

# Serial programming with Arduino Board

2015004375 김다빈

## - \*결과

```
Markers Properties Servers Data Source Explorer Snippets Console
Serial_Communication [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (2017. 4. 2. 오후 4:46:45)
WARNING: RXTX Version mismatch
        Jar version = RXTX-2.2-20081207 Cloudhopper Build rxtx.cloudhopper.net
        native lib Version = RXTX-2.2pre2
500
delay : 500ms
1000
delay : 1000ms
100
delay : 100ms
400
delay : 400ms
```

## - 코드 설명

### 1. Serial\_Communication.java( main 함수 클래스 )

```
SimpleRead.java SimpleWrite.java Serial_Communication.java
1 import java.io.*;
2
3
4
5
6
7 public class Serial_Communication {
8
9     static CommPortIdentifier portId;
10    static Enumeration portList;
11    static SerialPort serialPort;
12
13    public static void main(String[] args) {
14
15        // 시스템에 있는 가능한 드라이버의 목록을 받아온다.
16        portList = CommPortIdentifier.getPortIdentifiers();
17
18        // enumeration type 인 portList 의 모든 객체에 대하여
19        while (portList.hasMoreElements()) {
20            // enumeration 에서 객체를 하나 가져온다.
21            portId = (CommPortIdentifier) portList.nextElement();
22
23            // 가져온 객체의 port type 이 serial port 이면
24            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
25                if (portId.getName().equals("/dev/tty.usbmodem1421")) {
26
27                    try {
28                        serialPort = (SerialPort)portId.open("Serial_CommunicationApp", 2000);
29                    } catch (PortInUseException e) {}
30
31                    // 시리얼 통신 설정. Data Bit는 8, Stop Bit는 1, Parity Bit는 없음.
32                    try {
33                        serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
34                        //serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
35
36                    } catch (UnsupportedCommOperationException e) {}
37
38                    /* 시리얼 포트에 데이터가 도착하면 이벤트가 한 번 발생되는데
39                     * 이 때, 자신이 리스너로 등록된 객체에게 이벤트를 전달하도록 허용. */
40                    Serial_Communication.serialPort.notifyOnDataAvailable(true);
41
42                    SimpleWrite writer = new SimpleWrite();
43                }
44            }
45        }
46    }
47 }
48 }
```

- Serial port를 열고, 세팅을 해준 후에, 아두이노에게 input값을 넘길 수 있도록 writer 호출.

## 2. SimpleWrite.java( writer(OutputStream) 클래스 )



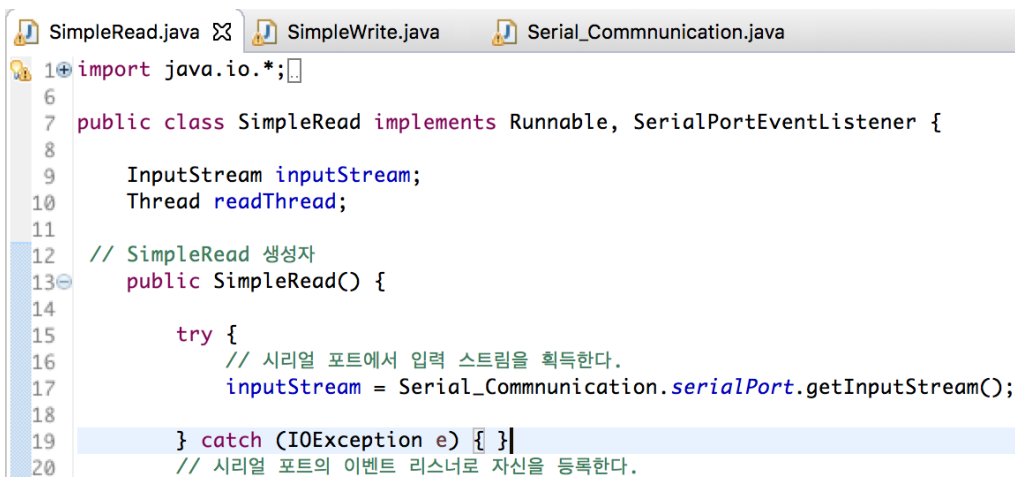
```

1 import java.io.*;
5
6 public class SimpleWrite{
7
8     String tmp;
9     OutputStream outputStream;
10
11
12 public SimpleWrite() {
13
14     try {
15         outputStream = Serial_Communication.serialPort.getOutputStream();
16
17     } catch (IOException e) { }
18
19     while(true){
20         try{
21             Scanner in= new Scanner(System.in);
22             tmp=in.nextLine();
23             outputStream.write(tmp.getBytes());
24
25             SimpleRead reader = new SimpleRead();
26
27         }catch(IOException e){}
28     }
29
30 }
31 }

```

- input을 아두이노에게 넘겨주는 클래스.
- 반복적으로 스캐너를 받아 input을 계속 넘겨준다.
- 넘겨준 후 아두이노에서 온 값을 바로바로 받을 수 있도록 reader 호출.

## 3. SimpleRead.java( Reader(InputStream) 클래스 )



```

1 import java.io.*;
6
7 public class SimpleRead implements Runnable, SerialPortEventListener {
8
9     InputStream inputStream;
10    Thread readThread;
11
12    // SimpleRead 생성자
13    public SimpleRead() {
14
15        try {
16            // 시리얼 포트에서 입력 스트림을 획득한다.
17            inputStream = Serial_Communication.serialPort.getInputStream();
18
19        } catch (IOException e) { }
20        // 시리얼 포트의 이벤트 리스너로 자신을 등록한다.

```

```

21     try {
22         Serial_Communication.serialPort.addListener(this);
23     } catch (TooManyListenersException e) { }
24
25     // 쓰레드 객체 생성
26     readThread = new Thread(this);
27
28     // 쓰레드 동작
29     readThread.start();
30 }
31
32 public void run() {
33     try {
34         Thread.sleep(20000);
35     } catch (InterruptedException e) { }
36 }
37 // 시리얼 포트 이벤트가 발생하면 호출. 시리얼 포트 이벤트를 전달한다.
38 public void serialEvent(SerialPortEvent event) {
39     // 이벤트의 타입에 따라 switch 문으로 제어.
40     switch (event.getEventType()) {
41     case SerialPortEvent.BI:
42     case SerialPortEvent.OE:
43     case SerialPortEvent.FE:
44     case SerialPortEvent.PE:
45     case SerialPortEvent.CD:
46     case SerialPortEvent.CTS:
47     case SerialPortEvent.DSR:
48     case SerialPortEvent.RI:
49     case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
50         break;
51     // 데이터가 도착하면
52     case SerialPortEvent.DATA_AVAILABLE:
53         byte[] readBuffer = new byte[300];    // byte 배열 객체 생성
54         int numBytes=0;
55
56         // 입력 스트림이 사용가능하면, 버퍼로 읽어 들인 후
57         // String 객체로 변환하여 출력
58         try {
59             while (inputStream.available() > 0) {
60                 numBytes = inputStream.read(readBuffer);
61             }
62             System.out.print(new String(readBuffer,0,numBytes));
63         } catch (IOException e) { }
64         break;
65     }
66 }
67 }

```

- 아두이노에서 나온 결과값을 받아온 후, 결과값이 발생했으면 이벤트 리스너를 호출하여 이벤트를 발생시킨다.
- 이벤트가 발생하면, readBuffer에 받아온 값을 전부 저장하여 String형태로 변환 후 콘솔에 결과값을 print한다.
- \* 결과는 위의 결과 콘솔창과 같다.

#### +) 프로그램 설명

총 위의 세 자바 파일과 하나의 ino파일(아두이노 내 업로드 되어 있음)로 구성되어 있으며, main 클래스에서 포트세팅을 통해 아두이노 프로그램과 자바 프로그램을 이어주면, 자바 프로세스(writer)를 통해 신호를 보내면 그 신호를 아두이노가 받아 처리 후 다시 자바프로세스에게 처리 결과를 보내주면(reader 프로세스가 결과값을 읽어온다) 콘솔 창에 결과를 보내주는 방식이다.

- 한줄평

이번 과제는 확실히 저번 과제보다 어려운 부분이 있었습니다. 과제를 이해하는 것부터 시작해서 어떻게 이어야 할 것인가, 많은 부분을 구글링을 통해 알아봐야 했고, 또 환경변수 설정, 디버그 등 꽤 오랜시간이 걸렸습니다. 그렇지만, 이 과정을 통하여 다른 언어임에도 불구하고 다른 두 프로세스가 작동 할 수 있다는 것을 알았고 직접 실행시켜 보았습니다. 어려웠지만, 꽤나 흥미로운 작업이었습니다.