CHAPTER 1: INTRODUCTION

RADAR (**R**adio **D**etection **A**nd **R**anging) 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.

The project has combined the hardware and software elements of RF engineering to produce a completely portable system which operates on the 2.4 GHz Wi-Fi band and is capable of two different modes. These modes are the Doppler mode which calculates a targets speed, and the Range mode which calculates its distance. The primary aims of this year's project was to improve the radars portability, decrease its cost, and increase its functionality. In this project we will formed the ultrasonic Radar system whose range is up to 4 meter (400cm).

This equipment is very useful in the distance measurement, obstacles finder robot, for surveying in civil engineering and also it is useful in to the parking of a vehicles and military applications.

This instrument will easy to handle and it's reliable. Due to the small size it is easy to carry any were.

CHAPTER 2: REVIEW AND LITERATURE SURVAY

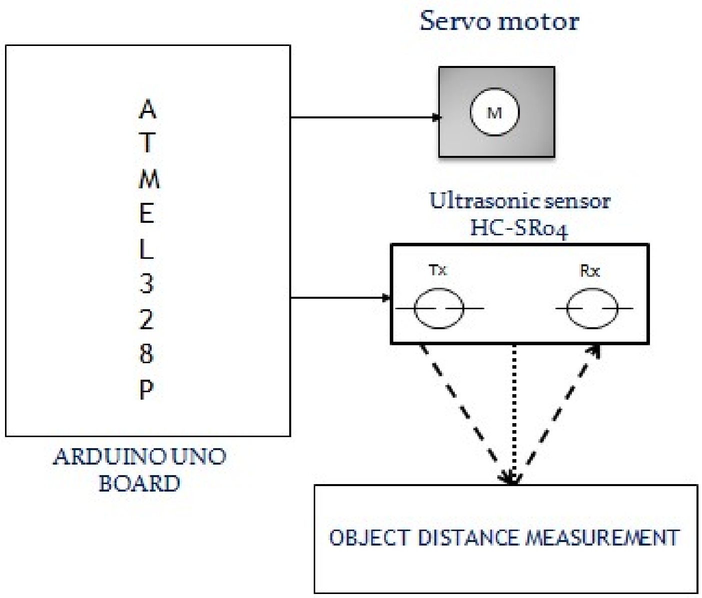
Future concept for FM detection, to identify main lines along which research and development effort should be oriented .Our project describes a prototype of portable military radar system. Information plays an important role in military operations and radar is thus required to detect, locate and identify numerous targets accurately in all weather conditions and other wide areas. In recent times misusing of unmanned aerial vehicles is becoming the hot and topical problem.

Portable RADAR system Means of air attack (MOAA) were, are and will be in the future, inseparable part of modern armed conflicts. That goes always when warring parties have MOAA in disposal. Highly developed armies are using the whole spectrum of MOAA and their quantity employment intensity and proportion among them depend on many factors such as disposition and situation of enemy, type of operation, phase of conflict, local condition of operation. Hence philosophy of defense against them should be formed. The main solution for this problem would be surface object detection in urban environmental condition using infrared and visible part of electromagnetic spectrum. M. Polasek in their paper compiled an algorithm for detection and selection of objects of interest in urban built-up background of civil automobiles which closely resemble similar military equipment. Those objects are captured by IR camera and visible camera in different outdoor conditions for instance during daytime in all seasons. The object detection in infrared spectrum is based on assumptions of object of interest are given definite color .The basic task of the object detection in image data is a selection of an optimal threshold value to be converted from intensity image to binary image.

In 1790, Lazzaro Spallanzo was first whose discovered the BAT movement with the help of hearing for movement not seeing forward. Jean-Dawel Col- ultrasonic security system discovered sonography 1826 using an underwater bell, and determine the speed of sound in liquid. Therefore further study and research work proceed slowly on time to time. In 1881, when Pierce Curie’s design the modern ultrasound transducer and he concluded that the relationship between electrical voltage and pressure on any crystalline material, and on that time TITANIC tragedy influences to take more interest to work in this field.

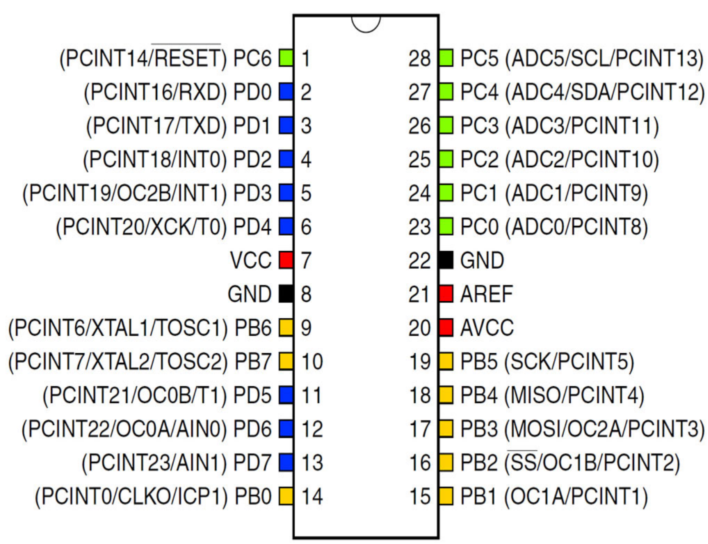
CHAPTER 3: SYSTEM DEVELOPMENT

Block diagram:



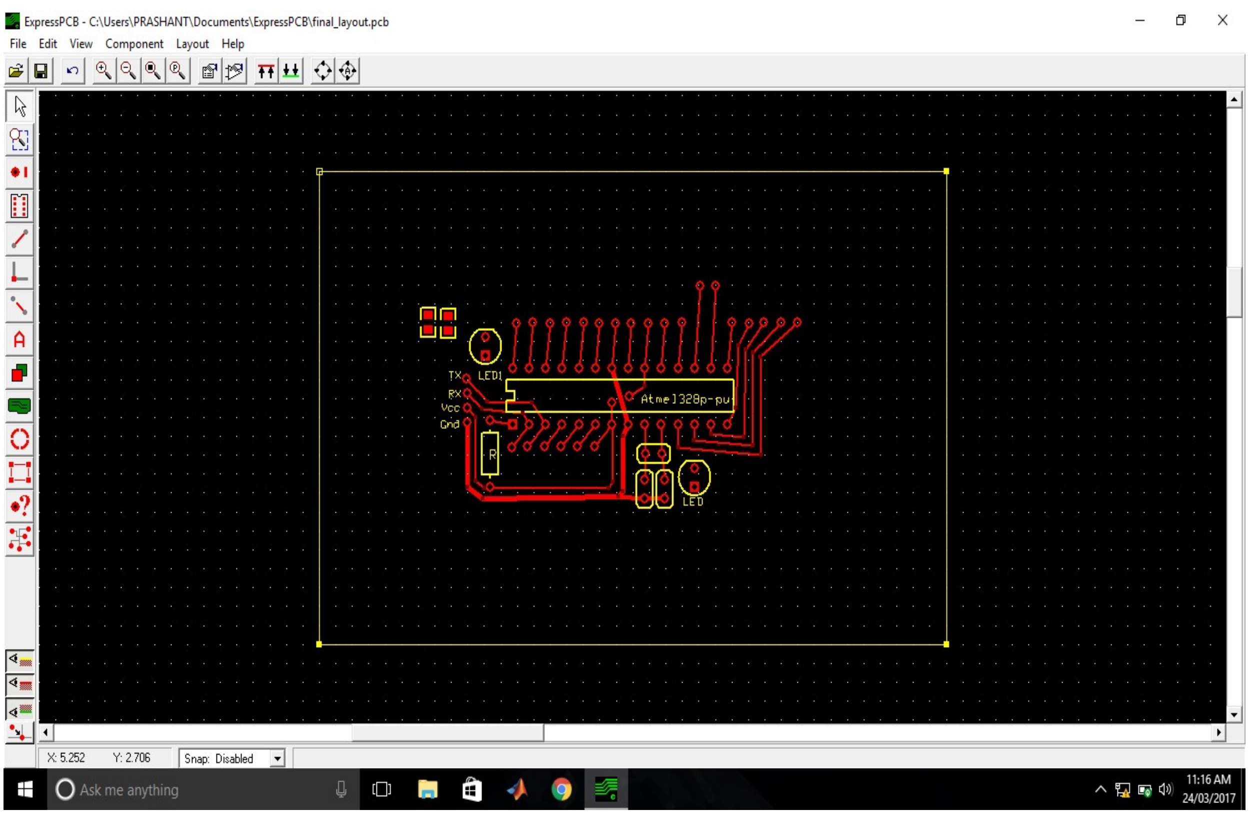
A. Hardware

1) Atmega328p-pu pin diagram:



ARDUINO UNO BOARD PROTOTYPE

2) PCB Layout:



3)Ultrasonic sensor(distance meter) :

Vcc = connect to 5v of positive voltage for power

Trig = A pulse is sent here for the sensor to go into ranging mode for object detection

Echo = The echo sends a signal back if an object has been detected or not. If a signal is returned, an object has been detected . if not , no object has been detected.

GND = completes electrical pathway of the power.

4) Servo motor:

This is nothing but a simple electrical motor controlled with the help of servomechanism.

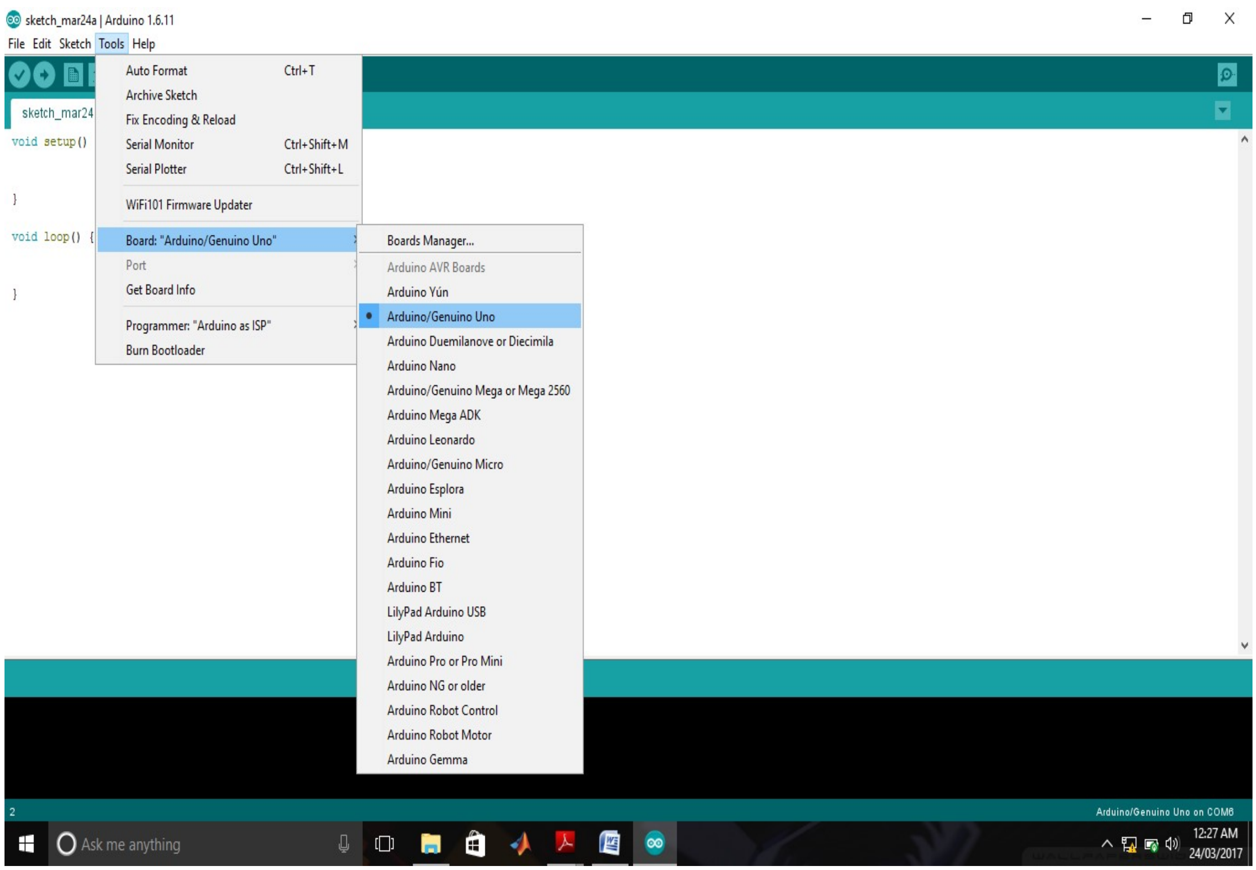
Some special types of motor are required with some special arrangement which makes the motor to rotate a certain angle for a given electrical input. For this purpose servo motor comes into picture

B. Software:

1) Arduino IDE :

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.



Arduino is an open source, computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself kits. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler tool chains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2003 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

The name Arduino comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

History:

The origin of the Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2003, Colombian student Hernando Barragán created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller. In 2003, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they copied the Wiring source code and renamed it as a separate project, called Arduino.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU tool chain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consist of only two functions:

Setup : This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.[44]

Loop : After setup has been called, function loop is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

Applications:

i) Xoscillo, an open-source oscilloscope

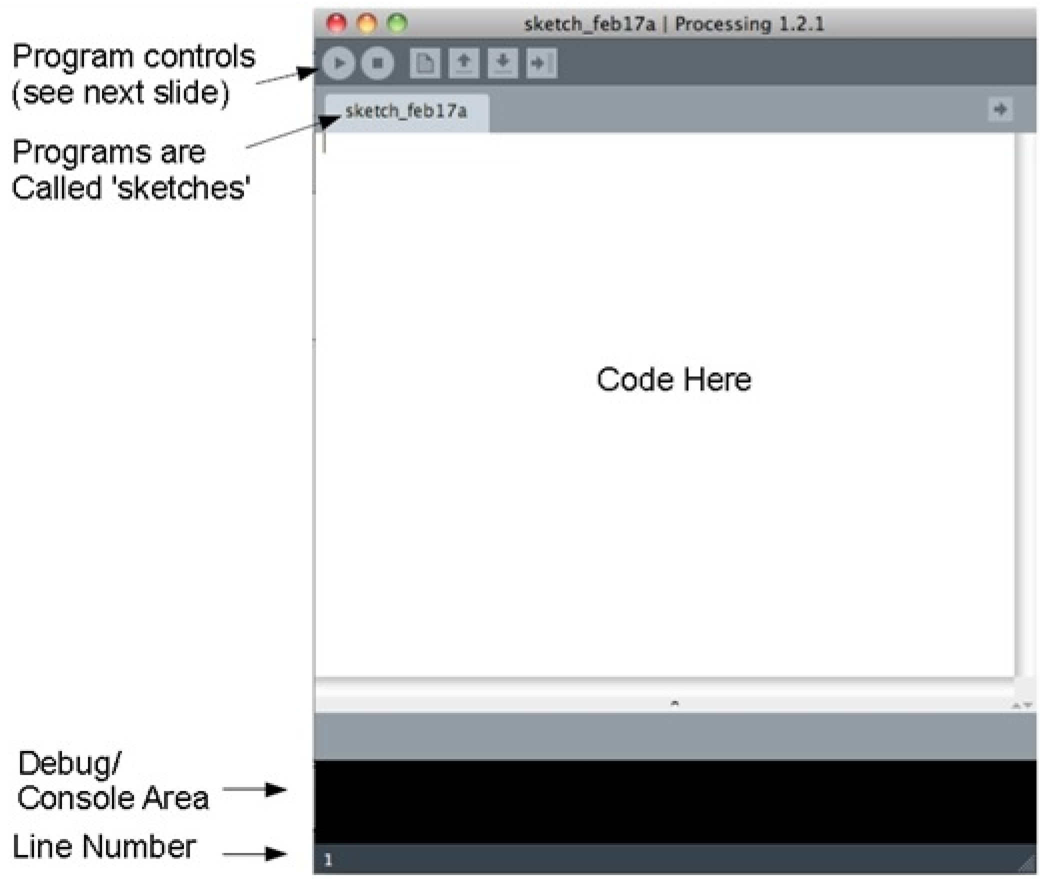
ii) Arduinome, a MIDI controller device that mimics the Monome

iii) OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars

iv) Ardupilot , drone software and hardware

2) Arduino processing software:

Arduino Processing is an open source language/development tool for writing programs in other computer . Useful when you want those other computers to talk with an arduino, for instance to display or save some data collected by arduino. Processing is a free, open-source Java-based framework as well as an Integrated Development Environment (IDE).It was initially developed by Casey Reas and Ben Fry at the MIT Media Lab in 2001 as a tool to help teach programming, with special attention given to artistic and visual applications. Processing has since gone on to become one of the most widely used tools for teaching introductory programming to noncomputer scientists, as well as a popular tool amongst artists and creative technologists for realizing their work.



Each processing sketch is actually a subclass of a PApplet superclass which defines most of Processing's behavior. i.e., Processing encapsulates much of the intricacies of developing in Java (don't need a main method,public/private distinctions). In fact, static methods are prohibited, unless you're programming in pure Java mode (more on this later)

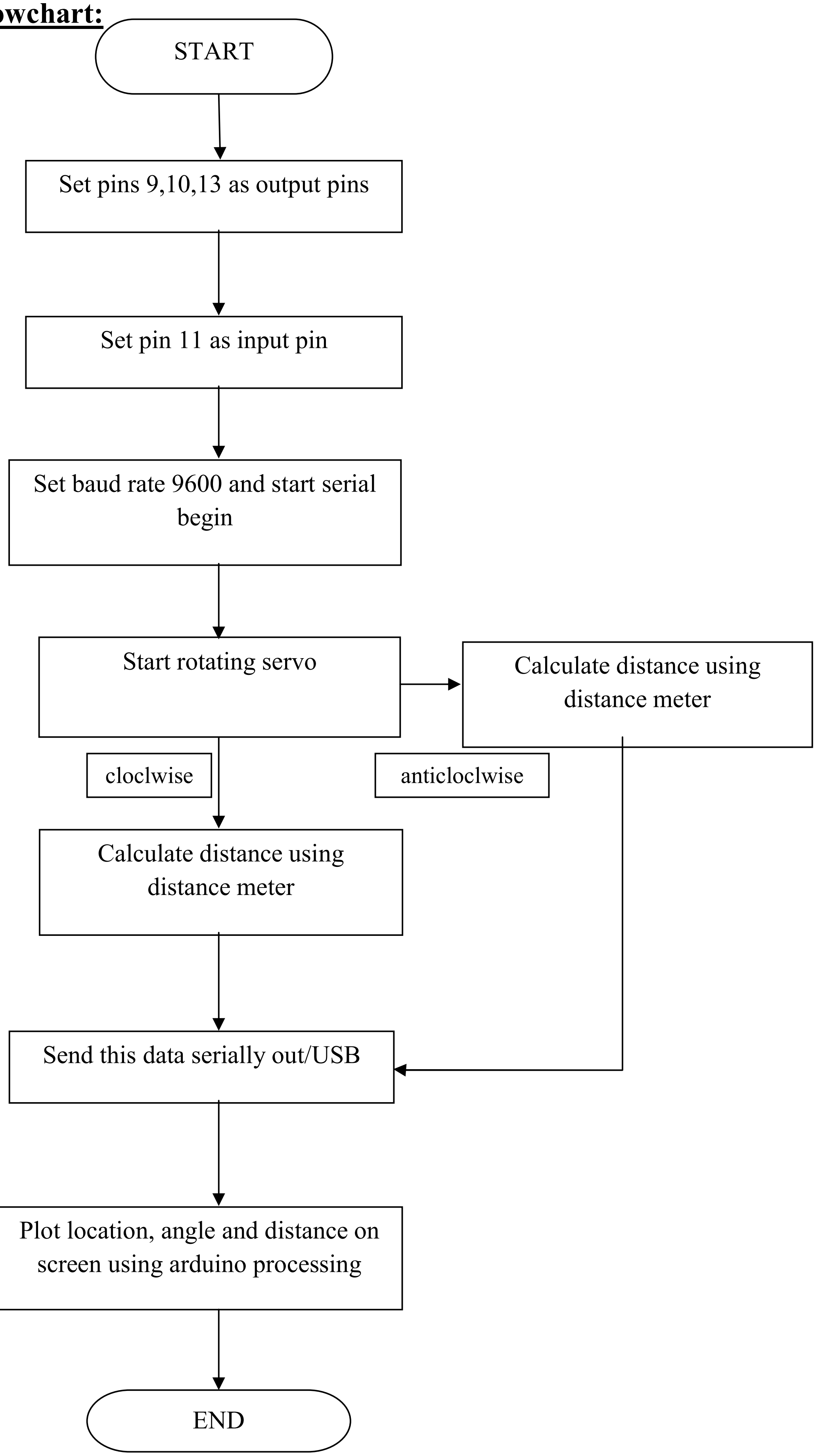
The setup() and draw() methods are common to most Processing sketches

i. Setup() is for setup - initializing variables and objects. Only gets run once on startup.

ii. Draw() is the main loop of your sketch – it will continue to loop unless a noLoop() method is called.

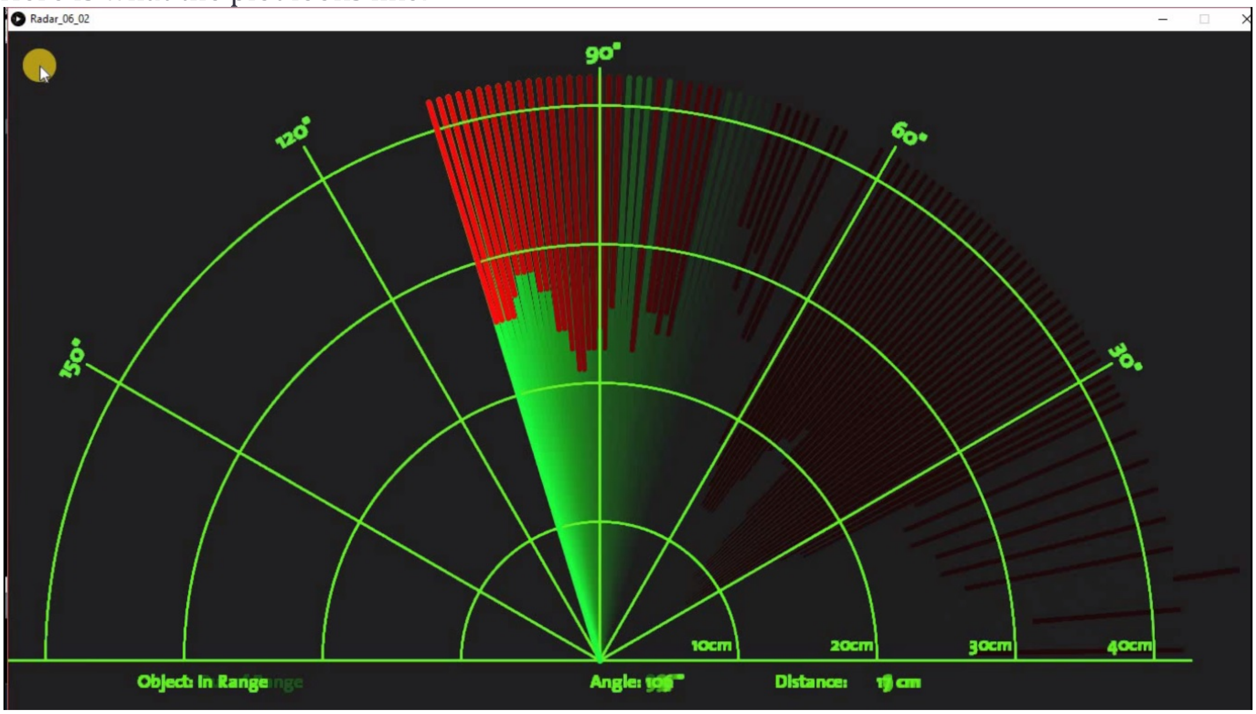
Basically, if it changes (like an animation), it needs to be called in the draw() method.

Programme flowchart:



CHAPTER 4 : RESULT AND DISCUSSION

As mentioned, The program triggers a PROCESSING Software to plot the detected objects. Here is what the plot looks like:



As shown in above figure we get the result of our project. Thus in the result of the project we get the exact location/direction of the object and distance from the system in cm.

Distance measurement accuracy :

Our sensor calibration and utilization gave us a distance measurement with roughly 1.5cm accuracy. Provided that we are using the sensor to detect presence of an object more so than the exact distance, this accuracy is tolerable.

Speed of detection :
It takes a maximum of three seconds for the system to detect a newly present object, and to turn on the LED. This is because the servo motor is set to cover 180 degree range, and thus it takes roughly 10 seconds for it to start from one angle and back. However, a user can easily configure the servo to a narrower Angle to focus at specific area (such as doorway), in which case the sensor will detect newly present object in less than a second.

Detection accuracy :
Through our tests, we believe our system is capable of detecting intruders 95% of the time, provided that sensor was placed at appropriate position. The few times that intruder get away are when they are capable of crossing past the sensor quicker than 200msec, which is our measurement interval hard-coded into our code.

Application :

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.

A. Air Force

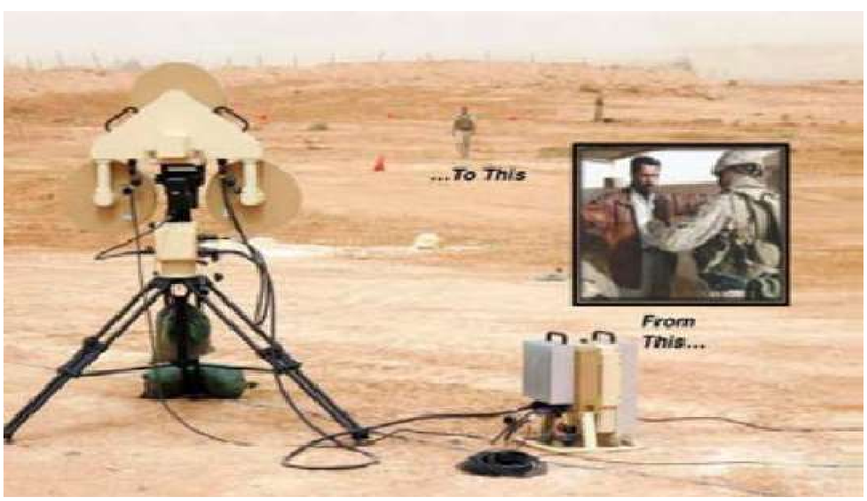
In aviation, aircraft are equipped with radar devices that warn of aircraft or other obstacles in or approaching their path, display weather information, and give accurate altitude readings. The first commercial device fitted to aircraft was a 1938 Bell Lab unit on some United Air Lines aircraft. Such 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.

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 system are used to monitor and regulate ship movement in busy water.

C. Applications 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.



CHAPTER 5 : CONCLUSION AND FUTURE SCOPE

Future Radar Concepts Will Result from Multiple interactions and tradeoffs between anticipated requirements, available technologies, and permanent basic limitations. In this paper, an overview of these key factors is given, in order to examine some promising future concepts and to identify main lines along which research and development effort should be oriented. Being highly prospective, this paper is certainly highly critic able. Rather than pretending to determine the future, the main objectives of this paper are to simulate reflection and discussions about significant evolutions in radar concepts, and possible breakthroughs The views presented here are the result of personal reflection, past experience, and multiple professional interactions, as such, they should not be considered as representative of any country or company opinion.

References :

[1]Future Concepts for Electromagnetic Detection - FranGois le Chevalier

[2] Mini UAVs Detection by Radar- Miroslav Krátký*, Luboš Fuxa

[3]ALGORITHM FOR MILITARY OBJECT DETECTION USING
IMAGE DATA - Iq. Pham and M. Polasek, University of Defence, Kounicova 65, 66210 Brno, the Czech Republic

[4] http://www.arduino.cc

[5] http://en.wikipedia.org

Functional Testing of Project :

Setup Atmel328p and IDE	completed
Order parts as needed	completed
Build basic test apparatus	completed
Configure HCSR-04 interface	completed
Interface with servo	completed
Test servo range of motion	completed
Write c program for IDE	completed
Write Processing sketch	completed
Calibrate ultrasonic sensor output	completed
Test completed code	completed
Run program	completed

List of Component with Price :

Sr No.	List of components	Price
1.	Atmel 328p-pu IC	**150 /-**
2.	16MHz crystal	**05 /-**
3.	Ultrasonic Sensor HCSR-04	**150 /-**
4.	Servo Motor	**250 /-**
5.	Servo Motor (Damaged during work)	**250 /-**
6.	Jumper wire and Connectors	**45 /-**
	TOTAL =	**850 /-**

Retrospective :

I am glad that I chose to complete this project on the Arduino. It was my first real coding experience on this platform, and I can say that compared to writing C for the Zilog ZNEO, writing Wiring libraries for Arduino makes for a much more fun and productive experience. I am grateful that my time on the ZNEO taught me a lot about what is happening behind the scenes, but quiet honestly it is nice to not have to worry about it so much.

Knowing what I know now, I might have tried to find a sensor that had a narrower beam pattern. It also would have been nice to integrate an infrared range finder, as the time required to capture an ultrasonic range sample is fairly long. This being said, I think a broad beam pattern might be appropriate if the sensor is mounted in a stationary position or if the servo is only moved sporadically to face the direction of travel, so the SRF10 will still prove useful in future projects.

One area that will require a bit more effort is the integration of Processing Software with the Arduino. I still have not gotten wireless uploading of Arduino sketches working, and this would certainly be a nice feature to have.

As stated previously, another area I need to look into is battery power. A better long term portable power supply would include a higher output rechargeable battery.

Attachment :

1. Arduino IDE software code
2. Arduino Processing software code
3. Atmel328p-pu IC data sheet

1. Arduino IDE software code :

```
#include <Servo.h>          // Includes the Servo library
Servo myservo;              // Creates a servo object for controlling the servo motor
const int trigPin = 10;     // Defines Tirg and Echo pins of the Ultrasonic Sensor
const int echoPin = 11;
long duration;              // Variables for the duration and the distance
int distance;
int pos = 0;

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);  // Sets the echoPin as an Input
  pinMode(13,OUTPUT);
  Serial.begin(9600);
  myservo.attach(9);       // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 0 to 180 degrees
  for (pos = 0; pos <= 180; pos++)
    { // goes from 0 degrees to 180 degrees
     // in steps of 1 degree
     myservo.write(pos);        // tell servo to go to position in variable 'pos'
     delay(60);                 // waits 60ms for the servo to reach the position
  distance = calculateDistance(); // Calls a function for calculating the distance measured by the
Ultrasonic sensor for each degree
  Serial.print(pos);            // Sends the current degree into the Serial Port
  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
 Serial.print("."); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
 if(distance<40)
 {
   digitalWrite(13,HIGH);
   delay(40);
   }
 else
 digitalWrite(13,LOW);
 delay(30);
 }
 // Repeats the previous lines from 180 to 0 degrees
 for (pos = 180; pos >= 0; pos--) { // goes from 180 degrees to 0 degrees
   myservo.write(pos);              // tell servo to go to position in variable 'pos'
   delay(60);                       // waits 60ms for the servo to reach the position
 distance = calculateDistance();
 Serial.print(pos);
 Serial.print(",");
 Serial.print(distance);
 Serial.print(".");
 if(distance<40)
 {
   digitalWrite(13,HIGH);
   delay(40);
   }
 else
 digitalWrite(13,LOW);
 delay(30);  }  }
```

```
// Function for calculating the distance measured by the Ultrasonic sensor
int 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;
  return distance;
}
```

2.Arduino Processing software 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 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);

 smooth();

 printArray(Serial.list());

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

 //myPort = new Serial(this,"COM6", 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("TheSans-Plain-12.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, 1010);

  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);
  //myPort.write(data);
  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(640,680); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,1200,1200,PI,TWO_PI);
  arc(0,0,900,900,PI,TWO_PI);
  arc(0,0,600,600,PI,TWO_PI);
  arc(0,0,300,300,PI,TWO_PI);
  // draws the angle lines
  line(-640,0,640,0);
  line(0,0,-640*cos(radians(30)),-640*sin(radians(30)));
  line(0,0,-640*cos(radians(60)),-640*sin(radians(60)));
  line(0,0,-640*cos(radians(90)),-640*sin(radians(90)));
  line(0,0,-640*cos(radians(120)),-640*sin(radians(120)));
  line(0,0,-640*cos(radians(150)),-640*sin(radians(150)));
  line(-640*cos(radians(30)),0,640,0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(640,680); // moves the starting coordinats to new location
```

```
  strokeWeight(6);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*22.5; // 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)),630*cos(radians(iAngle)),-630*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(6);
  stroke(30,250,60);
  translate(640,680); // moves the starting coordinats to new location
  line(0,0,630*cos(radians(iAngle)),-630*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, 1010, width, 1080);
fill(98,245,31);
textSize(20);
text("10cm",740,670);
text("20cm",890,670);
text("30cm",1040,670);
text("40cm",1190,670);
textSize(20);
text("Object: " + noObject, 140, 710);
text("Angle: " + iAngle +" °", 630, 710);
text("Distance: ", 830, 710);
if(iDistance<40) {
text("        " + iDistance +" cm", 900, 710);
}
textSize(25);
fill(98,245,60);
translate(641+640*cos(radians(30)),662-640*sin(radians(30)));
rotate(-radians(-60));
text("30°",0,0);
resetMatrix();
translate(634+640*cos(radians(60)),664-640*sin(radians(60)));
rotate(-radians(-30));
text("60°",0,0);
resetMatrix();
translate(625+640*cos(radians(90)),670-640*sin(radians(90)));
rotate(radians(0));
text("90°",0,0);
```

```
resetMatrix();
translate(615+640*cos(radians(120)),683-640*sin(radians(120)));
rotate(radians(-30));
text("120°",0,0);
resetMatrix();
translate(620+640*cos(radians(150)),698-640*sin(radians(150)));
rotate(radians(-60));
text("150°",0,0);
popMatrix();
}
```