

Chapter 5. 아두이노 기본 클래스

아두이노 스케치는 C/C++ 언어를 기반으로 하고 있다. 스케치는 setup과 loop 함수를 기반으로 구성되는 C 스타일로 작성하는 것이 일반적이지만 주변 장치 제어를 위한 라이브러리들은 C++ 스타일로 작성하는 것이 일반적이다. 스케치에서 흔히 사용되는 시리얼 통신과 문자열을 다루기 위한 Serial과 String 클래스는 아두이노의 기본 라이브러리로 제공되며 별도의 헤더 파일 없이도 사용할 수 있다.

1. 시리얼 연결

Serial은 아두이노와 기타 주변 장치의 시리얼 통신을 담당하는 클래스이다. 아두이노의 시리얼 통신은 RS-232C 프로토콜의 정의 중 RX (receive data), TX (transmit data), GND (ground) 세 개의 핀만을 사용하여 통신을 수행하는 방식으로 UART(Universal Asynchronous Receiver Transmitter)라고 불린다. 아두이노 메가2560에서 시리얼 통신을 위한 포트 중 첫 번째는 디지털 0번 (RX) 핀과 1번 (TX) 핀이며 이들 핀을 통해 이루어지는 UART 시리얼 통신을 담당하는 객체가 Serial이다. 아두이노 우노에서도 디지털 0번 핀과 1번 핀으로 유일한 UART 시리얼 통신이 이루어지며 이를 담당하는 객체 역시 Serial이다. 아두이노 우노와 메가2560은 USB 연결을 통해 컴퓨터와 연결된다. 하지만 아두이노 우노는 USB를 통해 전달되는 데이터를 직접 받아들이지 않으므로 USB 데이터를 UART 데이터로 변환해서 받아들이야 한다. USB-UART 데이터 변환을 위해 아두이노 우노와 메가2560에는 전용 마이크로컨트롤러인 ATmega16u2가 포함되어 있으며, 학습보드의 경우 FTDI 칩이 포함되어 있다. UART 시리얼 통신은 스케치 업로드 및 컴퓨터와의 UART 통신을 위해 사용된다는 점은 이미 이전 장에서 살펴보았다. 디지털 0번 및 1번 핀은 일반적인 디지털 데이터 입출력을 위해서도 사용할 수 있지만 스케치 업로드 및 시리얼 통신용으로 사용하고 디지털 입출력 핀으로 사용하는 경우는 흔하지 않다.

이전 장에서 이미 시리얼 모니터로 데이터를 출력하기 위해 Serial 클래스를 사용하였다. 시리얼 모니터는 아두이노의 Serial 객체를 통해 송수신되는 데이터를 확인하기 위한 용도로 사용된다. 시리얼 모니터에는 데이터 송수신을 위해 사용할 수 있는 영역 이외에도 아래쪽에 두 개의 콤보 박스를 확인할 수 있다. 첫 번째 콤보 박스는 문자열 전송시에 문자열 끝에 추가할 문자를 선택할 수 있도록 해준다. 디폴트값으로 추가 문자는 없는 것(No line ending)으로 설정되어 있지만 필요한 경우 '새 줄' ('\n'), '캐리지 리턴' ('\r') 또는 두 문자 모두를 추가할

수 있다. 두 번째 콤보 박스는 데이터 변조 속도를 설정하는 콤보 박스로 디폴트값은 9600 보율(baud rate)로 설정되어 있다.

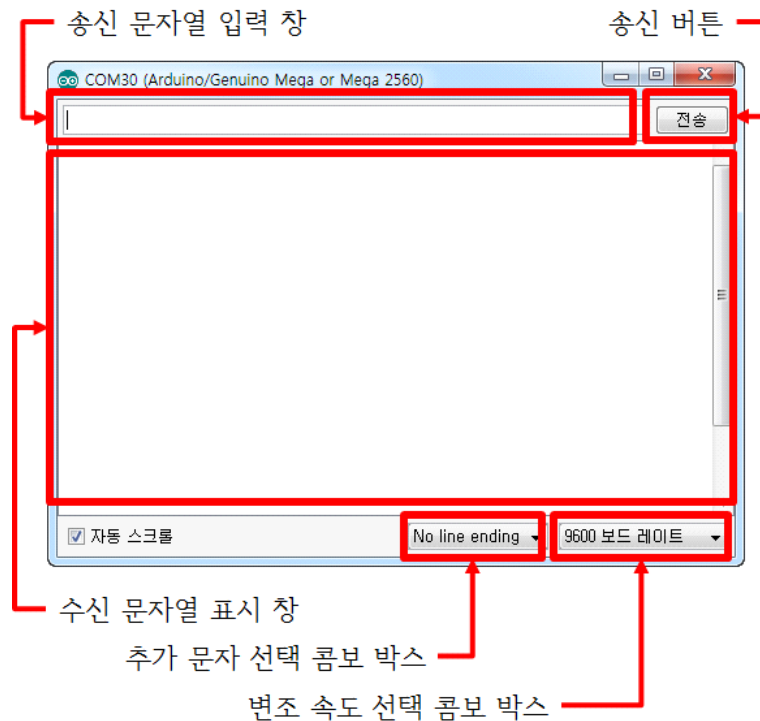


그림 1. 시리얼 모니터

2. Serial 클래스

Serial 클래스를 통해 컴퓨터로 데이터를 전송하는 방법을 살펴보자. 데이터 전송에 사용할 수 있는 멤버 함수에는 문자열 기반의 print 및 println 함수와 이진 데이터 기반의 write 함수가 있다. println 함수는 전송되는 문자열 끝에 개행문자를 추가하는 점을 제외하면 print 함수와 동일하다. print 함수는 인자로 주어지는 내용을 문자열로 변환하여 출력하며 문자나 문자열뿐만 아니라 정수와 실수 역시 문자열로 출력할 수 있다. 반면 write 함수는 바이트 단위의 데이터 출력을 위해 사용된다.

```
size_t print(value, format)
```

- 매개변수

value : 출력값 (char, char 배열, String, 정수, 실수 등)

format : 출력 형식

- 반환값 : 시리얼 포트에 출력된 바이트 수

```
size_t write(uint8_t ch)
```

- 매개변수

ch : 출력할 바이트 단위 데이터

- 반환값 : 시리얼 포트에 출력된 바이트 수

스케치 1은 print 및 write 함수를 사용하여 다양한 유형의 데이터를 시리얼 모니터로 출력하는 예이다.

스케치 1. print 함수와 write 함수

```
void setup() {
  Serial.begin(9600);           // 시리얼 포트 초기화
}

void loop() {
  Serial.print("String : ");
  Serial.println("Test String"); // 문자열 출력

  Serial.print("Char   : ");
  Serial.println('c');           // 문자 출력

  Serial.print("Integer : ");
  Serial.println(123);           // 정수 출력

  Serial.print("Float   : ");
  Serial.println(3.14);          // 실수 출력
```

```

byte data = 65;
Serial.println();
Serial.print("With print : ");
Serial.println(data);                // print 함수

Serial.print("With write : ");
Serial.write(data);                  // write 함수

while(true);
}

```

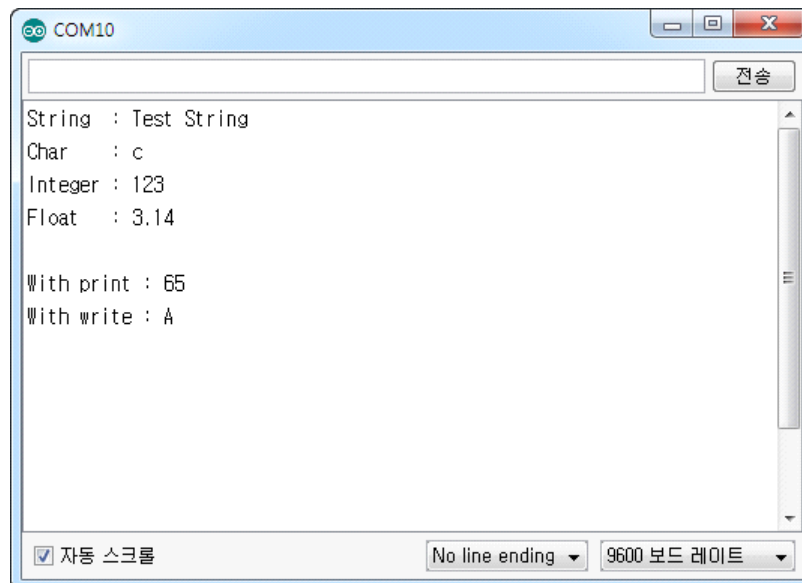


그림 2. 스케치 1 실행 결과

스케치 2는 정수를 이진수, 십진수 및 십육진수로 출력하는 방법과 실수의 소수점 이하 자리수를 지정하는 방법을 보여주는 스케치의 예이다.

스케치 2. 정수의 진법 변환 및 실수의 소수점 이하 자리수 지정

```

void setup() {
  Serial.begin(9600);                // 시리얼 포트 초기화

```

```
}  
  
void loop() {  
    int n = 123;  
    float f = 3.1415927;  
  
    Serial.println(n, BIN);           // 이진수  
    Serial.println(n, DEC);           // 십진수. 디폴트값으로 DEC는 생략 가능  
    Serial.println(n, HEX);           // 십육진수  
  
    Serial.println();  
    Serial.println(f);                // 디폴트값으로 소수점 이하 두 자리  
    Serial.println(f, 5);              // 소수점 이하 다섯 자리  
  
    while(true);  
}
```

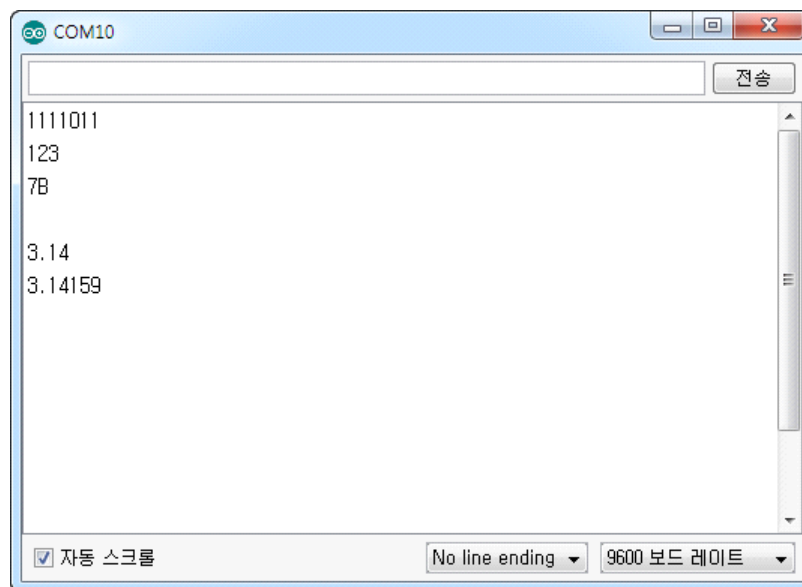


그림 3. 스케치 2 실행 결과

UART 시리얼 통신은 양방향 통신이므로 연결된 시리얼 장치로 데이터를 송신하는 함수와 더불어 데이터 수신을 위한 함수들도 마련되어 있다. 데이터 수신을 위한 함수에는 수신된 데이터가 존재하는지의 여부를 알아보는 `available`, 수신된 데이터를 바이트 단위로 읽어오는

read 함수와 peek 함수가 있다. read 함수의 경우 읽어온 데이터를 버퍼에서 삭제하는 반면 peek 함수는 버퍼에서 삭제하지 않는 차이가 있다.

```
int available(void)
```

- 매개변수 : 없음
- 반환값 : 시리얼 통신 수신 버퍼에 저장된 데이터의 바이트 수

```
int peek(void)
```

- 매개변수 : 없음
- 반환값 : 시리얼 통신 수신 버퍼의 첫 번째 바이트 데이터 또는 -1

```
int read(void)
```

- 매개변수 : 없음
- 반환값 : 시리얼 통신 수신 버퍼의 첫 번째 바이트 데이터 또는 -1

스케치 3은 시리얼 모니터에서 입력한 문자를 아두이노에서 수신하고 이를 다시 시리얼 모니터로 재전송하여 출력하는 스케치의 예이다.

스케치 3. 시리얼 모니터에서 전송된 데이터 에코 백

```
void setup() {
  Serial.begin(9600);           // 시리얼 포트 초기화
}

void loop() {
  if(Serial.available() > 0){    // 수신된 데이터 존재 여부 확인
    byte data = Serial.read();   // 바이트 단위로 읽기

    Serial.print("Echo back : ");
    Serial.write(data);          // 문자 출력
  }
}
```

```
Serial.print(" ");  
Serial.println(data);           // ASCII 값 출력  
}  
}
```

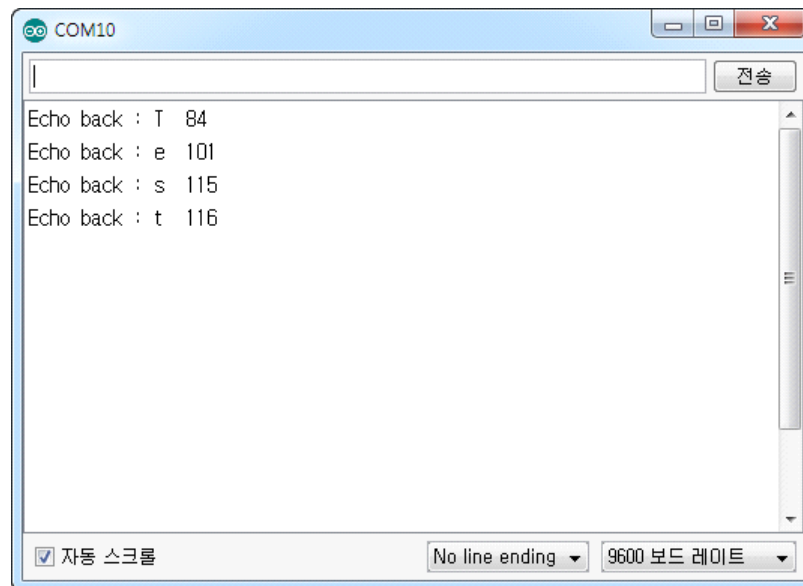


그림 4. 스케치 3 실행 결과

3. String 클래스

String 클래스는 문자열을 다루기 위한 클래스이다. C에서는 문자열을 다루기 위한 별도의 데이터 형이 존재하지 않으므로 문자 배열로 문자열을 처리한다. 반면 C++에서는 문자열을 위한 전용 클래스를 제공하고 있으며 아두이노 역시 String 클래스를 제공하고 있다. String 클래스를 사용하면 문자열은 물론 숫자도 문자열로 쉽게 변환하여 새로운 문자열로 만들 수 있다. 스케치 4는 문자열을 생성하고 기존의 문자열 또는 다른 유형의 데이터를 이용하여 새로운 문자열을 생성하는 스케치의 예이다.

스케치 4. 문자열 생성

```
void setup() {  
  Serial.begin(9600);           // 시리얼 포트 초기화
```

```
}

void loop() {
    String str1 = "One string", str2 = "Another string";
    int n = 1234;
    float f = 3.14;
    char c = 'A';

    Serial.println(str1);                // String 출력
    Serial.println(str1 + " " + str2);    // 문자열 연결

    Serial.println(String(n));            // 정수로부터 10진 문자열 생성
    Serial.println(String(n, BIN));       // 정수로부터 2진 문자열 생성
    Serial.println(String(n, HEX));       // 정수로부터 16진 문자열 생성

    // 실수의 경우 문자열로 직접 변환할 수 없다.
    // 단, Serial 클래스의 print 함수로 문자열 형식으로 출력은 가능하다.
    // Serial.println(String(f));
    Serial.println(f);

    // 다른 타입의 데이터를 연결하여 새로운 String 객체를 생성한다.
    Serial.println("String + integer : " + String(n));

    String str3 = "String + character : ";
    str3 += n;
    Serial.println(str3);

    while(true);
}
```

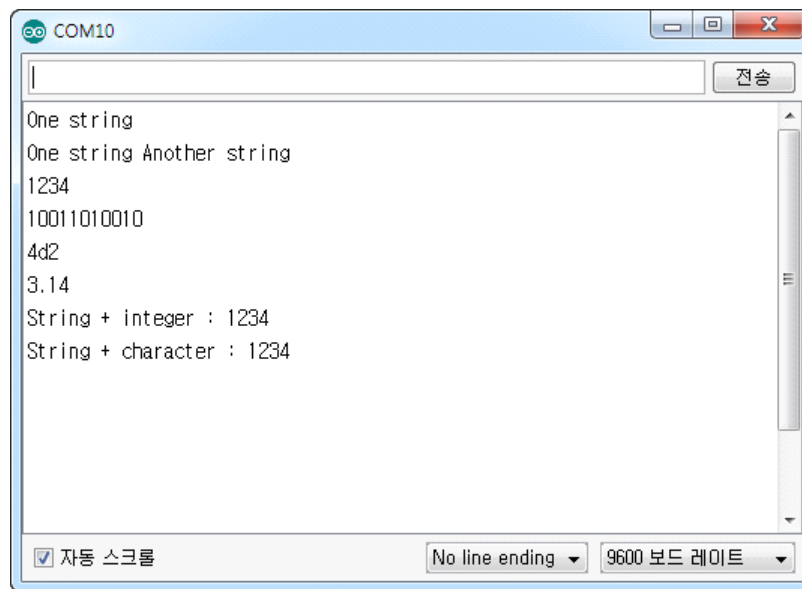


그림 5. 스케치 4 실행 결과

String 클래스에는 문자열의 비교 및 검색 등을 위해 다양한 멤버함수가 제공되고 있다. 흔히 사용되는 멤버 함수에는 사전 순서에 따른 정렬을 위한 `compareTo` 함수와 두 문자열의 내용일 동일한지 비교하는 `equals` 및 `equalsIgnoreCase` 함수가 있다. `equalsIgnoreCase` 함수는 알파벳 비교 시에 대소문자를 구별하지 않는 것을 제외하고는 `equals` 함수와 동일하다.

```
int compareTo(const String &string2)
```

- 매개변수

string2 : 비교 대상이 되는 문자열

- 반환값 : 순서상으로 string2가 먼저 나오면 양의 값, string2와 동일하면 0, string2가 뒤에 나오면 음의 값을 반환

스케치 5는 다섯 개의 문자열을 비교하여 오름차순으로 정렬한 결과를 보여주는 스케치의 예이다. 실행 결과에서 볼 수 있듯이 아두이노에서는 한글이 지원되지 않는다.

스케치 5. 문자열 정렬

```
void setup() {  
  Serial.begin(9600);           // 시리얼 포트 초기화
```

```
}

void loop() {
    // 정렬할 문자열 배열
    String str[5] = {"abc", "ABC", "!@#", "라라라", "123"};

    // 문자열 정렬
    for(int i = 0; i < 4; i++){
        for(int j = i + 1; j < 5; j++){
            int compare = str[i].compareTo(str[j]);
            if(compare > 0){                // 오름차순으로 정렬
                String temp = str[i];
                str[i] = str[j];
                str[j] = temp;
            }
        }
    }

    // 정렬된 문자열 출력
    for(int i = 0; i < 5; i++){
        Serial.println(String(i) + " : " + str[i]);
    }

    while(true);
}
```

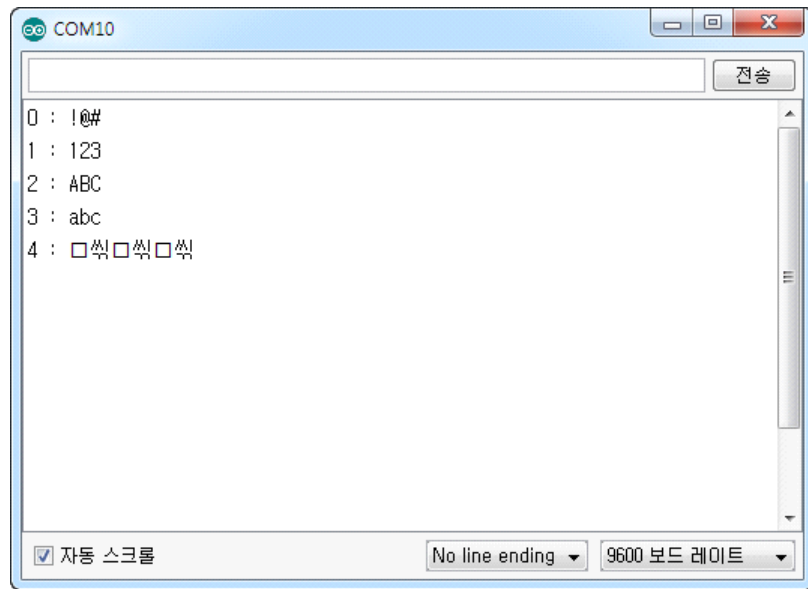


그림 6. 스케치 5 실행 결과

이외에도 Serial 클래스와 String 클래스에는 여러 가지 유용한 멤버 함수들이 다수 정의되어 있으며 다른 멤버 함수들은 부록을 참고하기 바란다.