Processing (programming language)

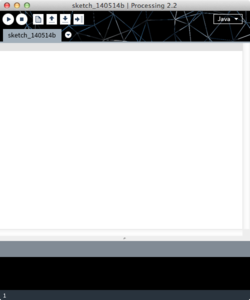
Processing is an open source computer programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks.

The project was initiated in 2001 by Casey Reas and Benjamin Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab. In 2012, they started the Processing Foundation along with Daniel Shiffman, who joined as a third project lead. Johanna Hedva joined the Foundation in 2014 as Director of Advocacy.[1]

One of the aims of Processing is to allow non-programmers to start computer programming aided by visual feedback. The Processing language builds on the Java language, but uses a simplified syntax and a graphics user interface.

**Features**

Processing includes a *sketchbook*, a minimal alternative to an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for organizing projects.

[](https://en.wikipedia.org/wiki/File:Processing_2.2_Mac_OS_X_Screenshot.png)

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

» 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, Windows, Android, and ARM

» Over 100 libraries extend the core software

» Well documented, with many books available

Every Processing sketch is actually a subclass of the [PApplet](https://github.com/processing/processing/blob/master/core/src/processing/core/PApplet.java) [Java class](https://en.wikipedia.org/wiki/Java_class) which implements most of the Processing language's features.

When programming in Processing, all additional classes defined will be treated as [inner classes](https://en.wikipedia.org/wiki/Inner_class) when the code is translated into pure Java before compiling. This means that the use of [static variables](https://en.wikipedia.org/wiki/Static_variables) and [methods](https://en.wikipedia.org/wiki/Static_methods) in classes is prohibited unless you explicitly tell Processing that you want to code in pure Java mode.

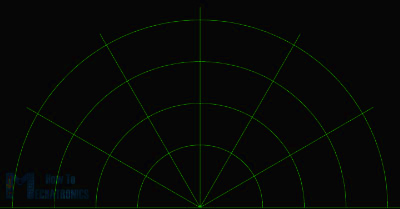
Processing also allows for users to create their own classes within the PApplet sketch. This allows for complex [data types](https://en.wikipedia.org/wiki/Data_types) that can include any number of arguments and avoids the limitations of solely using standard data types such as: [int](https://en.wikipedia.org/wiki/Integer_(computer_science)) (integer), [char](https://en.wikipedia.org/wiki/Character_(computing)) (character), [float](https://en.wikipedia.org/wiki/Floating_point) (real number), and color (RGB, [ARGB](https://en.wikipedia.org/wiki/RGBA_color_space), hex).

**Radar plot using Processing**

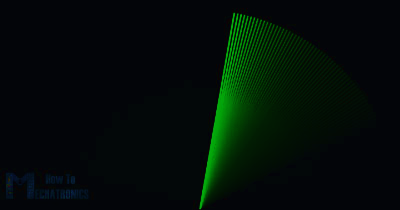
Now we need to make a code and upload it to the Arduino Board that will enable the interaction between the Arduino and the Processing IDE.

Write and Upload the program.

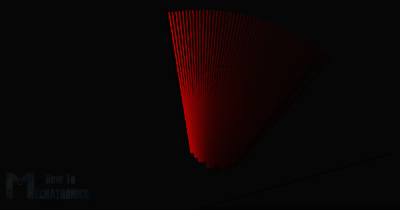
Now we will receive the values for the angle and the distance measured by the sensor from the Arduino Board into the Processing IDE using the SerialEvent() function which reads the data from the Serial Port and we will put the values of the angle and the distance into the variables iAngle and iDistance. These variable will be used for drawing the radar, the lines, the detected objects and some of the text.



For drawing the line that is moving along the radar I made this function drawLine(). Its center of rotation is set with the translate() function and using the line() function in which the iAngle variable is used the line is redrawn for each degree.



For drawing the detected objects I made this drawObject() function. It gets the distance from ultrasonic sensor, transforms it into pixels and in combination with the angle of the sensor draws the object on the radar.



Here’s the final appearance of the radar:

