Simple Arduino radar

Components required:

1. Arduino board

2. Servo motor

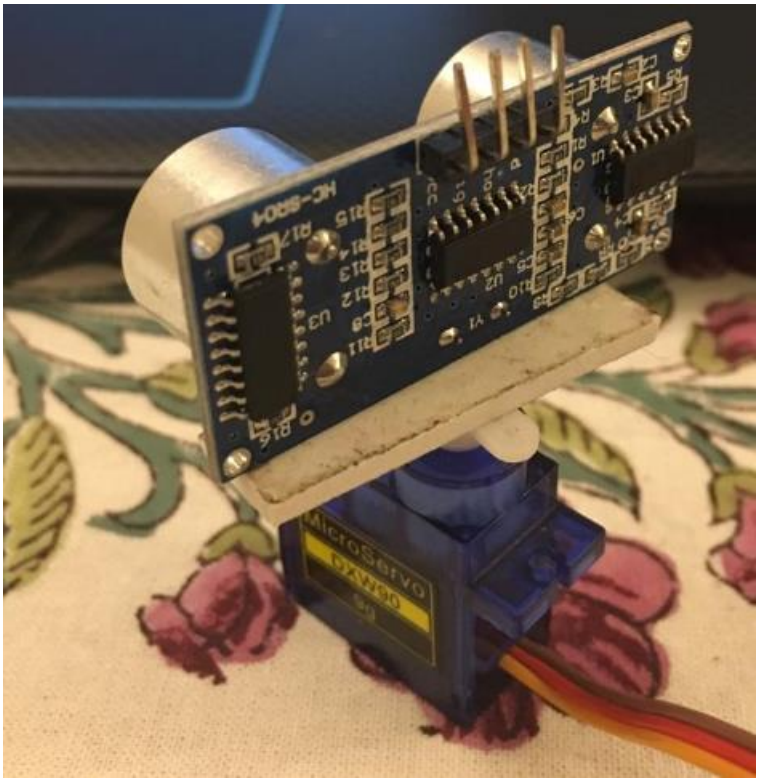
3. Bread board

4. HC-SR04 ultrasonic sensor

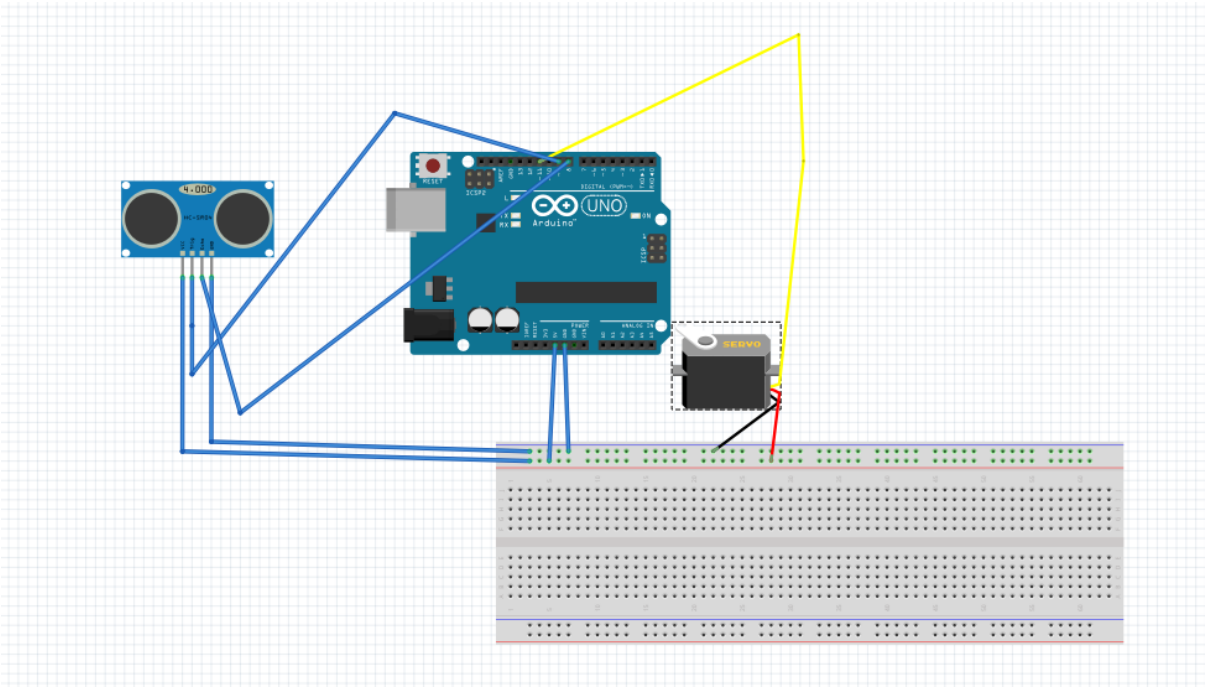
5. Jumper wires (male to female, male to male, female to female)
6. Arduino ide
7. Double sided tape

Hardware and connections

1. Use the double sided tape to connect the ultrasonic sensor and the servo as shown.



2. Make the connections as shown



Arduino code

```
//Declare variables before void setup
#include <Servo.h>//includes header file to communicate with servo
const int trig = 9;// sets the trigger pin of ultrasonic sensor to 9 of arduino
const int echo  = 8; // sets the echo pin of ultrasonic sensor to 8 of arduino
long dur;//duration variable is of the time measured by the sensor. Long is a data
type that stores 32 bits of data
int dist;
Servo srv;// creates servo object srv
void setup() {//sets the pins as input and output and basic setup
  pinMode(trig, OUTPUT); //sets trig as an output pin
  pinMode(echo, INPUT); //sets echo  as an input pin

  Serial.begin(9600);//serial communication baud rate set to 9600
  srv.attach(11); // Defines on which pin is the servo motor attached
// Baud rate refers to the number of signal or symbol changes that occur per second.
}
void loop() {//part of the program that repeats
  for(int i=15;i<=165;i++){
  srv.write(i);//writes the value of angle to the servo
  delay(30);//delays the execution of the next line of code by 30 miliseconds
  dist = caldist();//caldistance is the function used to calculate the distance using the
sensor

  Serial.print(i);
  Serial.print(",");
  Serial.print(dist);
```

```
  Serial.print(".");
  }
  for(int i=165;i>15;i--){
  srv.write(i);
  delay(3);
  dist = caldist();
  Serial.print(i);
  Serial.print(",");
  Serial.print(dist);
  Serial.print(".");
  }
}
int caldist(){

  digitalWrite(trig, LOW); //writes a low to clear out the transmitter of the sensor
  delayMicroseconds(2);
  digitalWrite(trig, HIGH); // writes a high to send a pulse out of transmitter of the
sensor
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  dur = pulseIn(echo, HIGH); //pulsein gives the value of microseconds it takes for the
pulse to hit an obstacle and return
  dist= dur*0.0340/2;//time taken is 2t. to calculate time we divide by 2 and calculate
using the formula distance=speed*time. Distance is given in cm.
  return dist;
}
```

Processing code

```
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 prt; // creates serial object prt
String angle="";
String distance="";
String data="";
float pdistance;
int intangle, intdistance;
int index1=0;
int index2=0;
void setup() {

 size (1000,1000);
```

```
 background(0,0,0);
 prt = new Serial(this,"COM3", 9600); // starts the serial communication
 prt.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So basically it reads : angle, distance.
}
void draw() { // the part of the code that runs repeatedly
  radar(); //calls the function that draws the outline of the radar
  write();//calls the function that writes basic data on the outline of the radar
object();//calls the function that plots the object on the radar

}
void serialEvent (Serial prt)
  {
  data = prt.readStringUntil('.');
  data = data.substring(0,data.length()-1);
  index1 = data.indexOf(",");
  angle= data.substring(0, index1);
  distance= data.substring(index1+1, data.length());
  intangle = int(angle);
  intdistance = int(distance);
  }
void radar() {
  pushMatrix();// starts storing the changes only ahead of the command
  translate(500,900); // moves the starting coordinates to new location. The processing screen is like a graph with the origin on the top left corner. So basically the fourth quadrant. Translate shifts origin to 500px ahead and 900px down
  noFill();
  strokeWeight(5);//makes the lines thinner or thicker
  stroke(102, 255, 102);
  {
    // draws the arc lines
  arc(0,0,800,800,PI,TWO_PI);
  arc(0,0,600,600,PI,TWO_PI);
  arc(0,0,400,400,PI,TWO_PI);
  arc(0,0,200,200,PI,TWO_PI);
  // draws the angle lines. This is done using simple trigonometry
  line(-500,0,500,0);
  line(0,0,-500*cos(radians(30)),-500*sin(radians(30)));
  line(0,0,-500*cos(radians(60)),-500*sin(radians(60)));
  line(0,0,-500*cos(radians(90)),-500*sin(radians(90)));
  line(0,0,-500*cos(radians(120)),-500*sin(radians(120)));
  line(0,0,-500*cos(radians(150)),-500*sin(radians(150)));
  }
```

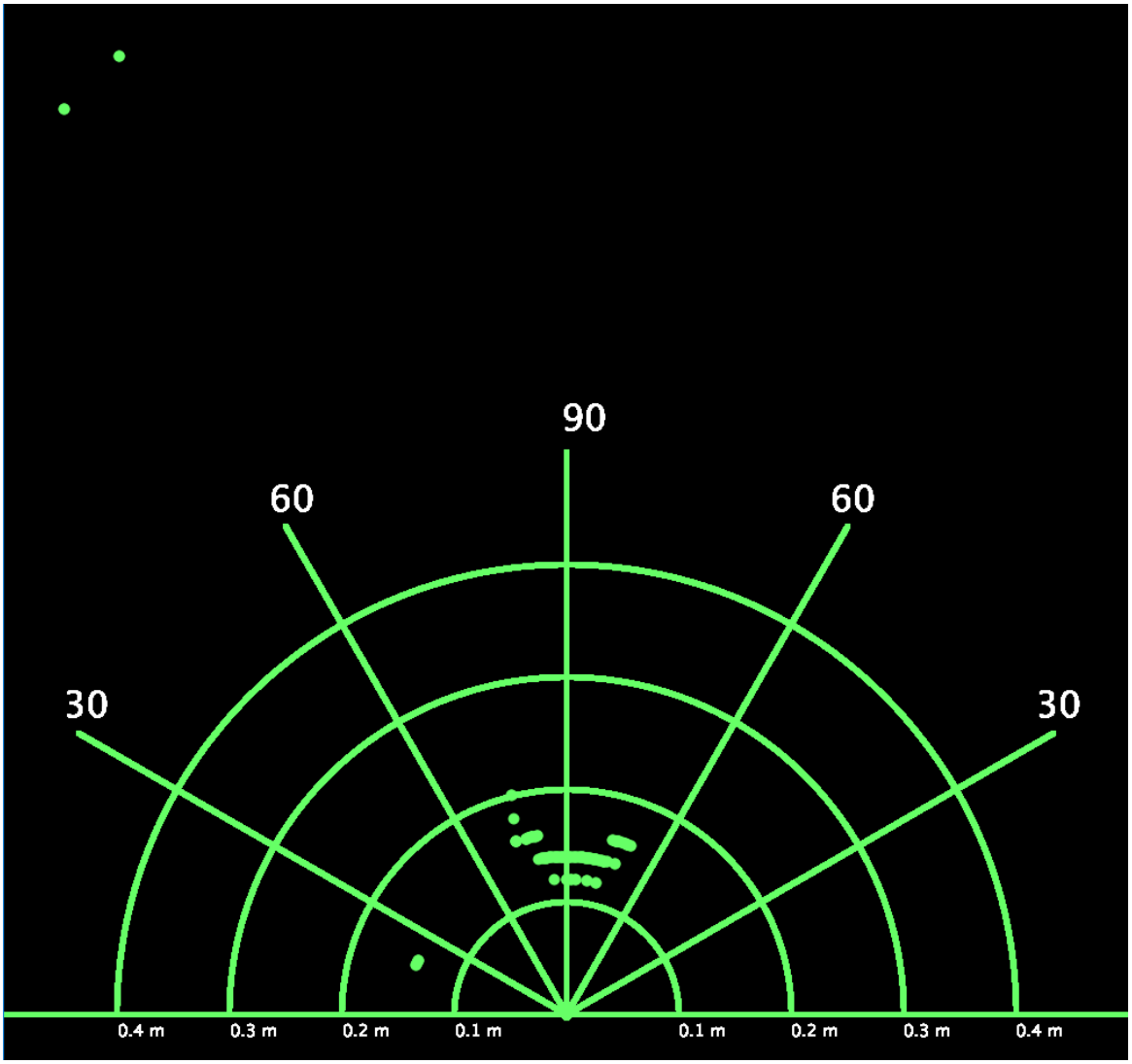
```
  popMatrix();// the changes between the pushmatrix and popmatrix are lost and
applied only to the code in between
}
void write()
{
  pushMatrix();
  translate(500,900);
  textSize(15);
  fill(255,255,255);
  {
  text("0.1 m",100,20);
  text("0.2 m",200,20);
  text("0.3 m",300,20);
  text("0.4 m",400,20);
  text("0.1 m",-100,20);
  text("0.2 m",-200,20);
  text("0.3 m",-300,20);
  text("0.4 m",-400,20);
  textSize(32);
  }
  fill(255,255,255);
  {
  text("30",-500*cos(radians(30))-15,-500*sin(radians(30))-15);
  text("60",-500*cos(radians(60))-15,-500*sin(radians(60))-15);
  text("90",-5,-520);
  text("60",-500*cos(radians(120))-15,-500*sin(radians(120))-15);
  text("30",-500*cos(radians(150))-15,-500*sin(radians(150))-15);
  popMatrix();
 }
}
void object()

 {
   pushMatrix();
  translate(500,900);
  pdistance=intdistance*20;//I'm going to explain the pixel conversion ahead
 {
   fill(204, 102, 255);
  strokeWeight(10);
   point(-pdistance*cos(radians(intangle)),-pdistance*sin(radians(intangle)));

 }
 popMatrix();
  }
```

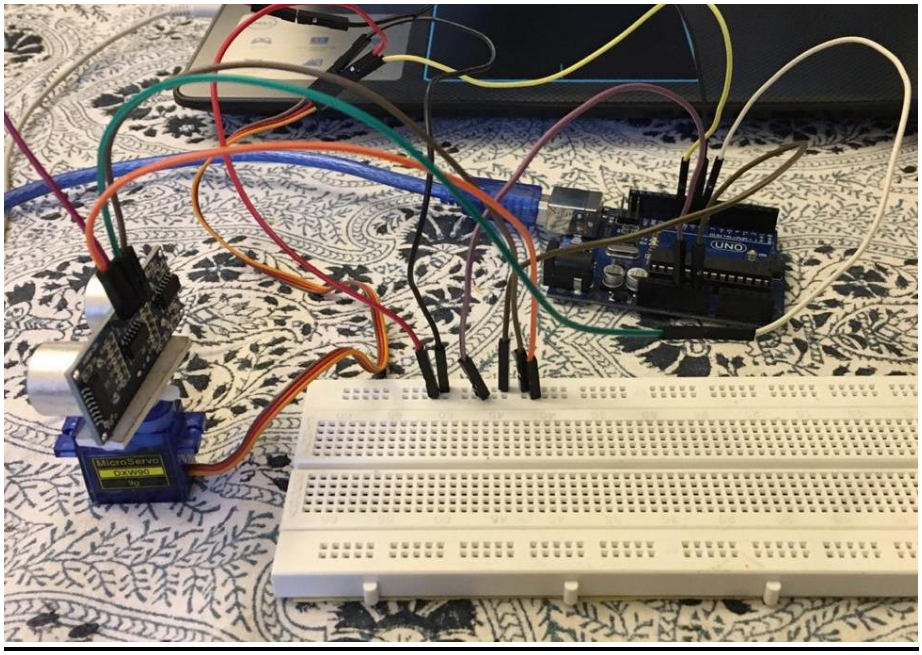
Pixel conversion

We have decided to give our radar a range of 40 cm. (range can go up to 400 cm). Because our outer circle is 800px in radius 800px corresponds to 40cm. and therefore 1 cm corresponds to 20 px. Hence the multiplication by 20.

Reference: https://howtomechatronics.com/projects/arduino-radar-project/

Implementation




MicroServo
UNO