Introduction to Creative Coding

Week 7 - Connecting Processing and Arduino using Serial Communications

Thomas Deacon, 2019

Topics

Week 7

- Recap week 6
- Serial Communications
- Sending Data to Processing
- Controlling Arduino from Processing

Recap wk 6

Introduction to Arduino

All materials available at:

https://github.com/JohnMechatronics/Introduction-To-Arduino

- The arduino board
- Arduino Language
- Digital Input
- Digital Output
- Analog Input

Serial Communication

Serial communications gives an easy way for an Arduino board to interact with a computer and other devices.

It works as a communication protocol because each side of the process can understand how messages are sent, and therefore, how to be received.

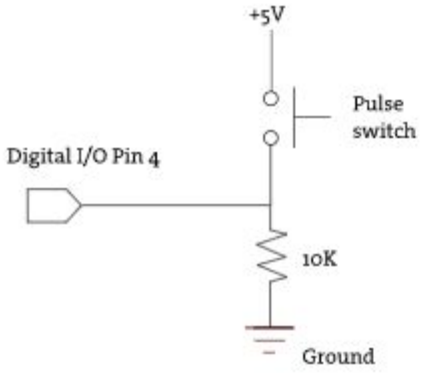
Simple Read

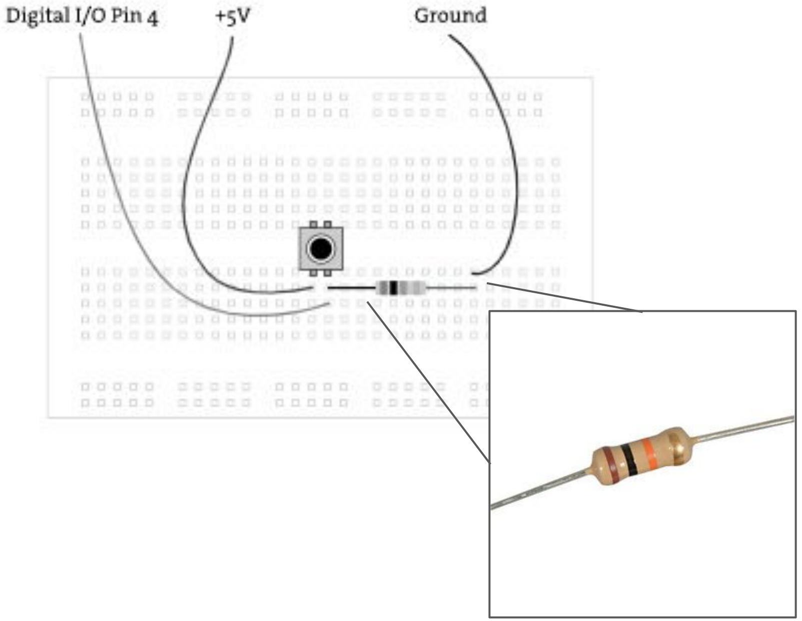
Arduino > Processing

This is the most basic setup that allows Processing to respond to events in the external world.

Depending on the state of the button a square changes its color.

Simple Read Circuit





Simple Read Code

Arduino

```
int switchPin = 4;
void setup() {
 pinMode(switchPin, INPUT);
 Serial.begin(9600);
}
void loop() {
 if (digitalRead(switchPin) == HIGH) {
   Serial.write(1);
 } else {
   Serial.write(0);
 }
 delay(100);
}
```

Processing

```
import processing.serial.*;
Serial myPort;  // Create object from Serial class
int val;       // Data received from the serial port
void setup()
{
 size(200, 200);
 String portName = Serial.list()[4]; // < Fix this
 myPort = new Serial(this, portName, 9600);
}
void draw()
{
 if ( myPort.available() > 0)
   val = myPort.read();
 background(255);
 if (val == 0)  fill(0);
 else   fill(204);
 rect(50, 50, 100, 100);
}
```

Data Send/Read

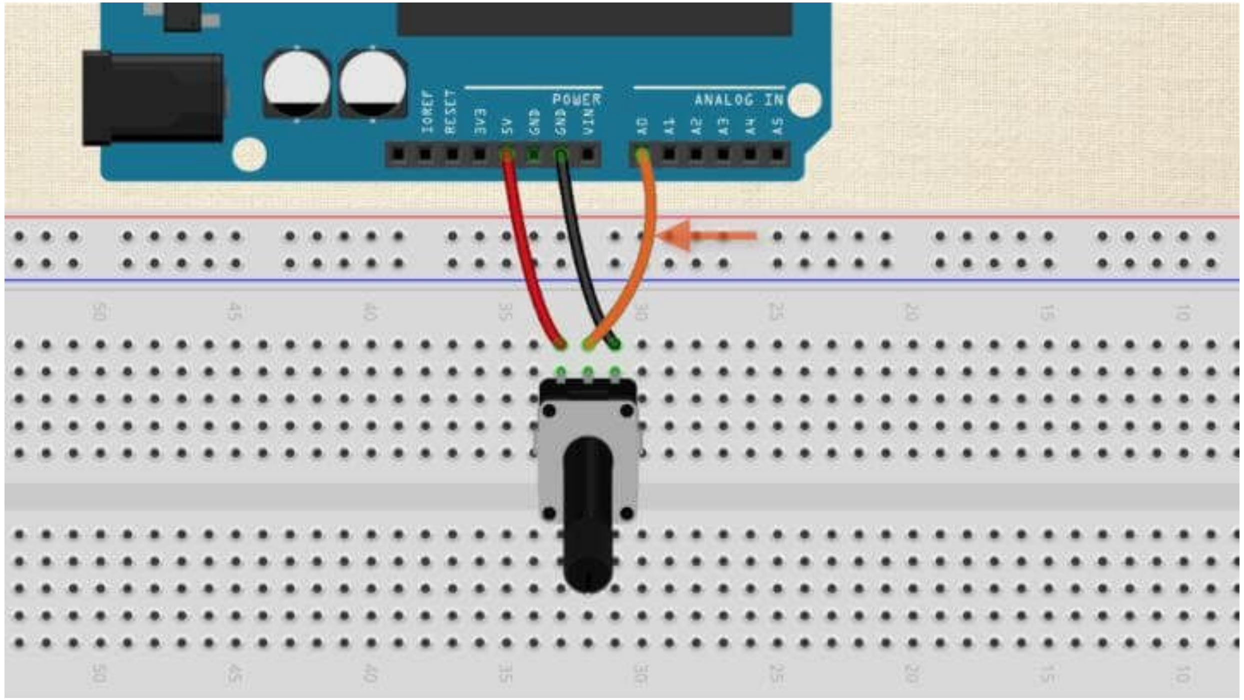
Arduino > Processing

Next we use analog input data from arduino and send it directly to Processing.

To do this we must read the whole line from the serial port, and tell Processing to chunk lines based on “newline” ASCII symbols.

Also introduce SerialEvent method in Processing as more effective management of input data.

Simple Read Circuit



Data Read Arduino Code

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

void loop() {
 int sensorValue = analogRead(A0);
 Serial.println(sensorValue/4);
 delay(5);
}
```

Data Read Processing

```
import processing.serial.*;

int linefeed = 10; // new line ASCII = 10
Serial port;
int val;

void setup() {
 size(200, 200);
 noStroke();
 frameRate(10);
 String portName = Serial.list()[4];
 port = new Serial(this, portName, 9600);
 port.bufferUntil(linefeed);
}
```

```
void draw() {
 background(204);
 fill(val);
 rect(50, 50, 100, 100);
}


void serialEvent (Serial myPort)
{
 String myString = myPort.readString();
 if (myString != null)
 {
   myString = trim(myString);
   val = int(myString);
   println(val);
 }
}
```

Multiple Data Send/Read

Arduino > Processing

Now we read multiple analog input pins from arduino and send that data in a single message.

To do this we must add the data using a “delimiter” like a comma “,”.

When reading it back in processing, we have to split that data up to retrieve the numbers.

Multi Read Arduino Code

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

void loop() {
 // Pretend read the analog pins:
 int sensorValueA = random(1024);
 int sensorValueB = random(1024);
 Serial.print(sensorValueA/4);
 Serial.print(",");
 Serial.println(sensorValueB/4);
 delay(50);
}
```

Multi Read Processing

```
import processing.serial.*;
int linefeed = 10; // new line ASCII = 10
Serial port;  // Create object from Serial class
int valA, valB;      // Data received from the serial
port
void setup() {
 size(200, 200);
 noStroke();
 frameRate(10);  // Run 10 frames per second
 String portName = Serial.list()[4];
 println(portName);
 port = new Serial(this, portName, 9600);
 port.bufferUntil(linefeed);
}
```

```
void draw() {
 background(204);
 fill(valA);
 rect(50, 50, 100, 100);
 fill(valB);
 rect(75, 75, 50, 50);
}

void serialEvent (Serial myPort)
{
 String myString = myPort.readString();
 if (myString != null)
 {
   myString = trim(myString);
   String sensors[] = split(myString, ',');
   if(sensors.length>1)
   {
     valA = int(sensors[0]);
     valB = int(sensors[1]);
   }
 }
}
```

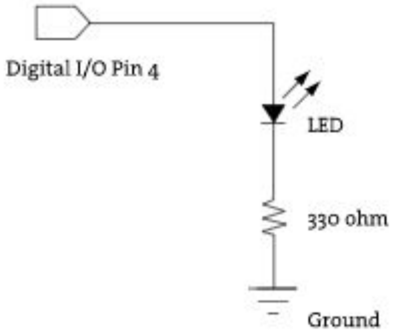
Transmitting to Arduino

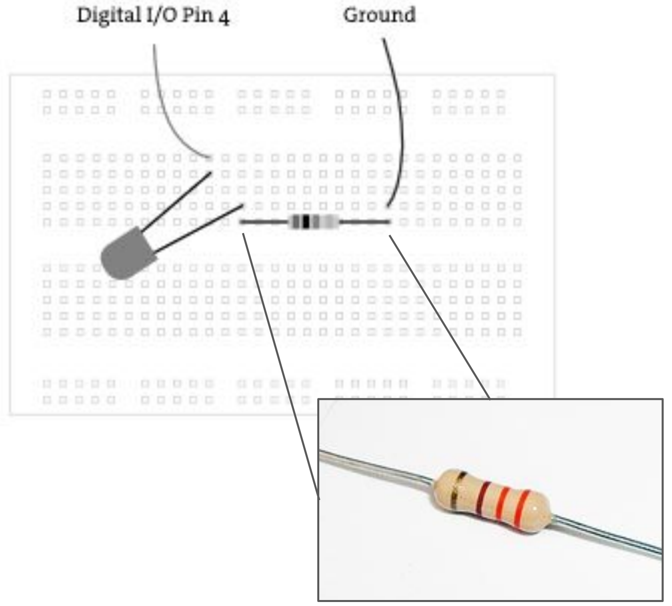
Processing > Arduino

We can also do the opposite. Control stuff in the real world with stuff on the computer. Keyboard, mouse, GUI, simulations, network etc.

In this bit, we will turn on and off a LED from interaction in Processing.

Transmit LED Circuit





Transmit LED Code Arduino

```
char val;
int ledPin = 4;

void setup() {
 pinMode(ledPin, OUTPUT);
 Serial.begin(9600);
}
```

```
void loop() {
 if (Serial.available()) {
   val = Serial.read();
 }
 if (val == 'H') {
   digitalWrite(ledPin, HIGH);
 } else {
   digitalWrite(ledPin, LOW);
 }
 delay(100);
}
```

Transmit LED Code Processing

```
import processing.serial.* ;
Serial port;

void setup() {
 size(200, 200);
 noStroke();
 frameRate(10);
 port = new Serial(this, Serial.list()[4],
9600);
}

boolean mouseOverRect() {
 return ((mouseX >= 50) && (mouseX <= 150)
&& (mouseY >= 50) && (mouseY <= 150));
}
```

```
void draw() {
 background(255);
 if (mouseOverRect() == true)  {
   fill(204);
   port.write('H');
 } else {
   fill(0);
   port.write('L');
 }
 rect(50, 50, 100, 100);
}
```

Resources for each week available at:

https://github.com/VizRCA/intro-to-creative-coding