

Chapter 18. EEPROM

1. EEPROM

아두이노에 사용된 ATmega328과 ATmega2560 마이크로컨트롤러에는 3종류의 메모리가 포함되어 있다. 이 중 스케치가 설치되고 실행되는 메모리는 플래시(flash) 메모리로 프로그램 메모리라고도 부른다. 스케치가 실행되는 동안 연산 결과를 임시로 저장하는 등의 용도로 사용되는 메모리는 SRAM이다. SRAM은 3종류의 메모리 중 가장 빠른 속도를 보이지만 휘발성 메모리로 전원이 꺼지면 내용은 모두 사라진다. 마지막 EEPROM은 SRAM과 마찬가지로 데이터를 저장하기 위한 용도로 사용되며 비휘발성 메모리라는 점에서 SRAM과 차이가 있다. 표 1은 ATmega328과 ATmega2560에 포함된 메모리의 종류와 크기를 요약한 것이다.

아두이노	우노	메가2560
플래시 메모리	23 KByte	256 KByte
SRAM	2 KByte	9 KByte
EEPROM	1 KByte	4 KByte

표 1. 아두이노의 메모리

EEPROM은 그 크기가 크지 않고 비휘발성 메모리라는 특성으로 특히 쓰는 시간이 많이 걸리는 단점이 있다. 따라서 EEPROM에는 한 번 데이터를 기록한 이후 자주 참조하는 작은 크기의 데이터를 저장하기 위해 사용된다.

2. EEPROM 라이브러리

아두이노에는 마이크로컨트롤러 내부의 EEPROM을 읽고 쓸 수 있는 EEPROM 라이브러리를 기본 라이브러리 중 하나로 제공하고 있다. EEPROM 라이브러리에는 EEPROM에 바이트 단위의 데이터를 기록하는 write와 바이트 단위로 데이터를 읽어 오는 read의 두 멤버 함수가 정의되어 있다.

```
uint8_t read(int address)
void write(int address, uint8_t value)
```

스케치 1은 마이크로컨트롤러 내부 EEPROM의 0번지에서 9번지까지 바이트 단위의 데이터를 쓰고 이를 다시 읽어서 시리얼 모니터로 출력하는 스케치의 예이다.

스케치 1. 내장 EEPROM 읽고 쓰기

```
#include <EEPROM.h>

void setup(){
  Serial.begin(9600);
}

void loop(){
  for(int i = 0; i < 10; i++){          // EEPROM에 쓰기
    EEPROM.write(i, i);                // write(주소, 값)
    delay(20);
  }

  for(int i = 0; i < 10; i++){          // EEPROM에서 읽기
    int value = EEPROM.read(i);        // read(주소)
    Serial.println("Address : " + String(i) + "\t Value : " + value);
    delay(20);
  }

  while(true);
}
```

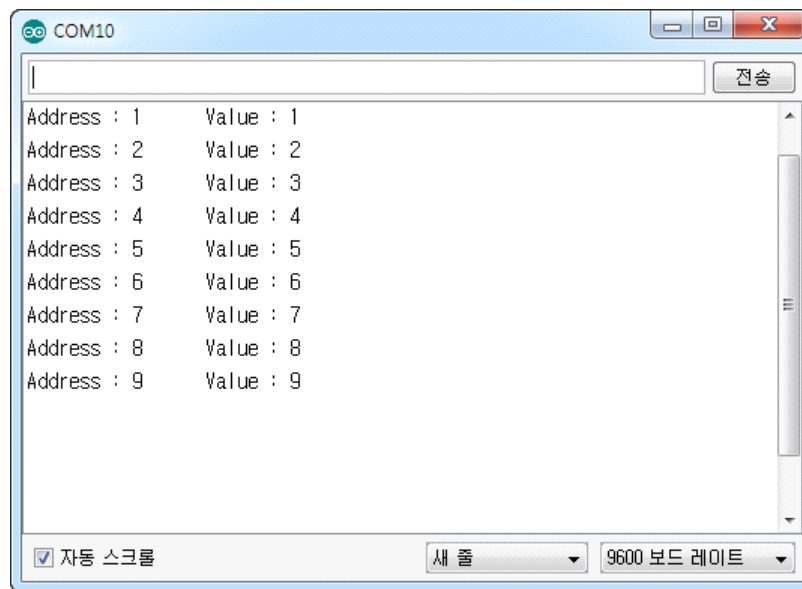


그림 1. 스케치 1 실행 결과

3. 외장 EEPROM

ATmega2560에도 4KByte의 EEPROM이 포함되어 있지만 용량이 그리 크지 않으므로 별도의 외부 기억 장치로 EEPROM을 사용하는 경우를 볼 수 있다. 외부 기억 장치로 사용하기에 EEPROM의 속도가 느린 단점은 있지만 비휘발성 메모리이고 간단히 데이터를 읽고 쓸 수 있어 널리 사용된다. EEPROM은 I2C나 SPI 방식 등 다양한 인터페이스를 사용하는 EEPROM이 사용되고 있으며 학습보드에는 SPI 방식을 사용하는 마이크로칩 사의 25LC640 EEPROM이 포함되어 있다. 25LC640 EEPROM은 8KByte(8192×8 bit)의 메모리를 가지고 있다.

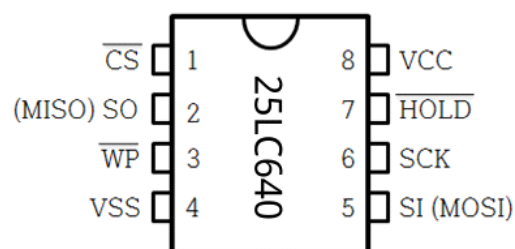


그림 2. 25LC640 핀 배치

25LC640 각 핀의 기능은 표 2와 같다. $\overline{WP}$ 는 쓰기 금지 기능으로 GND가 가해지면 EEPROM에 내용을 기록할 수 없도록 해준다. 이 장에서는 VCC에 연결하여 항상 쓰기가 가

능하도록 한다. $\overline{\text{HOLD}}$ 는 EEPROM과의 통신을 일시 중지시키는 기능을 하는 핀이다.

핀 번호	이름	설명	비고
1	$\overline{\text{CS}}$	Chip Select	SS (Slave Select)
2	SO	Serial Data Output	MISO
3	$\overline{\text{WP}}$	Write Protect	
4	VSS	Ground	
5	SI	Serial Data Input	MOSI
6	SCK	Serial Clock	
7	$\overline{\text{HOLD}}$	Hold Input	
8	VCC		

표 2. 25LC640 핀 설명

아두이노에서 SPI 연결을 위해 사용할 수 있는 핀이 표 3과 같이 정의되어 있다. 이 중 MOSI, MISO, SCK는 변경할 수 없다. SS는 임의의 핀을 사용할 수 있으며 학습보드에서는 디지털 42번 핀을 사용하였다.

SPI 신호	아두이노 우노 핀	아두이노 메가2560 핀
MOSI	디지털 11	디지털 51
MISO	디지털 12	디지털 50
SCK	디지털 13	디지털 52
SS	-	(디지털 42)

표 3. SPI 연결 핀

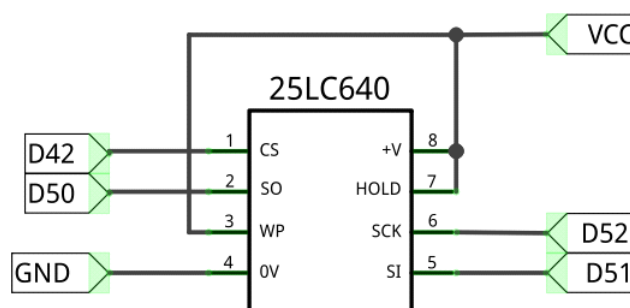


그림 3. 25LC640 EEPROM 연결 회로도

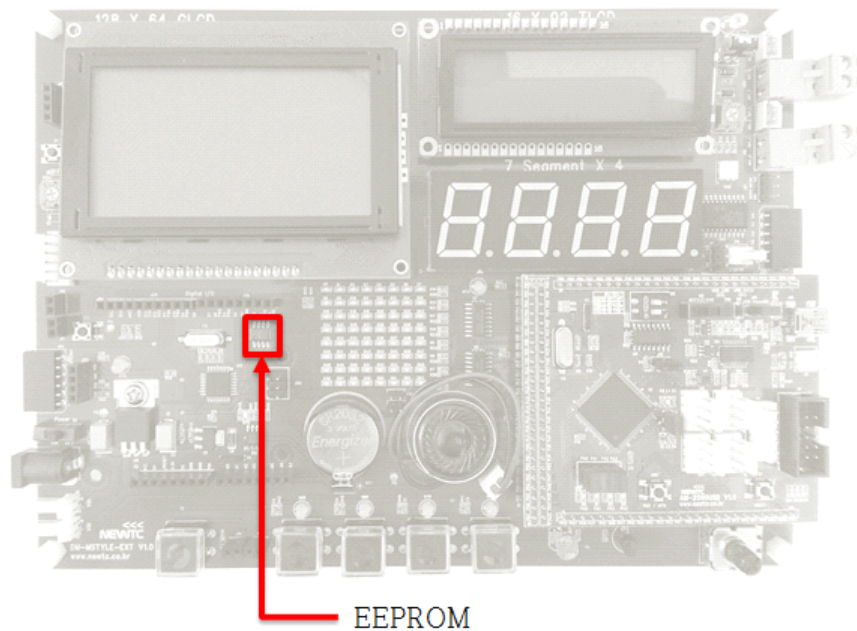


그림 4. 학습보드에의 25LC640 EEPROM

SPI 통신을 위해서는 먼저 SS 핀에 LOW로 가하여 해당 장치를 선택한 후 데이터를 전송하여야 한다. 데이터를 전송할 때 SPI는 항상 송신과 수신이 함께 발생하는 특징을 갖는다. MOSI는 Master Out Slave In의 약어로 마스터에서 슬레이브로 데이터를 전송하는 연결선을, MISO는 Master In Slave Out의 약어로 슬레이브에서 마스터로 데이터를 전송하는 연결선을 의미한다. 마스터가 (MOSI를 통해) 데이터를 송신하는 경우 마스터는 동일한 크기의 의미 없는 데이터를 (MISO를 통해) 수신하게 되며 수신된 데이터는 무시하면 된다. 마스터가 슬레이브로부터 (MISO를 통해) 데이터를 얻기 위해서도 동일한 크기의 (MOSI를 통해) 의미 없는 데이터를 보내야만 한다.

아두이노에는 SPI 통신을 위한 SPI 라이브러리가 기본 라이브러리 중 하나로 포함되어 있다. 먼저 '스케치 → Include Library → SPI' 메뉴 항목을 선택하여 해당 헤더 파일을 포함시킨다. SPI 라이브러리에서는 SPI 통신을 담당하는 객체를 생성해 주므로 별도의 객체 생성 없이도 SPI 통신을 사용할 수 있다.

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

SPI 통신을 초기화하는 함수는 begin이다. begin 함수에서는 SPI 통신에 사용되는 핀의 입출력을 설정하고 마스터 모드로 SPI를 활성화시킨다.

```
void begin(void)
- 매개변수 : 없음
- 반환값 : 없음
```

SPI 통신을 초기화 시킬 때 SS핀은 출력으로 설정하고 HIGH 상태로 지정해서 슬레이브가 선택되지 않은 상태로 해주어야 한다. 학습보드의 경우 EEPROM뿐만이 아니라 우노와도 SPI 통신을 위해 디지털 53번 핀이 사용되고 있으므로 53번 핀 역시 출력으로 설정하고 HIGH 값을 출력하여야 한다. 바이트 데이터 전송 함수는 transfer로 송신하는 데이터를 매개변수로 가지며 수신된 데이터를 반환값으로 가진다.

```
byte transfer(byte data)
- 매개변수
  data : 송신할 데이터
- 반환값 : 수신된 데이터
```

사용하는 함수는 간단하지만 SPI 장치에 따라 데이터를 주고받는 방법은 서로 다르므로 데이터시트를 참고하여 구현하여야 한다. EEPROM의 경우 데이터를 읽고 쓰는 두 함수만을 구현하면 된다. 데이터를 기록하기 위해서는 먼저 WREN(Write Enable, 0x06) 명령을 전송하여야 한다. 다음으로 WRITE(0x02) 명령을 전송하고, 16비트의 주소를 바이트 단위로 나누어서 전송한 후 기록할 바이트 단위의 데이터를 전송하면 된다. 스케치 2는 25LC640 EEPROM에 한 바이트의 데이터를 기록하는 함수이다.

스케치 2. 외장 EEPROM 쓰기

```
void write_eeprom(int address, byte data){
  // WREN 명령 전송
  digitalWrite(chipSelectPin, LOW);
```

```

SPI.transfer(WREN);
digitalWrite(chipSelectPin, HIGH);

// 데이터 기록
digitalWrite(chipSelectPin, LOW);
SPI.transfer(WRITE);           // WRITE 명령
SPI.transfer( (byte)(address >> 8) ); // 상위 8비트 주소
SPI.transfer( (byte) address ); // 하위 8비트 주소
SPI.transfer(data);           // 데이터
digitalWrite(chipSelectPin, HIGH);
}

```

데이터를 읽기 위해서는 먼저 READ(0x03) 명령, 주소 순서로 데이터를 전달하고 실제 데이터를 얻어오기 위해 의미 없는 바이트 단위의 데이터를 전송해 주어야 한다. 스케치 3은 1바이트의 데이터를 읽는 함수의 예이다.

스케치 3. 외장 EEPROM 읽기

```

byte read_eeprom(int address){
  digitalWrite(chipSelectPin, LOW);
  SPI.transfer(READ);           // READ 명령
  SPI.transfer( (byte)(address >> 8) ); // 상위 8비트 주소
  SPI.transfer( (byte) address ); // 하위 8비트 주소

  // 데이터를 읽기 위해 의미 없는 바이트 값을 전송
  int data = SPI.transfer(0x00);

  digitalWrite(chipSelectPin, HIGH);

  return data;
}

```

스케치 4는 25LC640 EEPROM의 0번지에서 99번지까지 데이터를 기록한 후 이를 다시 읽어 시리얼 모니터로 출력하는 스케치의 예이다. 1바이트의 데이터를 기록하기 위해서는 5ms 정

도의 시간이 필요하므로 데이터를 기록한 이후 delay 함수를 사용하고 있다. 스케치 4에는 EEPROM을 읽고 쓰기 위한 함수가 포함되어 있지 않으므로 스케치 2와 스케치 3 역시 포함시켜야 한다.

스케치 4. 외장 EEPROM 쓰기과 읽기

```
#include <SPI.h>

byte chipSelectPin = 42;           // EEPROM의 SS
byte chipSelectArduinoUno = 53;    // 아두이노 우노의 SS

#define READ          0x03         // 읽기 명령
#define WRITE         0x02         // 쓰기 명령
#define WREN          0x06         // 쓰기 허용 명령

void setup(){
    pinMode(chipSelectPin, OUTPUT);
    pinMode(chipSelectArduinoUno, OUTPUT);
    digitalWrite(chipSelectArduinoUno, HIGH);

    SPI.begin();
    Serial.begin(9600);
}

void loop(){
    for(int i = 0; i < 100; i++){    // 데이터 쓰기
        write_eeprom(i, i);
        delay(10);
    }

    for(int i = 0; i < 100; i++){    // 데이터 읽기
        byte data = read_eeprom(i);

        Serial.print("Reading address : ");
        Serial.print(i);
        Serial.print(", value : ");
```

```
Serial.println(data);  
}  
  
while(true);  
}
```

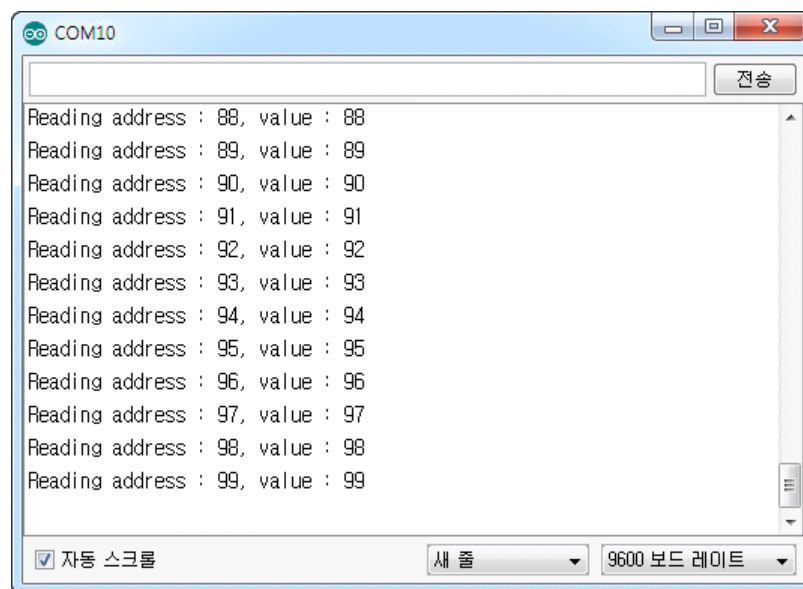


그림 5. 스케치 4 실행 결과