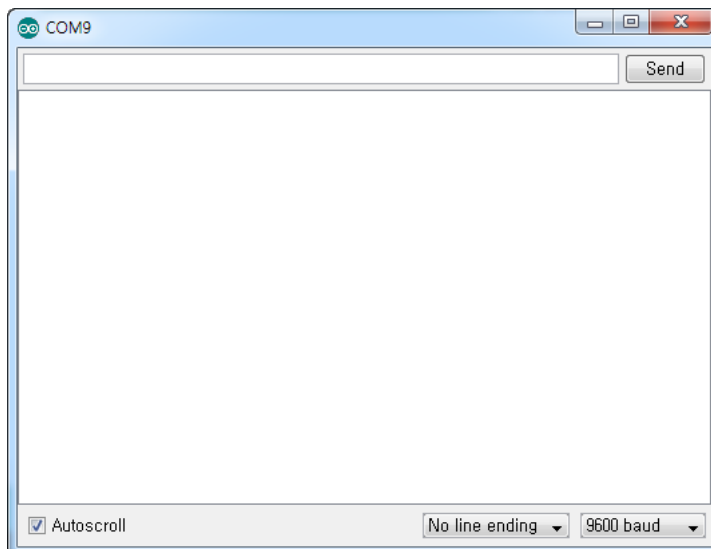
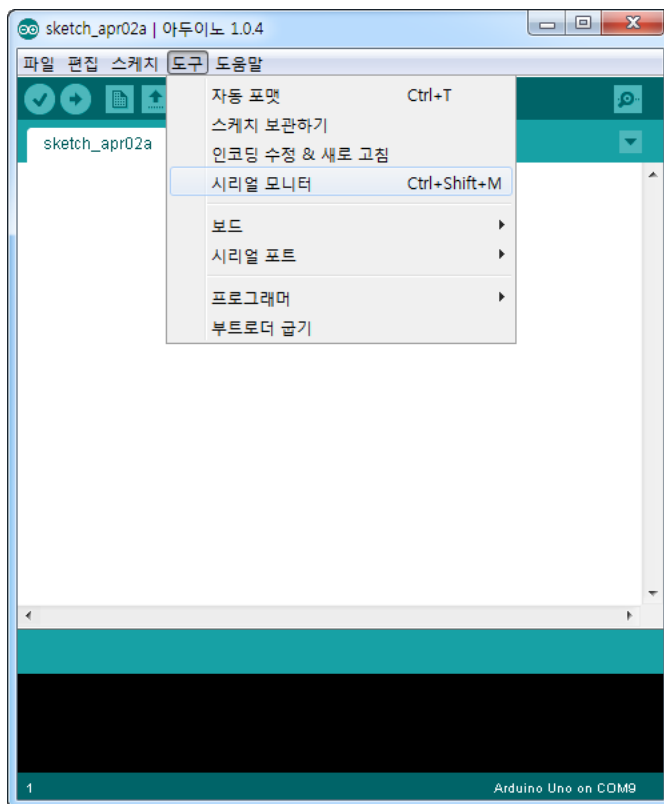


4. 시리얼 통신(Serial Communication)

시리얼 통신이란 하나의 선을 통하여 데이터를 순차적으로 보내고 이를 통해 두 기기의 의사소통이 이루어지는 것이다. 주위에서 흔히 볼 수 있는 USB 형태의 입력 장치들은 시리얼 통신 방식을 사용한다. 아두이노 보드는 시리얼 통신을 통해 컴퓨터와 다른 장치간에 연결을 할 수 있다. 이전의 작업에서 작성된 스케치 코드는 아두이노보드에 있는 USB-to-serial 칩을 통해 컴퓨터에 연결된 USB 포트를 시리얼 포트로 사용하여 업로드가 수행된 것이다. 시리얼 통신은 프로그램의 디버깅에 효율적으로 사용될 수 있다. 업로드된 프로그램이 실행되고 있을 때 각종 변수의 값 등을 확인하기 위해서는 아두이노에서 시리얼 통신을 통해 컴퓨터로 값을 보내주고 이 값을 시리얼 모니터를 통해서 확인하게 되는 것이다.



아두이노가 보낸 데이터는 시리얼 모니터의 화면에 나타나게 되며, 사용자 컴퓨터에서 아두이노로 데이터를 보낼 때는 값을 입력 후 Send 버튼을 누르면 된다. 이때 시리얼 모니터와 아두이노 보드의 통신 속도를 동일하게 맞추어 주어야 한다.

No line Ending은 전송되는 메시지의 끝에 캐리지 리턴을 추가할 것인지를 결정 짓는 것이다. 시리얼 통신을 구현하기 위해서는 하드웨어와 소프트웨어가 필요하며, 하드웨어는 아두이노와 대상 장치 간에 전기적인 신호를 통해 정보를 주고 받는 역할을 하며, 소프트웨어는 하드웨어 위에서 인식할 수 있는 비트나 바이트를 전송하게 되는 것이다. 하드웨어와 관련한 복잡한 사항들은 아두이노의 시리얼 라이브러리가 대부분 처리하므로 크게 신경쓰지 않아도 된다. 아두이노와의 통신을 통해 주고 받는 정보를 시각적 혹은 의미있게 표현하기 위해서는 시리얼 모니터로는 한계가 있으며, 추가로 시리얼 통신 부분을 프로그래밍할 수 있는 프로그래밍 언어를 통해서 아두이노와 통신을 하게 된다. 아두이노가 보내는 온도 센서의 값을 그래프로 표현하기 위해서는 적합한 언어를 선택하고 해당 프로그램에서 센서의 값을 시각적으로 표현하는 작업을 추가로 해주어야 한다.

가. 기본 명령

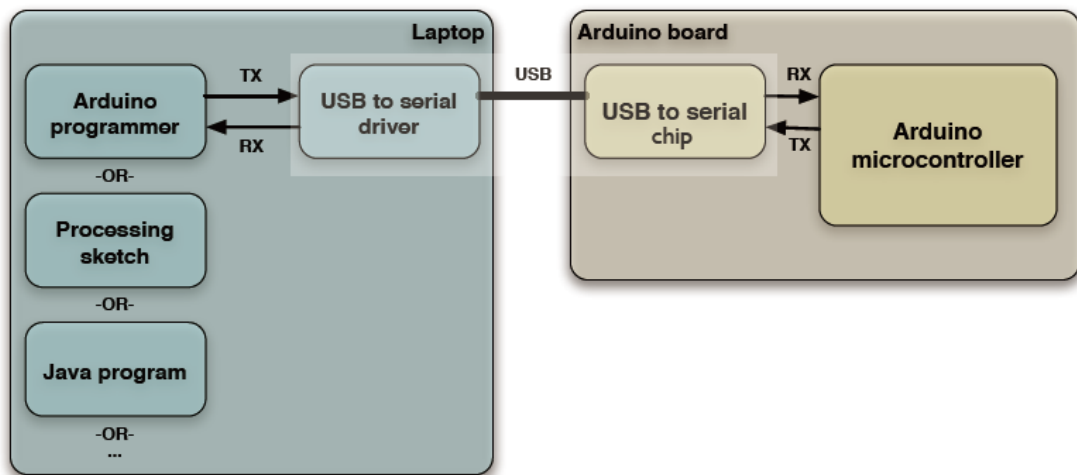
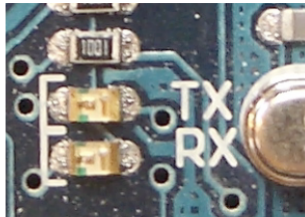
시리얼 통신을 사용하기 위해서 아두이노에 사용되는 기본 명령은 다음과 같다.

Serial.begin() - 시리얼 통신을 사용하기 위한 준비단계

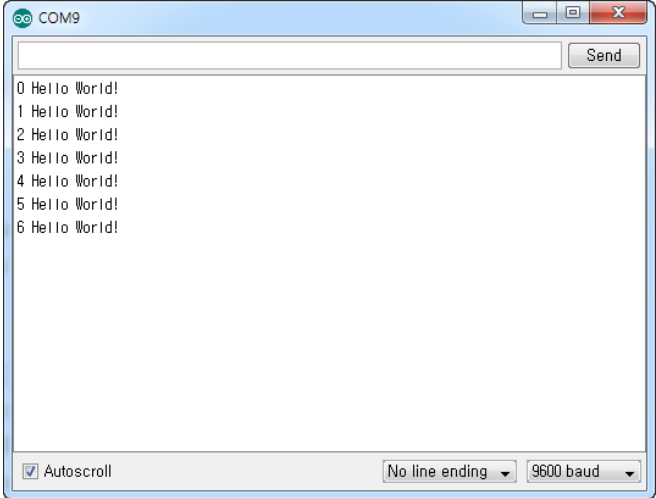
Serial.print() - 시리얼 통신을 통해 컴퓨터로 데이터를 보낸다.

Serial.read() - 시리얼 통신을 통해 컴퓨터로부터 데이터를 읽는다.

- TX – sending to PC
- RX – receiving from PC
- Used when programming or communicating



프로그램 - Serial 출력

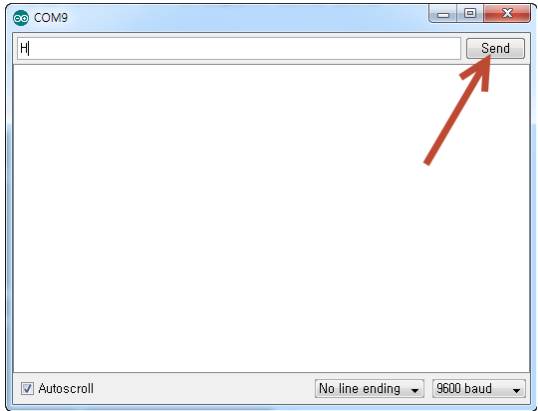
<pre> int led = 13; int i = 0; void setup() { pinMode(led, OUTPUT); Serial.begin(9600); } void loop() { Serial.println(i+ +); Serial.println(" Hello World!"); digitalWrite(led, HIGH); delay(500); digitalWrite(led, LOW); delay(500); } </pre>	
--	--

프로그램 분석

위 예제코드는 시리얼 포트에 1씩 증가하는 `i` 값과 문자열을 보내며, 13번 포트에 연결된 LED를 점멸하고 있다. 아두이노 보드를 보면 일정한 시간 별로 TX버튼이 점멸하는 것을 볼 수 있으며, 이는 시리얼 통신을 통해 아두이노 보드가 컴퓨터로 자료를 전송하고 있음을 의미한다.

프로그램 - Serial 입력

이번에는 컴퓨터에서 아두이노로 시리얼 포트를 통해 값을 보내고 아두이노에서 이 값을 받아 H이면 LED를 점멸시키는 코드를 작성해보자.

<pre> int led = 13; int val = 0; void setup() { pinMode(led, OUTPUT); Serial.begin(9600); } void loop() { if (Serial.available()) { val = Serial.read(); if (val == 'H') { digitalWrite(led, HIGH); delay(1000); digitalWrite(led, LOW); } } } </pre>	
--	--

프로그램 분석

시리얼 포트에서 데이터를 받아들이는 부분만 달라졌다. loop문 안에서 Serial에 이용할 수 있는 데이터가 있다면 이를 읽고 이 값을 H와 비교한 후에 LED를 일정 시간동안 켜주는 코드이다.

Serial.read()의 리턴 값은

the first byte of incoming serial data available(or -1 if no data is available) - int

나. Serial.print() & Serial.write()

Serial.print(val), Serial.print(val, format)

format 부분은 전송되는 값의 데이터 타입을 결정짓는다.

Serial.print(78, BIN) : "1001110"

Serial.print(78, OCT) : "116"

Serial.print(78, DEC) : "78"

Serial.print(78, HEX) : "4E"

Serial.println(1.23456, 0) : "1"

Serial.println(1.23456, 2) : "1.23"

Serial.println(1.23456, 4) : "1.2346"

Serial.write()는 시리얼 포트를 통해 바이너리 데이터를 보낸다는 차이가 있다.

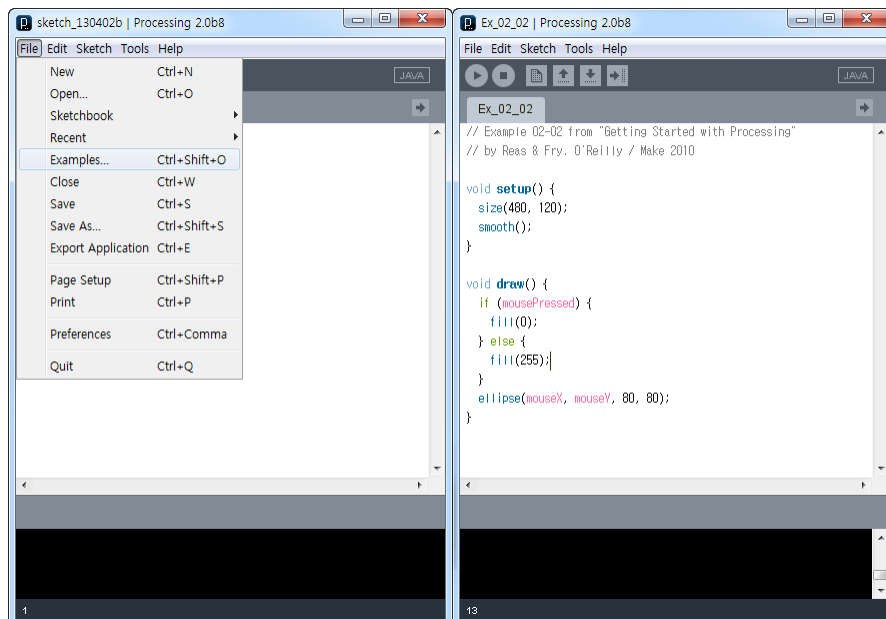
Serial.write(65); : A

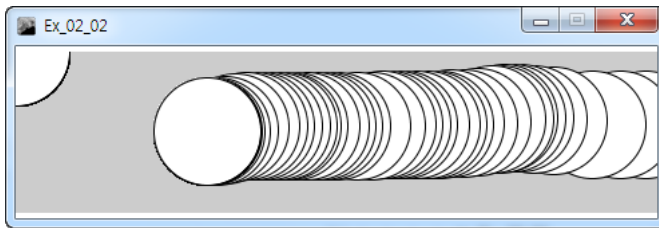
다. 데이터의 시각화

아두이노가 시리얼 포트에 보내온 데이터를 시각화해보자. 이는 다양한 센서의 값들을 시각적으로 표현해야 하는 작업이 많기 때문이다. 이번 작업에서는 편리하게 그래픽 영상 작업을 할 수 있는 프로세싱이라는 언어를 이용하여 작업을 한다. 프로세싱은 아두이노와 유사한 IDE 환경을 가지고 있으며, 자바를 기반으로 개발이 되었다. 프로세싱은 setup()과 draw()라는 두 가지 기본 함수를 가진다.

1) 프로세싱 다운로드

프로세싱을 processing.org에서 다운 받을 수 있으며 설치하는 아두이노와 마찬가지로 압축을 해제하는 과정으로 끝이 난다. 다양한 예제파일들이 포함되어 있어 스스로 학습을 할 수 있다.





2) 시리얼 데이터 표현하기

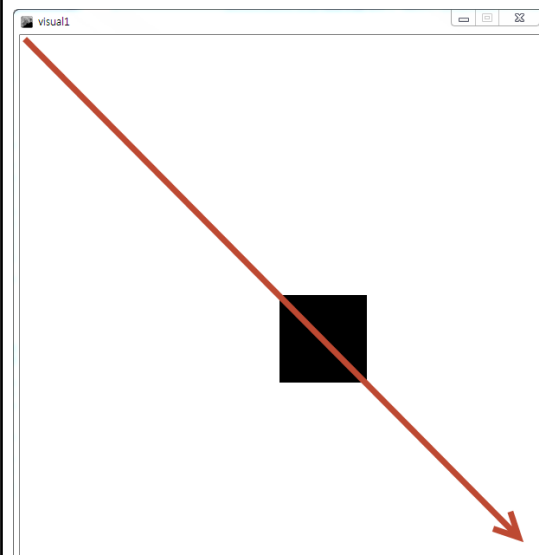
아두이노가 보내오는 시리얼 데이터를 읽기 위해 프로세싱은 시리얼 포트를 열고 해당 자료를 읽어 표시해야 한다. 이번 작업에서는 아두이노가 1씩 증가시키는 값을 입력으로 받아 화면상의 직사각형을 움직여 보는 작업이다.

```
import processing.serial.*;

int lf = 10;
Serial myPort;
String myString;
int value;

void setup() {
  size(600, 600);
  println(Serial.list());
  myPort = new Serial(this, Serial.list()[4], 9600);
}

void draw() {
  while (myPort.available() > 0) {
    myString = myPort.readStringUntil(lf);
    if (myString != null) {
      myString = trim(myString);
      value = int(myString);
      println(value);
      background(255);
      fill(0);
      rect(value*5, value*5, 100, 100);
    }
  }
}
```



프로그램 분석

setup 함수에서 화면의 크기를 선언하고, 0번 포트 번호를 사용한다. 이는 컴퓨터 별로 다를 수 있다. draw 함수에서 읽을 데이터가 있다면 해당 값을 출력하고 화면에 정사각형을 그린다. value 값이 증가함에 따라 정사각형은 화면의 대각선을 따라 움직이게 된다.