Object Detection using SONAR based system

Report submitted for

Mini Project
(5th Semester)

in partial fulfillment of the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

By

Resu Nikhil Reddy, 14UEC080
Sumit Sapra, 14UEC108
Surya Prakash Venkat, 14UEC109

Under the guidance of

Dr. M. V. Deepak Nair, Assistant Professor
Department of Electronics and Communication Engineering
The LNM Institute of Information Technology
Jaipur, India

The LNM Institute of Information Technology
Jaipur, India
http://www.lnmiit.ac.in

Title:
Object Detection using SONAR based system

Theme:
Embedded Systems

Project Period:
Odd Semester 2016

Project Group:
Resu Nikhil Reddy
Sumit Sapra
Surya Prakash Venkat

Supervisor:
Dr. M. V. Deepak Nair

Page Numbers: 16

Date of Completion:
December 9, 2016

Abstract:

The goal is to develop an object detection system which could determine the presence of objects in it's vicinity.

The system leverages the *HC-SR04 Ultrasonic Sensor* mounted on top of a *Micro-Servo*motor, both being controlled by an *Arduino UNO*. The HC-SR04 detects its separation from an obstacle placed in front of it and sends the data to the Arduino, while the servo rotates the sensor about its axis in order to have a field of view that sweeps over a sector with a radius specified within 4 metres.

The *Arduino* recieves sensor readings (distance measurements) from the sensor and sends it via *USART* to a computer for graphical display using the *Processing* IDE.

Contents

Introduction

The basic approach of our project, just like any other based on embedded systems (*microcontrollers*) largely depends on three main questions:

- Which *sensor*(s) should be used to fulfill the goals of the project? ...
- What actuators might be needed for introducing mechanical action, if needed? ...
- Which microcontroller board to use? ...

We had a literature survey through the internet about deciding the components needed for our project. This included studying in detail, several project ideas and instructions posted on blogs and YouTube.
We selected the following major components for having a *minimal*, *cost effective* and *efficient* apparatus that could do the job for us:

- *Sensor - HC-SR04 Ultrasonic Sensor*
- *Actuator - Micro-Servo* Motor
- *Microcontroller Board - Arduino UNO*

The next few sections will explain the basis of selection of hardware for this project.

The *Arduino UNO*

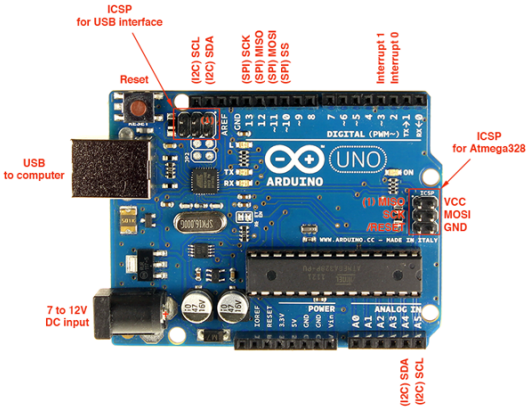


Figure 1.1: Arduino UNO Pinouts

Arduino is an open source electronics platform for fast prototyping of projects for users with minimal knowledge or experience in electronics and programming. We have used the *Arduino UNO*, the most widely used variant of the Arduino.
Technically, Arduino Uno is a microcontroller board based on the 8-bit ATMega328P microcontroller. With 14 digital input/output pins (including 6 PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button, it has everything needed to support the microcontroller; we simply have to connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
Arduino also comes with the Arduino IDE, in which we can write code and upload to the Arduino. The programming language used for an Arduino, called the Arduino Programming language, is very similar to C/C++ except that we can use inbuilt functions of the Arduino libraries which keep the code very simple and short.

Why Arduino?

We had the following reasons strongly backing our decision to choose the *Arduino UNO*:

Support - Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux.
This has enabled a very large support base for Arduino, more than any other microcontroller board ever released in the market. Every problem/bug/question that we ever had throughout the course of this project was promptly answered by questions in online forums, as there was a very high probability that scores of people had already posted the same question and the issue was resolved by other experienced users.

Cost Effective - The Arduino UNO, its basic variant that every beginner to embedded systems usually starts with, is available for less than $25. Being an open source hardware and due to its simplicity, it is easily and widely replicable, which is also responsible for the growth of such a large community as mentioned in the earlier reason.

Features - It is generally observed that the computing power and inbuilt memory of the Arduino easily supports a wide variety of circuits with multiple components along with continuous serial communication with another device/computer. Compared to the system configuration of the board, our project would require fairly low amount of resources.

The no. of GPIO pins on the UNO is more than sufficient considering the need for the circuitry in our project, as we shall see when we look at the pinouts of the servo motor and the sensor - the only two other devices being used in the project.

One important advantage of using the UNO is that unlike most previous programmable circuit boards, the *Arduino UNO does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – we can simply use a USB cable.* The ATmega328 on the Arduino/Genuino Uno comes preprogrammed with a *bootloader* that allows us to upload new code to it without the use of an external hardware programmer.

Moreover, the components we have used work under the same operating voltage and current conditions as an arduino, especially considering the fact that such components are

Microcontroller	ATmega328P(8-bit)
Operating Voltage	5V
Input Voltage	(recommended) 7-12V; (limits) 6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Need of External Programmer	NO
Built-in LED	pin13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Table 1.1: Features of Arduino UNO

nowadays sold with their compatibility with arduino in mind.

Considering the above factors, we found Arduino UNO to be the most appropriate microcontroller board to work with for this project.

Examples

You can also have examples in your document such as in example 1.1.

Example 1.1 (An Example of an Example)
Here is an example with some math

$$0 = \exp(i\pi) + 1 \,. \tag{1.1}$$

You can adjust the colour and the line width in the `macros.tex` file.

A Subparagraph Moreover, you can also use subparagraph titles which look like this. They have a small indentation as opposed to the paragraph titles.

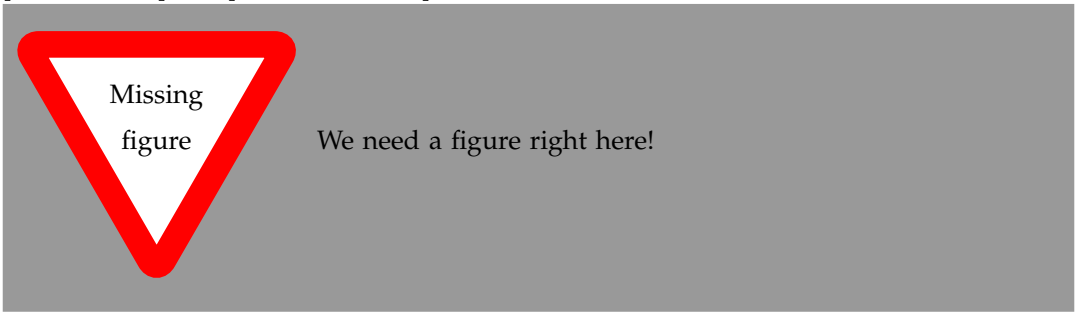
Is it possible to add a subsubparagraph?

I think that a summary of this exciting chapter should be added.

Chapter 2 name

Here is chapter 2. If you want to leearn more about LaTeX 2_ε, have a look at [**Madsen2010**], [**Oetiker2010**] and [**Mittelbach2005**].

I think this word is mis-pelled



Conclusion

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

Jesper Kjær Nielsen
jkn@es.aau.dk
http://kom.aau.dk/~jkn
Fredrik Bajers Vej 7
9220 Jaipur XD Ø

Appendix A : *Arduino* Code

Here is the code uploaded on to the *Arduino UNO*:

Listing A.1: The Arduino Code

```
/*ver 1.1 (Final)*/

// Includes the Servo library
#include <Servo.h>.

// Defines Trig and Echo pins of the Ultrasonic Sensor
const int trigPin = 2;
const int echoPin = 4;

// Variables for the duration & distance measured from the sensor
float duration;
float distance;

//Angle offset between graph and motor's XY axes
int offset = 0;
//Initial and final angles of motor's rotation
//Note that the motor can only take values 0-180.
//Make sure init_angle-offset >=60 to avoid jumper wire
    obstruction.
int init_ang = 70+offset;
int fin_ang = 180+offset;

// Declares a structure variable/object of the type Servo,
//(defined in the servo library) for controlling the servo motor.
Servo myServo;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); //Sets baud rate of serial communication
myServo.attach(9); // Defines on which pin is the servo motor
    attached
```

```
}

void loop() {
// if (Serial.available() > 0)  /*piece of code to be included in
    case of serial communication with IoT board*/
// {
// rotates the servo motor from init_ang to fin_ang degrees
for(int i=init_ang;i<=fin_ang;i++)
{
myServo.write(i-offset);  //angle value to be passed to the servo
    library object for writing into the motor
delay(17);//DELAY #1:for time taken in motor rotation for one
    degree before calculating distance
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
    for graphical representation
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 for the graph
Serial.print(";"); // Sends addition character right next to the
    previous value needed later in the Processing IDE for indexing
}
// Repeats the previous lines from fin_ang to init_ang degrees
for(int i=fin_ang;i>init_ang;i--)
{
myServo.write(i);
delay(17);  //DELAY #1 //can be minimised to 17 or 1667 microsec
distance = calculateDistance();
Serial.print(i);
Serial.print(",");
Serial.print(distance);
Serial.print(";");
}
//}
}
// Function for calculating the distance measured by the
    Ultrasonic sensor
float calculateDistance(){
unsigned long T1 = micros();
digitalWrite(trigPin, LOW); // trigPin needs a fresh LOW pulse
    before sending a HIGH pulse that can be detected from echoPin
delayMicroseconds(2);//DELAY #2:time for which low trig pulse is
    maintained before making it high
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);//DELAY #3:Sets the trigPin on HIGH state
    for 10 micro seconds
digitalWrite(trigPin, LOW);
```

```
duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns
    the sound wave travel time in microseconds
//distance= duration*0.034/2;
distance = (duration/2)/29.1;     //in cm,  datasheet gives "
    duration/58" as the formula

//To avoid sending data at variable time intervals due to varying
    time duration taken between execution of above code inside
    this function depending on distance of obstacle
//if no object, echo pulse is 38ms long HIGH
while(micros()-T1<38000)
{
;
}

return distance;
}
```

Appendix B : *Processing* Code

The *Java* code that we ran in *Processing* is given below :

Listing B.1: The Processing Code

```
/*   Arduino Radar Project
*    1.1 [Final and complete]
*   Initialise every variable to null value to avoid null pointer
    exception
*/

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
// defines variables
String angle="";
String distance="";
String data="";
String noObject="";
float pixsDistance=0.0;
int iAngle=0;
float iDistance=0.0;
int index1=0;
int index2=0;
int objCount=0,obctr=0;
PFont orcFont;
float angFlag=1.0;
float prevAng=0.0,deltaAng=0.0;
float maxDIST=50.0; //max distance of object in cms
int wctr=0;
float objWidth=0.0,prevWidth=0.0;
void setup() {
```

```
size (600, 700); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
smooth();

myPort = new Serial(this,"/dev/ttyACM0", 9600); // Enter the COM
    Port address as COM4 or COM 22.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("CenturySchL-Ital-20.vlw");

}

void draw() {

fill(237,13,245);
textFont(orcFont);
// simulating motion blur and slow fade of the moving line
noStroke();
fill(184,13);
rect(0, 0, width, height-height*0.065);

fill(211,27,228); //  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".
try{
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 = float(distance);
```

```
deltaAng=iAngle-prevAng;

if(deltaAng*angFlag<0.0){
objCount=0;
objWidth=0;
}
//anticlockwise--deltaAng>0
//clockwise--deltaAng<0
if(deltaAng>0.0)
angFlag=1.0;
else
angFlag=-1.0;
}
catch(Exception e){
println("Error parsing:");
e.printStackTrace();
}
}

void drawRadar() {
pushMatrix();
translate(width/2,height-height*0.507); // moves the starting
    coordinates to new location
noFill();
strokeWeight(2);
stroke(407,409,305);
// draws the arc lines
arc(0,0,(width-width*0.0385),(width-width*0.0385),PI,TWO_PI);
arc(0,0,(width-width*0.26),(width-width*0.26),PI,TWO_PI);
arc(0,0,(width-width*0.498),(width-width*0.498),PI,TWO_PI);
arc(0,0,(width-width*0.754),(width-width*0.754),PI,TWO_PI);
arc(0,0,(width-width*0.0385),(width-width*0.0385),0,PI);
arc(0,0,(width-width*0.26),(width-width*0.26),0,PI);
arc(0,0,(width-width*0.498),(width-width*0.498),0,PI);
arc(0,0,(width-width*0.754),(width-width*0.754),0,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(0,0,(-width/2)*cos(radians(180)),(-width/2)*sin(radians(180))
    );
line(0,0,(-width/2)*cos(radians(210)),(-width/2)*sin(radians(210))
    );
line(0,0,(-width/2)*cos(radians(240)),(-width/2)*sin(radians(240))
    );
```

```
line(0,0,(-width/2)*cos(radians(270)),(-width/2)*sin(radians(270))
    );
line(0,0,(-width/2)*cos(radians(300)),(-width/2)*sin(radians(300))
    );
line(0,0,(-width/2)*cos(radians(330)),(-width/2)*sin(radians(330))
    );
line((-width/2)*cos(radians(30)),0,width/2,0);
popMatrix();
}

void drawLine() {
pushMatrix();
strokeWeight(8);
stroke(32,65,174); //color for blue line
translate(width/2,height-height*0.507); // moves the starting
    coordinates to new location
line(0,0,(height-height*0.575)*cos(radians(iAngle)),-(height-
    height*0.575)*sin(radians(iAngle))); // draws the line
    according to the angle
popMatrix();
}

void drawObject() {
pushMatrix();
translate(width/2,height-height*0.508); // moves the starting
    coordinats to new location
strokeWeight(8);
stroke(240,1,40); // red color
pixsDistance = iDistance*((height-height*0.1666)*0.013/3); //
    covers the distance from the sensor from cm to pixels
if(iDistance<=maxDIST){
if(wctr>2){    //Assuming that an object will be thick enough to
    be detected for 2 degrees of rotation.
objWidth=0.0;
// 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)));
obctr++;
}
}
popMatrix();
}

void drawText() { // draws the texts on the screen
pushMatrix();
if(iDistance>maxDIST) {
noObject = "No object within Range";
if(wctr==0)
objWidth=prevWidth;
```

```
else
objWidth=float(wctr)*iDistance*0.0174;  //width of object in cm
prevWidth=objWidth;
wctr=0;
if(obctr>0){
objCount++;
obctr=0;
}
}
else {
wctr++;
if(wctr>2)  //Assuming that an object will be thick enough to be
    detected for 2 degrees of rotation.
{
noObject = "Object in Range";

}
}

fill(0,0,0);  //black background of bottom text
noStroke();
rect(0, height-height*0.0521, width, height);
fill(251,255,249);
textSize(15);

text("30cm",width-width*0.4041,height-height*0.4793);
text("60cm",width-width*0.281,height-height*0.4792);
text("90cm",width-width*0.177,height-height*0.4792);
text("120cm",width-width*0.0729,height-height*0.4792);
textSize(16);
text(noObject, width-width*0.634, height-height*0.0218);
text("Angle: " + iAngle +" °", width-width*0.97, height-height
    *0.0232);
text("Distance: ", width-width*0.26, height-height*0.0235);
textSize(16);
text(" No. of objects: "+ objCount +"", width-width*0.986, height-
    height*0.0714);
text(" Object Width: "+ objWidth +"", width-width*0.986, height-
    height*0.1714);
if(iDistance<=maxDIST) {
if(wctr>2)
text("        " + iDistance + " cm", width-width*0.185, height-
    height*0.0237);
}
textSize(19);
fill(7,7,6); //color for degrees text
translate((width-width*0.5020)+width/2*cos(radians(30)),(height-
    height*0.5283)-width/2*sin(radians(30)));
rotate(-radians(-60));
text("30°",0,0);
```

```
resetMatrix();
translate((width-width*0.507)+width/2*cos(radians(60)),(height-
    height*0.5139)-width/2*sin(radians(60)));
rotate(-radians(-29));
text("60°",-4,-5);
resetMatrix();
translate((width-width*0.507)+width/2*cos(radians(90)),(height-
    height*0.5149)-width/2*sin(radians(90)));
rotate(radians(0));
text("90\degree",-5,-2);
resetMatrix();
translate(width-width*0.513+width/2*cos(radians(120)),(height-
    height*0.51286)-width/2*sin(radians(120)));
rotate(radians(-30));
text("120°",0,0);
resetMatrix();
translate((width-width*0.5298)+width/2*cos(radians(150)),(height-
    height*0.4803)-width/2*sin(radians(150)));
rotate(radians(-60));
text("150°",0,0);
resetMatrix();
translate(width-width*0.475+width/2*cos(radians(180)),(height-
    height*0.47791)-width/2*sin(radians(180)));
rotate(radians(-90));
text("180°",0,0);
resetMatrix();
translate(width-width*0.494+width/2*cos(radians(210)),(height-
    height*0.4797)-width/2*sin(radians(210)));
rotate(radians(-120));
text("210°",0,0);
resetMatrix();
translate(width-width*0.484+width/2*cos(radians(240)),(height-
    height*0.4867)-width/2*sin(radians(240)));
rotate(radians(-150));
text("240°",0,0);
resetMatrix();
translate(width-width*0.474+width/2*cos(radians(270)),(height-
    height*0.50151)-width/2*sin(radians(270)));
rotate(radians(-180));
text("270°",0,0);
resetMatrix();
translate(width-width*0.470+width/2*cos(radians(300)),(height-
    height*0.51167)-width/2*sin(radians(300)));
rotate(radians(-210));
text("300°",0,0);
resetMatrix();
translate(width-width*0.478+width/2*cos(radians(330)),(height-
    height*0.5174)-width/2*sin(radians(330)));
rotate(radians(-240));
text("330°",0,0);
```

```
resetMatrix();
translate(width-width*0.523+width/2*cos(radians(360)),(height-
    height*0.52132)-width/2*sin(radians(360)));
rotate(radians(-270));
text("0°",0,0);
popMatrix();
prevAng = iAngle;
}
```