Graph

This example shows you how to send a byte of data from the Arduino to a personal computer and graph the result. This is called serial communication because the connection appears to both the board and the computer as a serial port, even though it may actually use a USB cable, a serial to USB and a USB to serial converter.

You can use the serial monitor of the Arduino Software (IDE) to view the sent data, or it can be read by Processing (see code below), Flash, PD, Max/MSP, etc.

Hardware Required

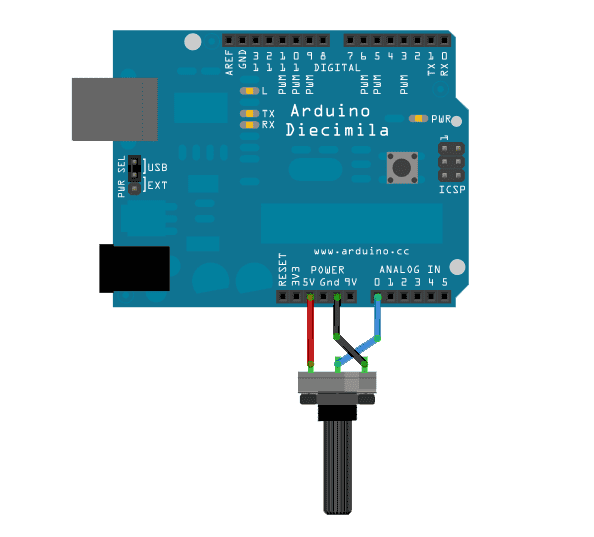
* Arduino Board
* Analog Sensor (potentiometer, photocell, FSR, etc.)

Software Required

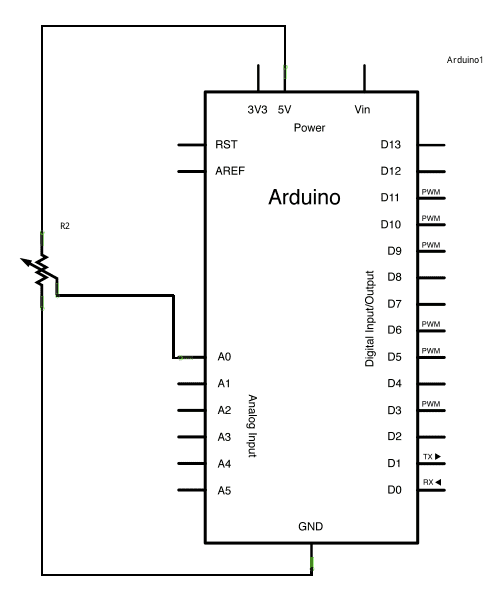
* [Processing](http://www.processing.org/) *( download at* [*https://processing.org/download/*](https://processing.org/download/)*)*

Circuit

Connect a potentiometer or other analog sensor to analog input 0.

[](https://www.arduino.cc/en/uploads/Tutorial/graph-circuit3.png)

Schematic

[](https://www.arduino.cc/en/uploads/Tutorial/AnalogReadSerial_sch.png)

**Guide:**

1. Upload **Arduino code** for Arduino

2. Run Processing.exe

3. Copy **Processing code** for Processing software

4. Click on Run icon in Processing software

5. To rotate the potentiometer and observe the result on the computer screen.

***Note:*** *Close the Arduino Serial Monitor window to give the serial port for the Processing software.*

Code

**Arduino code:**

/\* Graph

 A simple example of communication from the Arduino board to the computer: The value of analog input 0 is sent out the serial port. We call this "serial" communication because the connection appears to both the Arduino and the computer as a serial port, even though it may actually use a USB cable. Bytes are sent one after another (serially) from the Arduino to the computer.

 You can use the Arduino Serial Monitor to view the sent data, or it can be read by Processing, PD, Max/MSP, or any other program capable of reading data from a serial port. The Processing code below graphs the data received so you can see the value of the analog input changing over time.

\*/

void setup() {

 // initialize the serial communication:

**Serial**.begin(9600);

}

void loop() {

 // send the value of analog input 0:

**Serial**.println(analogRead(A0));

 // wait a bit for the analog-to-digital converter to stabilize after the last reading:

 delay(2);

}

---------------------------------------------------------------

**Processing code**

//\* Processing code for this example

 // Graphing sketch

 // This program takes ASCII-encoded strings from the serial port at 9600 baud

 // and graphs them. It expects values in the range 0 to 1023, followed by a

 // newline, or newline and carriage return

 // created 20 Apr 2005

 // updated 24 Nov 2015

 // by Tom Igoe

 // This example code is in the public domain.

 import processing.serial.\*;

**Serial** myPort;        // The serial port

 int xPos = 1;         // horizontal position of the graph

 float inByte = 0;

 void setup () {

   // set the window size:

   size(1000, 600);

   // List all the available serial ports

   // if using Processing 2.1 or later, use Serial.printArray()

   println(**Serial**.list());

   // I know that the first port in the serial list on my Mac is always my

   // Arduino, so I open Serial.list()[0].

   // Open whatever port is the one you're using.

   myPort = new **Serial**(this, **Serial**.list()[0], 9600);

   // don't generate a serialEvent() unless you get a newline character:

   myPort.bufferUntil('\n');

   // set initial background:

   background(0);

 }

 void draw () {

   // draw the line:

   stroke(127, 34, 255);

   line(xPos, height, xPos, height - inByte);

   // at the edge of the screen, go back to the beginning:

   if (xPos >= width) {

     xPos = 0;

     background(0);

   } else {

     // increment the horizontal position:

     xPos++;

   }

 }

 void serialEvent (**Serial** myPort) {

   // get the ASCII string:

   String inString = myPort.readStringUntil('\n');

   if (inString != null) {

     // trim off any whitespace:

     inString = trim(inString);

     // convert to an int and map to the screen height:

     inByte = float(inString);

     println(inByte);

     inByte = map(inByte, 0, 1023, 0, height);

   }

 }

Processing Sketch

Using the Processing sketch in the code sample above, you'll get a graph of the sensor's value. As you change the value of the analog sensor, you'll get a graph something like this:

