mkdir gpio\_python\_code

ls -1

cd gpio\_python\_code

touch 1\_helloworld.py

nano 1\_helloworld.py

sudo python 1\_helloworld.py

touch 9\_pir.py

nano 9\_pir.py

#!/usr/bin/python

import RPi.GPIO as GPIO

from time import sleep

GPIO.setmode(GPIO.BCM)

GPIO.setup(27,GPIO.OUT)

GPIO\_PIR = 7

print "PIR Module Test (CTRL-C to exit)"

# Set pin as input

GPIO.setup(GPIO\_PIR,GPIO.IN)

Current\_State = 0

Previous\_State = 0

try:

print "Waiting for PIR to settle ..."

# Loop until PIR output is 0

while GPIO.input(GPIO\_PIR)==1:

Current\_State = 0

print " Ready"

# Loop until users quits with CTRL-C

while True :

# Read PIR state

Current\_State = GPIO.input(GPIO\_PIR)

if Current\_State==1 and Previous\_State==0:

# PIR is triggered

print " Motion detected!"

# Record previous state

GPIO.output(27,GPIO.HIGH)

sleep(1)

GPIO.output(27,GPIO.LOW)

Previous\_State=1

elif Current\_State==0 and Previous\_State==1:

# PIR has returned to ready state

print " Ready"

Previous\_State=0

# Wait for 10 milliseconds

sleep(0.01)

# clean up gpio pins when we exit the script

except KeyboardInterrupt:

print " Quit"

# Reset GPIO settings

GPIO.cleanup()

sudo python 9\_pir.py

PIR Module Test (CTRL -C to exist)

Waiting for PIR to settle…

Ready

Motion detected!

Ready

Motion detected!

Ready

^C Quit

Arduino

**void** setup()

{

//initialize serial communications at a 9600 baud rate

Serial.begin(9600);

}

**void** loop()

{

//send 'Hello, world!' over the serial port

Serial.println("Hello, world!");

//wait 100 milliseconds so we don't drive ourselves crazy

delay(100);

}

Processing

Serial myPort; // Create object from Serial class

**String** val; // Data received from the serial port

**void** setup()

{

// I know that the first port in the serial list on my mac

// is Serial.list()[0].

// On Windows machines, this generally opens COM1.

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

**String** portName **=** Serial.list()[0]; //change the 0 to a 1 or 2 etc. to match your port

myPort **=** **new** Serial(**this**, portName, 9600);

}

**void** draw()

{

**if** ( myPort.available() **>** 0)

{ // If data is available,

val **=** myPort.readStringUntil('\n'); // read it and store it in val

}

println(val); //print it out in the console

}

HELLO WORLD!

Next step processing

**import** processing.serial.**\***;

Serial myPort; // Create object from Serial class

**void** setup()

{

size(200,200); //make our canvas 200 x 200 pixels big

**String** portName **=** Serial.list()[0]; //change the 0 to a 1 or 2 etc. to match your port

myPort **=** **new** Serial(**this**, portName, 9600);

}

**void** draw() {

**if** (mousePressed **==** true)

{ //if we clicked in the window

myPort.write('1'); //send a 1

println("1");

} **else**

{ //otherwise

myPort.write('0'); //send a 0

}

}

Arduino

**char** val; // Data received from the serial port

**int** ledPin **=** 13; // Set the pin to digital I/O 13

**void** setup() {

pinMode(ledPin, OUTPUT); // Set pin as OUTPUT

Serial.begin(9600); // Start serial communication at 9600 bps

}

**void** loop() {

**if** (Serial.available())

{ // If data is available to read,

val **=** Serial.read(); // read it and store it in val

}

**if** (val **==** '1')

{ // If 1 was received

digitalWrite(ledPin, HIGH); // turn the LED on

} **else** {

digitalWrite(ledPin, LOW); // otherwise turn it off

}

delay(10); // Wait 10 milliseconds for next reading

}

If we load up this code onto our Arduino, and run the Processing sketch from the previous page, you should be able to turn on an LED attached to pin 13 of your Arduino, simply by clicking within the Processing canvas

Shaking hands….. Arduino

**char** val; // Data received from the serial port

**int** ledPin **=** 13; // Set the pin to digital I/O 13

**boolean** ledState **=** LOW; //to toggle our LED

**void** setup()

{

pinMode(ledPin, OUTPUT); // Set pin as OUTPUT

//initialize serial communications at a 9600 baud rate

Serial.begin(9600);

establishContact(); // send a byte to establish contact until receiver responds

}

**void** loop()

{

**if** (Serial.available() **>** 0) { // If data is available to read,

val **=** Serial.read(); // read it and store it in val

**if**(val **==** '1') //if we get a 1

{

ledState **=** **!**ledState; //flip the ledState

digitalWrite(ledPin, ledState);

}

delay(100);

}

**else** {

Serial.println("Hello, world!"); //send back a hello world

delay(50);

}

}

**void** establishContact() {

**while** (Serial.available() **<=** 0) {

Serial.println("A"); // send a capital A

delay(300);

}

}

Shaking hands on processing

**import** processing.serial.**\***; //import the Serial library

Serial myPort; //the Serial port object

**String** val;

// since we're doing serial handshaking,

// we need to check if we've heard from the microcontroller

**boolean** firstContact **=** false;

**void** setup() {

size(200, 200); //make our canvas 200 x 200 pixels big

// initialize your serial port and set the baud rate to 9600

myPort **=** **new** Serial(**this**, Serial.list()[4], 9600);

myPort.bufferUntil('\n');

}

**void** draw() {

//we can leave the draw method empty,

//because all our programming happens in the serialEvent (see below)

}

Now for the big one: serialEvent(). Each time we see a carriage return this method gets called. We need to do a few things each time to keep things running smoothly:

* read the incoming data
* see if there’s actually anything in it (i.e. it’s not empty or ‘null’)
* trim whitespace and other unimportant stuff
* if it’s our first time hearing the right thing, change our firstContact boolean and let Arduino know we’re ready for more data
* if it’s *not* our first run, print the data to the console and send back any valid mouse clicks (as 1’s) we got in our window
* finally, tell Arduino we’re ready for more data
* **void** serialEvent( Serial myPort) {
* //put the incoming data into a String -
* //the '\n' is our end delimiter indicating the end of a complete packet
* val **=** myPort.readStringUntil('\n');
* //make sure our data isn't empty before continuing
* **if** (val **!=** null) {
* //trim whitespace and formatting characters (like carriage return)
* val **=** trim(val);
* println(val);
* //look for our 'A' string to start the handshake
* //if it's there, clear the buffer, and send a request for data
* **if** (firstContact **==** false) {
* **if** (val.equals("A")) {
* myPort.clear();
* firstContact **=** true;
* myPort.write("A");
* println("contact");
* }
* }
* **else** { //if we've already established contact, keep getting and parsing data
* println(val);
* **if** (mousePressed **==** true)
* { //if we clicked in the window
* myPort.write('1'); //send a 1
* println("1");
* }
* // when you've parsed the data you have, ask for more:
* myPort.write("A");
* }
* }
* }