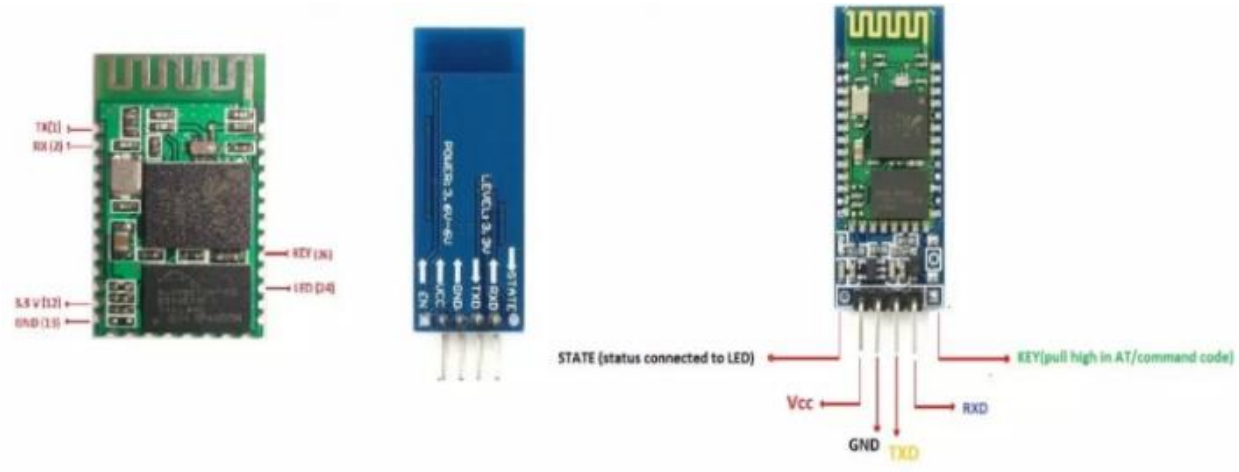
- Thiết kế mạch phần cứng sử dụng vi điều khiển họ AVR.
- Xây dựng hệ thống điều khiển thiết bị từ xa thông qua công nghệ truyền thông không dây Bluetooth.
- Thực hành cấu hình và sử dụng giao tiếp nối tiếp UART để truyền nhận dữ liệu giữa vi điều khiển và module Bluetooth.

Trong học phần này, em lựa chọn đề tài "*Thiết kế mạch điều khiển thiết bị từ xa qua Bluetooth*" làm nội dung vận dụng kiến thức vào thực tế.

Mục tiêu: *Thiết kế và chế tạo một mạch điện cho phép người dùng điều khiển bật/tắt thiết bị từ xa thông qua kết nối Bluetooth*, sử dụng module *HC-05* kết hợp với vi điều khiển *ATmega16A*. Hệ thống đảm bảo hoạt động ổn định trong phạm vi kết nối Bluetooth tiêu chuẩn, có giao diện điều khiển đơn giản và dễ sử dụng, phù hợp với các ứng dụng điều khiển thiết bị dân dụng cơ bản.

4.2 Module Bluetooth HC-05

Module HC-05 là một module Bluetooth phổ biến, được sử dụng rộng rãi trong các ứng dụng truyền thông không dây tầm ngắn. Với ưu điểm như kích thước nhỏ gọn, giá thành thấp, dễ cấu hình và khả năng giao tiếp nối tiếp UART trực tiếp với vi điều khiển, HC-05 rất phù hợp cho các hệ thống điều khiển từ xa cơ bản.



Hình 4.1 Module Bluetooth HC-05

HC-05 hoạt động ở chế độ Bluetooth cổ điển (Bluetooth Classic), cho phép kết nối không dây giữa vi điều khiển và các thiết bị ngoại vi như điện thoại thông minh hoặc máy tính. Module hỗ trợ hai chế độ hoạt động là Master và Slave, trong đó chế độ Slave thường được sử dụng để nhận lệnh điều khiển từ thiết bị bên ngoài. Dữ liệu điều khiển được truyền dưới dạng chuỗi ký tự thông qua giao tiếp UART, giúp việc lập trình và xử lý dữ liệu trở nên đơn giản.

Thông số kĩ thuật HC-05:

- Chuẩn Bluetooth: Bluetooth 2.0 + EDR
- Khoảng cách truyền: Tối đa 10 mét (trong điều kiện lý tưởng)
- Tần số hoạt động: 2.4 GHz
- Điện áp hoạt động: 3.6V đến 6V (thích hợp với 5V)
- Dòng tiêu thụ:
    - Khi kết nối: khoảng 30 mA
    - Khi ngủ: khoảng 0.1 mA
- Tốc độ truyền dữ liệu: Từ 1200 bps đến 115200 bps (thường sử dụng 9600 bps)
- Giao tiếp: UART (Serial Communication)
- Kích thước: Khoảng 27mm x 13mm x 2.2mm
- Đặc điểm khác: Dễ dàng cấu hình qua lệnh AT.

4.3 Thiết kế

4.3.1. Yêu cầu đặt ra

- Điều khiển phản hồi nhanh và chính xác
- Mạch hoạt động ổn định
- Màn hình hiển thị rõ nét

4.3.2. Các khối trong hệ thống

- Khối nguồn cung cấp điện áp 5V cho toàn bộ mạch
    - Vi điều khiển ATmega16A
    - Module Bluetooth HC-05
    - Màn hình LCD
    - LED và LED 7 thanh

- Khối xử lý trung tâm ATmega16A được có cấu hình:
    - PORTB: nhận tín hiệu từ các nút nhấn
    - PORTC: điều khiển LED 7 thanh và dữ liệu LCD
    - PORTD: điều khiển LED đơn và giao tiếp UART

- Khối Bluetooth
  Module HC-05 giao tiếp với ATmega16A thông qua UART:
    - TX (HC-05) → RX (ATmega16A)

- RX (HC-05) → TX (ATmega16A)

Hoạt động:

- Điện thoại kết nối Bluetooth với HC-05
- Ứng dụng gửi ký tự điều khiển thay cho công tắc bật/tắt
- ATmega16A nhận dữ liệu UART và giải mã lệnh:
  - '1' → Bật thiết bị
  - '0' → Tắt thiết bị

- Khối nút nhấn
  Các nút nhấn được nối với các chân PB0 – PB3 của ATmega16A, hoạt động theo nguyên lý:
  - Khi nhấn nút, chân I/O bị kéo xuống mức logic 0
  - Khi không nhấn, chân ở mức logic 1 nhờ điện trở kéo lên nội

  Trong mạch:

  - Nút PB1: đưa hệ thống về trạng thái bật
  - Nút PB2: đưa hệ thống về trạng thái tắt

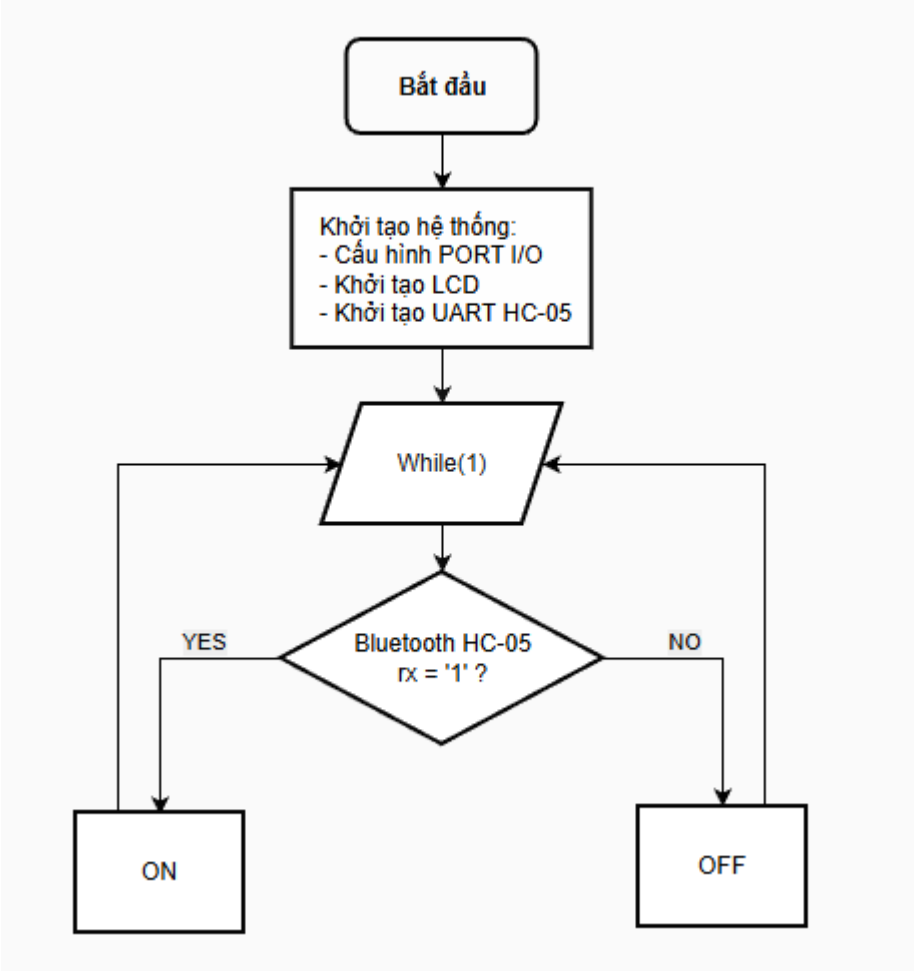
- Khối hiển thị LCD
  Màn hình LCD 16×2 hoạt động ở chế độ 8 bit, nhận dữ liệu từ PORTC và các chân điều khiển:
  - RS: chọn lệnh hoặc dữ liệu
  - RW: chọn chế độ đọc/ghi
  - E: cho phép truyền dữ liệu

  LCD hiển thị trạng thái hệ thống:

  - "DEVICE: ON"
  - "DEVICE: OFF"

4.3.3. Phần mềm

4.3.3.1. Lưu đồ thuật toán



Hình 4.2 Lưu đồ thuật toán hệ thống

Thuật toán hệ thống bắt đầu bằng việc khởi tạo các ngoại vi cần thiết như LCD, UART và các cổng vào/ra của vi điều khiển ATmega16A. Sau khi khởi tạo, chương trình hiển thị trạng thái ban đầu của thiết bị và đi vào vòng lặp vô hạn.

Trong vòng lặp, hệ thống liên tục kiểm tra trạng thái nút nhấn và dữ liệu nhận được từ module Bluetooth HC-05. Dựa vào lệnh điều khiển nhận được, hệ thống cập nhật trạng thái ON/OFF của thiết bị, điều khiển LED, LED 7 thanh và hiển thị trạng thái tương ứng lên màn hình LCD.

4.3.3.2. File main.c

Code main.c

```
#include "thu_vien_rieng.h"

int main(void)

{

	/* DEVICE LED ON */

	DDRD |= (1<<PD2);

	/* BUTTON */
```

```
DDRB &= ~((1<<PB0)|(1<<PB1));
PORTB |= (1<<PB0)|(1<<PB1);

LCD_Init();
UART_Init();

LCD_String("VA BLUETOOTH");
LCD_GotoXY(1,0);
LCD_String("DEVICE: READY");
char rx;
while (1)
{
	/* Bluetooth */
	rx = UART_Read();
	if (rx == '1')
	{
		LED_On();
		LCD_GotoXY(1,0);
		LCD_String("DEVICE: ON ");
	}
	else if (rx == '0')
	{
		LED_Off();
		LCD_GotoXY(1,0);
		LCD_String("DEVICE: OFF");
	}
	/* Button */
	if (button_press(PB0))
	{
		LED_On();
		LCD_GotoXY(1,0);
		LCD_String("DEVICE: ON ");
	}
	if (button_press(PB1))
	{
```

```
                    LED_Off();
                    LCD_GotoXY(1,0);
                    LCD_String("DEVICE: OFF");
                }
            }
}
```

Các hàm chính được sử dụng trong chương trình

- Hàm khởi tạo
  - `LCD_Init()`: khởi tạo màn hình LCD 16×2, chuẩn bị cho việc hiển thị thông tin.
  - `UART_Init()`: khởi tạo giao tiếp UART để truyền nhận dữ liệu với module Bluetooth HC-05.
- Hàm giao tiếp Bluetooth
  - UART_Read(): đọc dữ liệu nhận được từ module HC-05 thông qua giao tiếp UART.
- Hàm điều khiển thiết bị
  - LED_On(): bật thiết bị (LED mô phỏng).
  - LED_Off(): tắt thiết bị.
- Hàm xử lý nút ấn
  - button_press(): kiểm tra trạng thái nút nhấn và thực hiện chống dội phím bằng phần mềm.
- Hàm hiển thị
  - `LCD_String()`: hiển thị chuỗi ký tự lên LCD.
  - `LCD_GotoXY()`: điều khiển vị trí con trỏ hiển thị trên LCD.

Vòng lập vô hạn

Vòng lặp while(1) cho phép hệ thống hoạt động liên tục, luôn sẵn sàng tiếp nhận lệnh điều khiển từ Bluetooth hoặc nút nhấn. Mọi thay đổi trạng thái của thiết bị đều được cập nhật tức thời lên LCD, đảm bảo tính trực quan và dễ theo dõi cho người sử dụng.

4.3.3.3. File thu_vien_rieng

❖ Code *thu_vien_rieng.h*

```
#ifndef THU_VIEN_RIENG_H
#define THU_VIEN_RIENG_H
#include <avr/io.h>
#include <util/delay.h>
```

```
/* ========= LCD ========= */
#define LCD_DATA_PORT PORTC
#define LCD_DATA_DDR  DDRC

#define LCD_CTRL_PORT PORTD
#define LCD_CTRL_DDR  DDRD
#define LCD_RS PD6
#define LCD_RW PD5
#define LCD_EN PD7

void LCD_Init(void);
void LCD_Command(unsigned char cmd);
void LCD_Data(unsigned char data);
void LCD_String(char *str);
void LCD_Clear(void);
void LCD_GotoXY(unsigned char row, unsigned char col);

/* ========= UART - HC05 ========= */
void UART_Init(void);
char UART_Read(void);
void UART_Write(char data);

/* ========= BUTTON ========= */
uint8_t button_press(uint8_t pin);

/* ========= LED ========= */
void LED_On(void);
void LED_Off(void);

#endif
```

❖ Code *thu_vien_rieng.c*

- Nhóm điều khiển LCD

Nhóm này đảm nhiệm việc **hiển thị thông tin trạng thái hệ thống** lên màn hình LCD 16×2, giúp người dùng theo dõi tình trạng thiết bị.

- LCD_Command(): Dùng để gửi các **lệnh điều khiển** đến LCD như xóa màn hình, thiết lập chế độ hiển thị, dịch con trỏ,…
- LCD_Data(): Dùng để gửi **dữ liệu ký tự** cần hiển thị lên LCD.
- LCD_String(): Hiển thị **chuỗi ký tự** bằng cách gửi từng ký tự lên LCD, phục vụ việc hiển thị thông báo như *DEVICE: ON/OFF*.
- LCD_Clear(): Xóa toàn bộ nội dung hiển thị trên LCD để chuẩn bị cho nội dung mới.
- LCD_GotoXY(): Đặt vị trí con trỏ tại hàng và cột mong muốn trên LCD, giúp hiển thị thông tin đúng vị trí.
- LCD_Init(): Khởi tạo LCD ở chế độ **8 bit, 2 dòng**, bật hiển thị và cấu hình chế độ hoạt động ban đầu.

```
/* ================ LCD ================ */
void LCD_Command(unsigned char cmd)
{
    LCD_DATA_PORT = cmd;
    LCD_CTRL_PORT &= ~(1<<LCD_RS);
    LCD_CTRL_PORT &= ~(1<<LCD_RW);
    LCD_CTRL_PORT |=  (1<<LCD_EN);
    _delay_ms(1);
    LCD_CTRL_PORT &= ~(1<<LCD_EN);
    _delay_ms(2);
}

void LCD_Data(unsigned char data)
{
    LCD_DATA_PORT = data;
    LCD_CTRL_PORT |=  (1<<LCD_RS);
    LCD_CTRL_PORT &= ~(1<<LCD_RW);
    LCD_CTRL_PORT |=  (1<<LCD_EN);
    _delay_ms(1);
    LCD_CTRL_PORT &= ~(1<<LCD_EN);
    _delay_ms(2);
```

```
}

void LCD_String(char *str)
{
    while(*str)
    LCD_Data(*str++);
}

void LCD_Clear(void)
{
    LCD_Command(0x01);
    _delay_ms(2);
}

void LCD_GotoXY(unsigned char row, unsigned char col)
{
    if (row == 0)
    LCD_Command(0x80 + col);
    else
    LCD_Command(0xC0 + col);
}

void LCD_Init(void)
{
    LCD_DATA_DDR = 0xFF;
    LCD_CTRL_DDR |= (1<<LCD_RS)|(1<<LCD_RW)|(1<<LCD_EN);

    _delay_ms(20);
    LCD_Command(0x38);   // 8-bit, 2 line
    LCD_Command(0x0C);   // Display ON
    LCD_Command(0x06);   // Auto increment
    LCD_Clear();
}
```

- **Nhóm giao tiếp UART – Bluetooth HC-05**

Nhóm này thực hiện việc nhận lệnh điều khiển không dây từ điện thoại thông qua module Bluetooth HC-05.

- UART_Init(): Khởi tạo giao tiếp UART với tốc độ 9600 baud, phù hợp với cấu hình mặc định của HC-05.
- UART_Read(): Kiểm tra và đọc ký tự nhận được từ Bluetooth.
  Trong chương trình, ký tự '1' tương ứng lệnh bật thiết bị, '0' tương ứng lệnh tắt thiết bị.
- UART_Write(): Gửi dữ liệu từ vi điều khiển đến Bluetooth khi cần phản hồi.

```
/* ================= UART – HC05 ================= */
void UART_Init(void)
{
    UBRRL = 51; // 9600 baud @ 8MHz
    UCSRB = (1<<RXEN)|(1<<TXEN);
    UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
}

char UART_Read(void)
{
    if (UCSRA & (1<<RXC))
    return UDR;
    return 0;
}

void UART_Write(char data)
{
    while(!(UCSRA & (1<<UDRE)));
    UDR = data;
}
```

- **Nhóm xử lý nút nhấn**

Nhóm này cho phép **điều khiển thiết bị thủ công**, đồng thời đảm bảo độ ổn định khi nhấn nút.

- button_press(): Kiểm tra trạng thái nút nhấn tại chân tương ứng, sử dụng **delay để chống dội phím**, đảm bảo mỗi lần nhấn chỉ được ghi nhận một lần.

```
/* ================ BUTTON ================ */
```

```
uint8_t button_press(uint8_t pin)
{
    if (!(PINB & (1<<pin)))
    {
          _delay_ms(20);
          if (!(PINB & (1<<pin)))
          {
                while (!(PINB & (1<<pin)));
                return 1;
          }
    }
    return 0;
}
```

- **Nhóm điều khiển thiết bị (LED mô phỏng)**

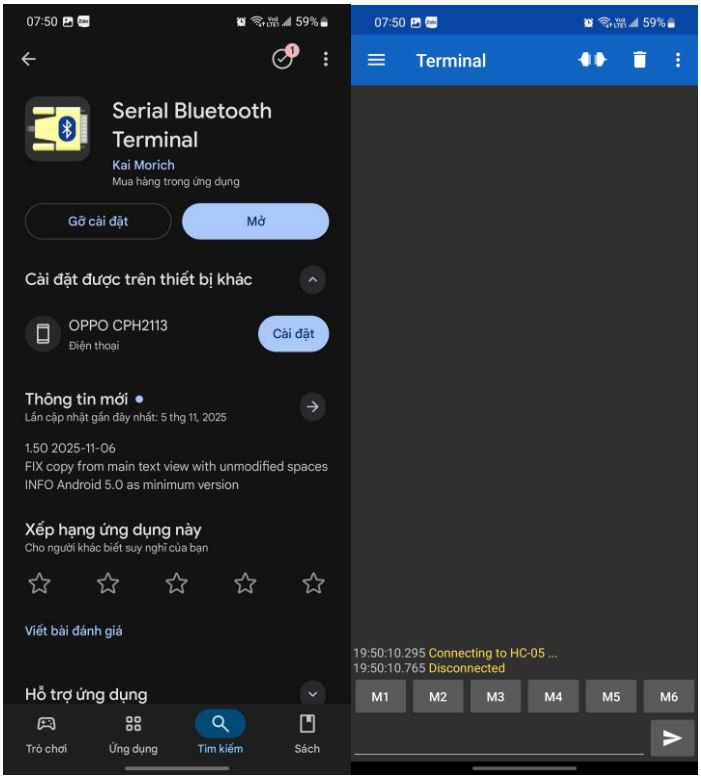
  Nhóm này thực hiện việc **bật/tắt thiết bị** được mô phỏng bằng LED.

  - LED_On(): Đặt mức logic cao tại chân điều khiển LED, tương ứng với trạng thái thiết bị **ON**.
  - LED_Off(): Đặt mức logic thấp tại chân điều khiển LED, tương ứng với trạng thái thiết bị **OFF**.

```
/* ================= LED ================= */
void LED_On(void)
{
    PORTD |= (1<<PD2);
}
void LED_Off(void)
{
    PORTD &= ~(1<<PD2);
}
```

4.3.3.4. App trên Android

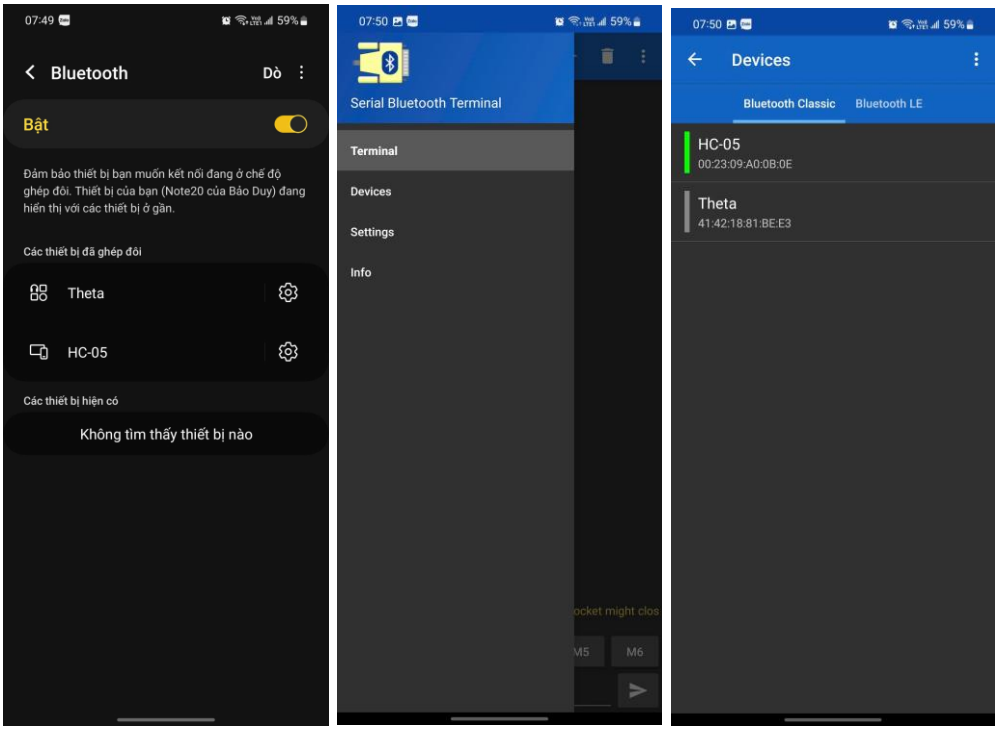
Sử dụng app Serial Bluetooth Terminal của Kai Morich



Hình 4.3 App Serial Bluetooth Terminal

Kết nối Bluetooth với HC-05 (máy Android)

- Nhập mã pin 0000 hoặc 1234 để ghép nối
- Vào app chọn Devices, chọn HC-05



Hình 4.4 Kết nối App Serial Bluetooth Terminal

4.3.4. Phần cứng

Nối dây Module Bluetooth HC-05 với ATmega16A

Module Bluetooth HC-05 giao tiếp với vi điều khiển thông qua chuẩn UART (nối tiếp không đồng bộ). ATmega16A sử dụng UART phần cứng với hai chân:

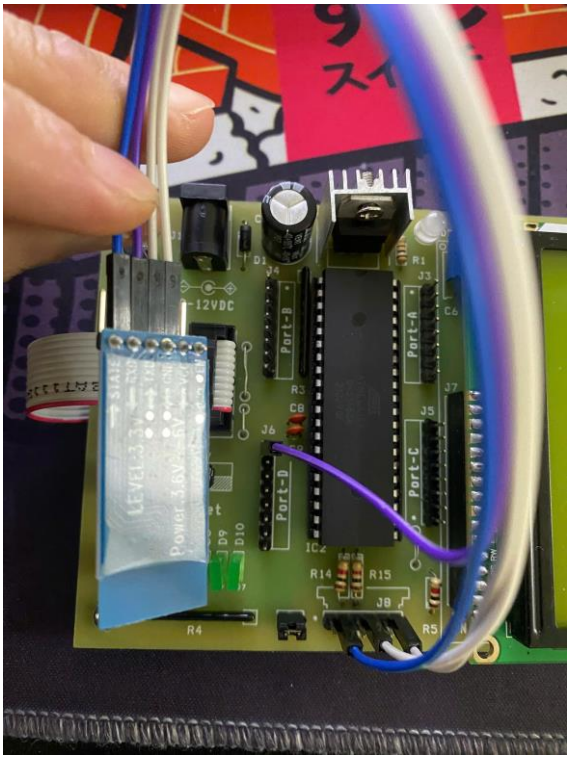
- TXD (PD1): truyền dữ liệu
- RXD (PD0): nhận dữ liệu

Khi kết nối cần đảm bảo:

- TX của HC-05 nối với RX của ATmega16A
- RX của HC-05 nối với TX của ATmega16A
- Chung mass (GND)

Sơ đồ nối chân HC-05 với ATmega16A

HC-05	Chức năng	ATmega16A
VCC	Nguồn 5V	VCC (5V)
GND	Mass	GND
TXD	Truyền dữ liệu	PD0 (RXD)
RXD	Nhận dữ liệu	PD1 (TXD) (chân RX trong mạch đã qua trở R14)_



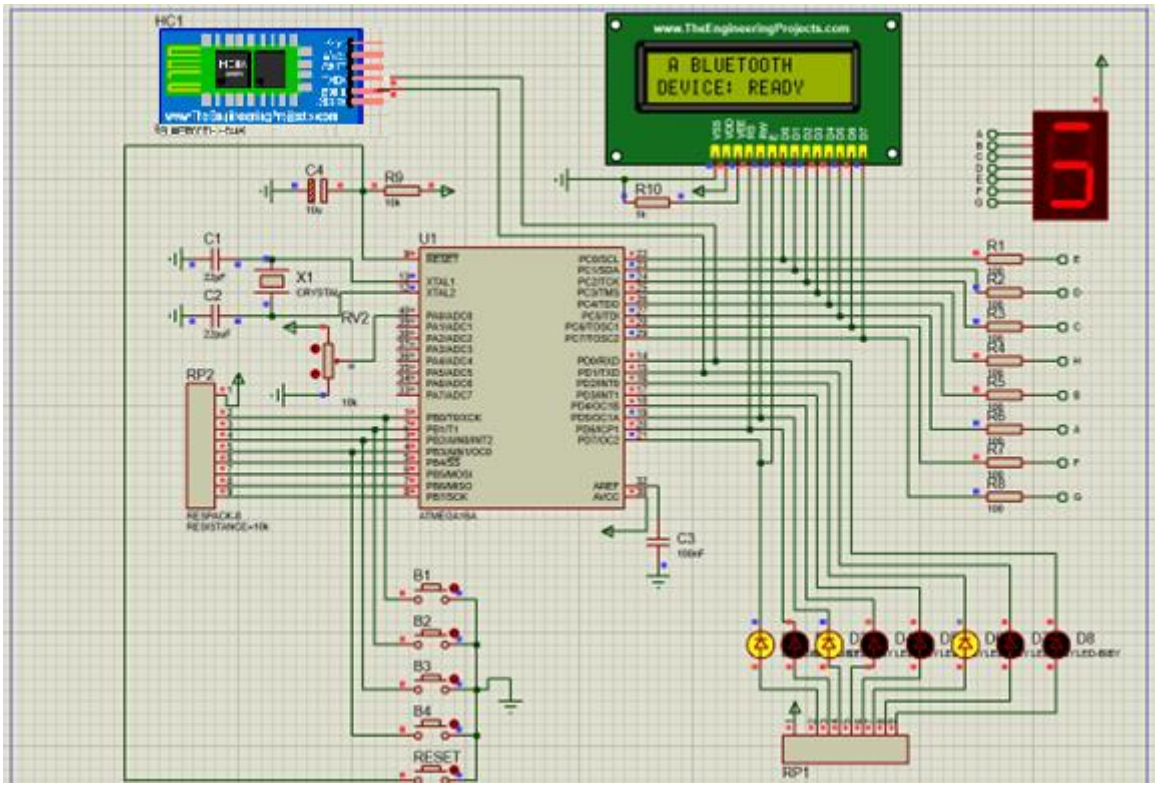
Hình 4.5 Sơ đồ nối chân

4.3.5. Nguyên lý hoạt động của hệ thống

- Sau khi cấp nguồn, vi điều khiển ATmega16A tiến hành **khởi tạo các ngoại vi** gồm:
  - Màn hình LCD 16×2
  - Giao tiếp UART để kết nối với Bluetooth HC-05
  - Các chân vào/ra điều khiển LED và nút nhấn
- Module Bluetooth HC-05 thiết lập kết nối không dây với điện thoại thông minh thông qua ứng dụng Bluetooth Terminal.
- Người dùng gửi lệnh điều khiển từ điện thoại:
  - Ký tự **'1'**: lệnh bật thiết bị
  - Ký tự **'0'**: lệnh tắt thiết bị
- Vi điều khiển nhận dữ liệu từ HC-05 thông qua giao tiếp UART, sau đó:
  - Phân tích ký tự nhận được
  - Thực hiện bật hoặc tắt LED tương ứng với trạng thái thiết bị
- Trạng thái hoạt động của hệ thống được **hiển thị trực tiếp trên LCD**, giúp người dùng theo dõi:
  - DEVICE: ON
  - DEVICE: OFF
- Ngoài điều khiển qua Bluetooth, hệ thống còn hỗ trợ **điều khiển bằng nút nhấn**:
  - Nút ON: bật thiết bị
  - Nút OFF: tắt thiết bị
    Chức năng này đảm bảo hệ thống vẫn hoạt động khi không có kết nối Bluetooth.
- Chương trình hoạt động liên tục trong vòng lặp vô hạn, cho phép hệ thống **phản hồi tức thời** với các lệnh điều khiển từ Bluetooth hoặc nút nhấn.

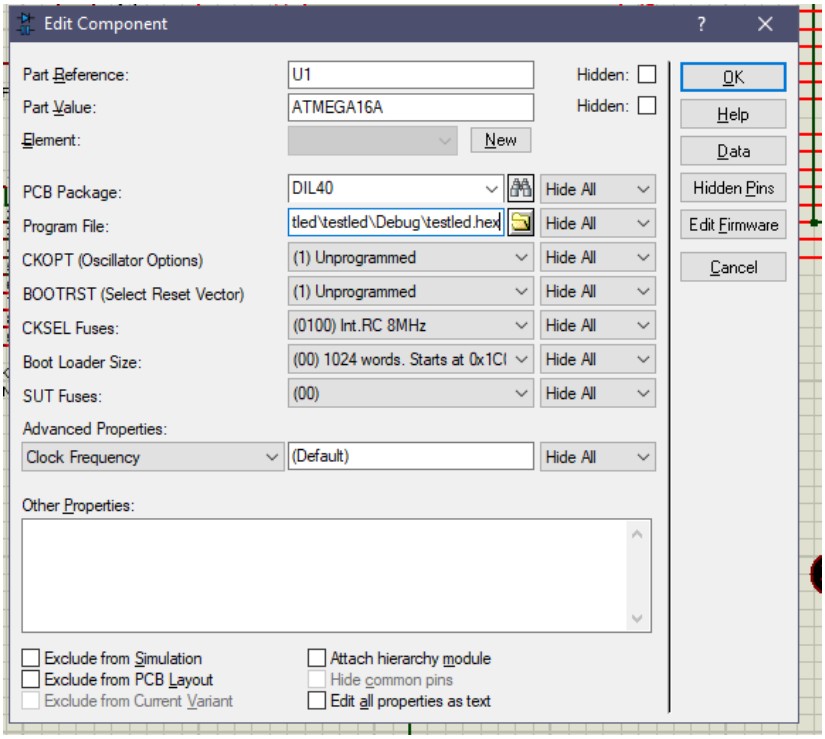
4.4 Mô phỏng

4.4.1. Mô phỏng trên phần mềm proteus 8

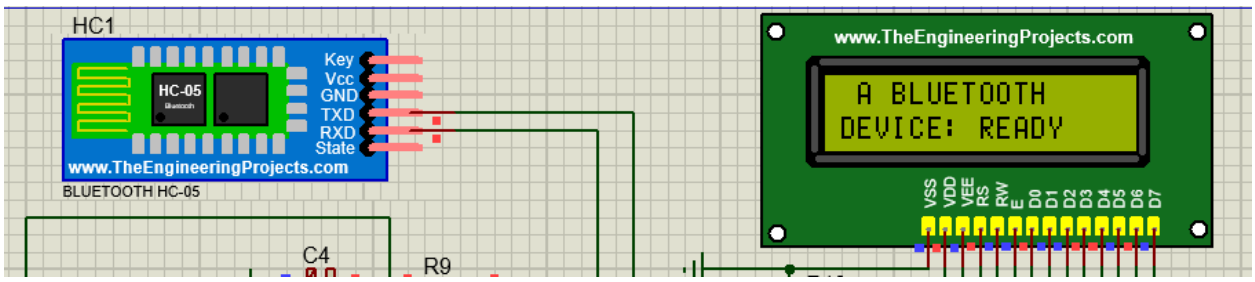


Hình 4.6 Mô phỏng mạch Blutooth trên Proteus

Lưu ý: Tiến hành biên dịch sang mã máy và mô phỏng bằng cách, click chuộtvào VĐK trong Proteus, trong bảng hiện ra tại Program file chọn đường dẫn đến file.hex vừa được dịch. Sau đó bấm OK và Debug > Run Simulation.

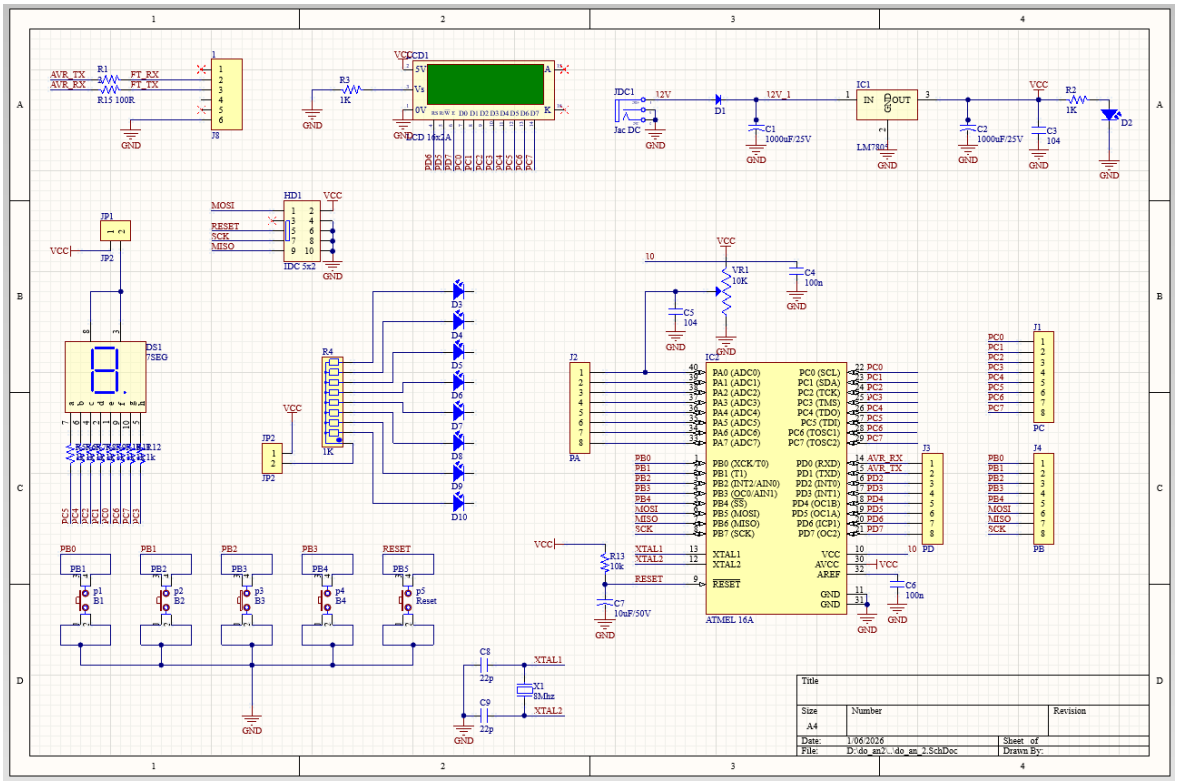


Hình 4.7 Thiết lập chọn file mã máy cho VĐK để mô phỏng trong Proteus

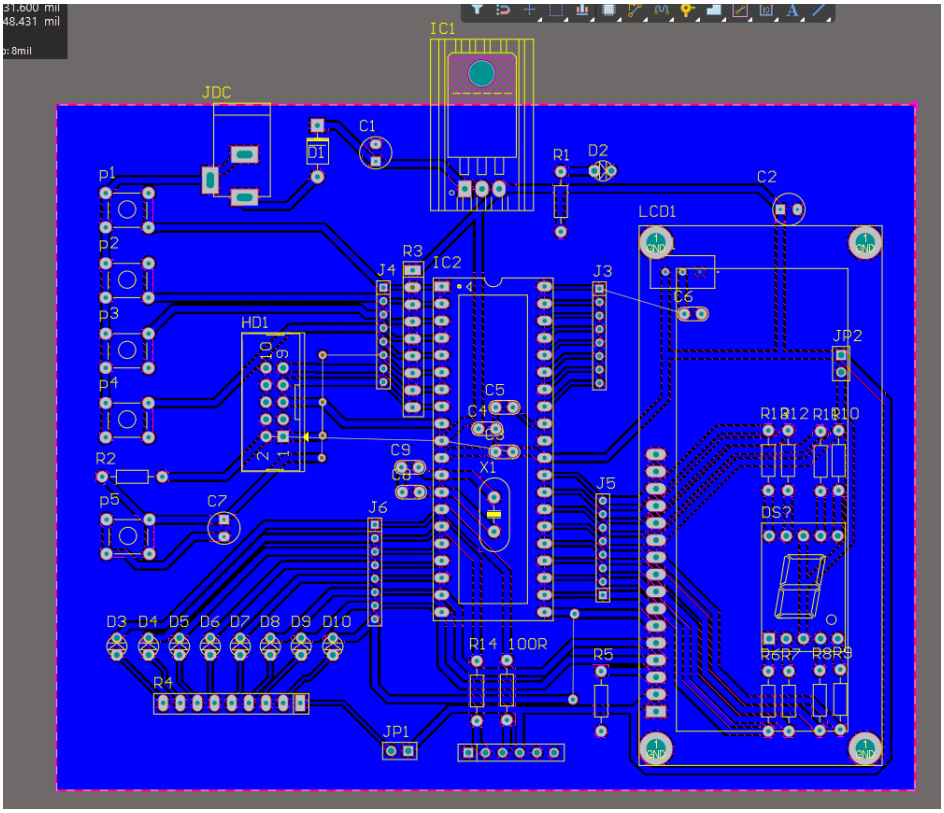


Hình 4.8 Kết quả mô phỏng trên Proteus

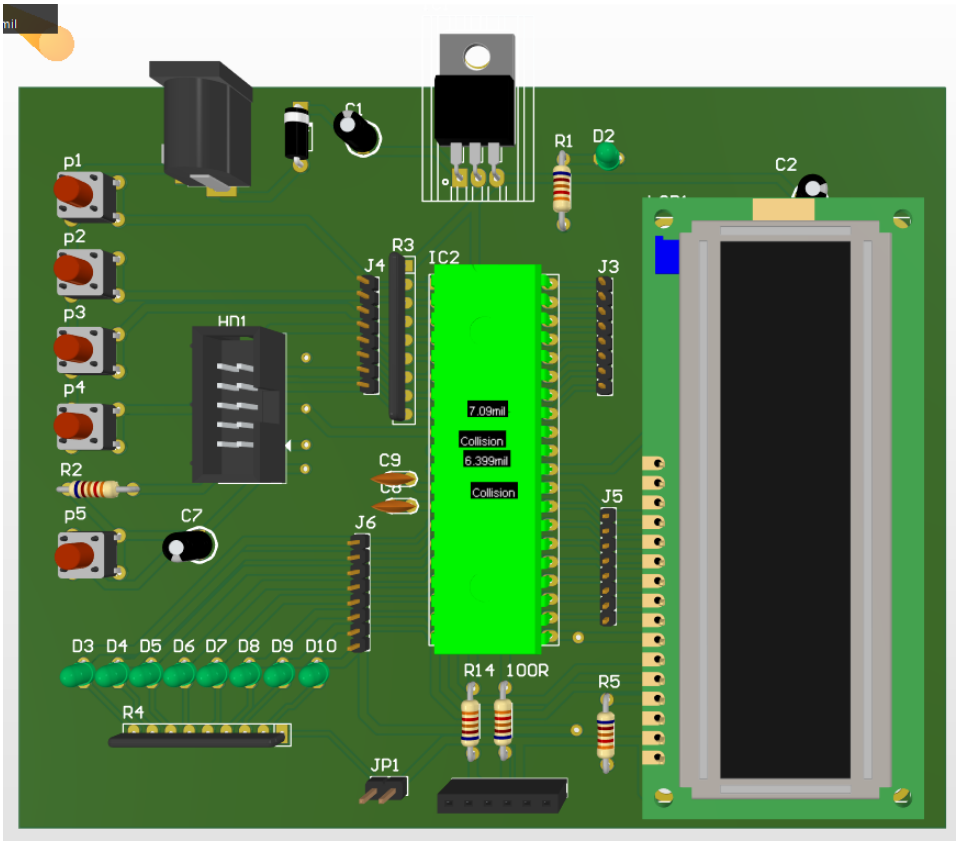
4.4.2. Thiết kế sơ đồ mạch kit AVR trên phần mềm Altium



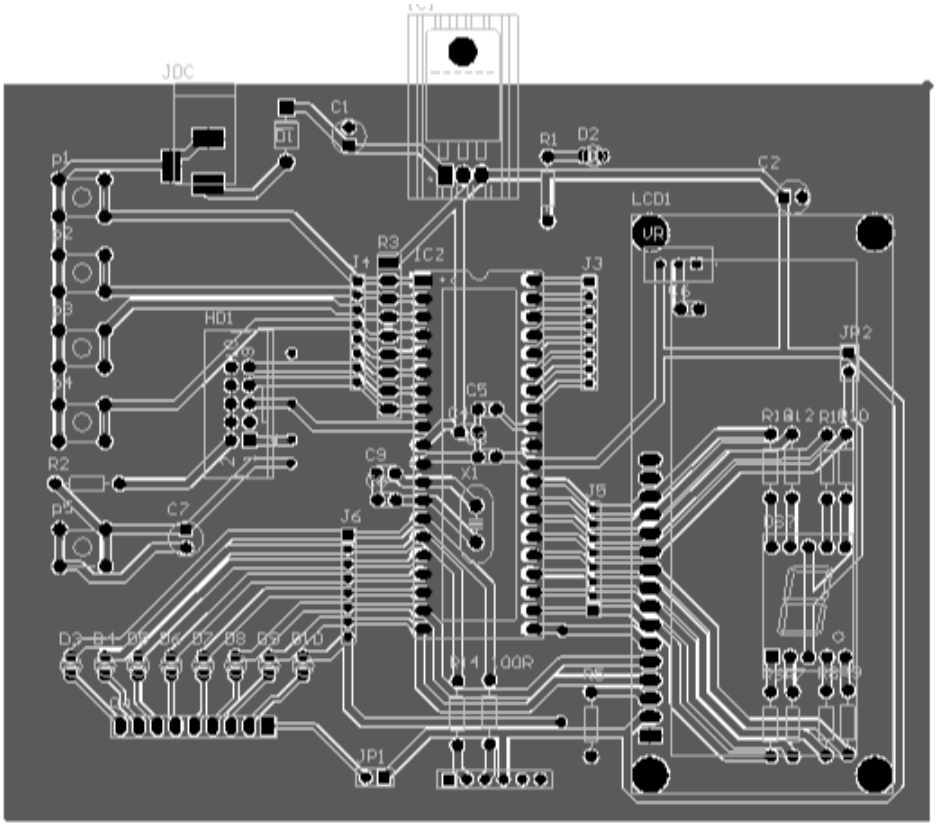
Hình 4.9 Sơ đồ nguyên lí trên Altium



Hình 4.10 Mạch PCB trên Altium



Hình 4.11 Mô hình 3D trên Altium

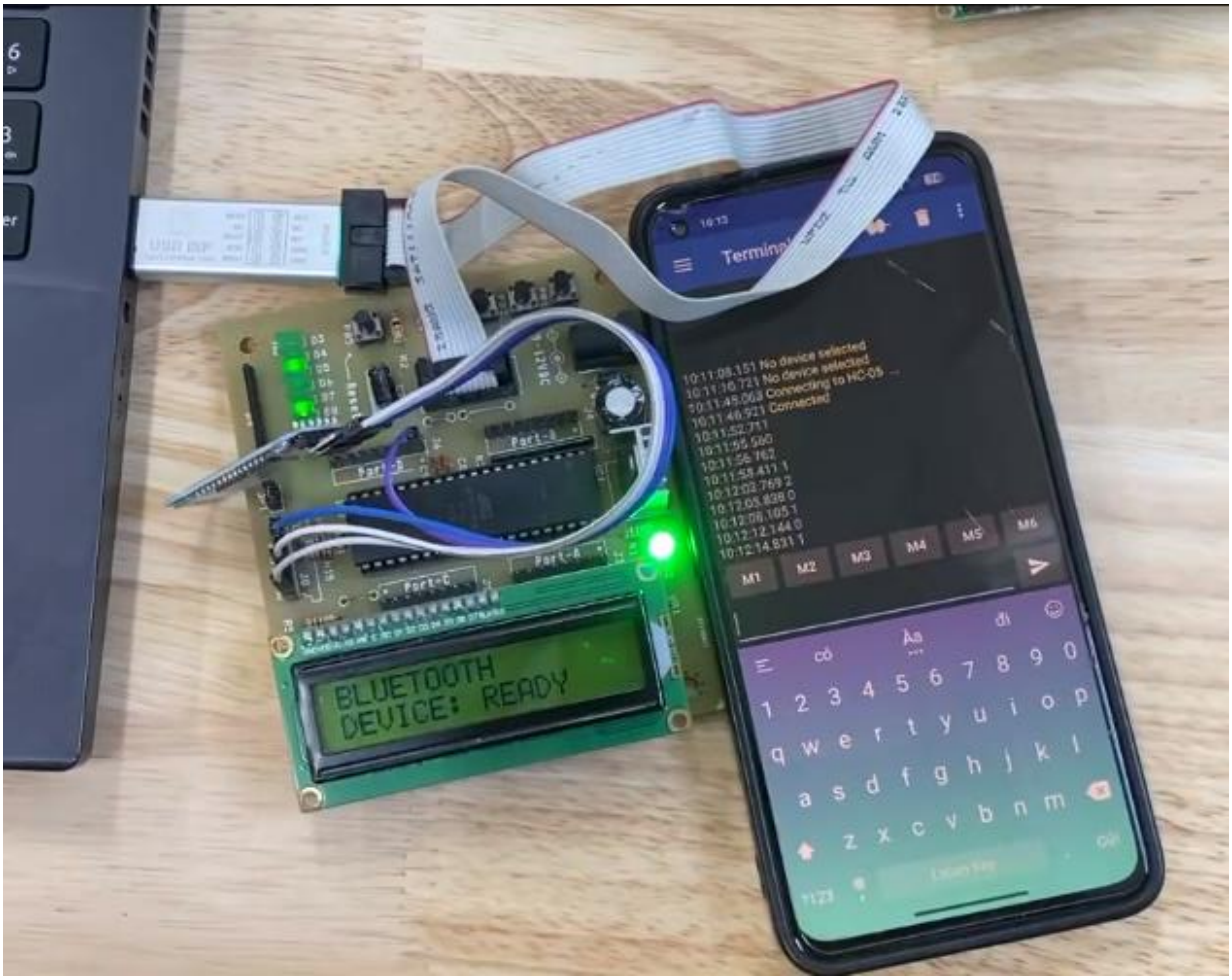


Hình 4.12 Mạch in PCB

4.5 Kết quả và nhận xét

4.5.1. Kết quả thực hiện thực tế

Mạch sau khi hoàn thiện hoạt động đúng theo yêu cầu đề ra. Module Bluetooth HC-05 kết nối ổn định với điện thoại, dữ liệu truyền qua UART chính xác. Vi điều khiển ATmega xử lý đúng lệnh điều khiển, bật/tắt LED theo dữ liệu nhận được và hiển thị trạng thái tương ứng trên màn hình LCD.



Hình 4.13 Kết quả thực hiện thực tế

4.5.2. Nhận xét

Qua quá trình thiết kế và thử nghiệm, mạch hoạt động ổn định và đáp ứng được các yêu cầu đặt ra ban đầu. Việc kết nối Bluetooth giữa điện thoại và vi điều khiển thông qua module HC-05 đạt hiệu quả tốt, dữ liệu truyền nhận chính xác. Tuy nhiên, hệ thống vẫn còn một số hạn chế về chức năng và giao diện điều khiển. Trong tương lai, đề tài có thể được phát triển thêm để nâng cao tính ứng dụng thực tế.

CHƯƠNG 5. ỨNG DỤNG

Điều khiển thiết bị trong gia đình

Ứng dụng trong điều khiển bật/tắt các thiết bị điện như: Cửa cuốn, điều hoà,… thông qua kết nối Bluetooth.

Hình 5.1 Thiết bị điện tử trong nhà

Hệ thống có thể được ứng dụng trong điều khiển thiết bị điện thông qua Bluetooth. Ngoài ra, đề tài còn có giá trị trong học tập và nghiên cứu, giúp sinh viên hiểu rõ hơn về giao tiếp Bluetooth và vi điều khiển, đồng thời làm nền tảng để phát triển các hệ thống điều khiển thông minh trong tương lai.

Hình 5.2 Ứng dụng trong SmartHouse

Đề tài có thể được mở rộng và ứng dụng trong các hệ thống nhà thông minh (Smart House) ở mức cơ bản. Từ việc điều khiển bật/tắt thiết bị bằng Bluetooth, hệ thống có thể tích hợp thêm nhiều thiết bị và cảm biến để giám sát và điều khiển trong không gian nhà ở, tạo nền tảng cho các hệ thống Smart House trong tương lai.

KẾT LUẬN

Qua quá trình tìm hiểu, nghiên cứu và thực hiện đề tài "Thiết kế mạch điều khiển thiết bị từ xa qua Bluetooth", em đã hoàn thành việc thiết kế và xây dựng một hệ thống điều khiển hoạt động ổn định, đáp ứng được các yêu cầu đề ra. Kết quả đạt được cho thấy hệ thống có khả năng truyền nhận dữ liệu không dây chính xác, các lệnh điều khiển được xử lý đúng theo nguyên lý đã phân tích và thiết kế ban đầu.

Trong quá trình thực hiện đồ án, em đã có cơ hội vận dụng và củng cố các kiến thức đã học về vi điều khiển họ AVR, cụ thể là ATmega16A, cũng như các kiến thức liên quan đến giao tiếp nối tiếp UART và công nghệ truyền thông không dây Bluetooth. Bên cạnh đó, em cũng được làm quen và sử dụng các công cụ hỗ trợ như phần mềm mô phỏng mạch điện Proteus và phần mềm thiết kế mạch in Altium, góp phần nâng cao kỹ năng thiết kế và triển khai hệ thống điện tử trong thực tế.

Thông qua đồ án, em nhận thấy đề tài không chỉ có tính ứng dụng cao trong các hệ thống điều khiển thiết bị dân dụng mà còn giúp em rèn luyện tư duy thiết kế, khả năng nghiên cứu tài liệu kỹ thuật và kỹ năng trình bày, thuyết minh sản phẩm. Đây là nền tảng quan trọng giúp em nâng cao năng lực học tập và nghiên cứu trong các học phần chuyên ngành tiếp theo.

Cuối cùng, em xin bày tỏ lòng biết ơn chân thành tới thầy Vũ Sinh Thượng đã tận tình hướng dẫn, chỉ bảo và tạo điều kiện thuận lợi nhất để em hoàn thành đồ án này.

TÀI LIỆU THAM KHẢO

- https://github.com/mohammad4kh/atmega32-atmega16-LCD-library
- https://www.alldatasheet.com/datasheet-pdf/pdf/78532/ATMEL/ATMEGA16.html
- https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf