

11 Introduction to Processing Program

1. What is Processing?

PROCESSING is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool.

Processing has come to be used for more advanced production-level work in addition to its sketching role. Originally built as a domain-specific extension to Java targeted towards artists and designers, Processing has evolved into a full-blown design and prototyping tool used for large-scale installation work, motion graphics, and complex data visualization. Processing is based on Java, but because program elements in Processing are fairly simple, you can learn to use it even if you don't know any Java.

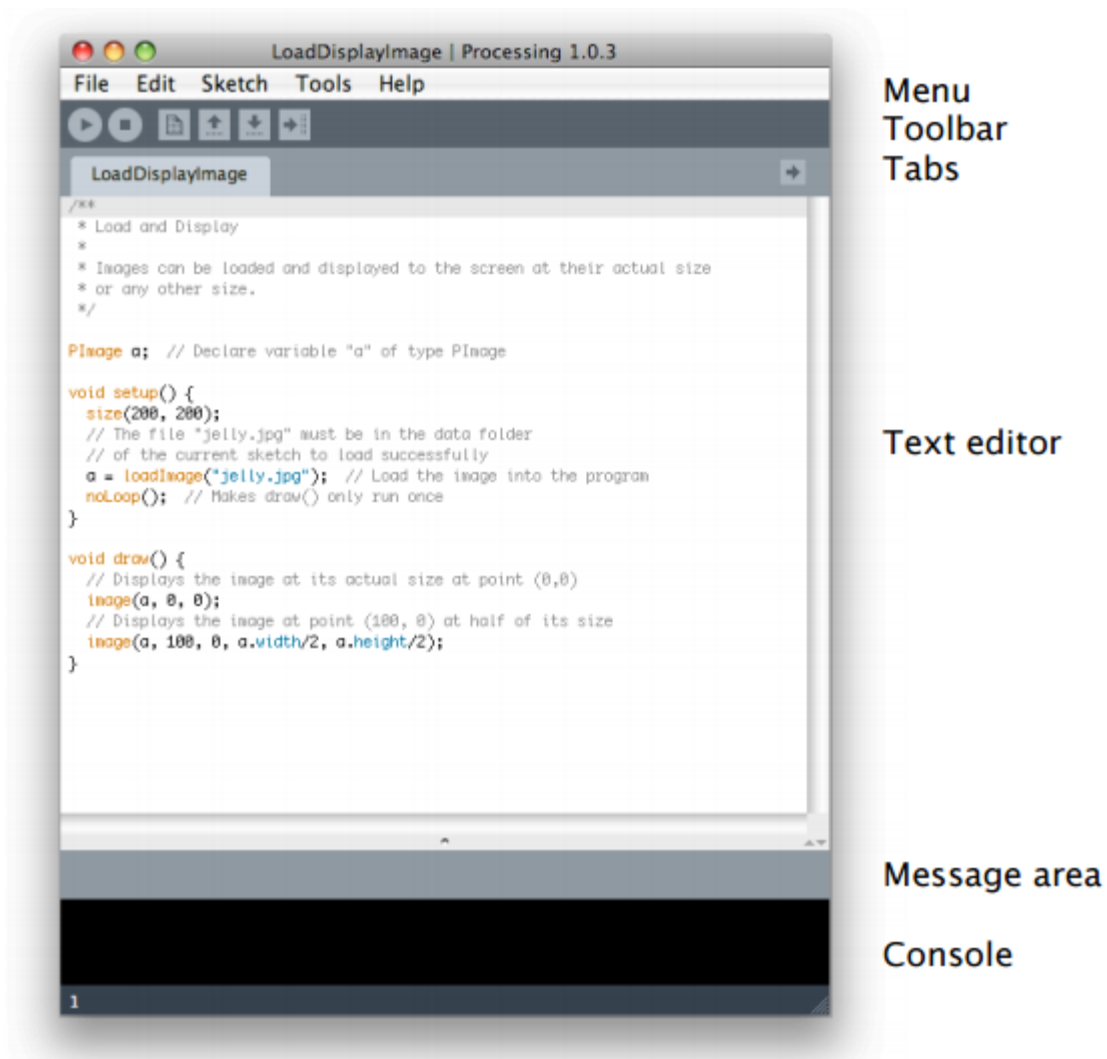
2. Why Processing?

There are many reasons Processing makes you more productive when using Arduino:

- Free to download and open source.
- Interactive programs with 2D, 3D, or PDF output.
- OpenGL integration for accelerated 2D and 3D.
- For GNU/Linux, Mac OS X, and Windows
- Over 100 libraries extend the core software.
- Well [documented](#), with many [books](#) available.

3. Download Processing

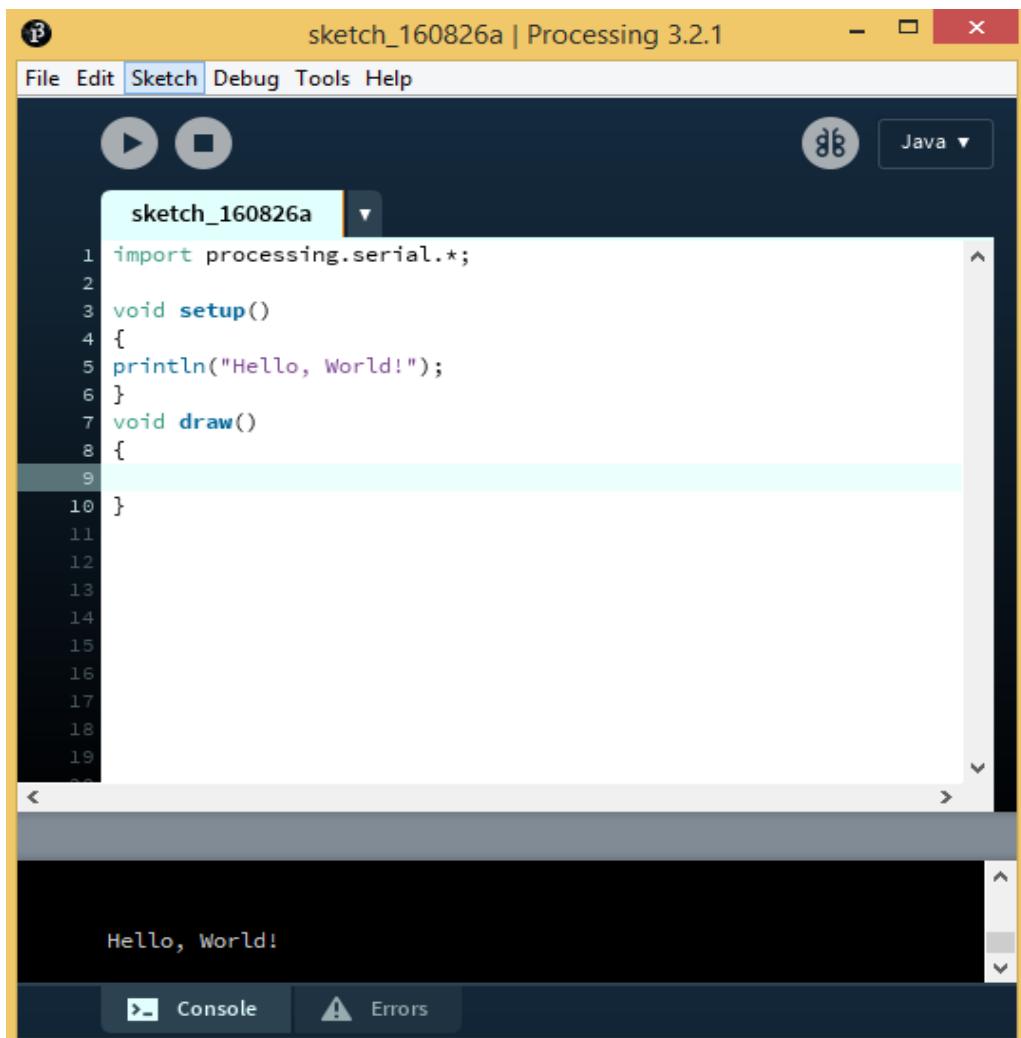
You can **install Processing** from the following link (<https://www.processing.org>).



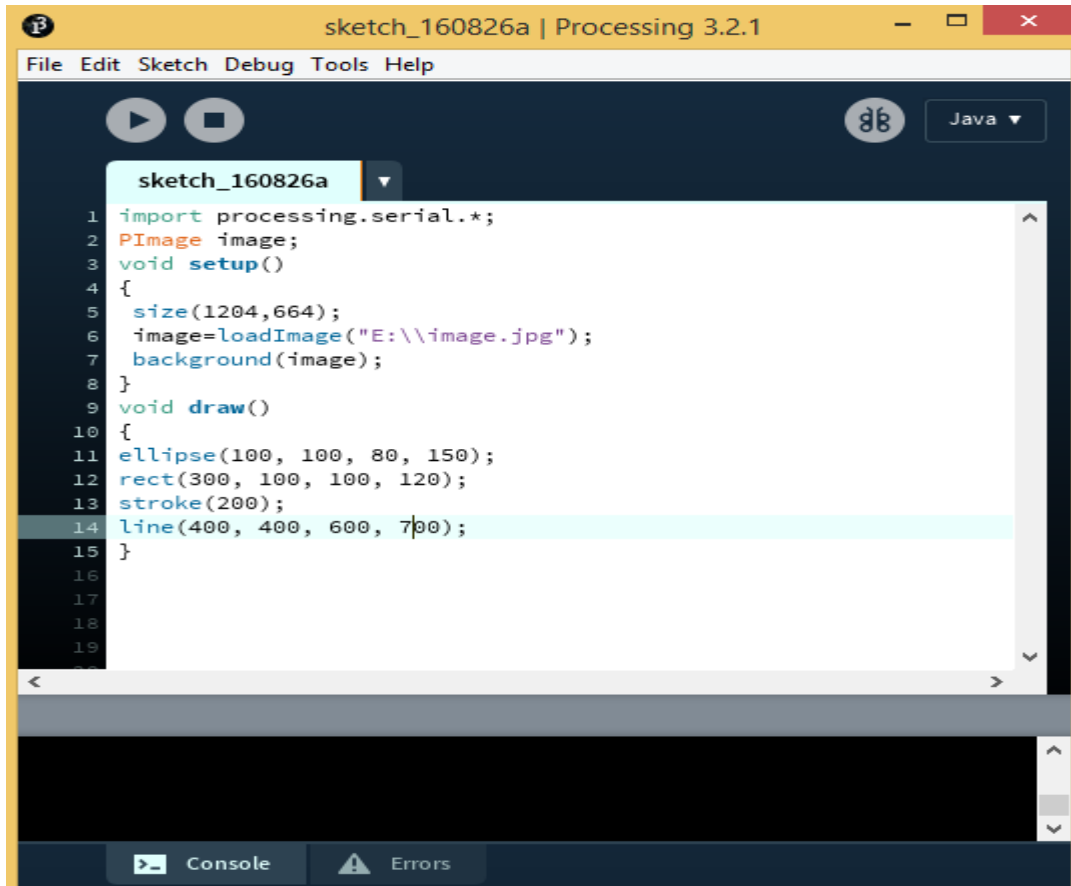
4. Sketch with Processing

A Processing program is called a *sketch*. The idea is to make Java-style programming feel more like scripting, and adopt the process of scripting to quickly write code. Sketches are stored in the *sketchbook*, a folder that's used as the default location for saving all of your projects. Sketches that are stored in the sketchbook can be accessed from File → Sketchbook. Alternatively, File → Open... can be used to open a sketch from elsewhere on the system.

4.1 Processing “Hello, world!”

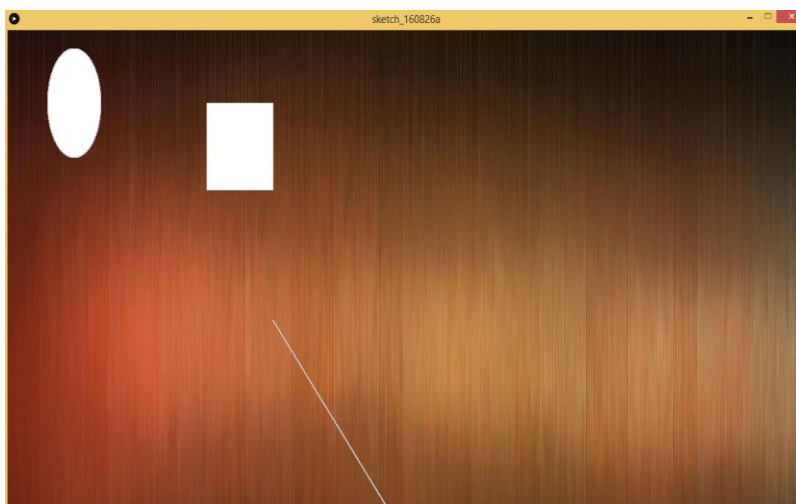


4.2 Sketch Rectangle, Ellipse, and Line.

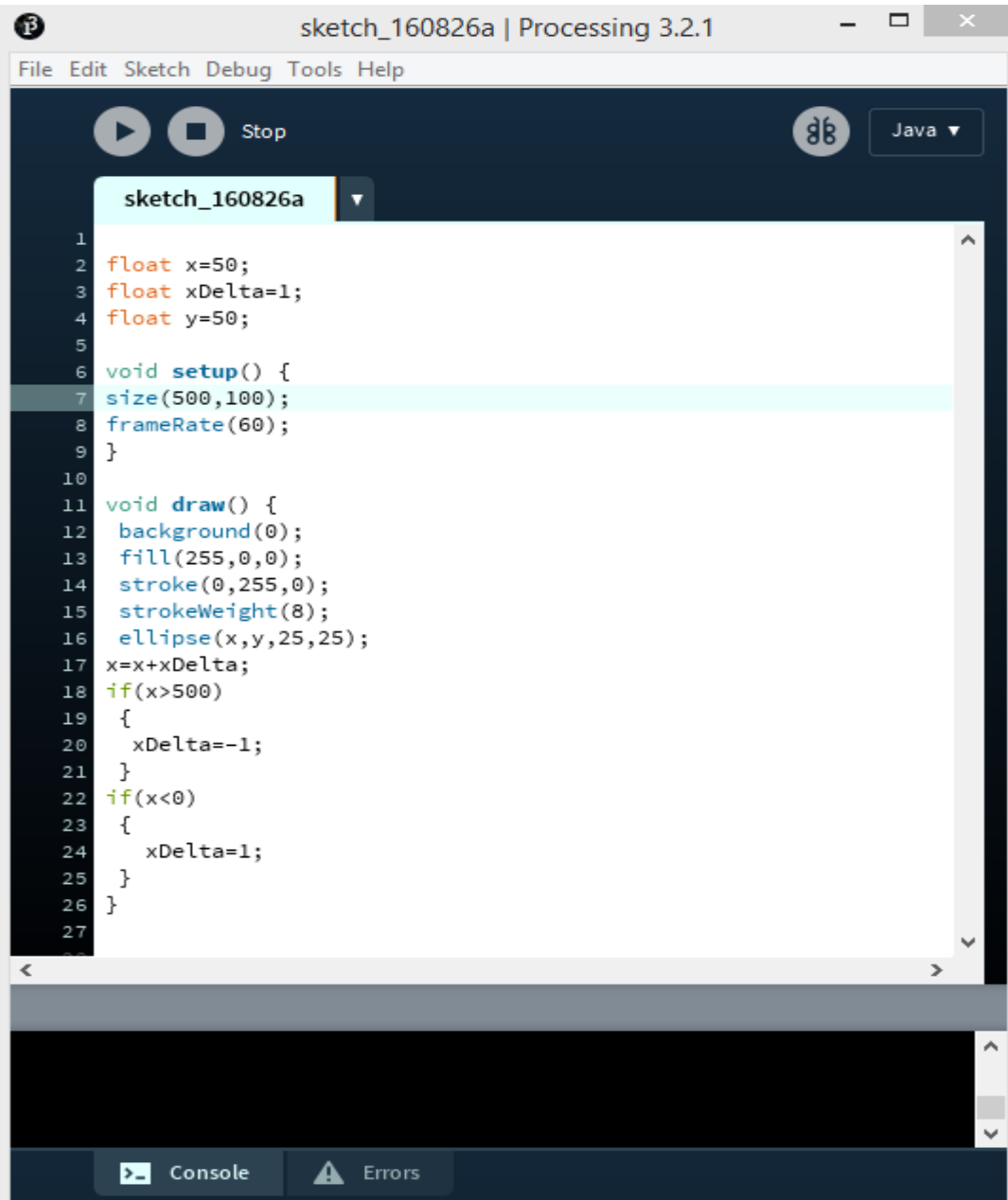


```
sketch_160826a | Processing 3.2.1
File Edit Sketch Debug Tools Help
sketch_160826a
1 import processing.serial.*;
2 PImage image;
3 void setup()
4 {
5   size(1204,664);
6   image=loadImage("E:\\image.jpg");
7   background(image);
8 }
9 void draw()
10 {
11   ellipse(100, 100, 80, 150);
12   rect(300, 100, 100, 120);
13   stroke(200);
14   line(400, 400, 600, 700);
15 }
16 }
17 }
18 }
19 }
```

Output:



4.3 Animation

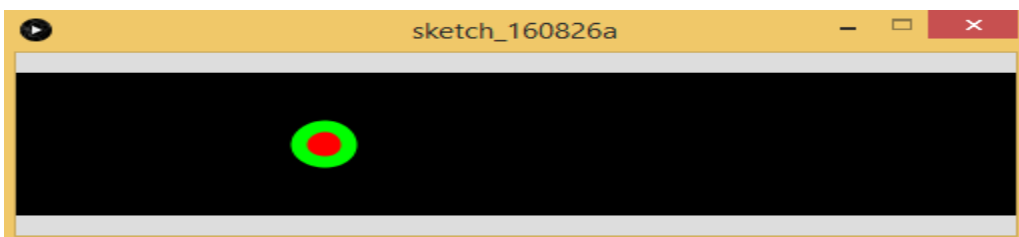


The screenshot shows the Processing IDE window titled "sketch_160826a | Processing 3.2.1". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". The toolbar contains a play button, a stop button, and a "Java" dropdown menu. The sketch name "sketch_160826a" is displayed in a dropdown menu. The code editor shows the following code:

```
1
2 float x=50;
3 float xDelta=1;
4 float y=50;
5
6 void setup() {
7 size(500,100);
8 frameRate(60);
9 }
10
11 void draw() {
12 background(0);
13 fill(255,0,0);
14 stroke(0,255,0);
15 strokeWeight(8);
16 ellipse(x,y,25,25);
17 x=x+xDelta;
18 if(x>500)
19 {
20 xDelta=-1;
21 }
22 if(x<0)
23 {
24 xDelta=1;
25 }
26 }
27
```

At the bottom of the IDE, there are tabs for "Console" and "Errors".

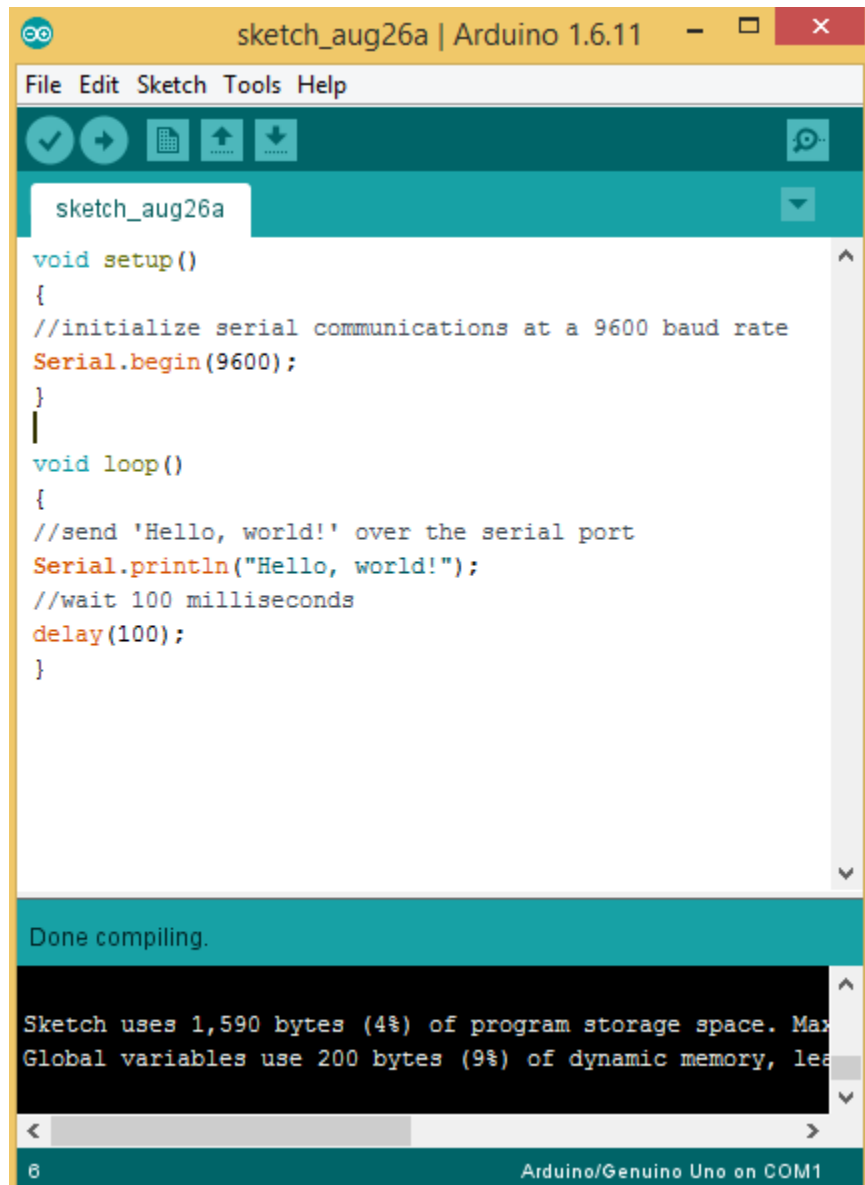
Output:



5. Send data from Arduino to Processing over the serial port

Let's start with the Arduino side of things. We'll show you the basics of how to set up your Arduino sketch to send information over serial.

- 1- Open Arduino software and type the following:



```
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
  delay(100);
}
```

Done compiling.

Sketch uses 1,590 bytes (4%) of program storage space. Max
Global variables use 200 bytes (9%) of dynamic memory, lea

6 Arduino/Genuino Uno on COM1

- 2- Plug in your Arduino board, select your board type (under Tools -> Board Type) and your Serial port (under Tools -> Serial Port) and hit the 'upload' button to load your code onto the Arduino.
- 3- Open Processing software and type the following.

```
sketch_161205a | Processing 3.2.3
File Edit Sketch Debug Tools Help

sketch_161205a
1 import processing.serial.*;
2 Serial myPort;
3 String val;
4 void setup()
5 {
6   String portName="COM4";
7   myPort=new Serial(this,portName,9600);
8 }
9 void draw()
10 {
11   if(myPort.available(>0)
12   {
13     val=myPort.readStringUntil('\n');
14     if(val != null)
15     print(val);
16   }
17 }
18
19

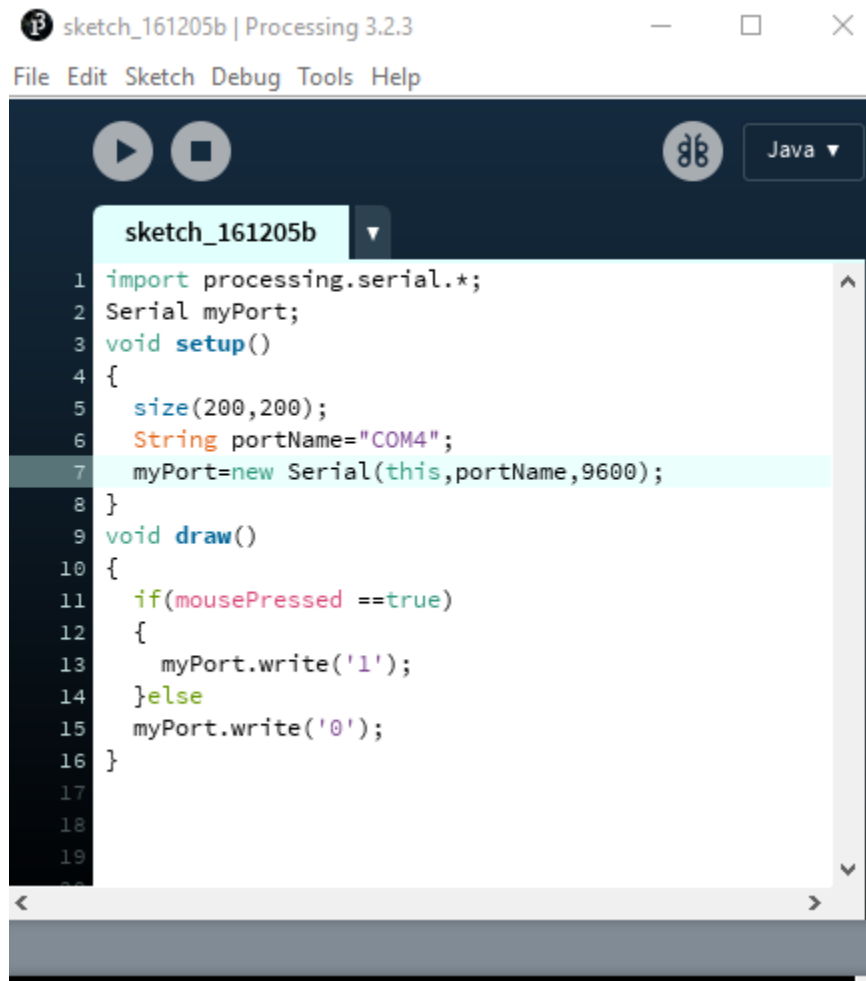
Hello, worl3d!
Hello, worl3d!
Hello, worl3d!
```

- 4- If you hit the 'run' button (and your Arduino is plugged in with the code on the previous page loaded up), you should see a little window pop-up, and after a sec you should see "Hello, World!" appear in the Processing console.

6. Send data from Processing to Arduino over the serial port

We've sent data from Arduino to Processing, but what if we want to send data the other way - from Processing to Arduino. The following example show how to send data from Processing to Arduino by click on mouse.

- 1- Open Processing software and type the following:



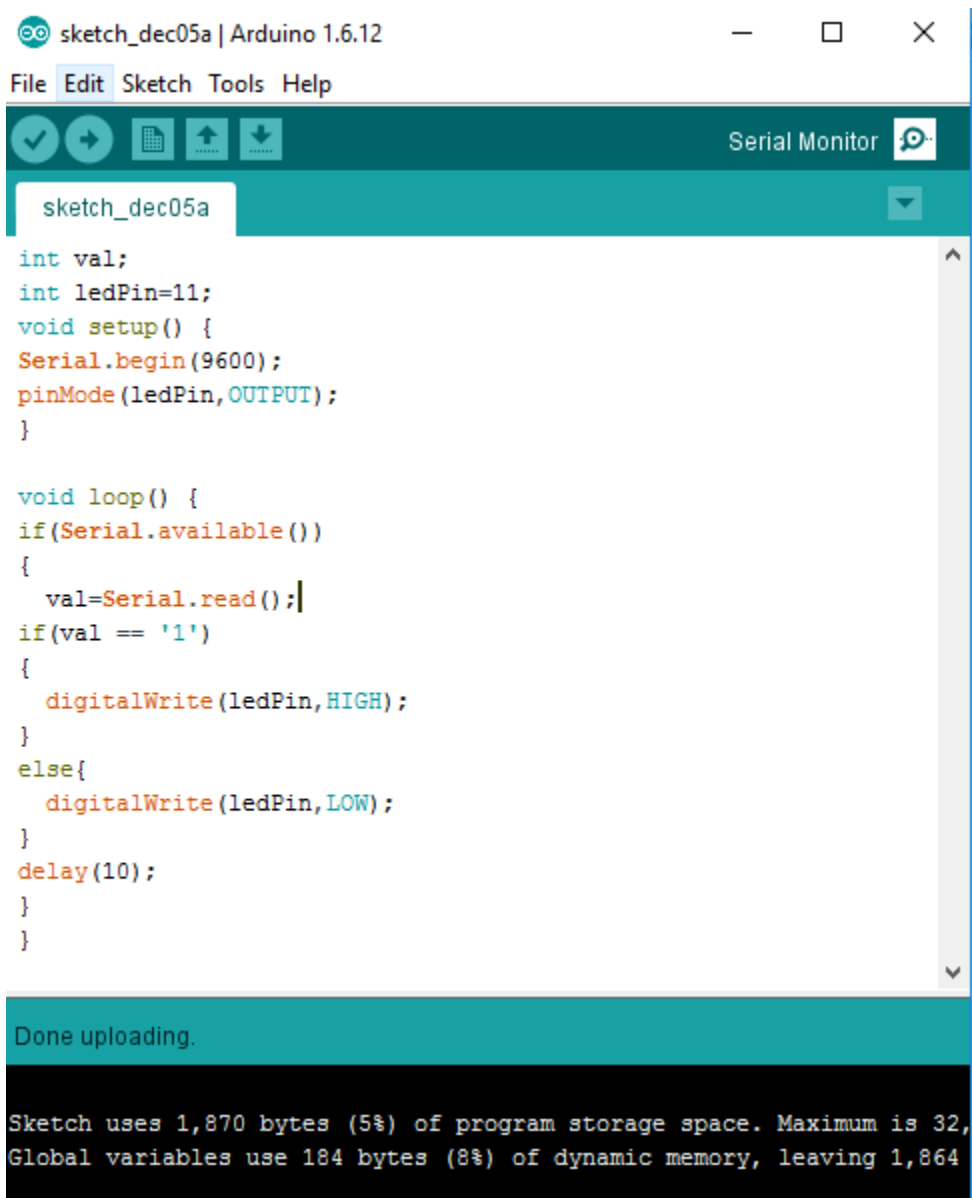
```
sketch_161205b | Processing 3.2.3
File Edit Sketch Debug Tools Help

sketch_161205b
1 import processing.serial.*;
2 Serial myPort;
3 void setup()
4 {
5   size(200,200);
6   String portName="COM4";
7   myPort=new Serial(this,portName,9600);
8 }
9 void draw()
10 {
11   if(mousePressed ==true)
12   {
13     myPort.write('1');
14   }else
15     myPort.write('0');
16 }
17
18
19
```

In our draw() loop, we send whatever we want over the serial port by using the write method from the Processing Serial library. For this sketch, we will send a '1' whenever we click our mouse in the Processing window. We'll also print it out on the console, just to see that we're actually sending something. If we aren't clicking we'll send a '0' instead.

If you run this code, you should see a bunch of 1's appear in the console area whenever you click your mouse in the window.

2- Open Arduino software and type the following:



```
sketch_dec05a | Arduino 1.6.12
File Edit Sketch Tools Help
Serial Monitor
sketch_dec05a
int val;
int ledPin=11;
void setup() {
  Serial.begin(9600);
  pinMode(ledPin,OUTPUT);
}

void loop() {
  if(Serial.available())
  {
    val=Serial.read();
    if(val == '1')
    {
      digitalWrite(ledPin,HIGH);
    }
    else{
      digitalWrite(ledPin,LOW);
    }
    delay(10);
  }
}

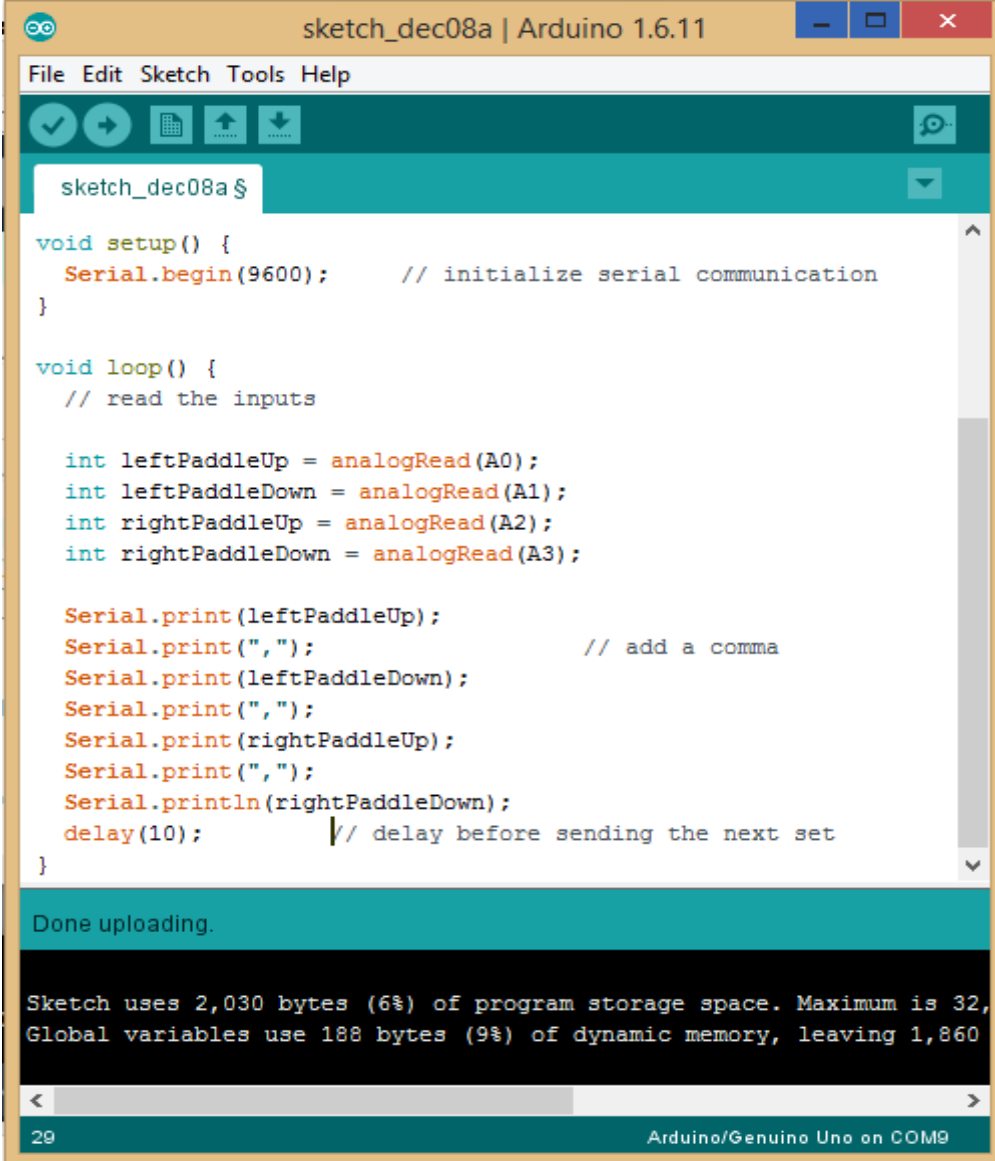
Done uploading.

Sketch uses 1,870 bytes (5%) of program storage space. Maximum is 32,
Global variables use 184 bytes (8%) of dynamic memory, leaving 1,864
```

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 11 of your Arduino, simply by clicking within the Processing canvas.

The following section show how to run **Pong** game using Processing and Arduino.

- 1- Open Arduino software and type the following:



```
sketch_dec08a | Arduino 1.6.11
File Edit Sketch Tools Help
sketch_dec08a $
void setup() {
  Serial.begin(9600);    // initialize serial communication
}

void loop() {
  // read the inputs

  int leftPaddleUp = analogRead(A0);
  int leftPaddleDown = analogRead(A1);
  int rightPaddleUp = analogRead(A2);
  int rightPaddleDown = analogRead(A3);

  Serial.print(leftPaddleUp);
  Serial.print(",");      // add a comma
  Serial.print(leftPaddleDown);
  Serial.print(",");
  Serial.print(rightPaddleUp);
  Serial.print(",");
  Serial.println(rightPaddleDown);
  delay(10);             // delay before sending the next set
}

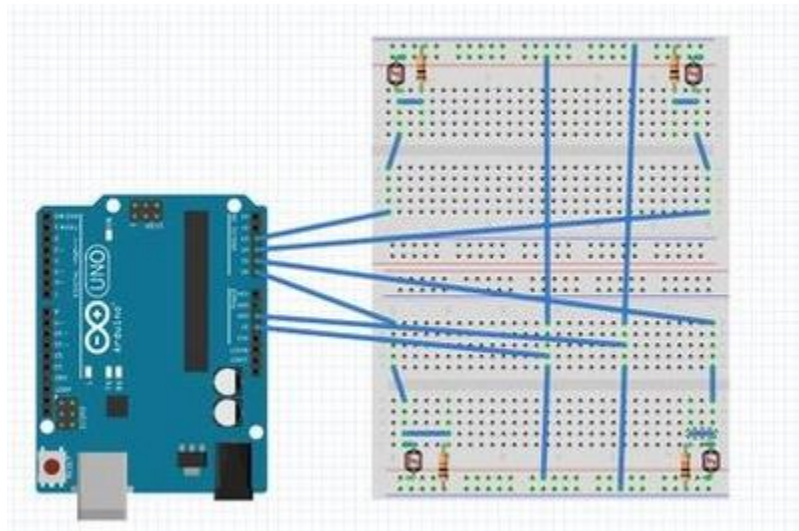
Done uploading.

Sketch uses 2,030 bytes (6%) of program storage space. Maximum is 32,
Global variables use 188 bytes (9%) of dynamic memory, leaving 1,860
29 Arduino/Genuino Uno on COM9
```

- 2- Open Processing software and type the content of file **Player2.txt**.

3- Connect the following circuit.

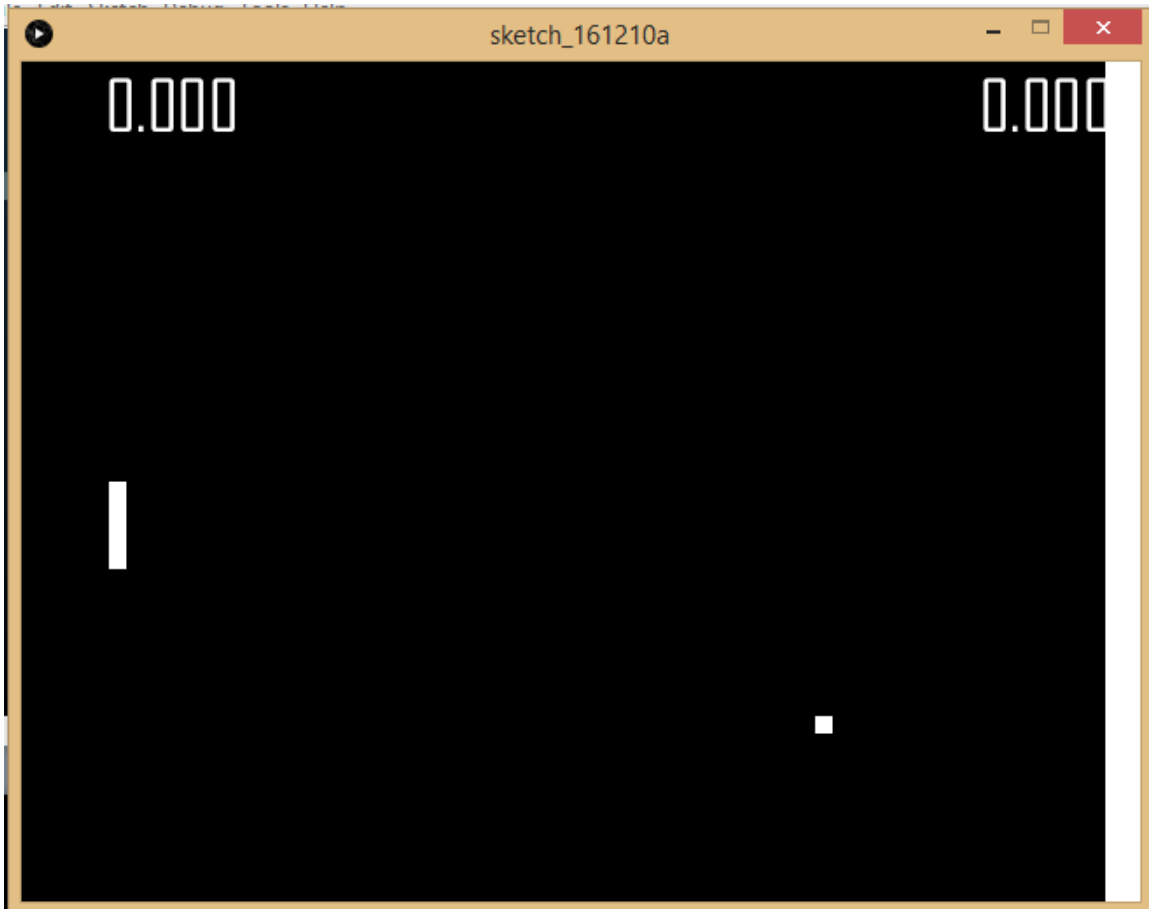
for our Game we have four inputs (two photocells per player).The Controller code works both with the one player mode and two player mode of the pong game, and it is fairly straightforward. The code reads the values which are output by the photocells and sends each value by serial communication to the processing engine.



4-Run Arduino and Processing softwares.the follwing screen should be appear.



You can play with one player by open Processing and type Player1.txt file instead of Player2.txt file. The following screen should be appear.



ToDo

This part will be given to you by the teacher assistant in the lab time.