American Journal of Modern Physics
2022; 11(2): 22-31
http://www.sciencepublishinggroup.com/j/ajmp
doi: 10.11648/j.ajmp.20221102.12
ISSN: 2326-8867 (Print); ISSN: 2326-8891 (Online)

Development and Realization of an Ultrasonic Ranging Detection and Tracking Device

Odaba Alphaeus, Alan Audu Ngyarmunta, Ohemu Monday Fredrick

Department of Electrical and Electronic Engineering, Air Force Institute of Technology Kaduna, Kaduna, Nigeria

Email address:
a.odaba@afit.edu.ng (O. Alphaeus), ngyarmuntaa@yahoo.com (A. A. Ngyarmunta), monfavour@gmail.com (O. M. Fredrick)

To cite this article:
Odaba Alphaeus, Alan Audu Ngyarmunta, Ohemu Monday Fredrick. Development and Realization of an Ultrasonic Ranging Detection and Tracking Device. *American Journal of Modern Physics*. Vol. 11, No. 2, 2022, pp. 22-31. doi: 10.11648/j.ajmp.20221102.12

Received: April 1, 2019; **Accepted**: December 15, 2021; **Published**: April 9, 2022

Abstract: There are several ways of contactless distance measurements. This research work made use of the principle of ultrasonic distance measurements and calculations as well as tracking of dynamic object by the techniques of distance comparison. When an electrical pulse of high voltage is applied to the ultrasonic transducer it vibrates across a specific spectrum of frequencies and generates a burst of sound waves. Each time any obstacle comes ahead of the ultrasonic sensor, the sound waves will reflect in the form of echo and generates an electric pulse. The time taken between sending sound waves and receiving the echo was calculated and the patterns of echo were compared with the patterns of sound waves to determine the detected signals condition. The device consists of two HC-SR04 ultrasonic sensors that sweep continuously through 1800 to detect and track object based on developed and installed codes in the Arduino Uno microcontroller and displays the range and angular position using Processing 3 Software on a computer screen display. The envisaged minimum and maximum range of object detections is 2cm and 39cm respectively using processed signal. However, the measured distance is from 5cm to 35cm and the corresponding calculated distances using waveforms from an oscilloscope were 6.8cm and 47.6cm. The discrepancies were attributable to the 0.2ms rise time of the trigger signals. The device was capable of tracking only relatively slow-moving objects and can be applicable in robotic vision, automatic guided vehicles, security surveillance, automobile anti-collision system and precision contactless measurements.

Keywords: Ultrasonic, Ranging, Tracking, Detection, Servo-Control, Frequencies, Surveillance, Sensors

1. Introduction

The detection, measurement and tracking of objects, their distances and directions have preoccupied engineers, scientists and technologist from the inception of human history [1, 2]. Many different techniques have been devised for the measurement of distance from the observer to a target, for the purposes of surveying, navigation, determining focus in photography, or accurately aiming a weapon in missile warfare [3]. Manual distance measuring and tracking are always done at the expense of human errors and risks [4]. Distance measurement has three important factors which are source, medium and target. The source is a point from where distance is to be measured and target is the object whose distance is to be measured. The medium lies between source and target. This research work designs and implements the measurement of dista and tracking of object without contacting the target. This is made possible through the use of sound waves of ultrasonic frequencies. Sound is a phenomenon that is a result of vibration of materials. Sound is characterized as a mechanical wave that carries mechanical energy [5]. For the transmission of this energy between transmitter and receiver, presence of a medium is necessary. The medium can be solids, liquids or gases. The sound energy travels by causing disturbance in the medium it is travelling and this is called propagation of sound waves. Under normal conditions, the velocity of sound is 330m/s [6]. Depending on the frequency of the sound wave, the SONAR (sound navigation and ranging) can be either Infrasonic or Ultrasonic [7]. Ultrasonic sensors produce sound waves with frequencies higher than the audible range (20Hz to 20 KHz), that is, greater than 20 kHz [9]. In case of an infrasonic sensor, the frequency of sound wave is less than 20Hz. Ultrasound frequency ranges from 20 kHz to several GHz. Ultrasound has its application in many fields which includes among others, liquid level indicator, car parking safety

system, Non-Destructive Testing (NDT) in detecting the flaws in plate composition as part of quality inspection and in bottom vessel to monitor coral seabed distance and to find schools of fishes [10].

2. Materials and Methods

This section covers hardware and software implementations for the ultrasonic system.

2.1. Hardware Implementation

The hardware implementation deals with the appropriate choice of materials based on the specification (Table 1) required for the ultrasonic device.

These include Arduino Uno microcontroller, HC-SR04 Ultrasonic Sensors, Tower Pro SG90 9g Servomotor, Piezoelectric Buzzer, Computer and integrated power supply via the USB port suitably interfaced.

2.1.1. Arduino Uno Specifications [11]

Microcontroller	ATmega328
Operating voltage	5V
Input voltage (recommended)	7-12V
Input voltage (limit)	6-20V
Digital I/O pins	14 (of which 6 provide PWM output
Analog input pins	6
DC current per I/O pins	40mA
DC current for 3.3v pins	50mA
Flash Memory	32KB of which 0.5KB used by the bootloader
SRAM	2KB
EEPROM	1KB
Clock Speed	16MHZ

2.1.2. Ultrasonic Specifications

Type	HC-SR04 Ultrasonic (US) sensor

2.1.3. Servomotor Specifications

Type	Tower pro SG90 9g servomotor
Weight:	9 g
Dimension:	22.2 x 11.8 x 31 mm approx.
Stall torque:	1.8 kgf·cm
Operating speed:	0.1 s/60 degree
Operating voltage:	4.8 V (~5V)
Dead band width:	10 µs
Temperature range:	0°C – 55°C

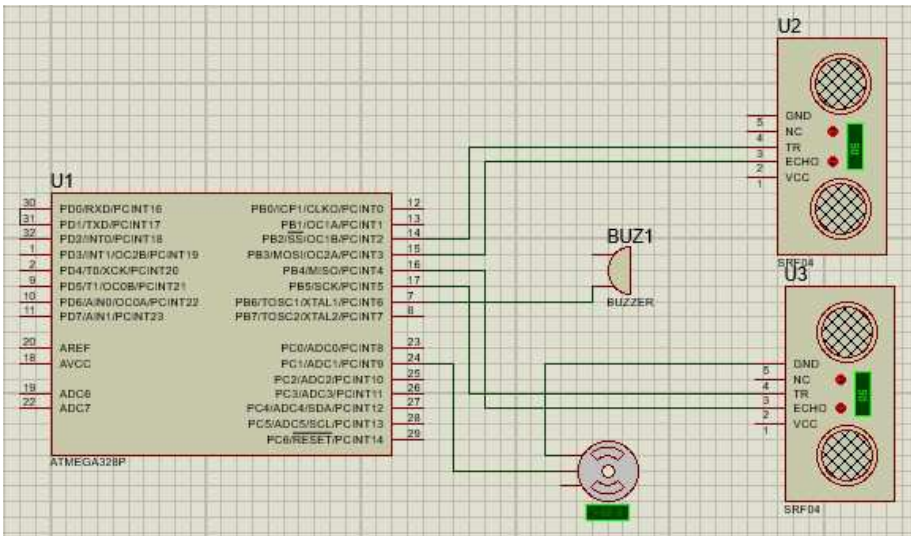


Figure 1. *Working Diagram.*

Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the left.

The display of the system is a desktop or laptop computer with a 5V USB port, Window Operating System, 1366 X 768

resolution and capable of multi-tasking is selected for graphic display and provision of integrated power supply in addition to being the platform for embedding the Arduino Uno. The computer must be 32 or 64 bits compliant to enable a serial communication of 9600 bits per second (bps) data transfer. The hardware was implemented according the working diagram as shown on figure 1.

2.1.4. Working Diagram

The working diagram is a schematic display of the device hardware implementation.

2.2. The Software Implementation

This includes writing code in C programming language for the Arduino board and a modified Java code for the Processing Software. The software implementation is based on the algorithm which is itemise in the following steps:

Step 1: Initiate a servo start up test.

Step 2 Emit 8 burst of 40kHz Ultrasounds signals.

Step 3: Listen for echoes.

Step 4: If there are echoes, calculate distances (Use the ultrasonic detection algorithm) from each of the sensors.

Step 5: Compare distances and if there is variation, increment or decrement servo position accordingly as detected.

Step 6: Go to Step 2.

Thus, the code constantly remains on the alert for object presence or movements.

The algorithm is graphically displayed as shown on the flowchart in figure 2.

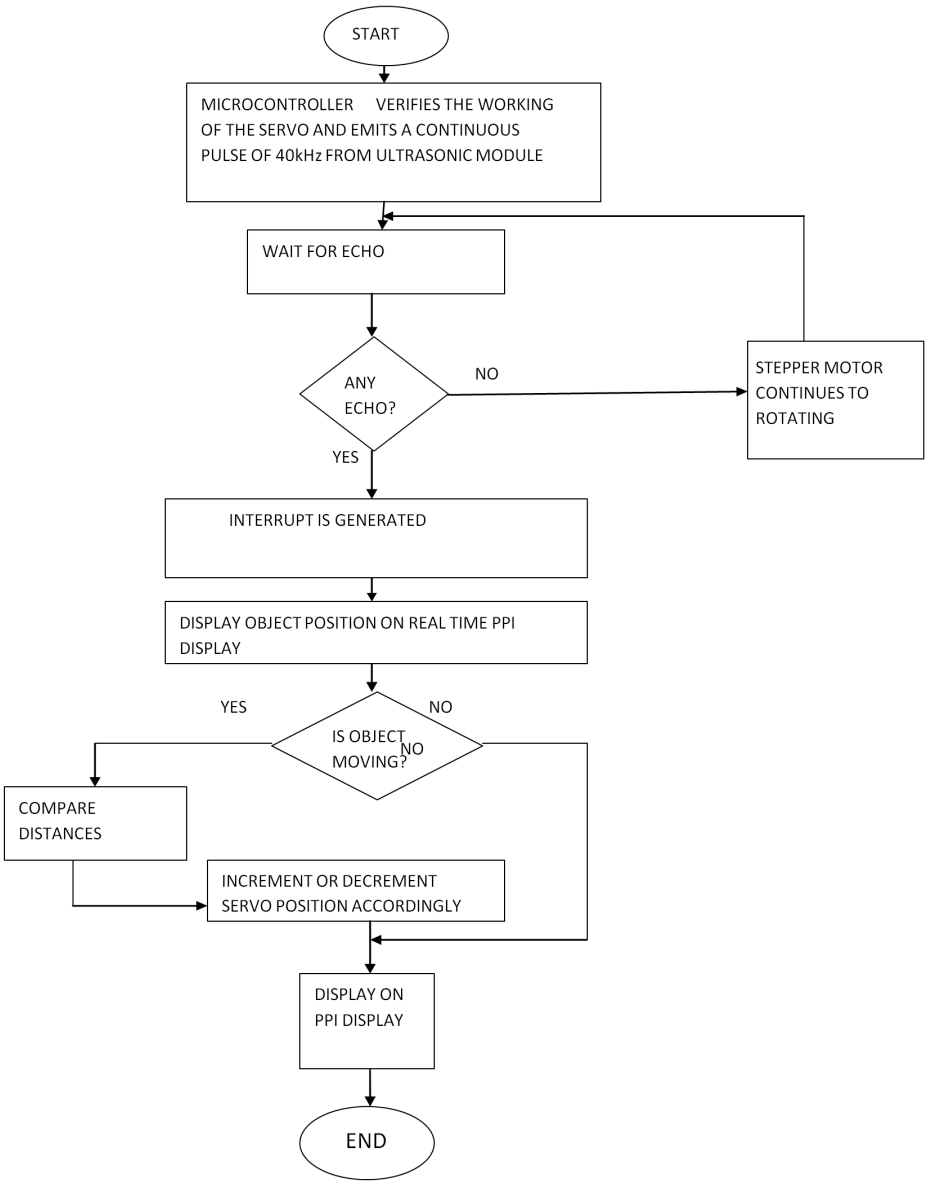


Figure 2. *Program flow chart.*

The program flow chart of figure 2 is a pictorial display of the algorithm. It covers integrating servo control while using two ultrasonic sensors to detect object and as well track the object's movement. The software implementations include:

1. Servo Control;
2. Ultrasonic Sensor Operation;
3. Object Detection;
4. Tracking;
5. Display.

2.2.1. Servo Control

The servo is controlled by three wires: ground, power, and control. The servo will move based on the pulses sent over the control wire, which set the angle of the actuator arm. The servo expects a pulse every 20mS to gain correct information about the angle [12]. The width of the servo pulse dictates the range of the servo's angular motion. The servo angle is incremented or decremented by the control code, for both ranging and tracking. The servo bandwidth of 10µS with capability to follow rapid changes in the commanded input is chosen to implement tracking of echoes of precisions in microseconds. The Tower Pro micro servo used has Stall torque of 1.8 kg/cm and thus can carry a 1.8kg load at a distance of 1cm. Considering the weight of the HC-SR04 sensor of 9g, the chosen servo is adequate. The servo control is effected using the servo.h Arduino library. The servo control command in Arduino is initiated as:

Servo myServo; // Creates a servo object for controlling the servo motor myServo.attach(9); // Defines on which pin is the servo motor attached

///servo startup test myServo.write(90);delay(500); myServo.write(180);delay(500); myServo.write(0);delay(500); myServo.write(90);delay(500);

2.2.2. Ultrasonic Sensor Operation

The two sensors are coded to transmit and receive signals concurrently based on the trigger signal from the microcontroller to the trigger pins. The echo expected within a preset timing are analyzed for detection and tracking. The operational code for trigger pulse and echo reception in Arduino is initiated with the following code and the pins are in accordance with Appendix I.

// Defines Trig and Echo pins of the Ultrasonic Sensor const int trigPin = 5; const int echoPin = 4; const int trigPin2 = 2; const int echoPin2 = 3; pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin, INPUT); // Sets the echoPin as an Input pinMode(trigPin2, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin2, INPUT); // Sets the echoPin as an Input

2.2.3. Object Detection

The measurement process is initiated by sending a trigger signal to the ultrasonic module. When the reset pulse is given to the processor, it produces a trigger pulse of 15 µs and transfers to the HC-SR04 ultrasonic module [13]. The trigger signal is a pulse with 10µS high time. When the module receives a valid trigger signal it issues 8 pulses of 40 KHz ultrasonic sound from the transmitter. The echo of this sound is picked by the receiver, after getting the echo of the ultrasonic sound, the module produces a signal at the echo pin such that the HIGH time is proportional to the distance to be measured.

$$Object\ distance\ =\ (high\ level\ time\ \times\ velocity\ of\ sound\ (340m/s))\ /\ 2 \tag{1}$$

$$Distance\ in\ cm\ =\ echo\ pulse\ width\ in\ \mu s/58 \tag{2}$$

The Arduino code for object detection is:

long 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 total duration from transmitter to object and back to receiver.

//wave travel time in microseconds

*distance= duration*0.034/2*;

Finally, the distance calculated based upon the pulse width of the echo signal is send to the laptop and the range is displayed using processing software in centimeters. To establish communication between the Arduino and the computer, the SPI port is enabled through the serial command:

Serial.begin(9600); void snd(){

Serial.print(i);Serial.print(",");Serial.print(distance2);Serial.print(".");

}

This ultrasonic device can measure distances of 2.0cm to 39.0cm at accuracy of 0.1 centimetre based on calculation from pulse width using UN-T digital storage oscilloscope and an accuracy of 1cm using the Processing software.

2.2.4. Tracking

Comparison of two measurements from each sensor is used to determine if an object is in front of the left, right, or both sensors. When the outputs of the sensors change, the servo rotate in the direction of the object, based on the differences in distance, thus tracking its motion. The tracking is achieved by the following lines of command:

void loop(){

#define acc 2 ///accuracy------difference in value of the 2 sensor start:

myServo.write(i); delay(30); if(dir==1)i++;

else if(dir==0)i--;

while(calculateDistance()<40 ||calculateDistance2()<40){ distance2 = calculateDistance2(); if(distance>distance2+acc){dir=1;i++;if(i>179)i=179;} else if(distance<distance2-acc){dir=0;i--;if(i<1)i=1;} else if(abs(distance-distance2)<=10){} myServo.write(i); delay(30); snd(); }

2.2.5. Buzzer

The buzzer indicates object presence by beeping. The code for the buzzer is, *pinMode(Buz, OUTPUT); digitalWrite(Buz,LOW); if(calculateDistance()<40*

```
||calculateDistance2()<40){ analogWrite(Buz,HIGH);
  } else{ analogWrite(Buz,LOW);
  }
```

2.2.6. Processing 3 Software

Processing 3 is an open source programming language and integrated development environment (IDE) for writing software to make images, animations, and interactions for the electronic arts, new media art, and visual design communities aimed at teaching the fundamentals of computer programming in a visual context, and as well serve as the foundation for electronic sketchbooks.

The software aids in the display of analysed echo.

The Processing 3 default window provides space for coding in Java as is shown in figure 3. The codes though basically Java, has some peculiarities and modification known as Processing language. Programs written with Processing are called sketches. When the PDE (Processing Development Environment) is started, a new sketch is created automatically. The Processing syntax is a dialect of Java. It hides certain parts of Java in order to allow for faster coding.

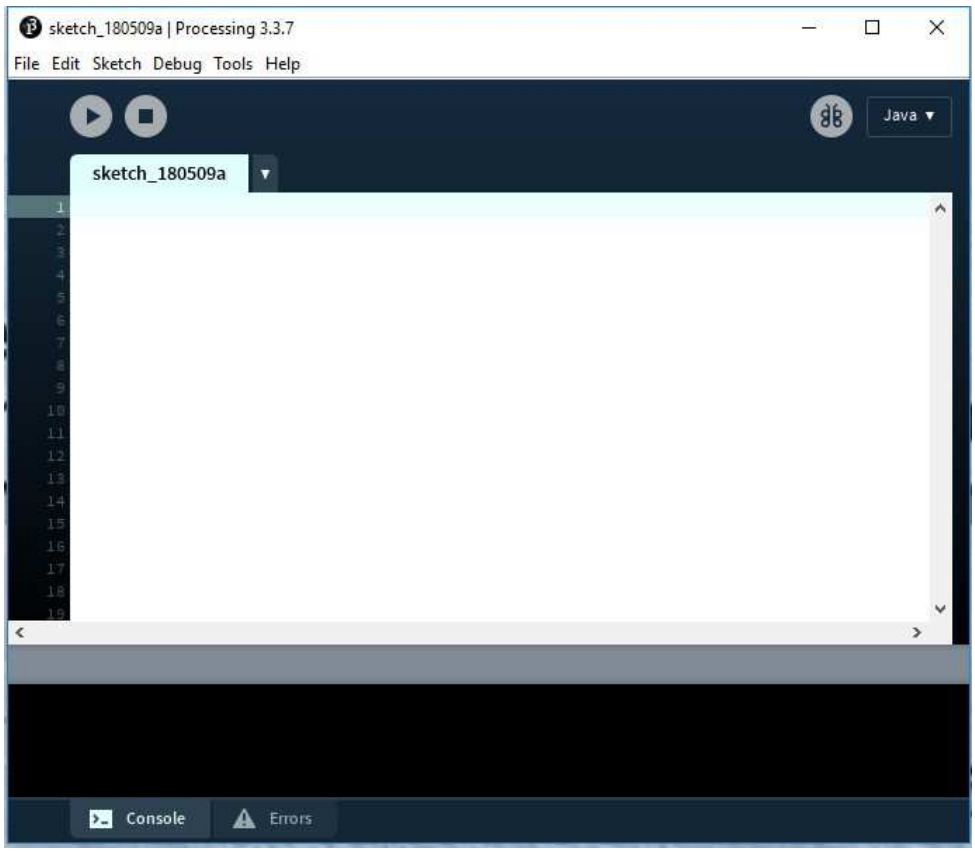


Figure 3. *Processing IDE Default window.*

If an object is detected, the signal is processed by the microcontroller and the signal is communicated to the computer for display by the Processing 3 software that shows the presence of the obstacle on the rotating PPI screen with distance and the angle at which it has been detected.

The Codes for the display using Java programming language is in appendix II.

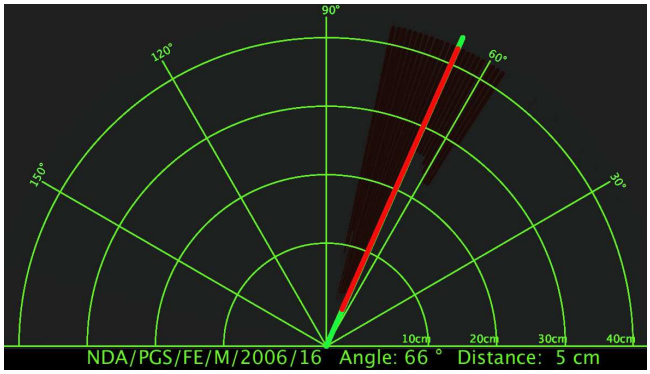
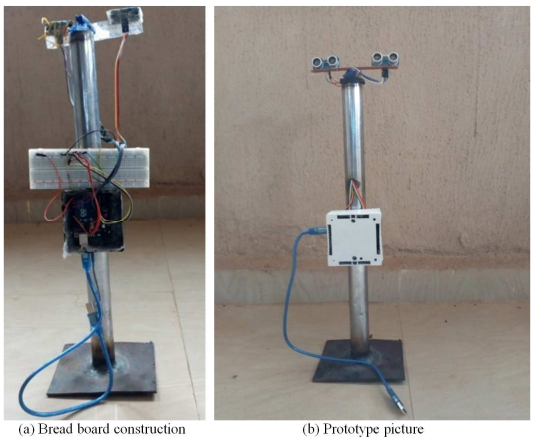


Figure 4. *PPI display using Processing Software.*



(a) Bread board construction (b) Prototype picture

Figure 5. *(a & b) Pictures of Construction.*

The complete integrated code that implements the technique in C programming language is in Appendix I and that for the display in Java is in Appendix II. Integration of both codes working correctly was achieved through the communication interface of the Arduino board.

2.3. Construction

The device was constructed using the working diagram of figure 1. The stand for the servo motor was constructed from a steel pipe and iron base plate. A plastic arm and mounting plastic ruler were used to attach the HC-SR04 ultrasonic sensor to the motor shaft. This was done to determine the effective separation of the pairs of the Ultrasonic sensor that yield track able distance variations. Several types of obstacles were used for verification of detection and tracking. The observed echo pulse had no amplitude differences. This observation is as a result of the object detection technique of the HC-SR04 ultrasonic sensor. The echo signal goes high at the same time as the trigger pulse and remains high until a reflection is detected within a preset distance. The HC- SR04 range of object detection is 2cm to 40cm, but for the purpose of clarity in PPI display a distance of 40cm was chosen as maximum detectable distance. The measurement set up for the oscilloscope and the processed values were concurrently done accordingly.

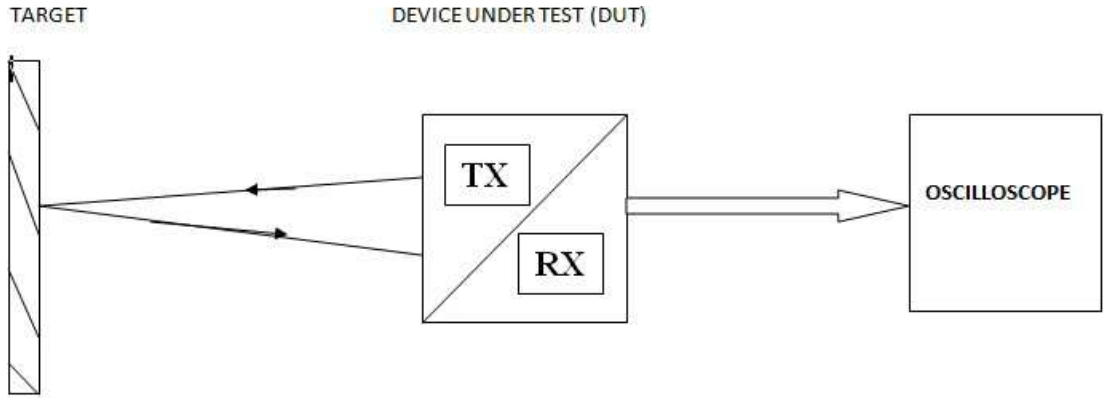


Figure 6. *Experimental Setup for Calculated Range Measurement [14].*

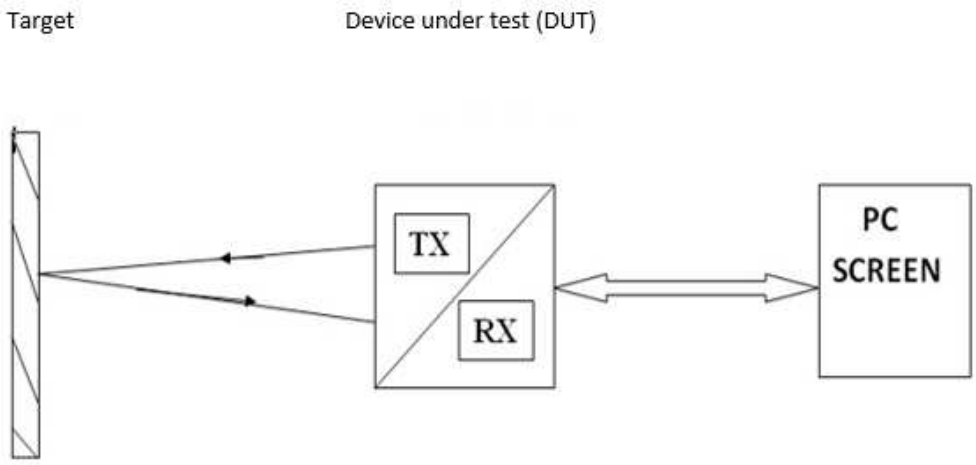


Figure 7. *Experimental Setup for processed Range Measurement [14].*

3. Results

Different distances were measured by the device to determine the accuracy of the measurements.

Using the oscilloscope, the time scale is 10.0 mS for object distance beyond 40cm. The oscilloscope scale is 2.0 mS for all other measurement. The PPI shows range as arcs and

azimuths (angle) as radial coordinates. From equation 2, the distances for various target positions are calculated and the values are displayed.

Table 1 shows some of the device measurements with their associated actual distances and the error percentage using copper board as obstacle.

Table 1. *The Difference between Actual Distance and the Calculated Distance.*

Actual Processed distance (cm)	Calculated distance (cm)	Difference (cm)	$Error = \frac{Difference}{Actual} \times 100$
5	6.8	1.8	36
10	13.6	3.6	36
15	20.4	5.4	36
20	27.4	7.4	37
25	34	9	36
30	40.8	10.8	36
35	47.6	12.6	26.5

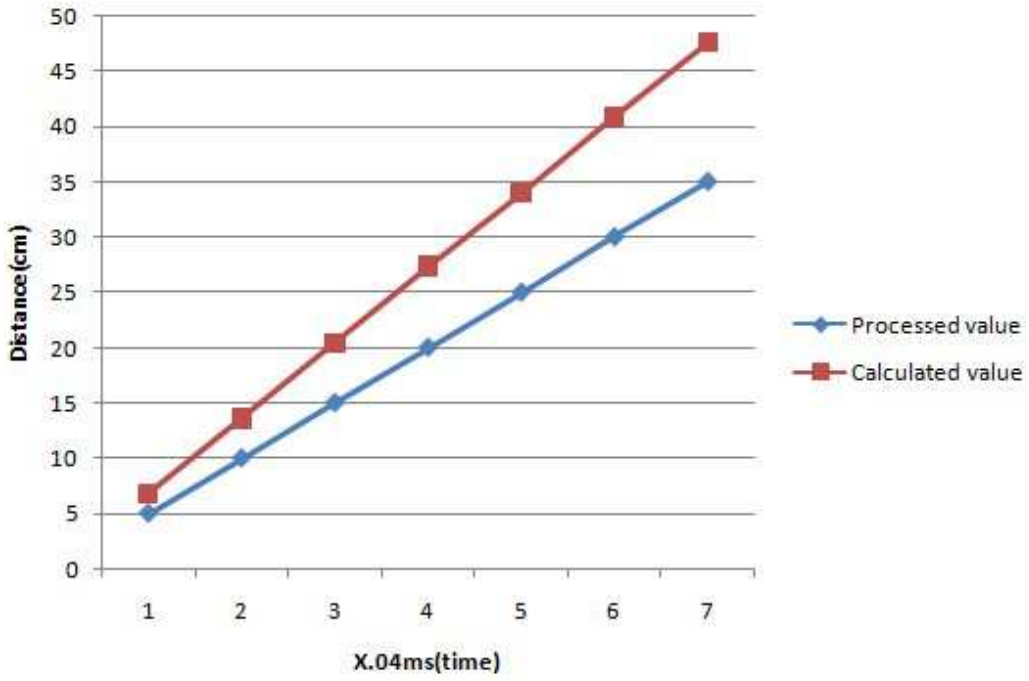


Figure 8. *Graphical display of the actual (processed) and calculated distance.*

The processed and calculated values are linearly dependent on time [15] The device displayed the range on PPI to the nearest centimetre and the azimuth to the nearest degree using the Processing 3 software. Physical measurement using ruler confirmed the distances to be actual. Leads from trigger pin and echo pin of the HR-SC04 modules were connected to the probes of digital oscilloscope and the trigger signal pattern and echo pulse pattern were analysed, hence the distances were calculated. While the precision of the calculated distance was in millimetre, the error in accuracy was 36% on the average. The discrepancies in calculated distance and measured distance are assumed to be traceable to the measured rise time of the trigger pulse of 0.2mS that provided additional delay time to echo pulse. The device effectively tracked a slow-moving object. The device is capable of tracking relatively slow-moving object within the sensing range.

4. Conclusion

As described earlier in this research, a system is developed to detect objects and calculate the distance of the object as well as track its movement at relatively slow speed. The implemented ultrasonic device differed from other works in incorporating tracking using distance comparison with the following conclusions.

The specified range of the HC-SR04 ultrasonic sensor used in the device is 2cm to 400cm, but to achieve a processed display visible on the laptop screen, 2cm to 40cm was achieved.

The device calculates distance of obstruction with sufficient accuracy using time of flight principle and pulse duration technique. A verification of pulse width using oscilloscope shows a calculated distance with an error of 36 % from the actual.

A moving target of relatively low speed has been tracked successfully.

The completion of this thesis according to the requirement of objectives that have been stated in the early chapters has been achieved.

Appendix

Appendix 1. (Arduino Pins) [16]

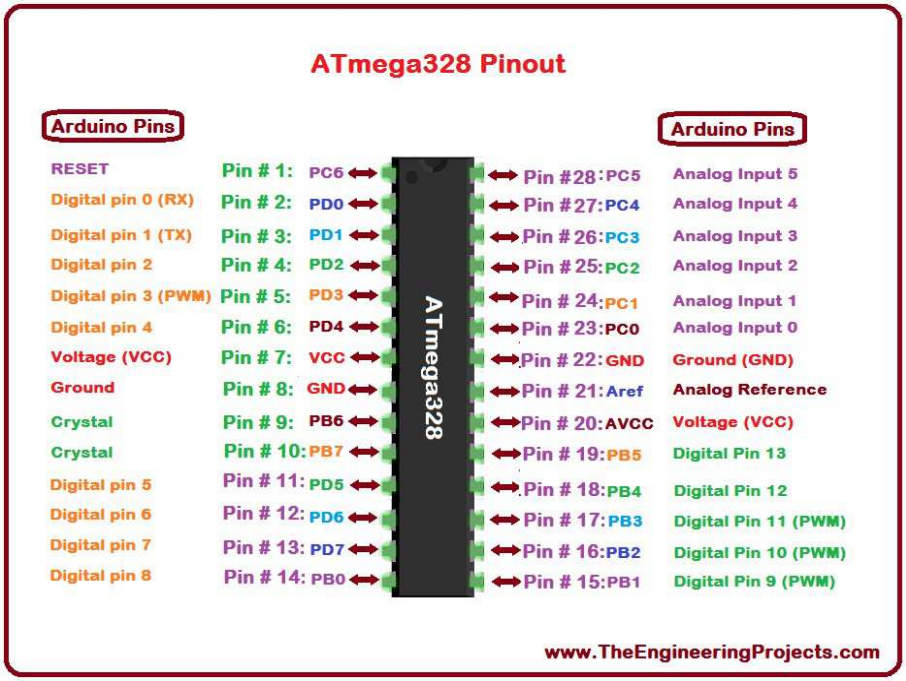


Figure 9. *Arduno (ATmega328) pins indicator.*

Appendix 2. (Source Code for the Arduino Board)

```
// Includes the Servo library #include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 5; const int echoPin = 4; const int trigPin2 = 2; const int echoPin2 = 3; int Buz = 6;
// Variables for the duration and the distance long duration; signed long distance; long duration2;
signed long distance2;
f
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(trigPin2, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin2, INPUT); // Sets the echoPin as an Input
pinMode(Buz, OUTPUT); Serial.begin(9600);
myServo.attach(9); // Defines on which pin is the servo motor attached digitalWrite(Buz,LOW);
///servo startup test myServo.write(90);delay(500); myServo.write(180);delay(500); myServo.write(0);delay(500);
myServo.write(90);delay(500);
}
#define sen 1 int i=0; bool dir; void loop(){
#define acc 2 ///accuracy------difference in value of the 2 sensor start:
myServo.write(i); delay(30); if(dir==1)i++; else if(dir==0)i--; if(calculateDistance()<40 ||calculateDistance2()<40){
analogWrite(Buz,HIGH);
}
else{
analogWrite(Buz,LOW);
}
while(calculateDistance()<40 ||calculateDistance2()<40){
distance2 = calculateDistance2(); if(distance>distance2+acc){dir=1;i++;if(i>165)i=165;} else if(distance<distance2-acc){dir=0;i--;if(i<15)i=15;} else if(abs(distance-distance2)<=10){} myServo.write(i); delay(30); snd(); } snd(); if(i<15){dir=1;} if(i>165){dir=0;}
goto start;
}
void snd(){
Serial.print(i);Serial.print(",");Serial.print(distance2);Serial.print("."); }
long 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; if (distance>100) return 100; return distance;
}
```

```
long calculateDistance2(){
digitalWrite(trigPin2, LOW); delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds digitalWrite(trigPin2, HIGH); delayMicroseconds(10); digitalWrite(trigPin2, LOW);
duration2 = pulseIn(echoPin2, HIGH); // Reads the echoPin, returns the sound
//wave travel time in microseconds
distance2= duration2*0.034/2; if (distance2>100) return 100; return distance2;
}
‘
```

Appendix 3. (Source Code for the Processing Software)

For the Processing graphic display on a laptop or desktop the code developed using examples of ‘Draw line’ is as follows.[48].

```
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; float pixsDistance; int iAngle, iDistance; int index1=0; int index2=0; PFont orcFont; void setup() {
size (1200, 700); // ***CHANGE THIS TO SCREEN RESOLUTION BEING USED*** 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.
} void draw() {
fill(98,245,31);
// 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)),(widthwidth*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(); 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(25);
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(40);
text(" StudentsHeart.com ", width-width*0.875, 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); 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();
  }
```

References

[1] Physics of music-notes. http://pages.mtu.edu,~suits, Speed of Sound in Air/ accessed 24 May, 2018.

[2] Yoon, Carol Kaesuk. "Donald R. Griffin, 88, Dies; Argued Animals Can Think", *The New York Times*, November 14, 2003. Accessed January 16, 2018.

[3] Ultrasonic Sensor by Anshul Thakur; https://www.engineersgarage.com/articles/ultrasonicsensors. Accessed January 16, 2018.

[4] Muller, R. (2004). "A numerical study of the role of the tragus in the big brown bat". JASA. 116 (6): 3701–3712. *Bibcode*: *2004ASAJ..116.3701M*. *doi*: *10.1121/1.181513*. *Accessed January 16, 2018.*

[5] What is Piezoelectric effect? By Carmen Emily Yang; http://www.electronicdesign.com/power/whatpiezoelectric-effect, Accessed January 16, 2018.

[6] The Radar Technology behind Autonomous Vehicles by Paul Pickering; http://www.design-technology.info/inventors/page28.htm, Accessed January 16, 2018.

[7] About Ultrasonic, Migatron Corp. www.migatron/ understanding ultrasonic technology/ accessed 24 April, 2018.

[8] L. P. Palma: Ultrasonic Distance Measurer, Freescale Semiconductor, application note, (2008).

[9] Mohammed Elmontasir Ibraheem Suliman, "Ultrasonic Range Meter" www.khartoumspace.uofk.edu. Accessed January 16, 2018.

[10] Basics of Microcontrollers by Vysakh. www.circuitstoday.com */accessed* 24 April, 2018.

[11] An Introduction to Processing and Music Visualization, by Christopher Pramerdorfer; Processing, 2010. Theoretical processing website. http://processing.org/; accessed 24 May, 2018.

[12] Gonzalo Rodr´ıguez-Canosa *et al*, "A Real-Time Method to Detect and Track Moving Object From Unmanned Aerial Vehicle Using a Single Camera"; www.ijrat.org. Accessed January 16, 2018.

[13] Deshmukh Gourav *et al* "Ultrasonic Radar For Object Detection, Distance And Speed Measurement"; www.mdpi.com/journal/sensors 2014, 14, 2911-2943; doi: 10.3390/s140202911.

[14] Rajan P Thomas et al. "Range Detection based on Ultrasonic Principle", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 2, February 2014. Copyright www.ijareeie.com

[15] Dai Juan et al, "Ultrasonic Automatic Tracking System", TELKOMNIKA Indonesian Journal of Electrical Engineering Vol. 12, No. 6, June 2014, pp. 4664 ~ 4670 DOI: 10.11591/telkomnika.v12i6.5449.

[16] Srijan Dubey *et al*. "Implementation Of Radar Using Ultrasonic Sensor." Indian J.Sci.Res. 14 (2): 482-485, 2017; ISSN: 2250-0138 (Online). Accessed March 14, 2018.