ELECTRONICS INSTRUMENTATION AND MEASUREMENT HARDWARE PROJECT

(SUBJECT CODE: EE 251)

TITLE: RADAR

SUBMITTED TO:	SUBMITTED BY:
DR. MALTI BANSAL	PARANGAT MITTAL
DEPT. OF ELECTRONICS	2K17/EC/121
& COMMUNICATION ENGINEERING	ECE-III-D3

INTRODUCTION

RADAR stands for **Radio Detection And Ranging System**. It is basically an electromagnetic system used to detect the location and distance of an object from the point where the RADAR is placed. It works by radiating energy into space and monitoring the echo or reflected signal from the objects. It operates in the UHF and microwave range.

A Basic Idea of RADAR-

The RADAR system generally consists of a transmitter which produces an electromagnetic signal which is radiated into space by an antenna. When this signal strikes any object, it gets reflected or reradiated in many directions. This reflected or echo signal is received by the radar antenna which delivers it to the receiver, where it is processed to determine the geographical statistics of the object. The range is determined by the calculating the time taken by the signal to travel from the RADAR to the target and back. The target's location is measured in angle, from the direction of maximum amplitude echo signal, the antenna points to. To measure range and location of moving objects, Doppler Effect is used.

EQUIPMENTS REQUIRED

Hardware Components:

1- Arduino Uno
2- Ultrasonic Sensor HCSR04
3- Servo Motor SG-90
4- Jumper Wires
5- USB Cables
6- Breadboard

Software Components:

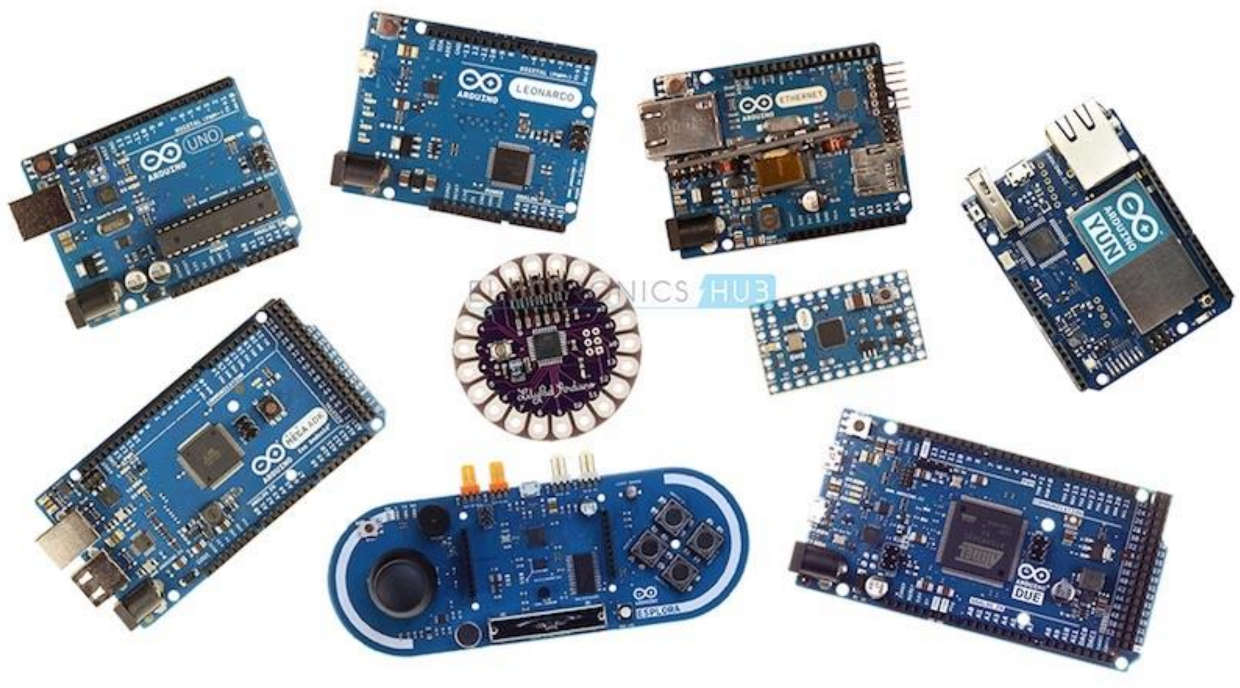
1- Arduino IDE
2- Processing IDE

ARDUINO UNO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide



ULTRASONIC SENSOR

DESCRIPTION-

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package.

From 2cm to 400 cm or 1" to 13 feet. Its operation is not affected by sunlight or black material like sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

FEATURES-

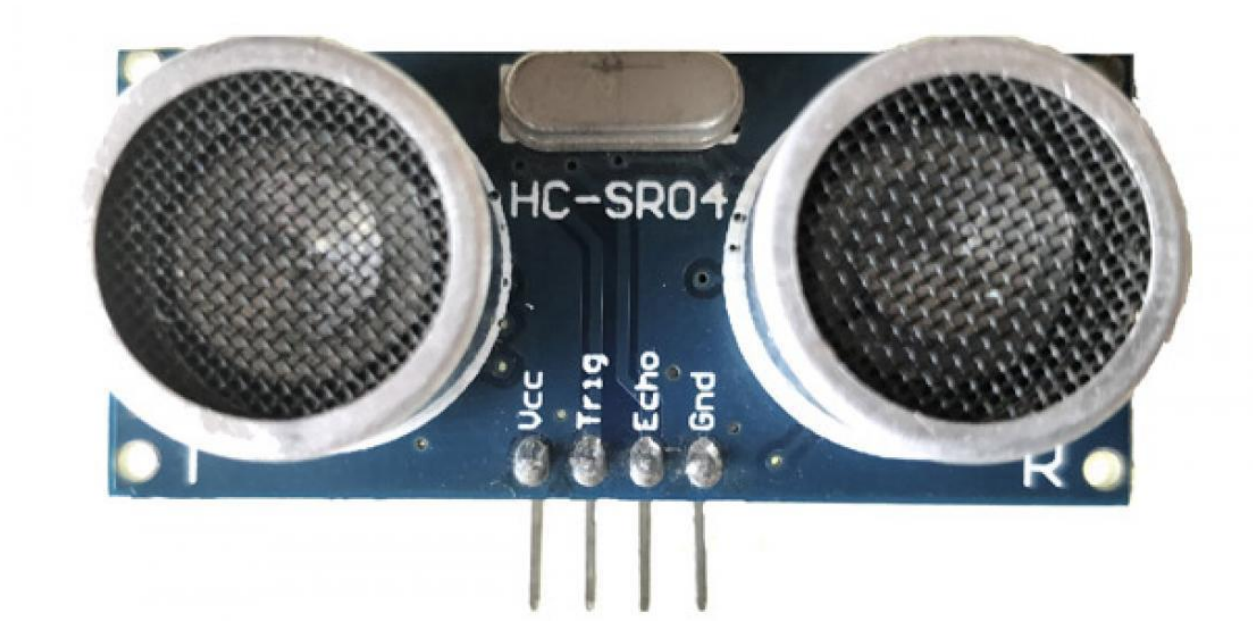
- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1″ – 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

WORKING-

The ultrasonic sensor uses sonar to determine the distance to an object. Here’s what happens:

1. The transmitter (trig pin) sends a signal: a high-frequency sound.
2. When the signal finds an object, it is reflected and…
3. … the transmitter (echo pin) receives it.

The time between the transmission and reception of the signal allows us to know the distance to an object. This is possible because we know the sound’s velocity in the air.



SERVO

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

Servomotors are generally used as a high-performance alternative to the stepper motor. Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate, but for this the controller needs to 'know' the position of the stepper motor on power up. Therefore, on first power up, the controller will have to activate the stepper motor and turn it to a known position, e.g. until it activates an end limit switch. This can be observed when switching on an inkjet printer; the controller will move the ink jet carrier to the extreme left and right to establish the end positions. A servomotor will immediately turn to whatever angle the controller instructs it to, regardless of the initial position at power up.

JUMPER WIRES

A **jumper wire** (also known as jumper wire, or jumper) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.
Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

TYPES-

Jump wires at the end of a multi-coloured ribbon cable are used to connect the pin header at the left side of a blue USB2Serial board to a white breadboard below. Another jumper cable ending in a USB micro male connector mates to the right side of the USB2Serial board. Red and black tinned jump wires can be seen on the breadboard.

There are different types of jumper wires. Some have the same type of electrical connector at both ends, while others have different connectors. Some common connectors are:

- Solid tips – are used to connect on/with a breadboard or female header connector. The arrangement of the elements and ease of insertion on a breadboard allows increasing the mounting density of both components and jump wires without fear of short-circuits. The jump wires vary in size and colour to distinguish the different working signals.
- Crocodile clips – are used, among other applications, to temporarily bridge sensors, buttons and other elements of prototypes with components or equipment that have arbitrary connectors, wires, screw terminals, etc.
- Banana connectors – are commonly used on test equipment for DC and low-frequency AC signals.

- Registered jack (RJnn) – are commonly used in telephone (RJ11) and computer networking (RJ45).
- RCA connectors – are often used for audio, low-resolution composite video signals, or other low-frequency applications requiring a shielded cable.
- RF connectors – are used to carry radio frequency signals between circuits, test equipment, and antennas.

USB CABLES

USB (abbreviation of **Universal Serial Bus**) is an industry standard that establishes specifications for cables, connectors and protocolsfor connection, communication and power supply between personal computers and their peripheral devices.
There have been three generations of USB specifications:

- USB 1.*x*
- USB 2.0, with multiple updates and additions
- USB 3.*x*

USB was designed to standardize the connection of peripherals like keyboards, pointing devices, digital still and video cameras, printers, portable media players, disk drives and network adapters to personal computers, both to communicate and to supply electric power. It has largely replaced interfaces such as serial ports and parallel ports, and has become commonplace on a wide range of devices.

USB connectors have been increasingly replacing other types for battery chargers of portable devices.

BREADBOARD

A **breadboard** is a construction base is a construction base for prototyping of electronics. Originally it was literally a bread board, a polished piece of wood used for slicing bread. In the 1970s the **solderless breadboard** (a.k.a. **plugboard**, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.

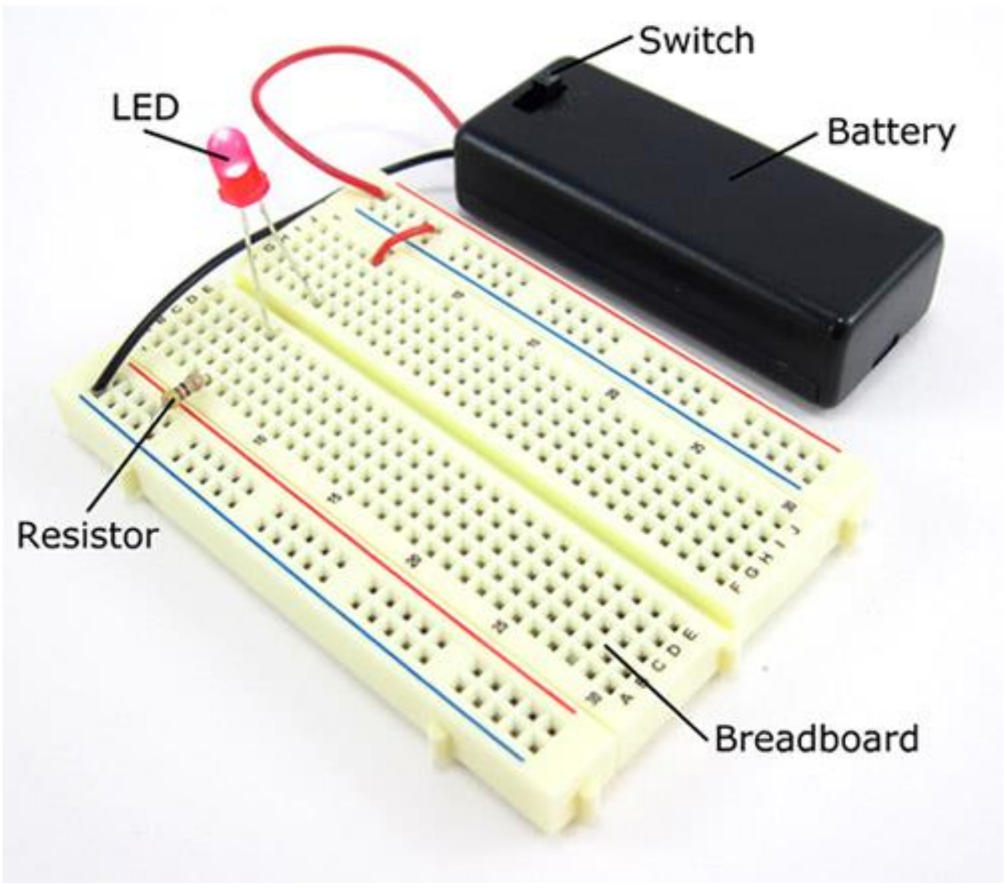
Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason, solderless breadboards are also popular with students and in technological education. Older breadboard types did not have this property. A stripboard (Veroboard) and similar prototyping printed circuit boards, which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

In the early days of radio, amateurs nailed bare copper wires or terminal strips to a wooden board (often literally a board to slice bread on) and soldered electronic components to them.[1] Sometimes a paper schematic diagram was first glued to the board as a guide to placing terminals, then components and wires were installed over their symbols on the schematic. Using thumbtacks or small nails as mounting posts was also common.

Breadboards have evolved over time, with the term now being used for all kinds of prototype electronic devices. For example, US Patent 3,145,483 was filed in 1961 and describes a wooden plate breadboard with mounted springs and other facilities. US Patent 3,496,419,was filed in 1967 and refers to a particular printed circuit board layout as a *Printed Circuit*

Breadboard. Both examples refer to and describe other types of breadboards as prior art.

The breadboard most commonly used today is usually made of white plastic and is a pluggable (solderless) breadboard. It was designed by Ronald J. Portugal in 1971.



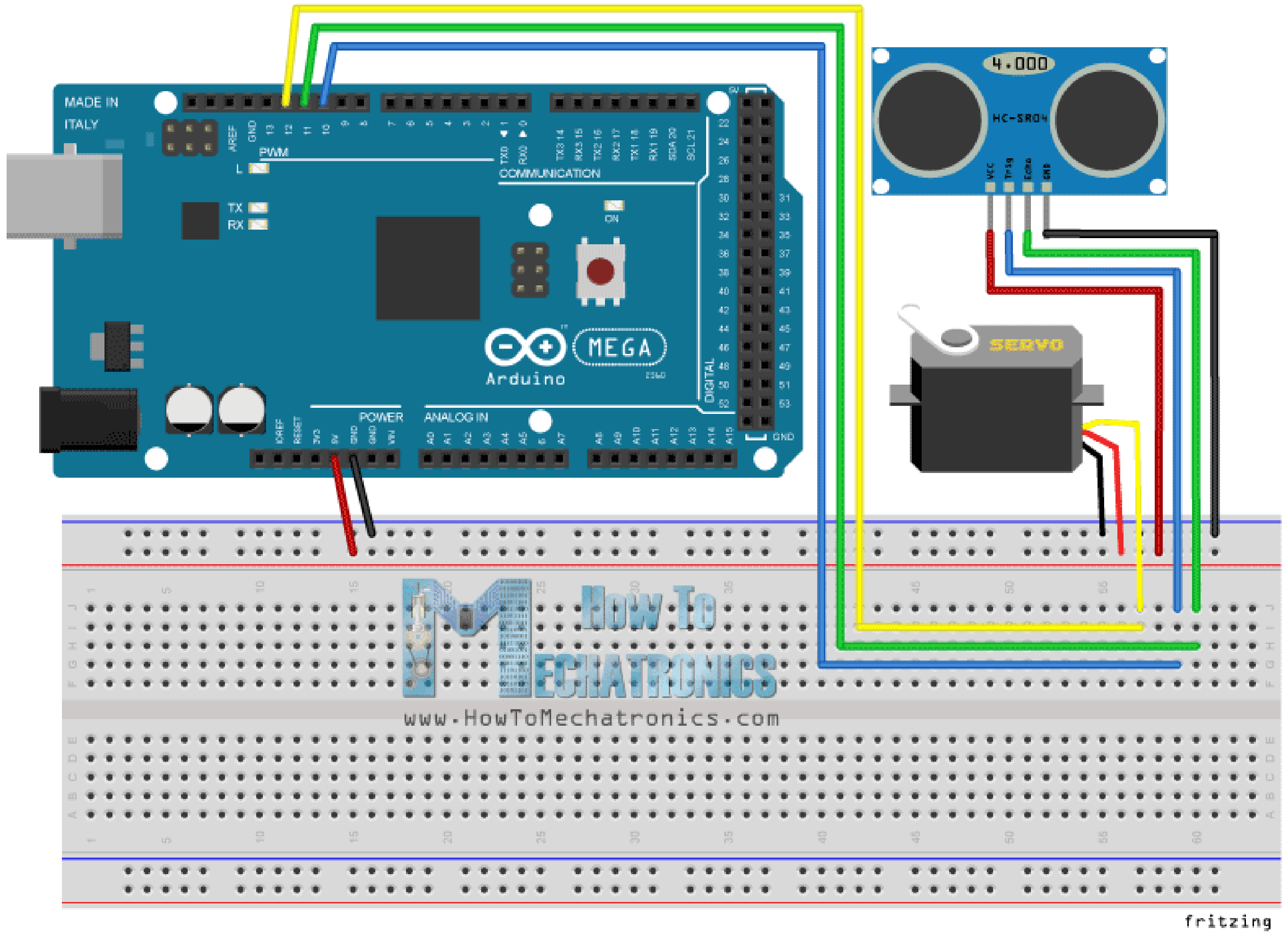
PROCESSING IDE

Processing is an open-source graphical library and integrated development environment (IDE) / playground built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

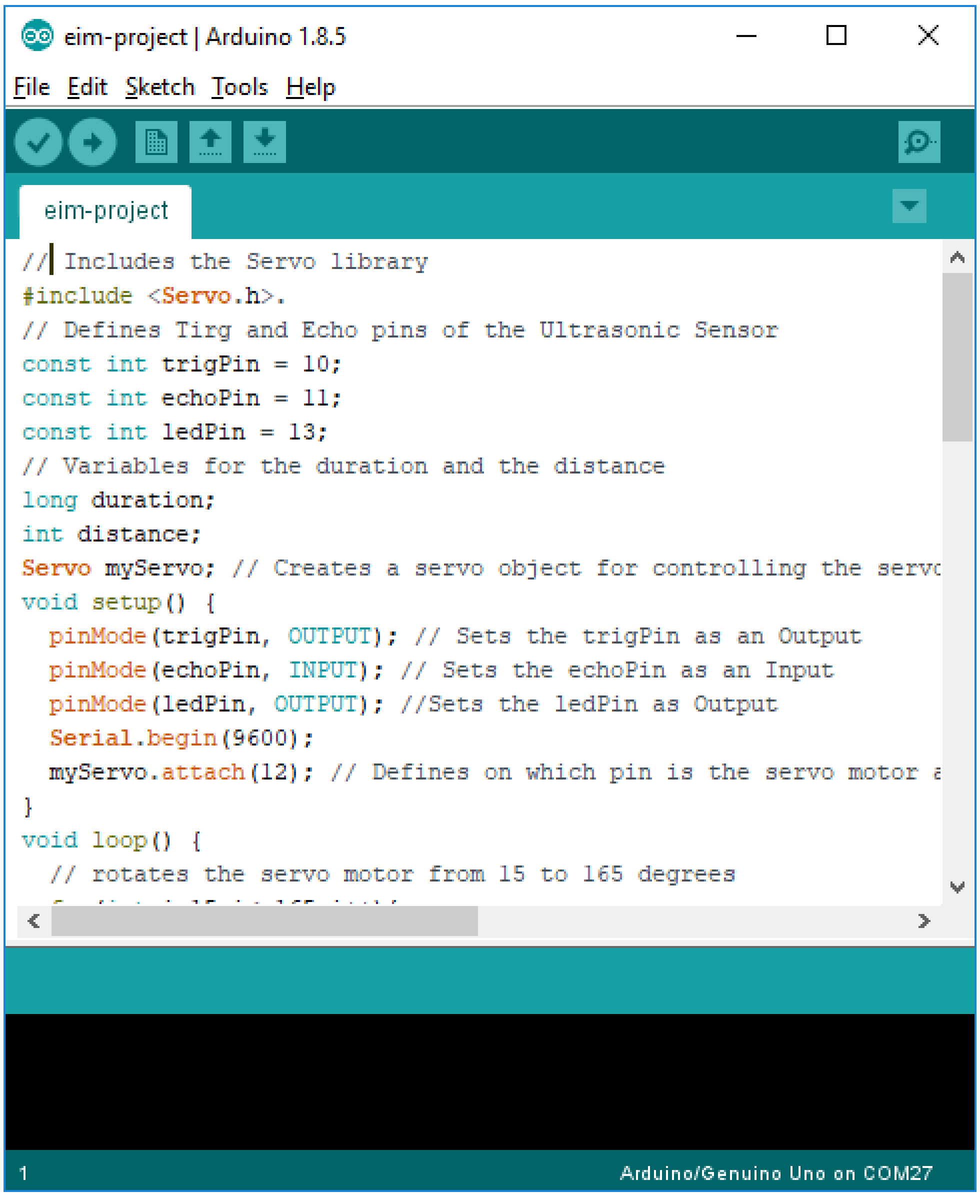
Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations. As well as this, it also has a graphical user interface for simplyfing the compilation and execution stage.

The Processing language and IDE were the precursor to numerous other projects, notably Arduino, Wiring and P5.js.

CIRCUIT DIAGRAM



ARDUINO CODE



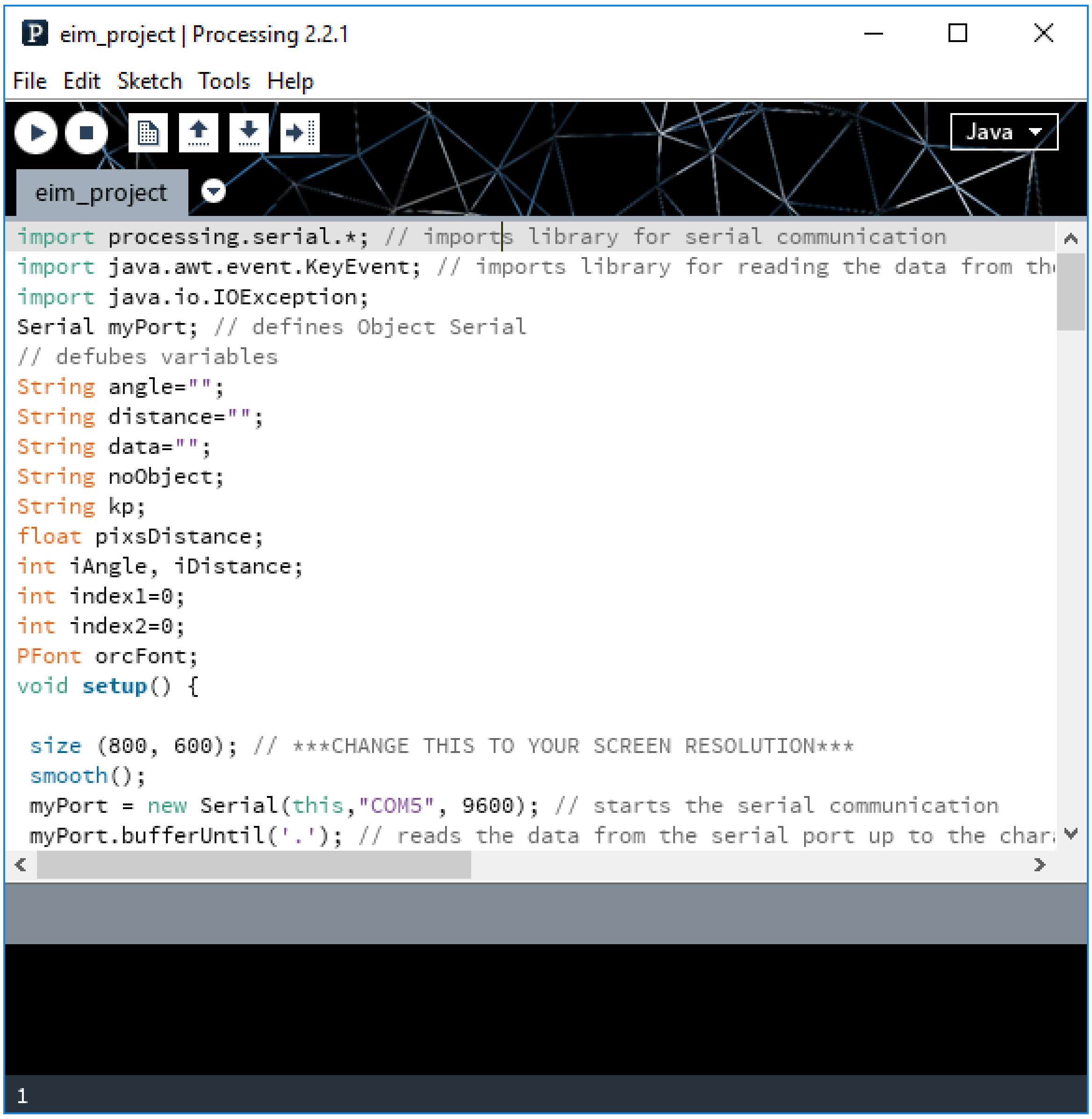
eim-project | Arduino 1.8.5
File Edit Sketch Tools Help
eim-project
1
Arduino/Genuino Uno on COM27

```
// Includes the Servo library
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
const int ledPin = 13;
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(ledPin, OUTPUT); //Sets the ledPin as Output
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor a
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
```

```
// Includes the Servo library
#include <Servo.h>
// Defines Trig and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
const int ledPin = 13;
// Variables for the duration and the distance
long duration;
int distance;
```

```
Servo myServo; // Creates a servo object for controlling
the servo motor

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an
Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(ledPin, OUTPUT); //Sets the ledPin as Output
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo
motor attached
}

void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Calls a function for
calculating the distance measured by the Ultrasonic sensor
for each degree

  Serial.print(i); // Sends the current degree into the
Serial Port
  Serial.print(","); // Sends addition character right
next to the previous value needed later in the Processing
IDE for indexing
  Serial.print(distance); // Sends the distance value into
the Serial Port
  Serial.print("."); // Sends addition character right
next to the previous value needed later in the Processing
IDE for indexing

  if(distance<40){
    digitalWrite(ledPin, HIGH);
    }
  else{
    digitalWrite(ledPin, LOW);
  }

  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");

  if(distance<40){
    digitalWrite(ledPin, HIGH);
    }
  else{
    digitalWrite(ledPin, LOW);
  }  }    }

// Function for calculating the distance measured by the
Ultrasonic sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin,
returns the sound wave travel time in microseconds
  distance= duration*0.034/2;
  return distance;
}
```

PROCESSING CODE



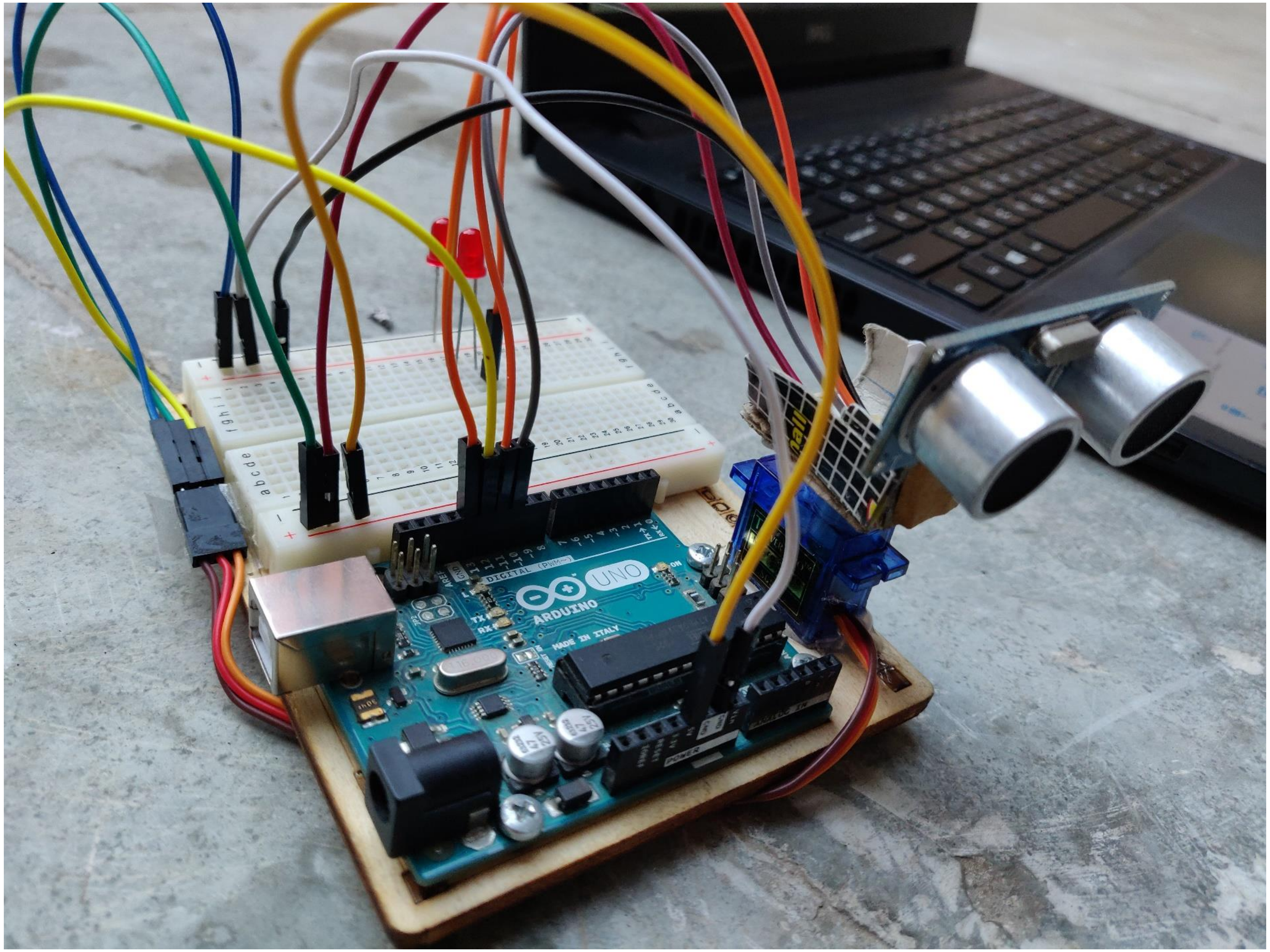
```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from th
import java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
String kp;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (800, 600); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM5", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the char
```

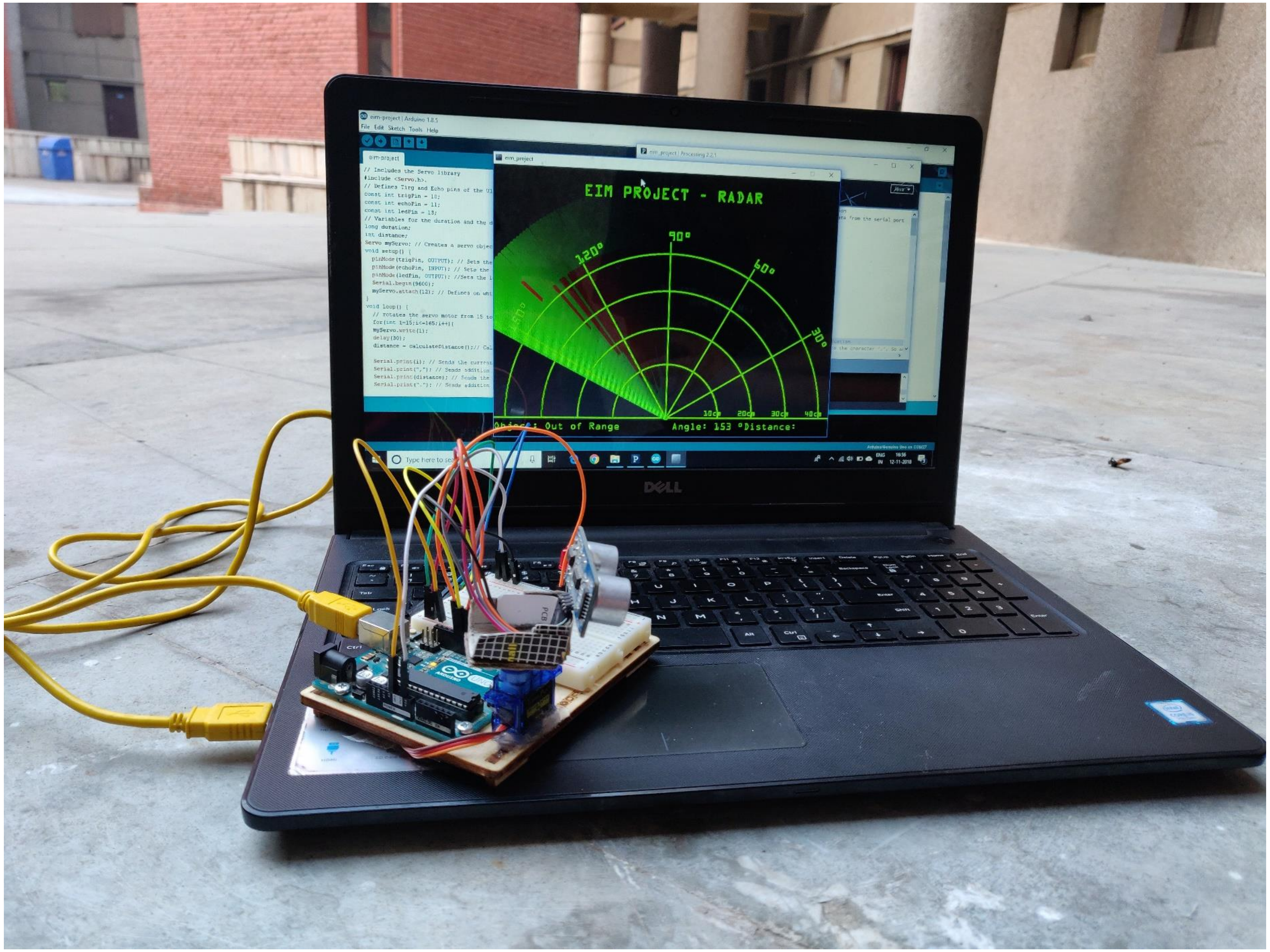
```
1
```

```
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;

Serial myPort; // defines Object Serial


// defubes variables
```

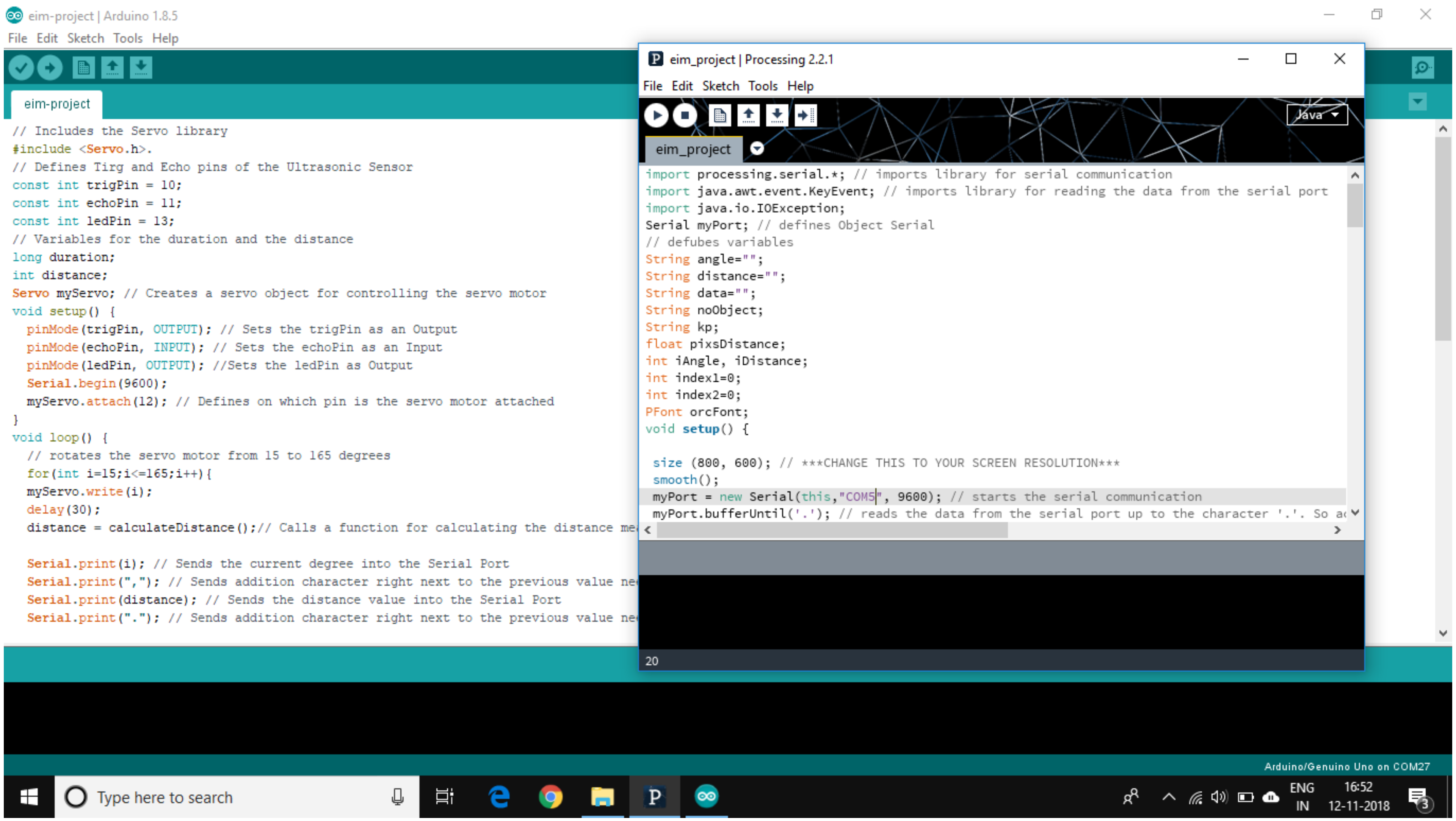
```
String angle="";
String distance="";
String data="";
String noObject;
String kp;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (800, 600);
 smooth();
 myPort = new Serial(this,"COM5", 9600); // starts the
serial communication
 myPort.bufferUntil('.'); // reads the data from the
serial port up to the character '.'. So actually it reads
this: angle,distance.
 orcFont = loadFont("OCRAExtended-30.vlw");
}

void draw() {

  fill(98,245,31);
  textFont(orcFont);
  // simulating motion blur and slow fade of the moving
line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent (Serial myPort) { // starts reading data
from the Serial Port
  // reads the data from the Serial Port up to the
character '.' and puts it into the String variable "data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ','
and puts it into the variable "index1"
  angle= data.substring(0, index1); // read the data from
position "0" to position of the variable index1 or thats
the value of the angle the Arduino Board sent into the
Serial Port
  distance= data.substring(index1+1, data.length()); //
read the data from position "index1" to the end of the
data pr thats the value of the distance

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the
starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-
width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-
width*0.27),PI,TWO_PI);
```

```
  arc(0,0,(width-width*0.479),(width-
width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-
width*0.687),PI,TWO_PI);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-
width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-
width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-
width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-
width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-
width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the
starting coordinats to new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025);
// covers the distance from the sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the
distance
  line(pixsDistance*cos(radians(iAngle)),-
pixsDistance*sin(radians(iAngle)),(width-
width*0.505)*cos(radians(iAngle)),-(width-
width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the
starting coordinats to new location
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-
(height-height*0.12)*sin(radians(iAngle))); // draws the
line according to the angle
  popMatrix();
}

void drawText() { // draws the texts on the screen

  pushMatrix();
  kp="EIM PROJECT";
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(15);

  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(20);
  text("Object: " + noObject, width-width*1, height-
height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-
height*0.0277);
  text("Distance: ", width-width*0.26, height-
height*0.0277);
  textSize(30);
  text("EIM PROJECT - RADAR",width-width*0.75, height-
height*0.9);
  if(iDistance<40) {
  text("        " + iDistance +" cm", width-width*0.225,
height-height*0.0277);
  }
  textSize(25);
  fill(98,245,60);
  translate((width-
width*0.4994)+width/2*cos(radians(30)),(height-
height*0.0907)-width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-
width*0.503)+width/2*cos(radians(60)),(height-
height*0.0888)-width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-
width*0.507)+width/2*cos(radians(90)),(height-
height*0.0833)-width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-
width*0.513+width/2*cos(radians(120)),(height-
height*0.07129)-width/2*sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();
  translate((width-
width*0.5104)+width/2*cos(radians(150)),(height-
height*0.0574)-width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```


UNO
ARDUINO
DIGITAL (PWM~)
MADE IN ITALY
POWER
ANALOG IN



```
// Includes the Servo library
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ul
const int trigPin = 10;
const int echoPin = 11;
const int ledPin = 13;
// Variables for the duration and the d
long duration;
int distance;
Servo myServo; // Creates a servo objec
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the
  pinMode(echoPin, INPUT); // Sets the
  pinMode(ledPin, OUTPUT); //Sets the l
  Serial.begin(9600);
  myServo.attach(12); // Defines on whi
}
void loop() {
  // rotates the servo motor from 15 to
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Cal

  Serial.print(i); // Sends the current
  Serial.print(","); // Sends addition
  Serial.print(distance); // Sends the
  Serial.print("."); // Sends addition
```



eim-project | Arduino 1.8.5

```
// Includes the Servo library
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
const int ledPin = 13;
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(ledPin, OUTPUT); //Sets the ledPin as Output
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Calls a function for calculating the distance me

  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous value ne
  Serial.print(distance); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous value ne
```

Arduino/Genuino Uno on COM27

eim_project | Processing 2.2.1

```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
String kp;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (800, 600); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM5", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So ac
```

20