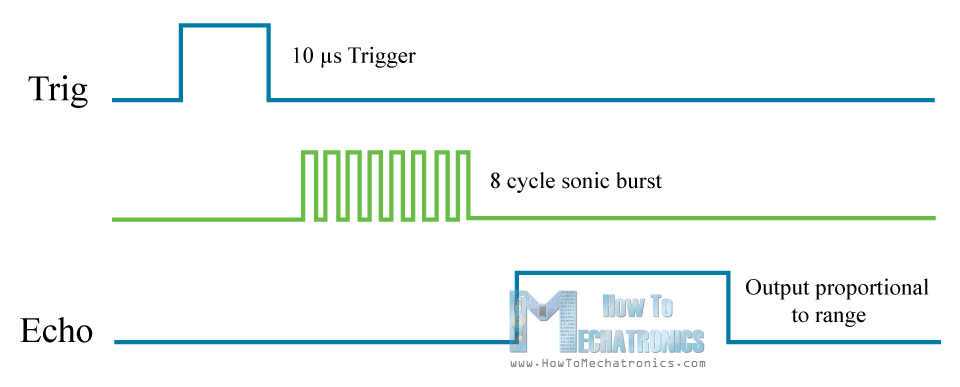
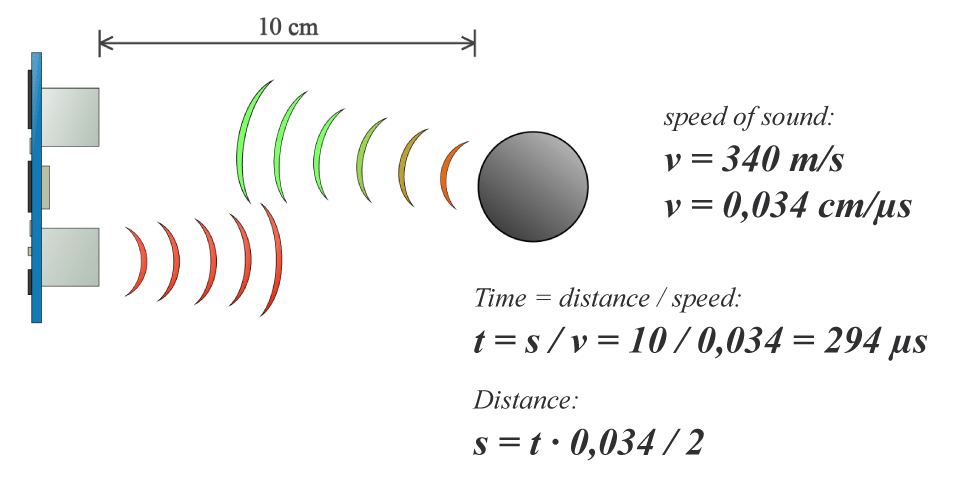
Introduction:

How It Works – Ultrasonic Sensor

It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance. In order to generate the ultrasound you need to set the Trig on a High State for 10 µs. That will send out an 8-cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.



For example, if the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 294 u seconds. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward.  So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.

****

First you have to define the Trig and Echo pins. In this case they are the pins number 9 and 10 on the Arduino Board and they are named trigPin and echoPin. Then you need a Long variable, named “duration” for the travel time that you will get from the sensor and an integer variable for the distance.

In the setup you have to define the trigPin as an output and the echoPin as an Input and also start the serial communication for showing the results on the serial monitor.

In the loop first, you have to make sure that the trigPin is clear so you have to set that pin on a LOW State for just 2 µs. Now for generating the Ultra sound wave we have to set the trigPin on HIGH State for 10 µs. Using the pulsIN() function you have to read the travel time and put that value into the variable “duration”. This function has 2 parameters, the first one is the name of the echo pin and for the second one you can write either HIGH or LOW. In this case, HIGH means that the pulsIN() function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance, we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously.  At the end we will print the value of the distance on the Serial Monitor.

Processing IDE:

We have used processing IDE for the display of our RADAR on PC as it shows the actual working of our project. In this IDE we have coded it in JAVA for the calculations of First Sweep, Second Sweep, Average, Motion.

Used processing and coded a code in JAVA for the visual display of our RADAR,

Code is below>

import processing.serial.\*;

Serial arduinoport;

float x, y;

int radius = 350;

int w = 300;

int degree = 0;

int value = 0;

int motion = 0;

int[] newValue = new int[181];

int[] oldValue = new int[181];

PFont myFont;

int radarDist = 0;

int firstRun = 0;

void setup(){

size(750, 450);

background (0);

myFont = createFont("verdana", 12);

textFont(myFont);

arduinoport = new Serial(this, Serial.list()[0], 9600);

}

void draw(){

fill(0);

noStroke();

ellipse(radius, radius, 750, 750);

rectMode(CENTER);

rect(350,402,800,100);

if (degree >= 179) {

motion = 1;

}

if (degree <= 1) {

motion = 0;

}

strokeWeight(7);

if (motion == 0) {

for (int i = 0; i <= 20; i++) {

stroke(0, (10\*i), 0);

line(radius, radius, radius + cos(radians(degree+(180+i)))\*w, radius + sin(radians(degree+(180+i)))\*w);

}

} else {

for (int i = 20; i >= 0; i--) {

stroke(0,200-(10\*i), 0);

line(radius, radius, radius + cos(radians(degree+(180+i)))\*w, radius + sin(radians(degree+(180+i)))\*w);

}

}

noStroke();

fill(0,50,0);

beginShape();

for (int i = 0; i < 180; i++) {

x = radius + cos(radians((180+i)))\*((oldValue[i]));

y = radius + sin(radians((180+i)))\*((oldValue[i]));

vertex(x, y);

}

endShape();

/\* second sweep \*/

fill(0,110,0);

beginShape();

for (int i = 0; i < 180; i++) {

x = radius + cos(radians((180+i)))\*(newValue[i]);

y = radius + sin(radians((180+i)))\*(newValue[i]);

vertex(x, y);

}

endShape();

fill(0,170,0);

beginShape();

for (int i = 0; i < 180; i++) {

x = radius + cos(radians((180+i)))\*((newValue[i]+oldValue[i])/2);

y = radius + sin(radians((180+i)))\*((newValue[i]+oldValue[i])/2);

vertex(x, y);

}

endShape();

if (firstRun >= 360) {

stroke(150,0,0);

strokeWeight(1);

noFill();

for (int i = 0; i < 180; i++) {

if (oldValue[i] - newValue[i] > 35 || newValue[i] - oldValue[i] > 35) {

x = radius + cos(radians((180+i)))\*(newValue[i]);

y = radius + sin(radians((180+i)))\*(newValue[i]);

ellipse(x, y, 10, 10);

}

}

}

for (int i = 0; i <=6; i++){

noFill();

strokeWeight(1);

stroke(0, 255-(30\*i), 0);

ellipse(radius, radius, (100\*i), (100\*i));

fill(0, 100, 0);

noStroke();

text(Integer.toString(radarDist+50), 380, (305-radarDist), 50, 50);

radarDist+=50;

}

radarDist = 0;

for (int i = 0; i <= 6; i++) {

strokeWeight(1);

stroke(0, 55, 0);

line(radius, radius, radius + cos(radians(180+(30\*i)))\*w, radius + sin(radians(180+(30\*i)))\*w);

fill(0, 55, 0);

noStroke();

if (180+(30\*i) >= 300) {

text(Integer.toString(180+(30\*i)), (radius+10) + cos(radians(180+(30\*i)))\*(w+10), (radius+10) + sin(radians(180+(30\*i)))\*(w+10), 25,50);

} else {

text(Integer.toString(180+(30\*i)), radius + cos(radians(180+(30\*i)))\*w, radius + sin(radians(180+(30\*i)))\*w, 60,40);

}

}

noStroke();

fill(0);

rect(350,402,800,100);

fill(0, 100, 0);

text("Degrees: "+Integer.toString(degree), 100, 380, 100, 50);

text("Distance: "+Integer.toString(value), 100, 400, 100, 50);

text("Radar screen code ", 540, 380, 250, 50);

fill(0);

rect(70,60,150,100);

fill(0, 100, 0);

text("Screen Key:", 100, 50, 150, 50);

fill(0,50,0);

rect(30,53,10,10);

text("First sweep", 115, 70, 150, 50);

fill(0,110,0);

rect(30,73,10,10);

text("Second sweep", 115, 90, 150, 50);

fill(0,170,0);

rect(30,93,10,10);

text("Average", 115, 110, 150, 50);

noFill();

stroke(150,0,0);

strokeWeight(1);

ellipse(29, 113, 10, 10);

fill(150,0,0);

text("Motion", 115, 130, 150, 50);

}

void serialEvent (Serial arduinoport) {

String xString = arduinoport.readStringUntil('\n');

if (xString != null) {

xString = trim(xString);

String getX = xString.substring(1, xString.indexOf("V"));

String getV = xString.substring(xString.indexOf("V")+1, xString.length());

degree = Integer.parseInt(getX);

value = Integer.parseInt(getV);

oldValue[degree] = newValue[degree];

newValue[degree] = value;

firstRun++;

if (firstRun > 360) {

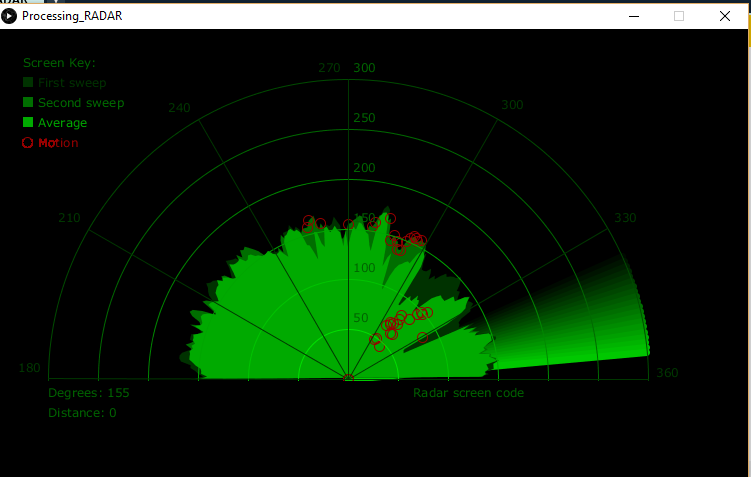
firstRun = 360;

}

}

}

And now we got a positive response from our project after taking test and burning program again and again we have output below:



The final project looked like this;