ARDUINO RADAR

A
Report Submitted
In Partial Fulfilments of the Requirements
For the Degree of

BACHELOR OF TECHNOLOGY
in
Electronics & Communication

by
Chanchal Jaiswal(1130434016)
Sana Bushra(1130434038)
Anusha Srivastava(1130434010)
Anjali Shukla(1130434008)

Under the supervision of
Raj Kumar Maddheshiya
Assistant Professor
BBD University, Lucknow

School Of Engineering
BABU BANARASI DAS UNIVERSITY
LUCKNOW
September,2016

ABSTRACT

RADAR is an object detection system which uses radio waves to determine the range, altitude, direction, or speed of objects. The radar dish or antenna transmits pulses of radio waves or microwaves which bounce off any object in their path. Arduino is a single-board microcontroller to make using electronics in multidisciplinary projects more accessible. This project aims at making a RADAR that is efficient, cheaper and reflects all the possible techniques that a radar consists of.

For some time, I have been interested in making some sort of robot based on the Arduino platform. This project is the first phase of this longer-term hobby project. The goal is to develop an object detection system which could be retrofitted to a semi-autonomous robot up the road.

The system leverages an ultrasonic range finder to detect obstacles located in proximity to the apparatus. The ultrasonic sensor is mounted on a sub micro servo in order to have a sweeping perspective. A Passive Infrared (PIR) sensor is also used to activate the SONAR when motion is detected.
The apparatus sends sensor readings and distance measurements over 802.15.4 to a computer for graphical display. Code running on the microcontroller generates audible alerts using a serial-driven voice synthesizer chip attached to a small speaker.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

RADAR is an object detection system which uses radio waves to determine the range, altitude, direction, or speed of objects. Radar systems come in a variety of sizes and have different performance specifications. Some radar systems are used for air-traffic control at airports and others are used for long range surveillance and early-warning systems. A radar system is the heart of a missile guidance system. Small portable radar systems that can be maintained and operated by one person are available as well as systems that occupy several large rooms.

Radar was secretly developed by several nations before and during the World War II. The term RADAR itself, not the actual development, was coined in 1940 by United States Navy as an acronym for Radio Detection and Ranging.

The modern uses of radar are highly diverse, including air traffic control, radar, astronomy, air-defense systems, antimissile systems, antimissile systems; marine radars to locate landmarks and other ships; aircraft anti-collision systems; ocean surveillance systems, outer space surveillance and rendezvous systems; meteorological precipitation monitoring; altimetry and flight control precipitation monitoring; altimetry and flight control systems; guided missile target locating systems; and ground-penetrating radar for geological observations. High tech radar systems are associated with digital signal processing and are capable of extracting useful information from very high noise levels.

II. LITERATURE SURVEY

A. "The Idea"

Army, Navy and the Air Force make use of this technology. The use of such technology has been seen recently in the self parking car systems launched by AUDI, FORD etc. And even the upcoming driverless cars by Google like Prius and Lexus.

This setup can be used in any systems the customer may

want to use like in a car, a bicycle or anything else. The use of Arduino in this provides even more flexibility of usage of the above-said module according to the requirements.
The idea of making an RADAR came as a part of a study carried out on the working and mechanism of "Automobiles of Future". Hence this time I was able to get a hold of one of the Arduino boards, Arduino UNO R3. So, knowing about the power and vast processing capabilities of the Arduino, I thought of making it big and a day to day application specific module that can be used and configured easily at any place and by anyone.
Moreover, in this fast moving world there is an immense need for the tools that can be used for the betterment of the mankind rather than devastating their lives.
Hence, from the idea of the self driving cars came the idea of self parking cars. The main problem of the people in the world is safety while driving. So, this gave up a solution to that by making use of this project to continuously scan the area for traffic, population etc. and as well as protection of the vehicles at the same time to prevent accidents or minor scratches to the vehicles.

III. COMPONENTS REQUIRED

A. Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital Input /Output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16MHz ceramic resonator, USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 programmed as a USB-to-serial converter.
Changes in Uno R3:

1. Pin out: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the reset pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5v and with the Arduino due that operates with 3.3v.
2. Stronger RESET circuit.
3. ATmega16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino Boards.

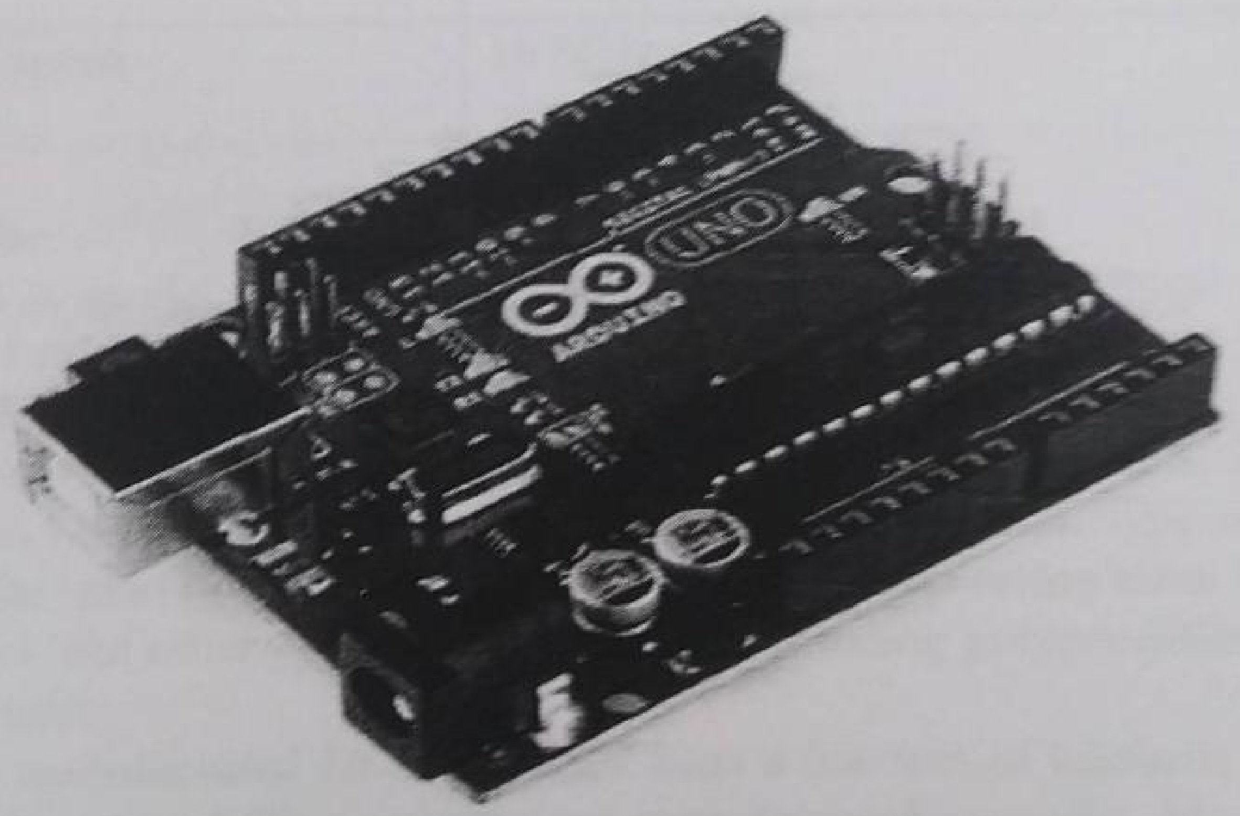

Figure 1.1 Arduino Board

TABLE I

ARDUINO UNO SPECIFICATIONS

Microcontroller	ATmega328
[1] Operating Voltage	5V
[2] Inout Voltage (Recommended)	7V-12V
[3] Input Voltage (Limits)	6V-20V
[4] Digital I/O Pins	14 (of which 6 provide PWM output)
[5] Analog Input Pins	6
[6] DC Current per I/O Pin	40 mA
[7] DC Current for 3.3V Pin	50 mA
[8] Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
[9] SRAM	2 KB (ATmega328)
[10] EEPROM	1 KB (ATmega328)
[11] Clock Speed	16 MHz

B. Crystal Oscillator

A crystal oscillator is an electronic oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise frequency. This frequency is commonly used to keep track of time (as in quartz wristwatches), to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The most common type of piezoelectric resonator used is the quartz crystal, so oscillator circuits incorporating them became known as crystal oscillators, but other piezoelectric materials including polycrystalline ceramics are used in similar circuits.

Quartz crystals are manufactured for frequencies from a few tens of kilohertz to hundreds of megahertz. More than two billion crystals are manufactured annually. Most are used for consumer devices such as wristwatches, clocks, radios, computers, and cell phones. Quartz crystals are also found inside test and measurement equipment, such as counters, signal generators, and oscilloscopes.

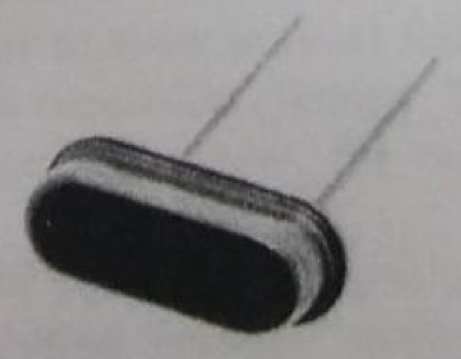

Figure1.2 Crystal Oscillator(16 MHz)

C. Servo Motor

A servomotor is a rotary actuator that allows for precise control of angular 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 different class of motor, on the basis of fundamental operating principle, but uses servomechanism to achieve closed loop control with a generic open loop motor.

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

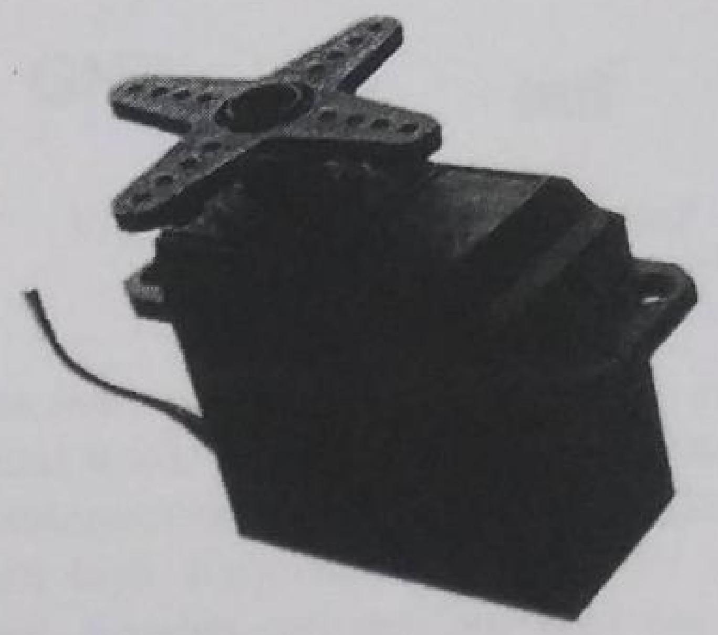

Figure 1.3 Servo Motor

TABLE II

DATASHEET FOR SG-90 MODULE OF SERVO MOTOR

Weight	**9g**
Dimension	**22.2 x 11.1 x 31 mm approx..**
Stall torque	**1.8 kgf.cm**
Operating speed	**0.1s/60 degree**
Operating voltage	**4.8 V (~5V)**
Dead bandwidth	**10 micro seconds**
Temperature range	**(0-55) degree celsius**

D. Voltage Regulator

A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level.

With the exception of shunt regulators, all modern electronic voltage regulators operate by comparing the actual output voltage to some internal fixed reference voltage. Any difference is amplified and used to control the regulation element. This forms a negative feedback servo control loop. If the output voltage is too low, the regulation element is commanded to produce a higher voltage

The 78XX series of three-terminal positive regulator are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe

operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current.
Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

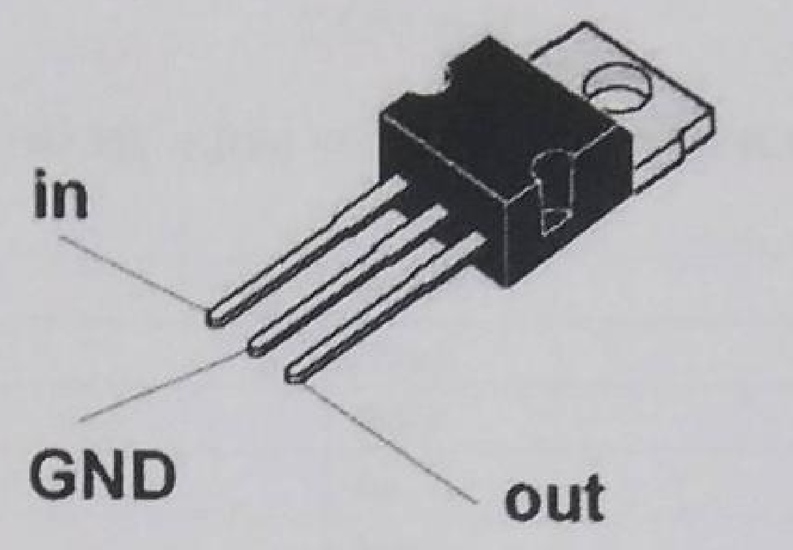

Figure 1.4 Voltage Regulator

E. Ultrasonic Sensor

Ultrasonic sensors (also known as transceivers when they both send and receive, but more generally called transducers) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.
This technology can be used for measuring wind speed and direction (anemometer), tank or channel level, and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water.
To measure tank or channel level, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultra sonography, burglar alarms and non-destructive testing.
Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed.

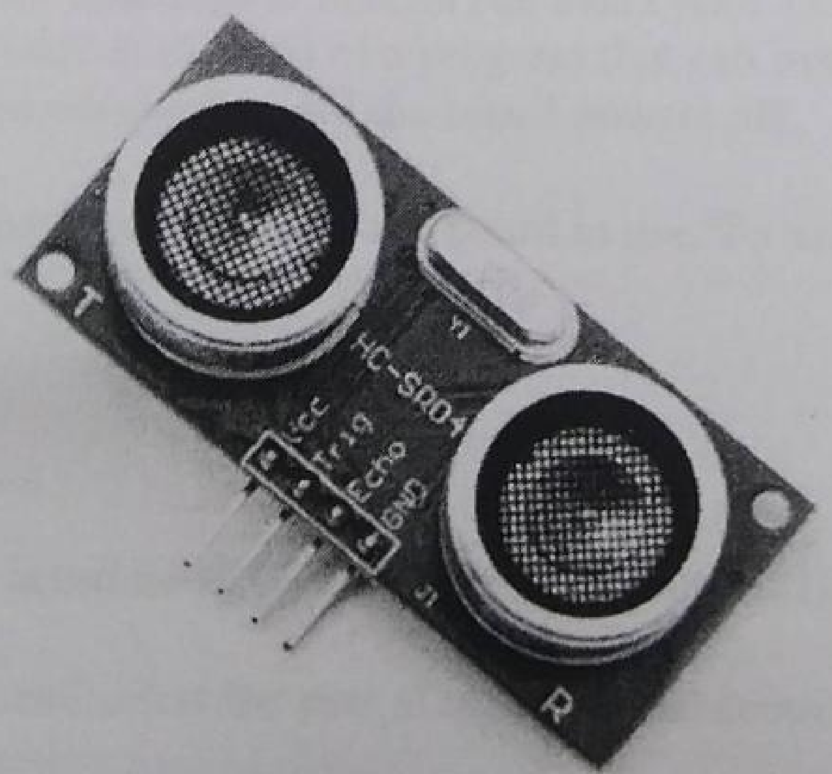

Figure 1.5 Ultrasonic Sensor

TABLE III

DATASHEET FOR HC-SR04 MODULE OF ULTRASONIC SENSOR

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

IV. USING ARDUINO SOFTWARE

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".
Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier.
Users only need define two functions to make a run able cyclic executive program:
• Setup(): a function run once at the start of a program that can initialize settings
• Loop(): a function called repeatedly until the board powers off.

Open the Arduino IDE software and select the board in use. To select the board:

• Go to Tools.

• Select Board.

• Under board, select the board being used, in this case Arduino Uno.

• Go to Tools and to Port and select the port at which the Arduino board is connected.

• Write the code in the space provided and click on compile. Once the code is compiled, click on upload to upload the sketch to the Arduino board.

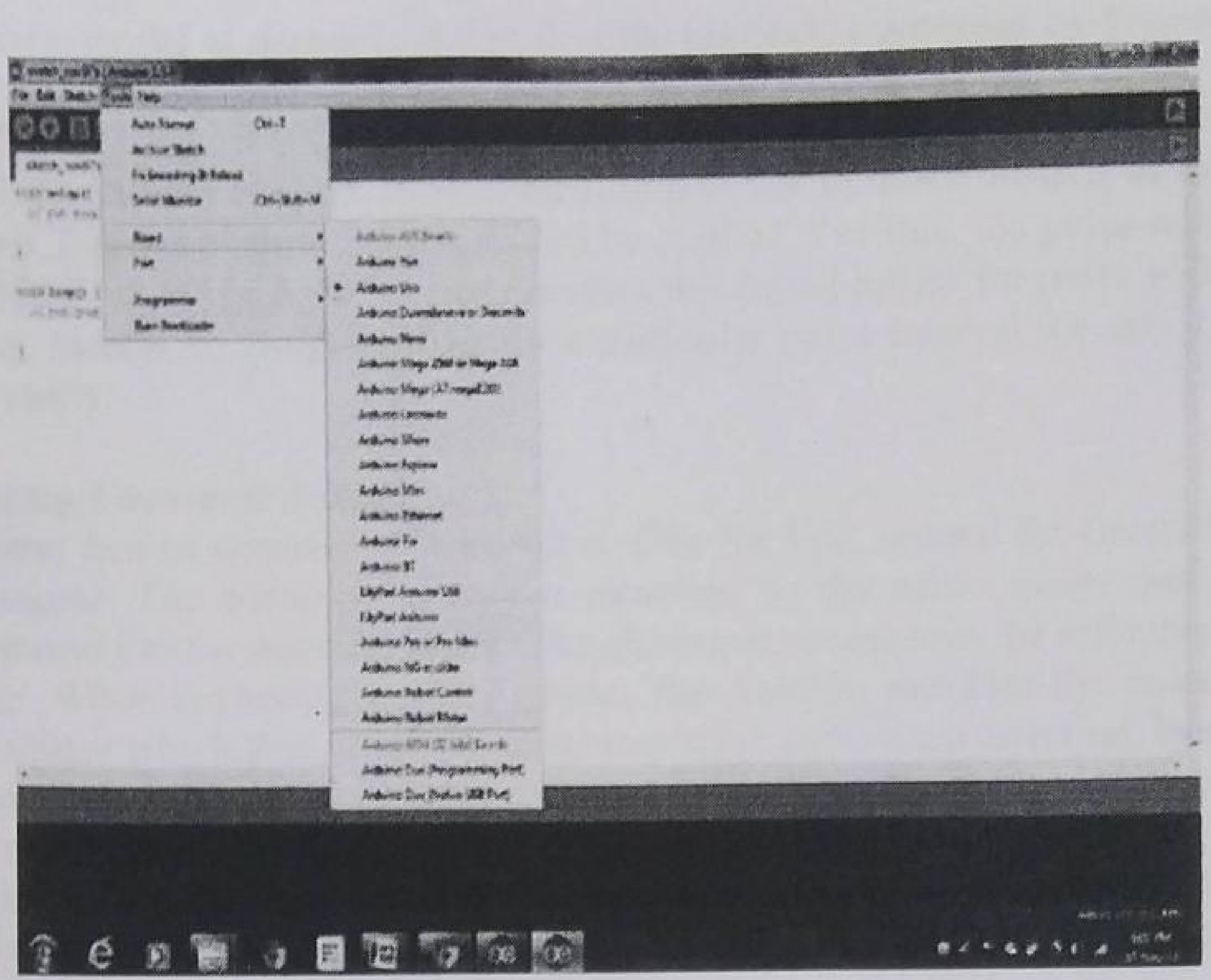

Figure 1.6 Arduino IDE

V. PRACTICAL IMPLEMENTATION

A. Connecting Servo Motor

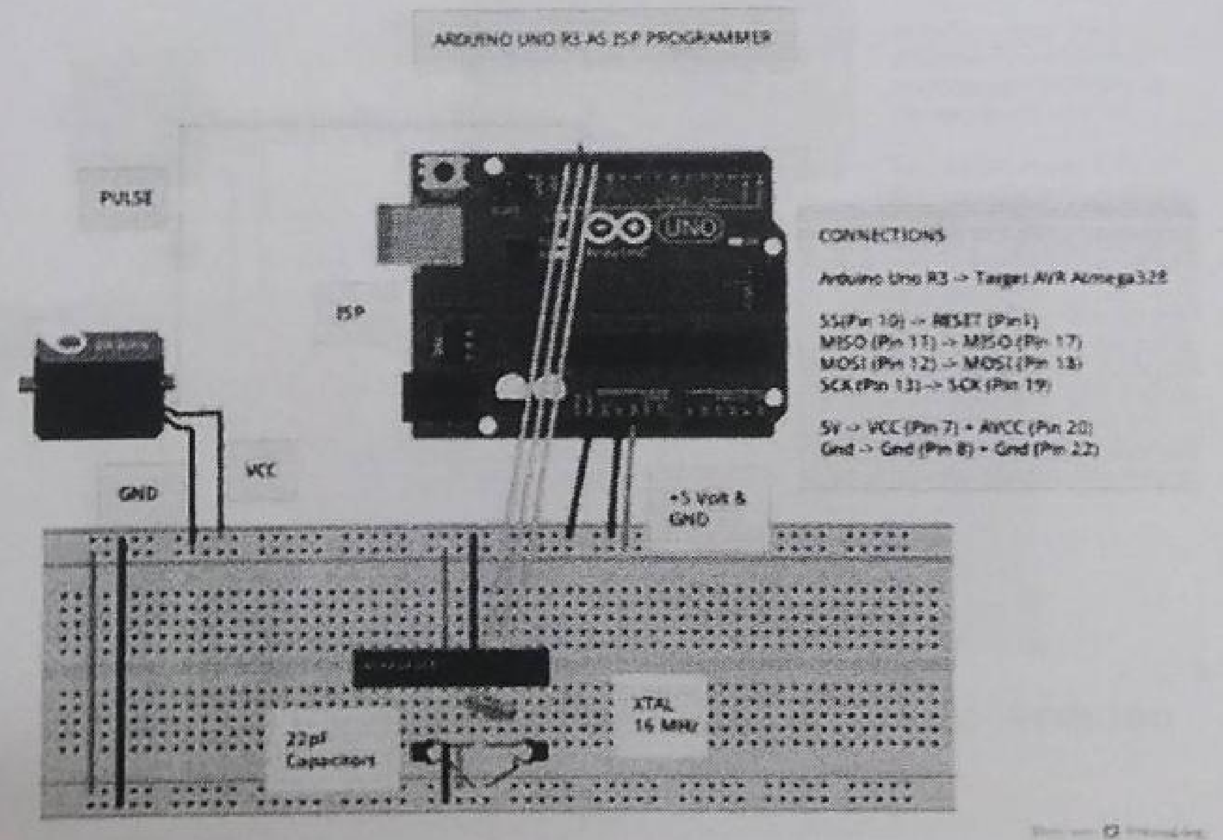
ARDUINO UNO R3 AS ISP PROGRAMMER
PULSE
ISP
VCC
GND
+5 Volt & GND
22pF Capacitors
XTAL 16 MHz
CONNECTIONS
Arduino Uno R3 -> Target AVR Atmega328
SS(Pin 10) -> RESET (Pin1)
MISO (Pin 11) -> MISO (Pin 17)
MOSI (Pin 12) -> MOSI (Pin 18)
SCK (Pin 13) -> SCK (Pin 19)
5V -> VCC (Pin 7) + AVCC (Pin 20)
Gnd -> Gnd (Pin 8) + Gnd (Pin 22)

Figure 1.7 Connecting Servo Motor

A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration.
A normal servo motor has three terminals:
1.VCC
2. GND
3. PULSE

A servo motor works at normally 4.8 to 6 volts. Ground is provided by connecting it to the Ground of the Arduino. The total time for a servo motor pulse is usually 20ms. To move it to one end of say 0 degree angle, a 1ms pulse is used and to move it to other end i.e 180 degrees, a 2ms pulse is applied. Hence, according to this to move the axis of the servo motor to the center, a pulse of time 1.5 ms should be applied. For this, the pulse wire of the servo motor is connected to the Arduino that provides the digital pulses for pulse width modulation of the pulse. Hence, by programming for a particular pulse interval the servo motor can be controlled easily.

B. Connecting Ultrasonic Sensor

An Ultrasonic Sensor consists of three wires. One for Vcc, second for Ground and the third for pulse signal. The ultrasonic sensor is mounted on the servo motor and both of them further connected to the Arduino board. The ultrasonic sensor uses the reflection principle for its working. When connected to the Arduino, the Arduino provides the pulse signal to the ultrasonic sensor which then sends the ultrasonic wave in forward direction. Hence, whenever there is any obstacle detected or present in front, it reflects the waves which are received by the ultrasonic sensor.

If detected, the signal is sent to the Arduino and hence to the PC/laptop to the processing software that shows the presence of the obstacle on the rotating RADAR screen with distance and the angle at which it has been detected.

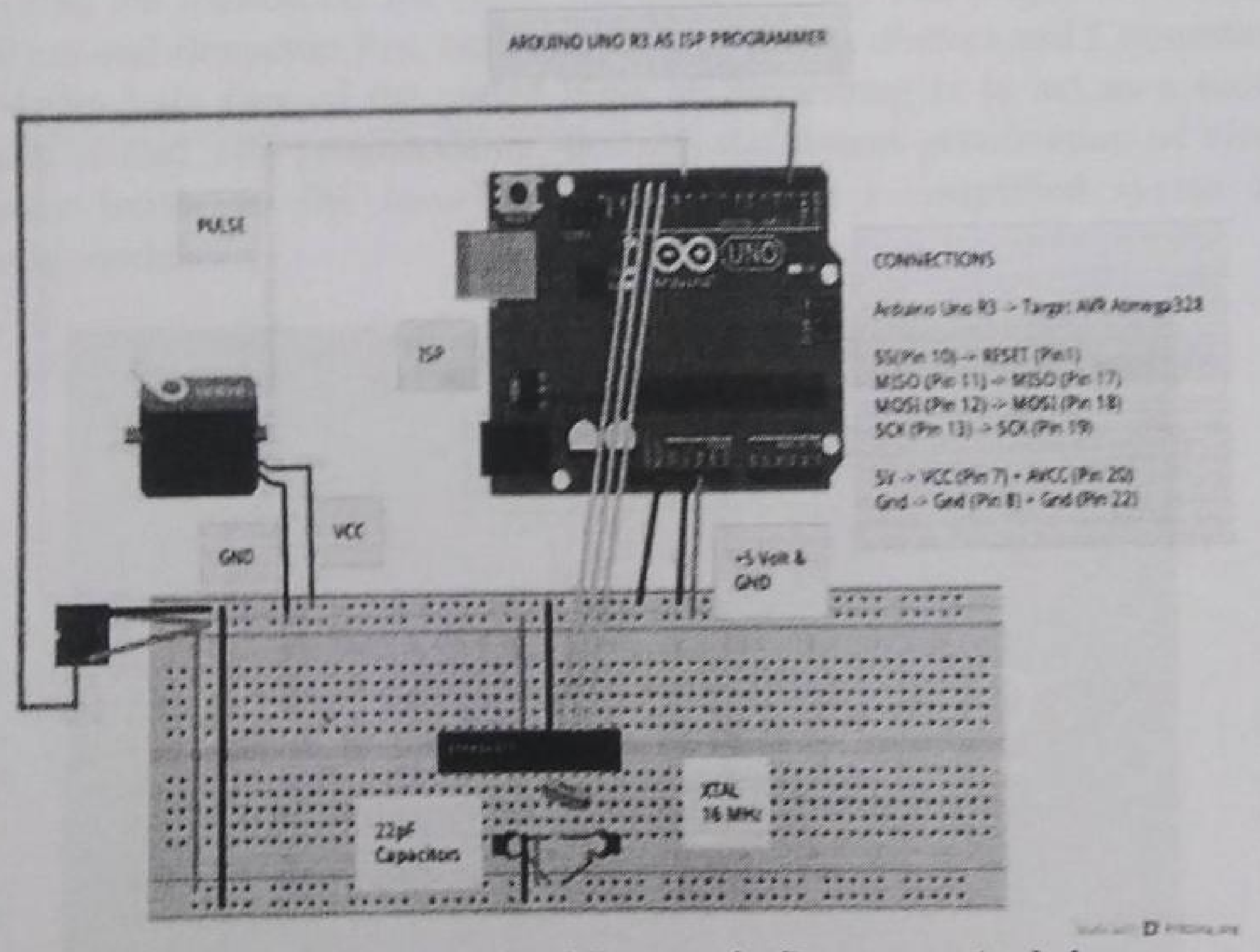
ARDUINO UNO R3 AS ISP PROGRAMMER
PULSE
ISP
VCC
GND
+5 Volt & GND
CONNECTIONS
Arduino Uno R3 -> Target AVR Atmega328
SS(Pin 10) -> RESET (Pin1)
MISO (Pin 11) -> MISO (Pin 17)
MOSI (Pin 12) -> MOSI (Pin 18)
SCK (Pin 13) -> SCK (Pin 19)
5V -> VCC (Pin 7) + AVCC (Pin 20)
Gnd -> Gnd (Pin 8) + Gnd (Pin 22)
XTAL 16 MHz
22pF Capacitors

Figure 1.8 Connecting Ultrasonic Sensor to Arduino

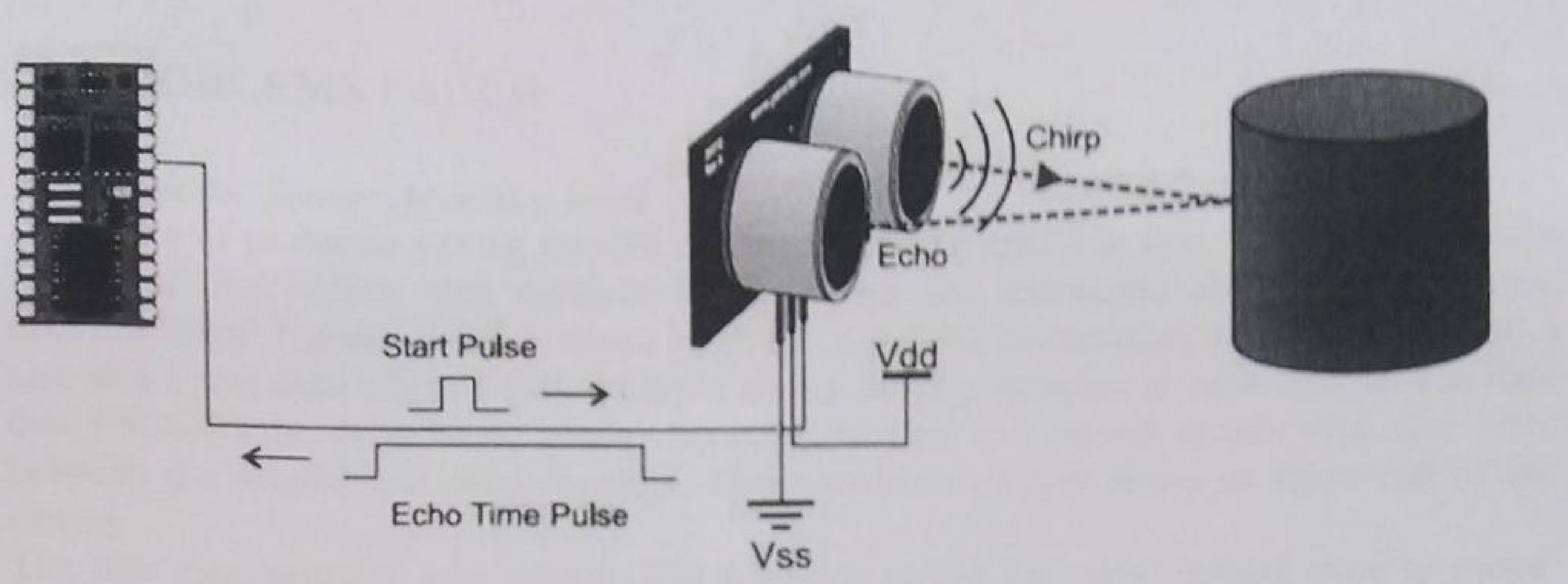

Figure 1.9 Working of Ultrasonic Sensor

VI. USING PROCESSING SOFTWARE

Processing is an open source programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks. The project was initiated in 2001 by Casey Reas and Benjamin Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab. One of the stated aims of Processing is to act as a tool to get non-programmers started with programming, through the instant gratification of visual feedback. The language builds on the Java language, but uses a simplified syntax and graphics programming models.

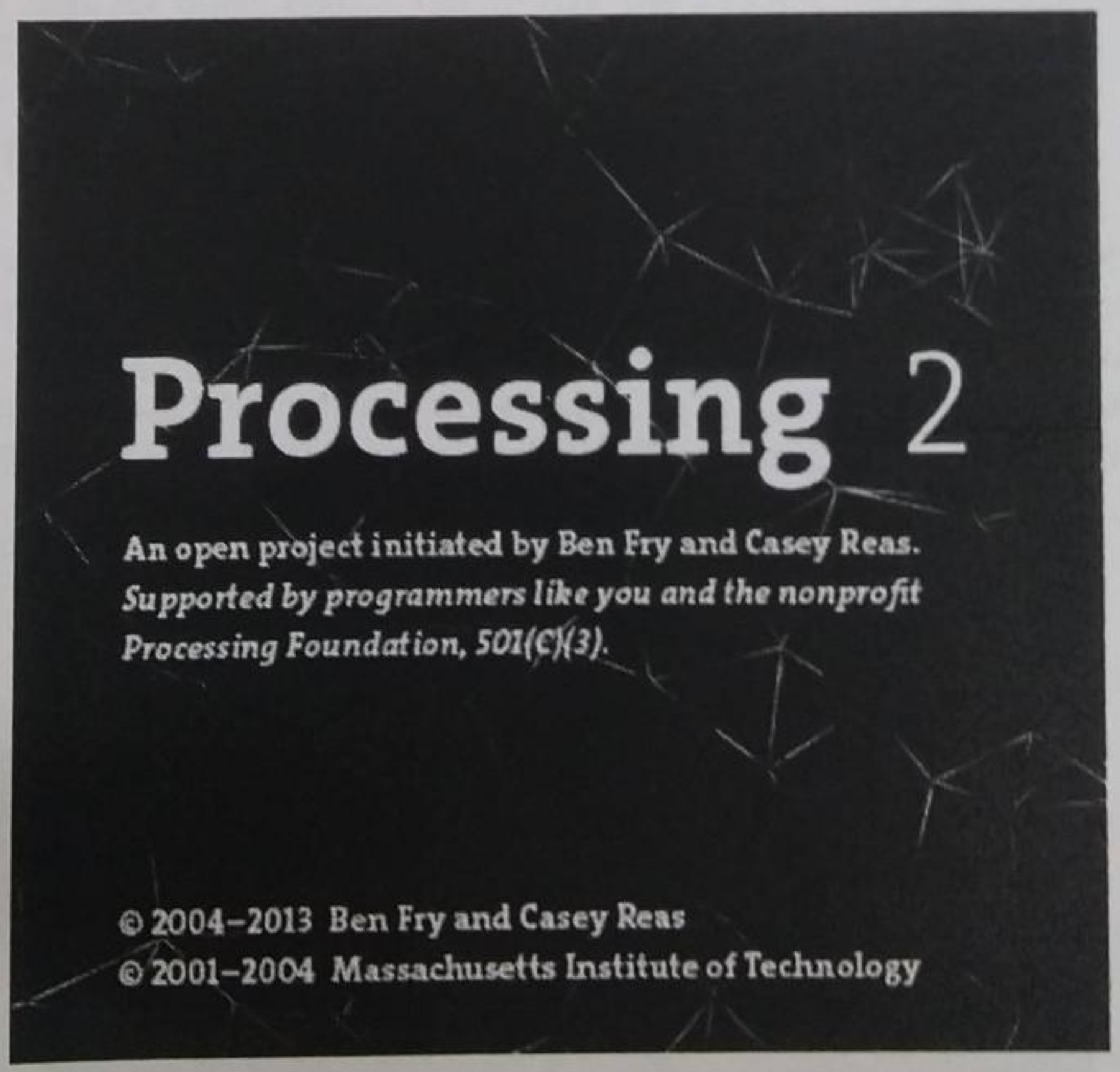
Processing 2
An open project initiated by Ben Fry and Casey Reas.
Supported by programmers like you and the nonprofit Processing Foundation, 501(C)(3).
© 2004–2013 Ben Fry and Casey Reas
© 2001–2004 Massachusetts Institute of Technology

Figure 1.10 Processing 2.0 Software

VII. PROBLEMS FACED

A. *Ultrasonic Sensor Accuracy Issue*

I had a lot of problems getting reliable readings from the sensor at first. I put together some functional tests which took multiple sonar sweeps and calculated absolute value deltas between them. I dumped these sensor readings and delta calculations into a spreadsheet. I also had a test case where I took multiple sensor ranging samples at each step. It was then that I noticed the deltas were greater between the first and second ranges than they were between the second and third readings. There were also larger deltas at either end of the sweep.

The data demonstrated very plainly that I wasn't giving the servo enough time to move. Increasing the delay in between readings (to allow the servo more time to step) and adding a delay at the beginning and end of the sweep helped to have much more reliable results. I also noticed during these tests that decreasing the analog gain sensitivity seemed to decrease anomalous readings between passes.

B. *Communicating with Arduino through PC*

Another major problem related to the Arduino board was the communication with it from PC. Since, there is a requirement of an RS-232 to TTL conversion for the communication, so try some methods:

[1] Firstly I used the MAX-232 IC to communicate with the Arduino as with the 8051 but due to large voltage drop and mismatch in the speed, it failed to communicate.

[2] Next. I tried to use a dedicated AVR as USB to Serial converter as in the original Arduino board, the difference being DIP AVR used by us instead of the SMD Mega16U2 controller. But, unfortunately I was unable to communicate through it.

[3] At last I had no other choice but to use the FTDI FT-232R chip for USB to Serial conversion. Finally IT WORKED!!!

VIII. APPLICATIONS

The idea of making an Ultrasonic RADAR appeared to us while viewing the technology used in defense, be it Army, Navy or Air Force and now even used in the automobiles employing features like automatic/driverless parking systems, accident prevention during driving etc. The applications of such have been seen recently in the self parking car systems launched by AUDI, FORD etc. And even the upcoming driverless cars by Google like Prius and Lexus.

Figure 1.11 Driverless Car by Google

A. Air Force

In aviation, aircraft are equipped with radar devices that warn of aircraft or other obstacles in or approaching their path, display weather information, and give accurate altitude readings. The first commercial device fitted to aircraft was a 1938 Bell Lab unit on some United Air Lines aircraft. Such aircraft can land in fog at airports equipped with radar-assisted ground-controlled approach systems in which the plane's flight is observed on radar screens while operators radio landing directions to the pilot.

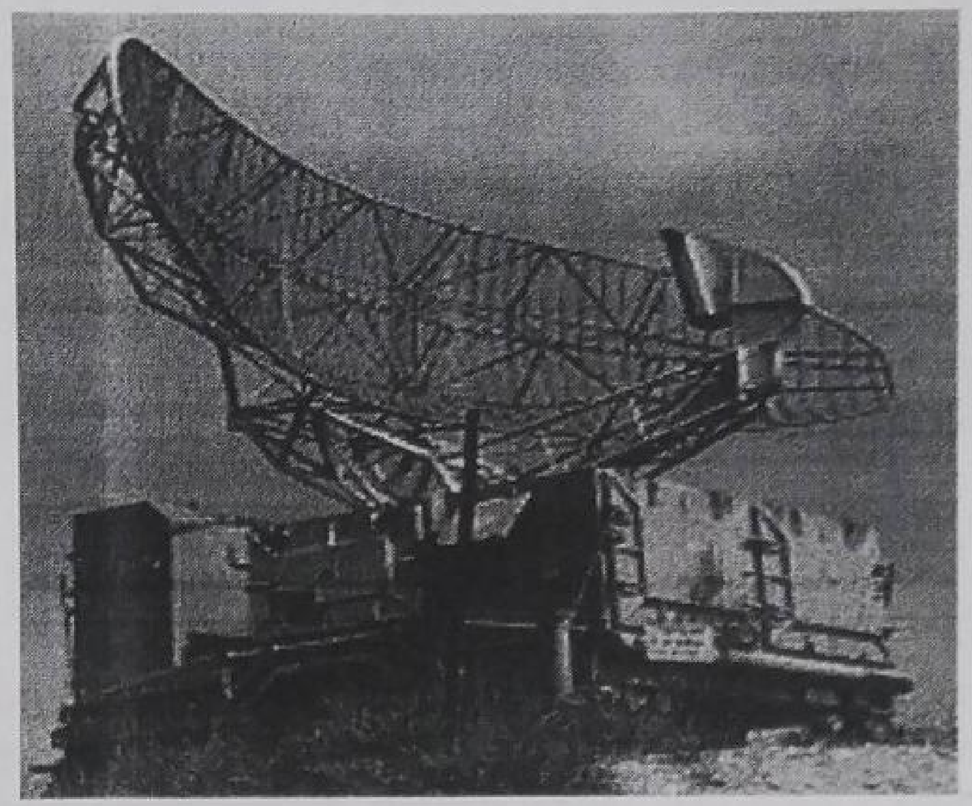

Figure 1.12 RADAR in Air Force

B. Naval Applications

Marine radars are used to measure the bearing and distance of ships to prevent collision with other ships, to navigate, and to fix their position at sea when within range of shore or other fixed references such as islands, buoys, and lightships. In port or in harbor, vessel traffic service radar systems are used to monitor and regulate ship movements in busy waters.

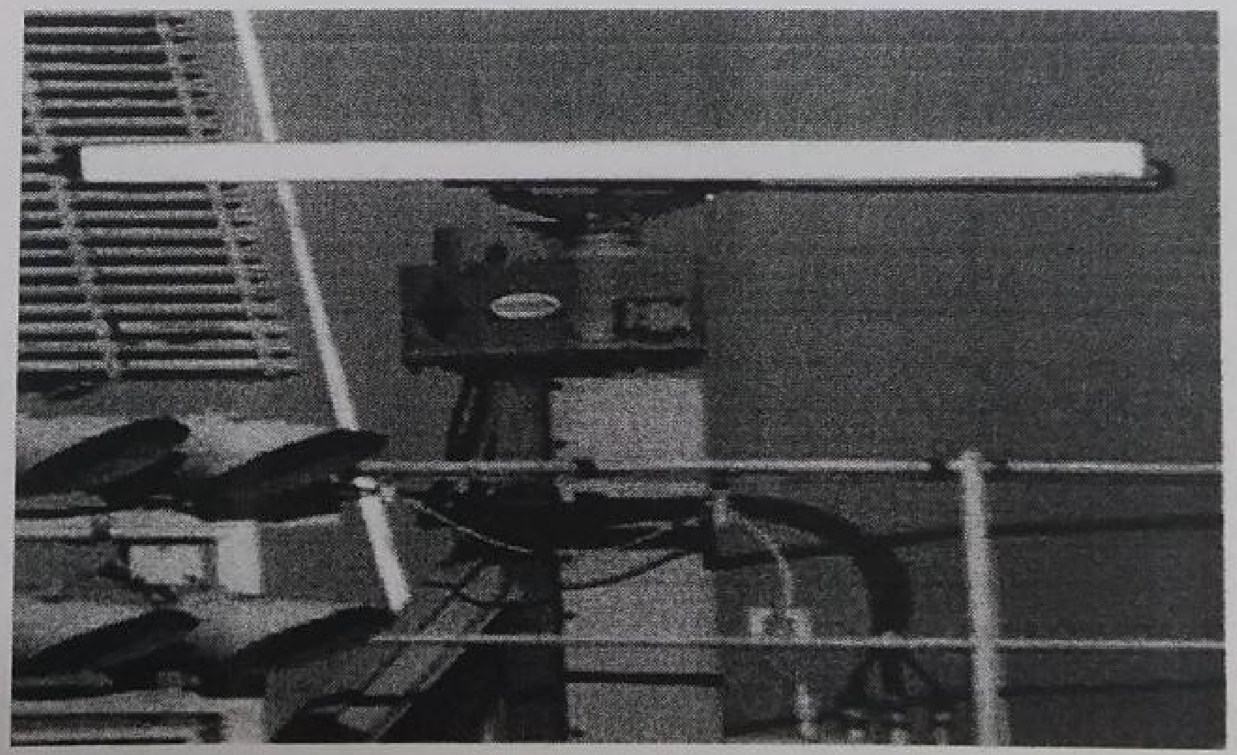

Figure 1.13 RADAR in Navy

C. Applications in Army

Scanned by TapScanner

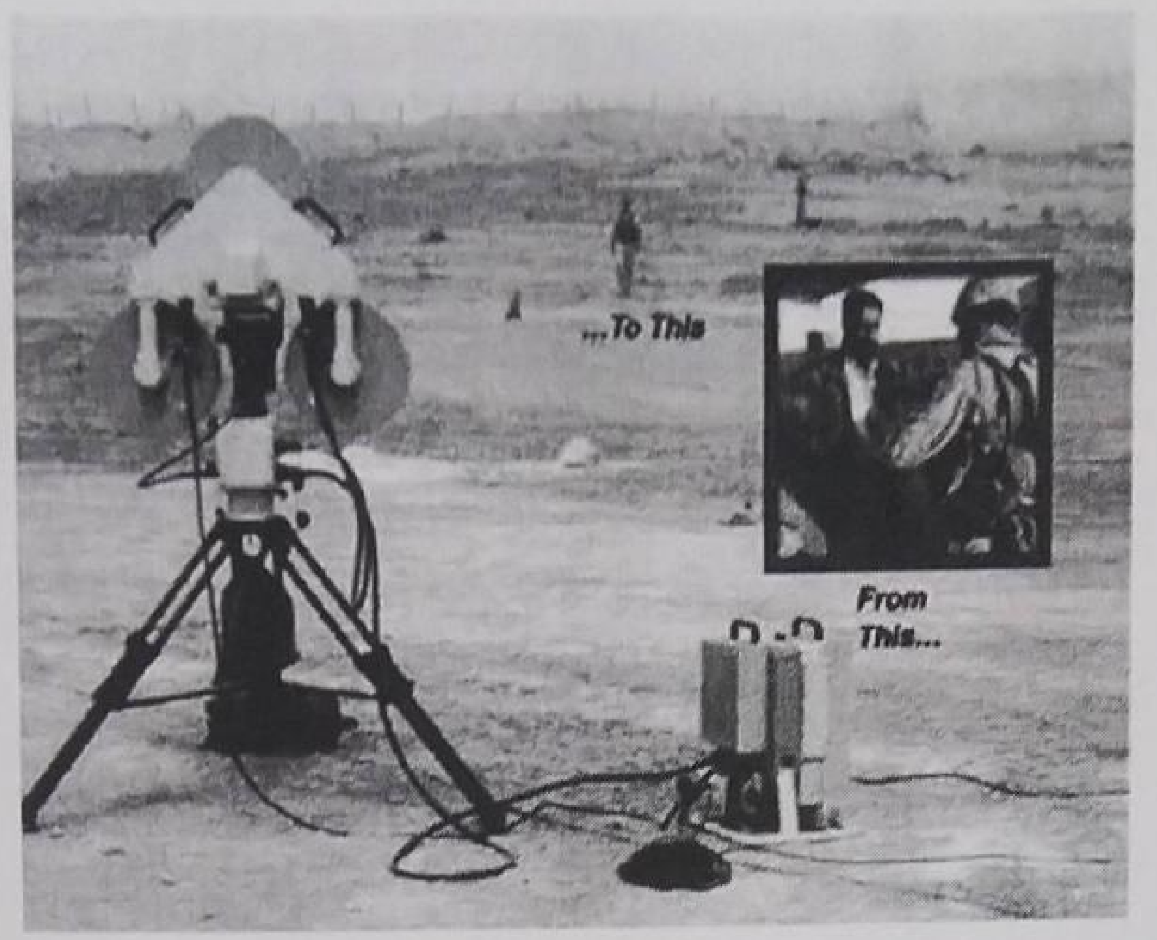

Figure 1.14 RADAR in Army

Two video cameras automatically detect and track individuals walking anywhere near the system, within the range of a soccer field. Low-level radar beams are aimed at them and then reflected back to a computer, which analyzes the signals in a series of algorithms. It does this by comparing the radar return signal (which emits less than a cell phone) to an extensive library of "normal responses."
Those responses are modeled after people of all different shapes and sizes (SET got around to adding females in 2009). It then compares the signal to another set of "anomalous responses" – any anomaly, and horns go off.
Literally, when the computer detects a threat, it shows a red symbol and sounds a horn. No threat and the symbol turns green, greeting the operators with a pleasant piano riff.

IX. A FINAL LOOK

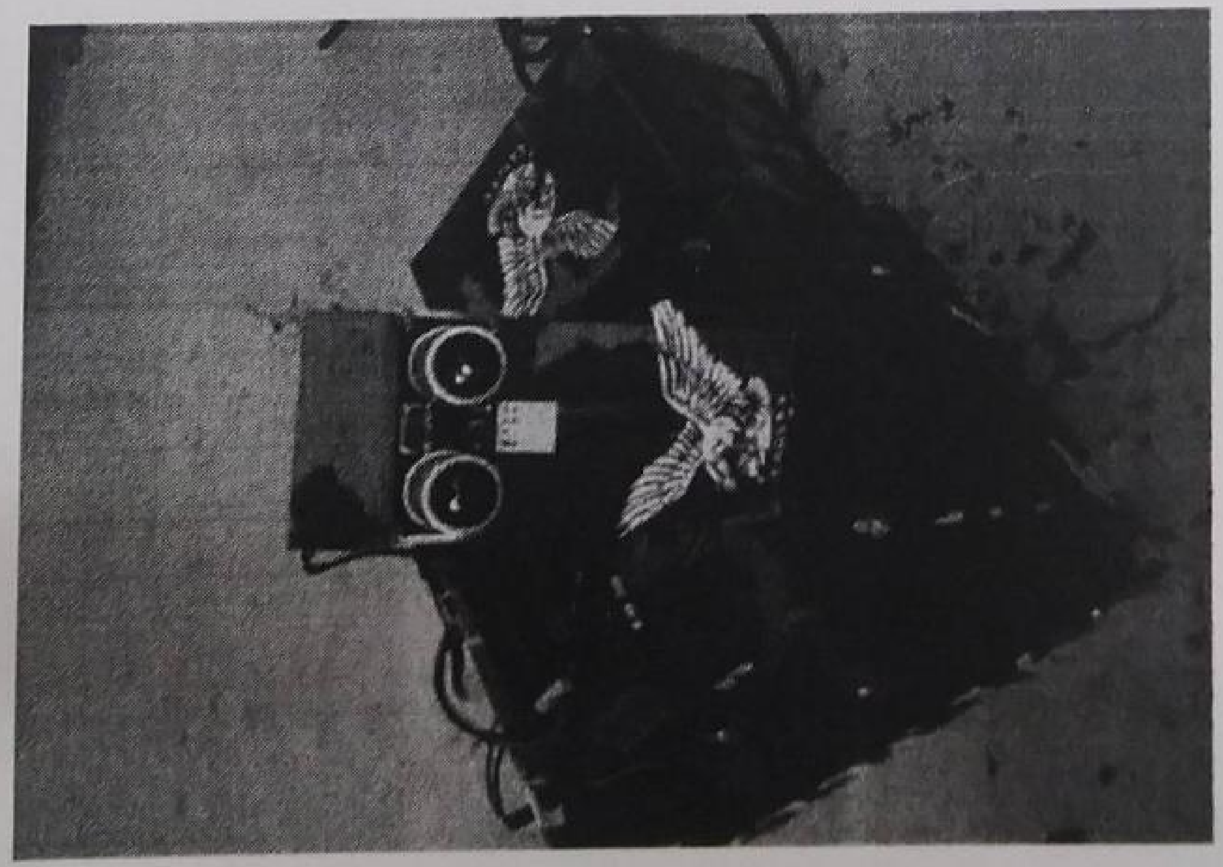

Figure 1.15 Final Project

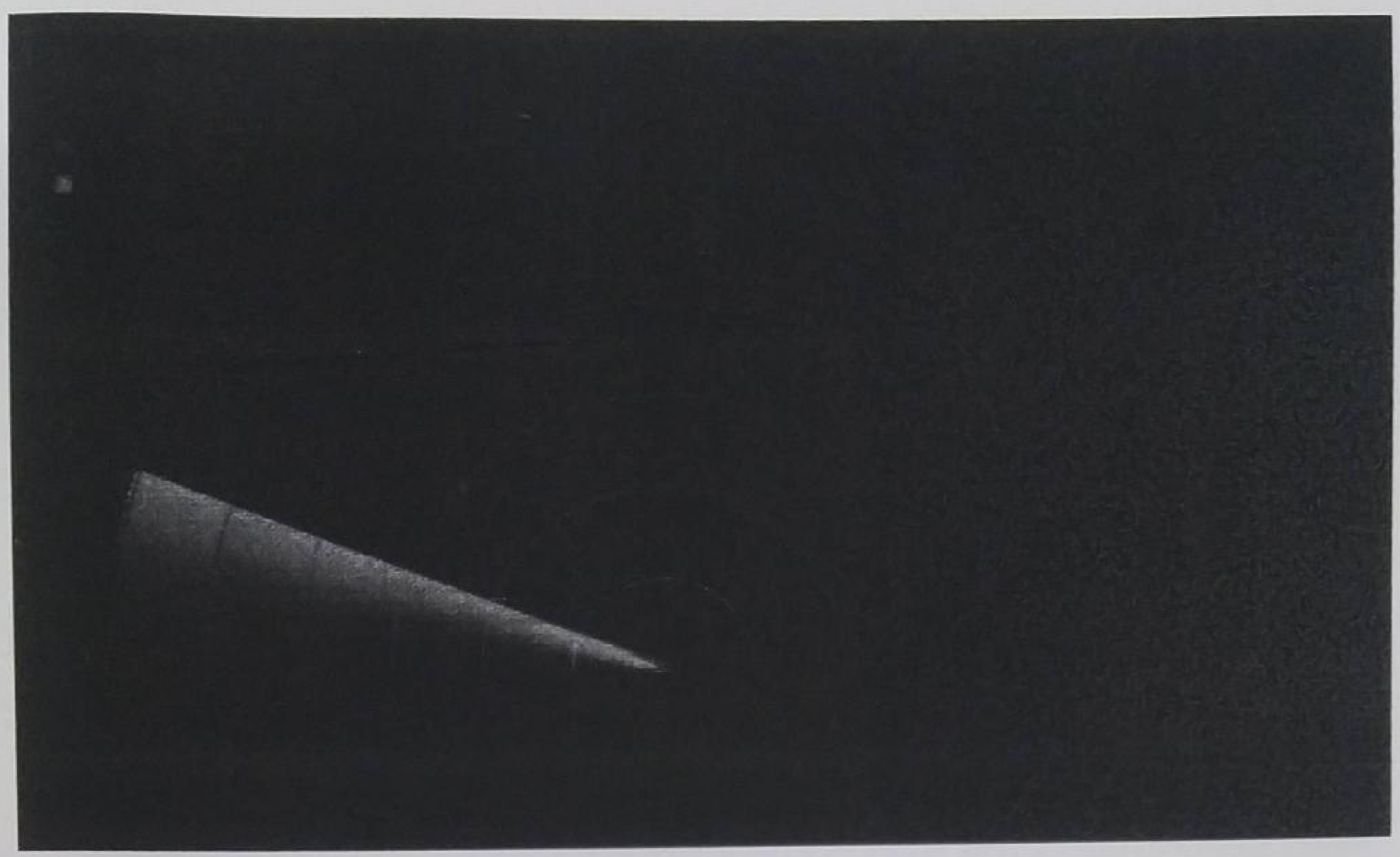

Figure 1.16 RADAR Screen using Processing 2.0

X. CIRCUIT SCHEMATICS

We connected the Ultrasonic Sensor HC-SR04 to the pins number 10 and 11 and the servo motor to the pin number 12 on the Arduino Board.

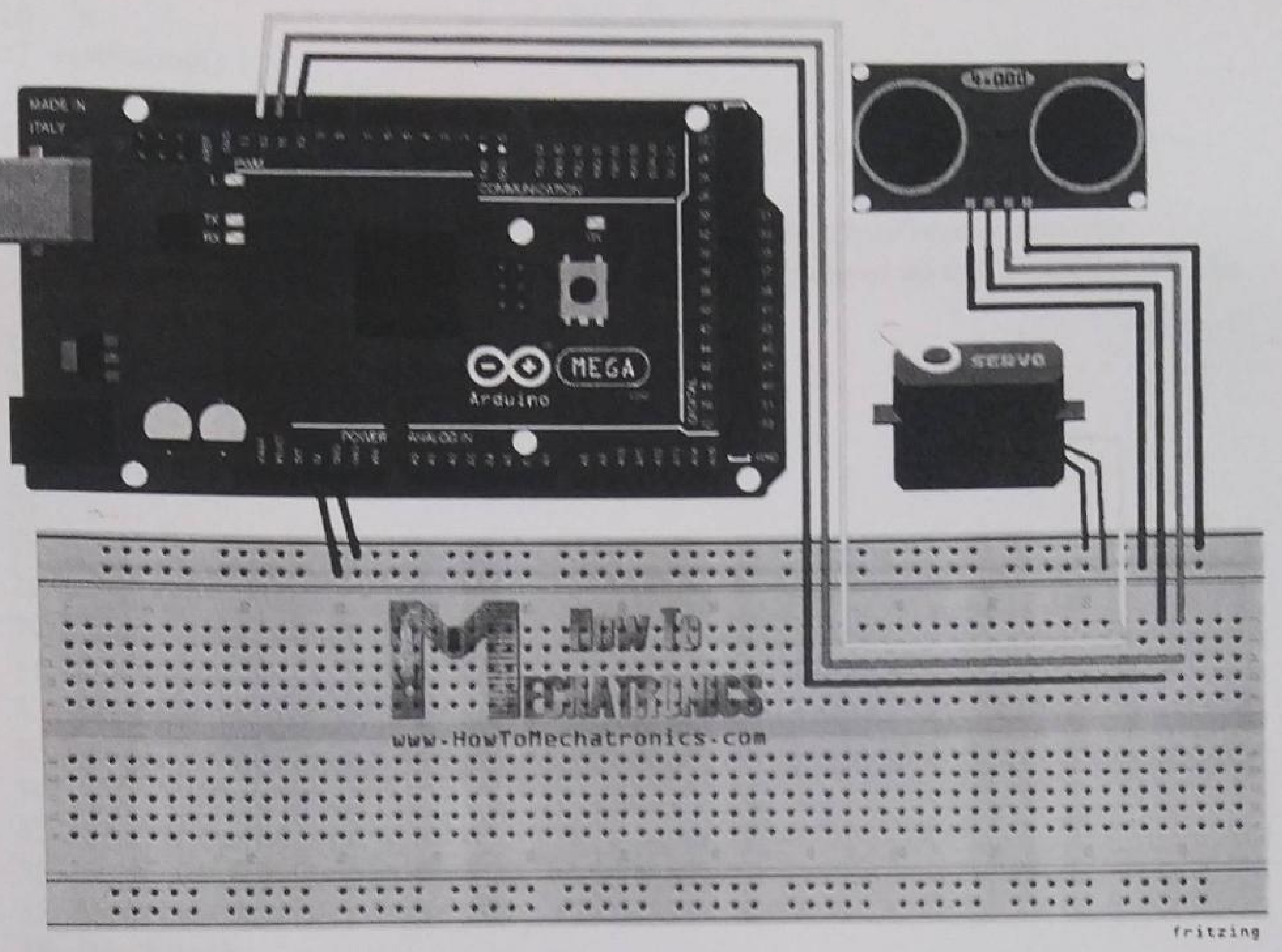

Figure 1.17 Circuit Schematics

XI. SOURCE CODE

Now we need to make a code and upload it to the Arduino Board that will enable the interaction between the Arduino and the Processing IDE.

```
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 (1920, 1080); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
smooth();
myPort = new Serial(this,"COM4", 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();
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("Object: " + noObject, 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();
}
```

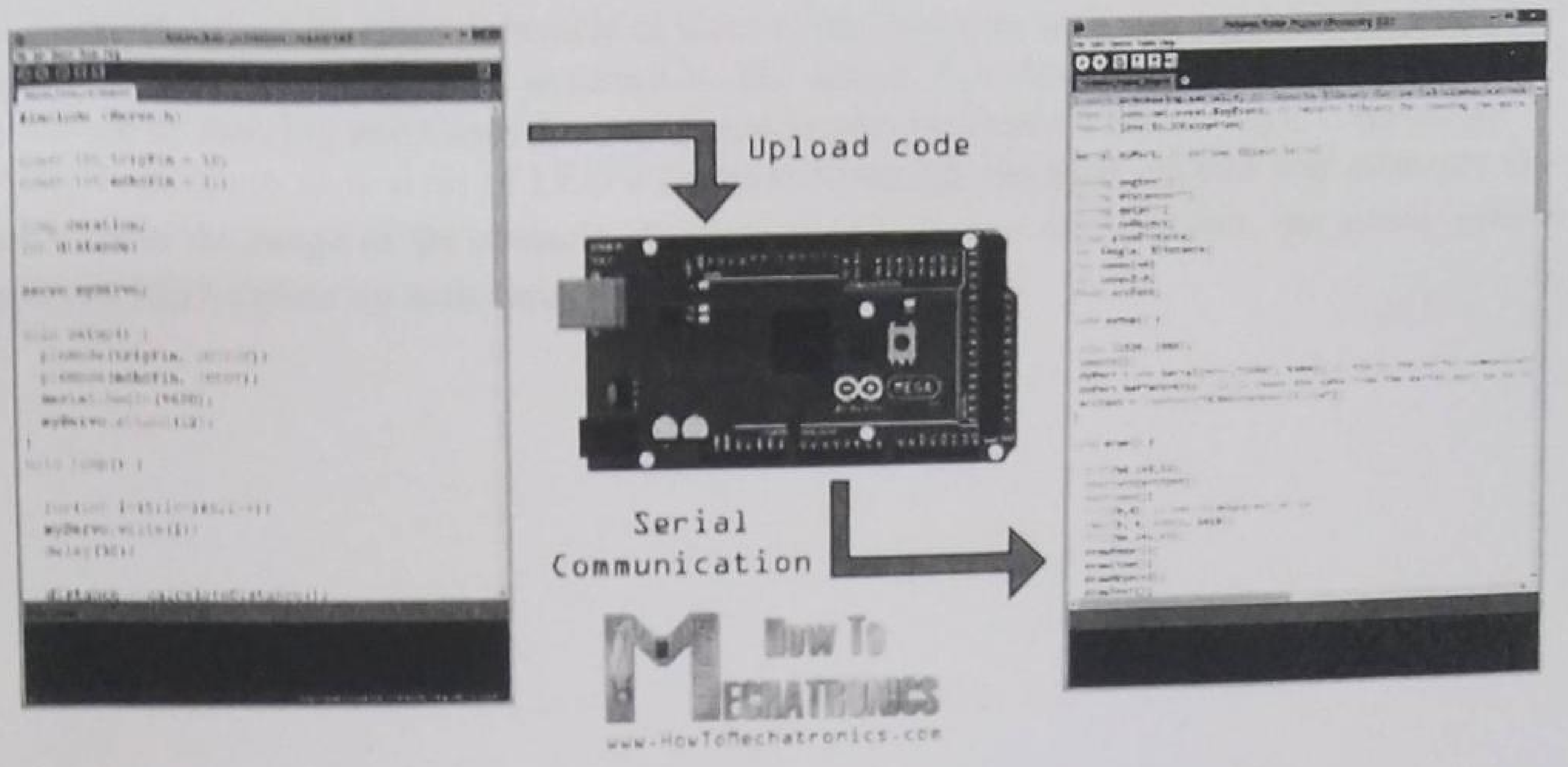

Figure 1.18 Interaction between the Arduino & Processing IDE

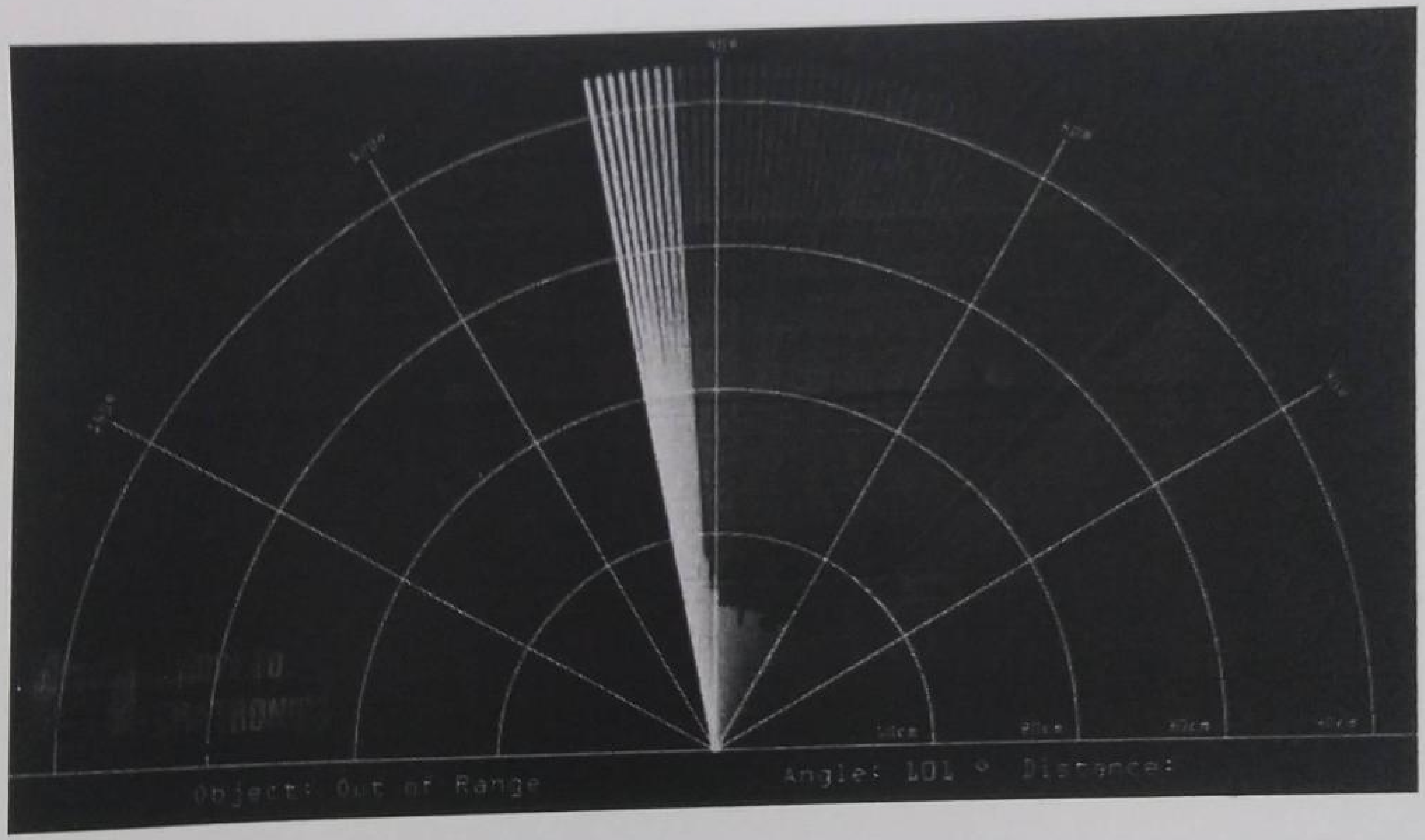

Figure 1.19 Final Appearance of the Radar

XII. CONCLUSION

This project aims on the use of Ultrasonic Sensor by connected to the Arduino UNO R3 board and the signal from the sensor further provided to the screen formed on the laptop to measure the presence of any obstacle in front of the sensor as well as determine the range and angle at which the obstacle is detected by the sensor. For this screen, we use Processing 2 software by Ben Fry and Casey Rease, Massachusetts Institute of Technology, Cambridge. Also, in addition to it, a set of LED's connected through the shift register and a buzzer also tells about the range of the obstacle. According to the range of the object, the green, yellow and red LED's glow up with variations in the buzzer output.

Scanned by TapScanner

REFERENCES

[1] http://www.arduino.cc/
[2] http://www.arduinoproducts .cc/
[3] http://www.atmel.com/atmega328/
[4] http://en.wikipedia.org/wiki/File:16MHZ_Crystal.jpg
[5] http://fritzing.org
[6] http//:www.sproboticworks.com/ic%20pin%20configurat ion/7805/Pinout.jpg/
[7] http://www.sproboticworks.com/ic%ultrasonicsensor%2 0pinout.jpg
[8] http://www.instructables.com/id/ ATMega328-using- Arduino-/
[9] http://www.motherjones.com/files/blog_google_driverles s_car.jpg
[10] http://www.google.co.in/imgres/Radar_antenna.jpg&w= 546&h=697&ei=wuuK
[11] http://www.radomes.org/museum/photos/equip/ANSPS1 7.jpg
[12] http://www.wired.com/dangerroom/2011/07/ suicide-
bombers-from-100-yards/
[13] http://upload.wikimedia.org/wikipedia/commons/Radara ccumulationseng.png
[14] http://arduino.cc/en/Tutorial/BarGraph/
[15] http://arduino.cc/en/Tutorial/LiquidCrystal/