Sketch Your Imagination with ARDUINO

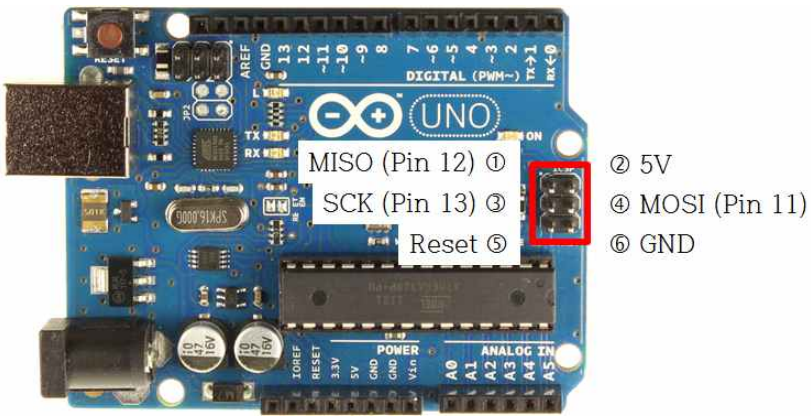
내용에 관련된 모든 피드백은 hgycap@hotmail.com으로 보내주세요.

Chapter 32. 부트로더

이 장에서는 마이크로컨트롤러에 프로그램을 다운로드 하고 설치하는 방법인 ISP(In-System Programming)와 UART(Universal Asynchronous Receiver/Transmitter) 시리얼 및 아두이노의 부트로더(bootloader) 역할에 대해 알아본다.

Arduino UNO는 ATmega328 마이크로컨트롤러를 기반으로 하고 있다. ATmega328에는 32KB의 플래시 메모리가 포함되어 있어 여기에 작성한 스케치가 설치된다. 하지만 Arduino UNO의 스펙을 자세히 살펴보면 32KB의 플래시 메모리 중 0.5KB를 부트로더가 이미 사용하고 있음을 볼 수 있다. ATmega328의 내장 플래시 메모리 중 약 1.6%를 차지하고 있는 부트로더의 정체는 무엇일까? 먼저 일반적으로 마이크로컨트롤러에 프로그램을 다운로드하고 설치하는 방법인 ISP에 대해 알아보자.

마이크로컨트롤러의 플래시 메모리에 프로그램을 다운로드 하여 설치하는 방법 중 흔히 사용되는 방법 중 하나가 ISP로 이는 글자 그대로 마이크로컨트롤러가 시스템에 설치된 이후 (In System) 마이크로컨트롤러의 플래시 메모리에 프로그램을 다운로드 할 수 있도록 해주는 방법을 가리킨다.[3] ISP와 유사한 단어로는 ISP를 SPI(Serial Peripheral Interface)를 통해 가능하도록 Microchip 사에서 개발한 ICSP(In-Circuit Serial Programming)가 있으며 ICSP는 ISP와 동일한 의미로 혼용되어 사용되고 있다. 일반적으로 ISP라는 용어가 더 많이 사용되므로 이 장에서는 ISP라는 용어를 사용한다. Arduino UNO에도 ISP를 위한 6핀 커넥터가 준비되어 있다. ATmega328을 위한 Arduino UNO의 ISP 커넥터 핀 배치는 그림 1과 같다.



3) ISP는 In-System Programming의 약어로 마이크로컨트롤러에 프로그램을 다운로드하는 과정이나 방법을 가리킬 뿐만이 아니라 ISP를 가능하게 해주는 장치인 In-System Programmer의 약어로도 사용된다. 따라서 ISP는 ISP를 가능하게 하는 ISP 장치를 가리키는 말로도 사용된다.

그림 1. Arduino UNO의 ICSP 커넥터

그림 1에서 알 수 있듯이 ISP를 위해 SPI 핀을 사용하지만 SPI와는 달리 슬레이브 선택을 위한 chip select 핀은 사용하지 않으며 대신 리셋(reset) 핀에 연결된다. 리셋 핀이 HIGH(VCC)인 상태에서 마이크로컨트롤러는 플래시 메모리에 설치된 프로그램을 실행하지만, LOW(GND)인 상태에서 마이크로컨트롤러는 ISP를 통해 프로그램을 다운 받아 설치한다. Arduino UNO에서 사용되는 ATmega328 마이크로컨트롤러에서 ISP를 위해 사용되는 핀은 그림 2와 같다.

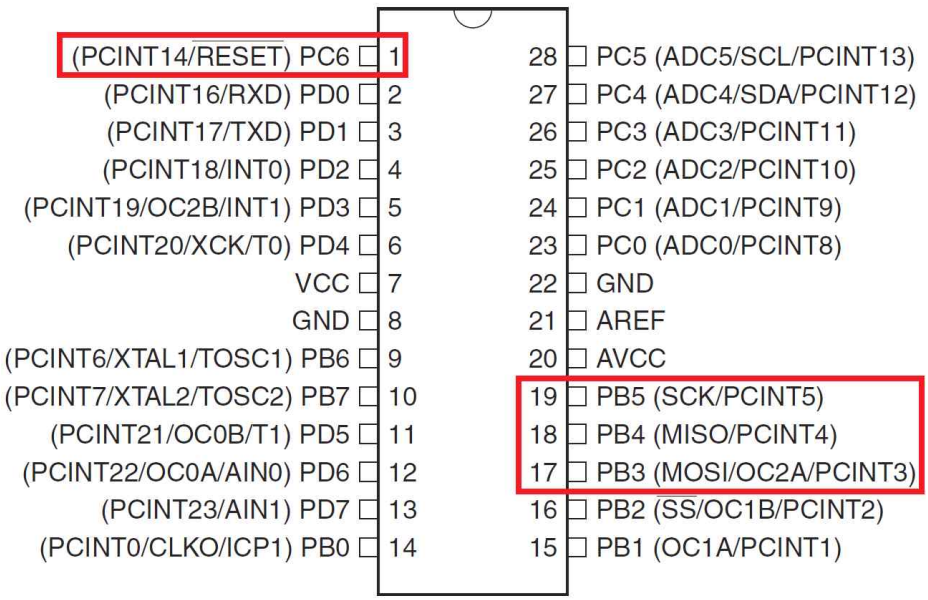


그림 2. ATmega328 핀 배치

ISP를 위해서는 아두이노에서 사용하는 6핀 커넥터 이외에도 그림 3의 10핀 커넥터가 흔히 사용되며 그림 3의 커넥터에서 3번, 4번, 6번, 8번 핀은 연결하지 않고 사용하거나 모두 GND에 연결하여 사용한다.

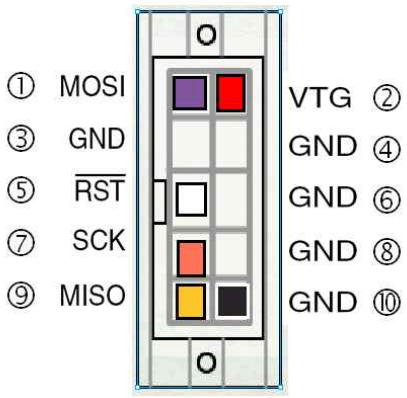


그림 3. ICSP 10핀 커넥터

ISP로 프로그램을 다운로드 하고 설치하기 위해서는 컴퓨터의 직렬 포트, 병렬 포트 또는 USB 포트를 마이크로컨트롤러의 ISP 커넥터와 연결하여야 하며 USB 포트를 이용한 연결이 흔히 사용된다. 컴퓨터의 해당 포트와 ISP 커넥터를 연결하기 위해서는 변환 장치가 필요하지만 아두이노 보드에는 ISP를 위한 변환 장치가 마련되어 있지 않으므로 ISP로 아두이노 보드에 프로그램을 다운로드 하여 설치하기 위해서는 그림 4와 같이 USB 신호를 ISP 신호로 변환하는 장치가 필요하다. 이러한 변환 장치는 다른 마이크로컨트롤러의 프로그래밍에서도 필요하다.

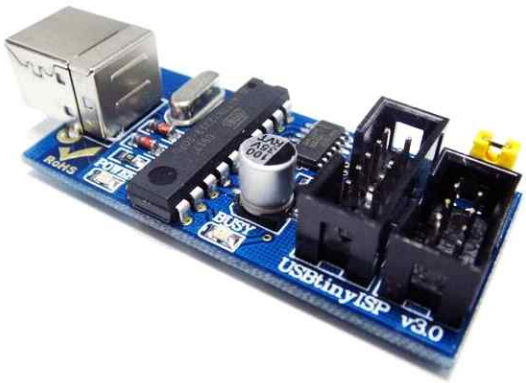


그림 4. USB 포트로 연결되는 ISP 장치

지금까지 아두이노를 위한 스케치를 작성하고 컴파일 하여 다운로드 할 때 별도의 ISP 없이 USB 연결만으로도 가능하였다. 이는 Arduiono UNO가 ISP 방식이 아닌 UART 시리얼 방식으로 프로그램을 다운로드 하여 설치하기 때문이다. UART 시리얼 방식 역시 전용 변환 장치가 필요하지만 Arduino UNO에는 ISP와는 달리 UART 시리얼 방식을 지원하기 위해 또 하

나의 마이크로컨트롤러가 포함되어 있다. Arduino UNO 보드의 USB 연결 단자 뒤쪽에는 MEGA16U2라는 작은 크기의 칩이 장착되어 있으며 이 칩이 바로 USB를 통해 전송된 데이터를 UART 프로토콜을 따르는 데이터로 변환하여 Atmega328의 하드웨어 시리얼 핀으로 전송하는 역할을 수행하는 마이크로컨트롤러이다. Arduino UNO 보드를 살펴보면 그림 1의 ISP 커넥터와 이외에 ISP 커넥터가 하나 더 준비되어 있음을 확인할 수 있으며 이 커넥터가 바로 MEGA16U2 마이크로컨트롤러에 프로그램을 다운로드 하기 위해 사용하는 ISP 커넥터이다.

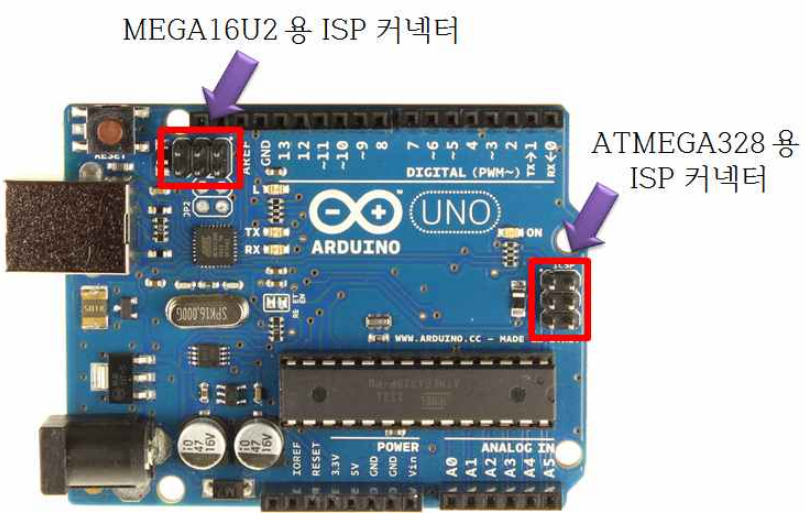


그림 5. Arduino UNO의 ICSP 커넥터

MEGA16U2 마이크로컨트롤러에서 ATmega328의 하드웨어 시리얼 포트로 전송되는 프로그램을 마이크로컨트롤러에 설치하기 위해서는 ISP에서와는 다른 방법이 필요하며 이 역할을 담당하는 프로그램이 ATmega328의 플래시 메모리에 내장되어 있는 부트로더(bootloader)이다. 마이크로컨트롤러에 전원이 인가되거나 리셋 버튼이 눌러지면 마이크로컨트롤러는 고정된 메모리 주소에 있는 프로그램으로부터 수행을 시작한다. ATmega328의 경우 고정된 메모리 주소는 0번지이거나 부트로더가 설치된 위치 중 하나이며 퓨즈 설정에 따라 변경할 수 있다. ISP 방식이나 UART 시리얼 방식 모두 사용자 프로그램[4]은 0번지부터 설치되므로 0번지부터 프로그램이 시작하도록 설정된 경우에는 리셋 버튼이 눌러지면 프로그램이 바로 실행된다. 부트로더로부터 실행되도록 설정되어 있는 경우라도 부트로더의 내용이 없다면 바로 0번지로 실행이 옮겨진다.

아두이노는 리셋 버튼이 눌러지면 0번지가 아닌 부트로더가 설치되어 있는 위치에서 프로그램

4) 부트로더도 일종의 프로그램이지만 그 용도가 다르므로 일반적으로 마이크로컨트롤러에서 실행되는 프로그램의 구별이 필요한 경우 '사용자 프로그램'이라 지칭한다.

실행이 시작되도록 설정되어 있다. 따라서 리셋 버튼이 눌러지면 아두이노는 부트로더를 실행하고 부트로더는 수행이 시작된 후 약간의 시간 동안 UART 시리얼 포트를 통해 컴퓨터와 통신을 시도한다. 통신에 실패하면 부트로더는 0번지부터 설치되어 있는 프로그램을 실행하고 통신에 성공하면 부트로더는 컴퓨터로부터 새로운 프로그램을 다운로드 받아 0번지부터 프로그램을 설치한다. 프로그램 다운로드 및 설치가 종료되면 부트로더는 새롭게 설치된 프로그램을 수행한다. 리셋 버튼이 눌러진 후 아두이노의 부팅 과정을 요약하면 그림 6과 같다.

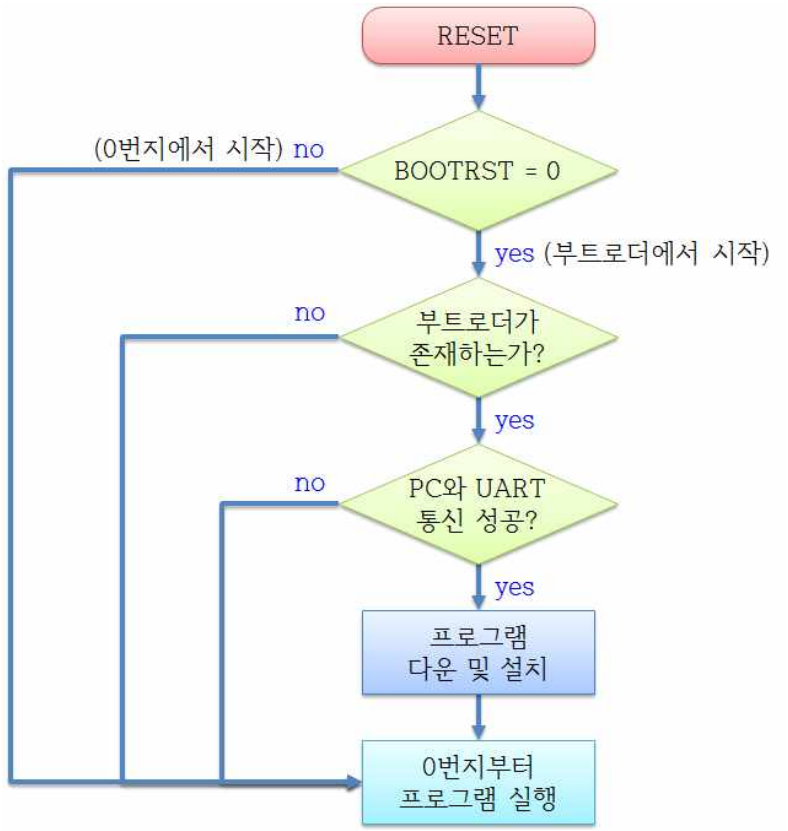


그림 6. 아두이노의 부팅 순서

이처럼 아두이노는 ISP를 위한 전용 하드웨어를 배제하고 USB를 통해 프로그램을 다운로드하고 설치할 수 있도록 부트로더를 사용하고 있다. 아두이노 보드에는 부트로더가 미리 포함되어 있으므로 별도로 부트로더를 설치해야 하는 경우는 거의 없지만 ATmega328 칩을 교환하거나 아두이노 보드를 직접 구성하는 경우에는 부트로더를 설치할 필요가 있다. 부트로더를 설치하는 과정을 일반적으로 '굽는다(burn)'고 표현하며 부트로더를 굽는 방법에 대해서는 아두이노 호환 보드를 제작하는 장에서 다룬다.

부트로더가 아두이노의 특징 중 한 가지이기는 하지만 부트로더를 사용함으로 인해 부트로더

를 위해 플래시 메모리의 일부를 사용하여야 하며 구동 시에 컴퓨터와의 통신을 위해 약간의 지연이 발생하는 단점이 있다. 이러한 단점은 부트로더를 제거함으로써 가능하며 부트로더를 제거하기 위해서는 프로그램이 시작되는 위치를 0번지부터 시작하도록 퓨즈 비트를 설정하고, 플래시 메모리 전체를 사용자 프로그램이 사용할 수 있도록 설정하여야 한다. 이 두 가지는 아두이노 설치 디렉터리 아래 'hardware/arduino' 디렉터리에 있는 boards.txt 파일을 수정함으로써 가능하다. Arduino UNO의 보드 정의 중 부트로더 사용과 관련된 항목은 다음 두 가지이다.

```
uno.upload.maximum_size=32256
uno.bootloader.high_fuses=0xDE
```

upload.maximum_size는 사용자 프로그램이 사용할 수 있는 최대 스케치의 크기로 Arduino UNO의 경우 32Kbyte 중에서 부트로더가 차지하는 0.5Kbyte를 뺀 값으로 설정되어 있다. $(32\times2^{10}-0.5\times2^{10}=32256)$ 전체 메모리를 사용하기 위해서는 부트로더 크기를 빼지 않은 $32\times2^{10}=32768$로 설정하면 된다. 프로그램 시작 위치는 high fuse의 최하위 비트(LSB, Least Significant Bit)인 BOOTRST (Boot Reset) 비트에 의해 결정되며 Arduino UNO의 경우 0으로 설정되어 부트로더로부터 수행이 시작하도록 설정되어 있다. high fuse 값을 0xDF로 설정하여 0번지로부터 수행이 시작하도록 설정함으로써 부트로더 영역을 사용자 프로그램 설치를 위해 사용할 수 있다. 부트로더를 사용하지 않도록 설정된 경우 아두이노에 프로그램을 설치하기 위해서는 반드시 ISP 방식을 사용하여야 하며 UART 시리얼 방식으로는 프로그램을 다운로드 할 수 없다는 점에 주의하여야 한다. 이처럼 부트로더 없이도 ATmega328에 프로그램을 설치하고 실행시킬 수 있지만 아두이노의 특징 중 하나가 부트로더인 점을 고려할 때 부트로더가 제거된 아두이노를 아두이노라고 부르는 데는 논란의 여지가 있을 수 있다.

ATmega328의 메모리에 대해 좀 더 자세히 살펴보자. ATmega328에는 다운로드 한 프로그램을 저장하기 위한 32Kbyte 크기의 비휘발성 플래시 메모리, 프로그램 수행 중 필요한 데이터를 저장하기 위한 2Kbyte 크기의 휘발성 SRAM (Static Random Access Memory), 데이터 영구 보관을 위한 1Kbyte 크기의 비휘발성 EEPROM (Electrically Erasable Programmable Read only Memory) 등 세 종류의 메모리가 포함되어 있다. 이 중 부트로더나 사용자 프로그램은 '프로그램 메모리'로 불리는 플래시 메모리에 설치된다. ATmega328은

8비트 마이크로컨트롤러이지만 명령어의 크기는 16비트 또는 32비트이므로 메모리 주소는 16비트 단위로 할당되어 ATmega328의 32KByte 메모리는 0x0000에서 0x3FFF까지 14비트 크기의 주소로 엑세스된다. ATmega328의 프로그램 메모리 구조는 그림 7과 같다.

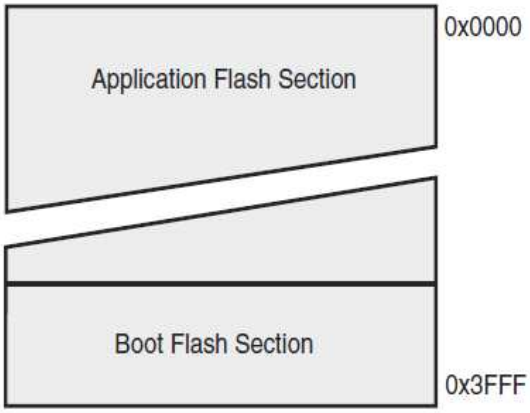


그림 7. ATmega328의 메모리 구조

그림 7에서 부트 영역(Boot Flash Section)에 부트로더가 위치한다. ATmega328은 self-programming이 가능하다. self-programming이란 마이크로컨트롤러가 프로그램을 다운받아 메모리에 직접 설치할 수 있는 기능을 말하며 이를 통해 부트로더는 프로그램을 다운받아서 플래시 메모리에 설치한다. 부트로더의 크기는 퓨즈 비트 BOOTSZ (Boot Size) 설정에 의해 최소 0.5Kbyte에서 최대 4Kbyte의 크기를 가질 수 있다. 부트로더의 크기에 따른 부트 영역의 크기와 주소는 표 1과 같으며 부트로더는 부트 영역의 시작 주소에서부터 적재된다.

BOOTSZ 퓨즈 비트	부트 영역 크기 (word)	프로그램 저장 영역 주소	부트 영역 주소
11	256	0x0000 - 0x3EFF	0x3F00 - 0x3FFF
10	512	0x0000 - 0x3DFF	0x3E00 - 0x3FFF
01	1024	0x0000 - 0x3BFF	0x3C00 - 0x3FFF
00	2048	0x0000 - 0x37FF	0x3800 - 0x3FFF

표 1. 부트 영역 크기에 따른 메모리 주소

Arduino UNO에서 사용하는 최적화된 부트로더는 0.5Kbyte의 크기를 가지며 BOOTSZ 비트는 high fuse에서 하위 두 번째 및 세 번째 비트에 해당한다. 표 2는 ATmega328의 high

fuse 비트의 의미와 디폴트값 및 Arduino UNO의 설정값을 나타낸 것으로 비트값이 1인 경우는 설정되지 않은 경우를, 비트값이 0인 경우는 설정된 경우를 나타낸다.

비트 이름	비트 번호	설명	디폴트값	Arduino UNO 설정값
RSTDISBL	7	외부 리셋 금지	1 (외부 리셋 가능)	1
DWEN	6	디버그 출력 가능	1 (디버그 출력 불가능)	1
SPIEN	5	SPI 프로그래밍 가능	0 (ISP로 프로그램 다운 가능)	0
WDTON	4	와치독 타이머 항상 켜기	1 (와치독 타이머 끔)	1
EESAVE	3	칩 내용을 지울 때 EEPROM 내용 보존	1 (EEPROM 내용도 지움)	1
BOOTSZ1	2	부트로더 크기	00 (4Kbyte)	11
BOOTSZ0	1			
BOOTRST	0	리셋 시 시작 시점	1 (0번지부터 시작)	0 (부트로더에서 시작)

표 2. ATmega328의 high fuse

2Kbyte 크기의 SRAM은 변수에 할당된 값을 저장하기 위해 주로 사용되며 데이터 메모리라고 불린다. SRAM을 사용함에 있어 주의하여야 하는 키워드로 PROGMEM이 있다. PROGMEM 키워드는 데이터가 일반적으로 저장되는 SRAM이 아닌 프로그램 메모리에 저장되도록 설정하기 위해 사용한다. 2Kbyte의 데이터 메모리는 크기가 작아 긴 문자열이나 크기가 큰 배열을 저장하는 경우 소진될 위험이 있다. 데이터 메모리가 소진되면 프로그램이 정상적으로 컴파일 되지 않거나 컴파일 된다고 하더라도 실행 중 잘못된 결과나 예상치 못한 동작을 수행하는 경우가 발생한다. 따라서 문자열과 같이 내용이 바뀌지 않으며 크기가 큰 데이터는 데이터 메모리가 아닌 상대적으로 큰 프로그램 메모리에 위치시키는 것이 안전하다. 프로그램 메모리에 데이터를 위치시키는 방법은 PROGMEM 키워드를 사용하는 방법과 F() 매크로를 이용하는 방법 두 가지가 있다. PROGMEM 키워드를 사용하여 정의된 변수의 값은 데이터 메모리가 아닌 프로그램 메모리에 저장되므로 값을 읽어오기 위해서는 전용 함수를 사용하여야 하며 프로그램 메모리는 실행 도중 내용을 바꿀 수 없으므로 변수의 값 변경은 불가능하다. F()

매크로 역시 값을 데이터 메모리가 아닌 프로그램 메모리에 저장하도록 지정하지만 F() 매크로는 함수 외부에서는 사용할 수 없다. 코드 1은 ROGMEM 키워드와 F() 매크로를 이용한 예를 보여준다. 많은 양의 데이터를 사용하고자 하는 경우에는 프로그램 메모리에 데이터를 저장하는 것을 고려해볼 필요가 있다.

코드 1

```
char str[] PROGMEM = "PROGMEM keyword";        // PROGMEM 키워드 사용

void setup(){
  Serial.begin(9600);
  Serial.println(F("F() macro"));                // F() 매크로 사용

  for(int i = 0; i < strlen(str); i++){
    // 프로그램 메모리의 변수값을 읽어오기 위한 전용 함수
    Serial.print((char)pgm_read_byte(str + i));
  }
  Serial.println();
}

void loop(){
}
```

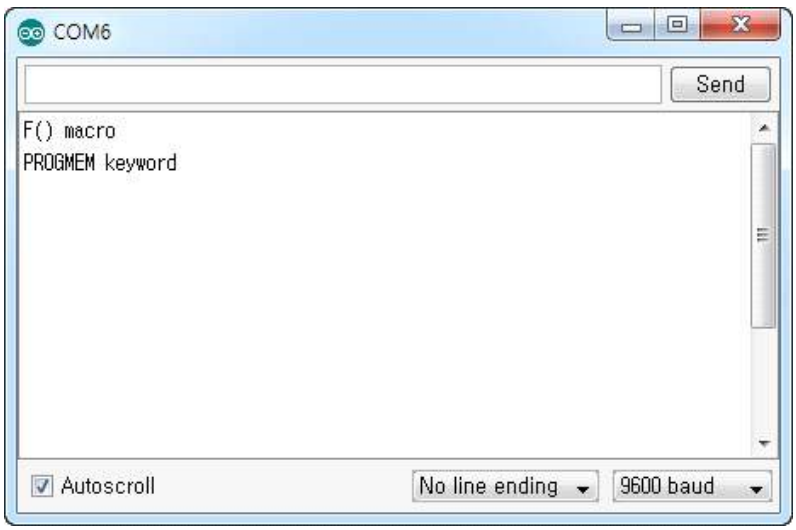


그림 7. 코드 1 실행 결과

데이터 메모리는 휘발성 메모리로 전원이 꺼지면 그 내용이 사라진다. 프로그램 메모리는 비휘발성 메모리이기는 하지만 프로그램 실행 도중에 내용을 기록하기 어렵다. 따라서 프로그램 실행 결과를 저장하고 전원이 꺼진 이후에도 보존하기 위해서는 EEPROM을 사용하여야 한다. ATmega328에는 1Kbyte 크기의 EEPROM이 준비되어 있으며 데이터를 읽고 쓰기 위해서는 전용 EEPROM 라이브러리를 사용하여야 한다. EEPROM 라이브러리 사용법은 해당 장을 참고하면 된다.