Radar System using Arduino

Pankhuri Varshney
B.Tech in Computer Science
Batch of 2023-2027
Division B, First Year
PRN: 23070122160

Om Ajit Ghule
B.Tech in Computer Science
Batch of 2023-2027
Division B, First Year
PRN: 23070122154

Mahi Sharma
B.Tech in Computer Science
Batch of 2023-2027
Division B, First Year
PRN: 23070122128

Omayr Muqarrab Yunus
B.Tech in Computer Science
Batch of 2023-2027
Division B, First Year
PRN: 23070122156

Nedha Nizamudeen
B.Tech in Computer Science
Batch of 2023-2027
Division B, First Year
PRN: 23070122147

***Abstract*—In the realm of modern technological advancements, the integration of Arduino microcontrollers has revolutionized the field of automation and control systems. This project introduces an innovative application of Arduino technology in the development of a radar system. Unlike traditional radar guidance systems, our Arduino-powered radar system utilizes ultrasound for precise aiming and employs sophisticated algorithms to enhance accuracy. The integration of ultrasound technology not only facilitates precise targeting but also incorporates a novel approach to obstacle avoidance, thus ensuring the safety and efficiency of the radar system.**

This project aims to showcase the potential of Arduino in advancing radar guidance systems, offering a cost-effective and versatile solution for applications in defense and beyond.

I. Introduction

Radar, short for Radio Detection and Ranging, is a technology that uses electromagnetic waves to detect the range, speed, and other characteristics of objects. Traditional radar systems have been complex and expensive, limiting their accessibility. The emergence of Arduino, a popular open-source microcontroller platform, has democratized the development of radar systems, enabling enthusiasts and researchers to experiment with radar technology.

A. Purpose of our Project

Radar systems have evolved significantly over the years, finding applications in various fields such as aerospace, defense, and automotive industries. With the rise of accessible and affordable microcontroller platforms like Arduino, there has been a growing interest in developing radar systems for educational, research, and hobbyist purposes. This literature review aims to explore the current state of radar systems implemented using Arduino platforms and highlight key developments, challenges, and potential applications.

Traditional radar guidance systems often rely on complex and expensive sensors, making them economically impractical for widespread use. Our approach seeks to address this limitation by harnessing the flexibility and cost-effectiveness of Arduino microcontrollers. The integration of ultrasound sensors not only enables precise aiming but also introduces a new dimension to missile systems by incorporating obstacle avoidance capabilities.

This project will explore the technical aspects of implementing Arduino in radar guidance, including the hardware and software components involved. Additionally, it will highlight the potential applications of such a system in defense scenarios where accuracy and safety are paramount. Through this endeavor, we aim to contribute to the ongoing discourse on the versatile applications of Arduino technology in the defense industry, paving the way for more accessible and efficient radar guidance systems.

II. Methodology Used in our Project

Designing a radar system project using Arduino and ultrasonic sensors involves a systematic approach, combining hardware setup, programming, and testing.

A. Componenets Used:

1) **Arduino Uno:** The Arduino Uno stands at the forefront of microcontroller technology, earning its reputation as a versatile and accessible board that has become a linchpin in the maker community. Fueled by the ATmega328P microcontroller, it offers 14 digital I/O pins, six of which support PWM, and an additional six analog input pins, providing a robust foundation for diverse projects. Operating at a clock speed of 16 MHz, the Uno delivers the processing power needed for applications ranging from fundamental LED projects to sophisticated robotics. Its USB interface simplifies both programming and power supply, while its compatibility with an extensive array of shields enhances its adaptability. Beyond its technical specifications, the Arduino Uno embodies the spirit of open-source innovation, fostering a dynamic and collaborative community. Whether enthusiasts are engaged in LED displays, sensor applications, home automation, or educational pursuits, the Arduino Uno continues to be a catalyst for exploration, creativity, and hands-on learning in the realms of electronics and programming.

2) **Ultrasonic Sensor:**The ultrasonic sensor, a remarkable device in the realm of electronic sensing technology, utilizes sound waves to measure distances and detect objects with exceptional precision. Working on the principle of echolocation, it emits ultrasonic waves and calculates the time taken for the waves to bounce back after hitting an object. This time measurement allows the sensor to determine the distance to the object. Ultrasonic sensors are characterized by their non-contact nature, making them invaluable in applications where physical touch is impractical or undesirable. Their versatility is evident in fields such as robotics, industrial automation, and automotive systems, where they facilitate object avoidance, distance measurement, and presence detection. Ultrasonic sensors have proven to be reliable, cost-effective tools, offering a crucial combination of accuracy and ease of implementation in a myriad of technological applications.
3) **Servo Motor:** A servo motor is a widely used electromechanical device that translates electrical signals into precise mechanical movements. Recognized for its accuracy and controllability, the servo motor is pivotal in robotics, automation, and various other applications. Comprising a motor, feedback mechanism, and control circuitry, the servo motor continuously receives signals to maintain or move to a specific angular position. This closed-loop control system enables the servo motor to provide precise motion control, making it an indispensable component in mechanisms demanding accuracy, such as robotic arms, remote-controlled vehicles, and even in industrial processes like CNC machining. The ability to rotate to a precise angle and maintain position, coupled with the ease of interfacing with microcontrollers, positions the servo motor as a fundamental element in the world of electromechanical systems.
4) **Arduino TFT Display:** The 1.8-inch SPI TFT Arduino monitor is a compact and feature-rich display module designed for seamless integration with Arduino microcontrollers through the SPI communication protocol. With a resolution of 128x160 pixels, this display provides vibrant and clear visual outputs, making it ideal for various embedded projects and DIY electronics. Its compatibility with different Arduino boards, coupled with user-friendly programming using dedicated libraries, facilitates the creation of graphical user interfaces and data visualizations. The lightweight and compact design of the 1.8-inch display is well-suited for applications with space constraints, offering an efficient and visually appealing solution for projects ranging from wearables to handheld devices.
5) **COM3 Serial Port:** COM3, a virtual serial port designation on a computer, plays a crucial role in facilitating serial communication between devices. Typically, COM3 represents a specific hardware or virtual serial port on the computer system, serving as an interface for data exchange between the computer and external peripherals. Serial communication through COM ports is fundamental in various applications, from interfacing with microcontrollers and embedded systems to connecting with legacy hardware devices. The COM3 serial port, like other COM ports, operates on the RS-232 standard, offering bidirectional communication for transmitting and receiving data. Its utilization is common in scenarios where a reliable and standardized method of communication is necessary, contributing to the seamless integration of diverse electronic devices with computer systems.

B. Programming Languages and IDEs Used:

1) **Arduino IDE:** The Arduino Integrated Development Environment (IDE) stands as the central hub for programming and developing applications for Arduino microcontrollers. This open-source software provides a user-friendly platform that simplifies the coding process, making it accessible for both beginners and experienced developers. Offering a simple yet powerful interface, the Arduino IDE supports the *C and C++ programming languages*, enabling users to write and upload code to their Arduino boards seamlessly. With features such as a code editor, compiler, and a serial monitor for real-time communication with the connected Arduino, the IDE streamlines the development workflow. Its extensive library of pre-written functions and example codes further accelerates the prototyping process. The Arduino IDE's cross-platform compatibility and continuous community-driven updates contribute to its widespread adoption, serving as an essential tool for enthusiasts, educators, and professionals engaged in the ever-expanding world of Arduino projects.
   The libraries used in our project apart from Arduino's in-built functionalities include:
   - Servo.h: Library to manipulate servo motors
   - SPI.h: Library to use default SPI settings of a display moniter
   - TFT.h: Library to manipulate the TFT display screen and make our radar appear on it
2) **Processing IDE:** The Processing IDE is a versatile and user-friendly software environment designed for creative coding and visual arts projects. Developed with a focus on simplicity and accessibility, Processing empowers artists, designers, and beginners to explore programming concepts and create interactive visual experiences. Based on the *Java programming language*, the IDE provides a simplified syntax that allows users to quickly translate their ideas into visual elements. Processing seamlessly integrates graphics and interactive elements, making it an ideal choice for generative art, animations, and interactive installations. With a vibrant and supportive community, extensive documentation, and a range of libraries, Processing encourages experimentation and collaboration in the realm of creative coding. Its intuitive interface, coupled with real-time feedback, facilitates

an engaging and iterative coding experience, making Processing a powerful tool for those seeking to merge programming with visual expression.
The libraries used in our project apart from the in-built functionalities include:

- processing.serial.*; to access and manipulate the COM3 serial port communication
- java.awt.*; to make a Graphic-User Interface
- java.io.IOException; to handle any unwanted exceptions or errors

C. Hardware Architecture:

In our radar system, the flow of the working is such that we have connected the servo motor, ultrasonic sensor and tft display module to our arduino board in the following manner:

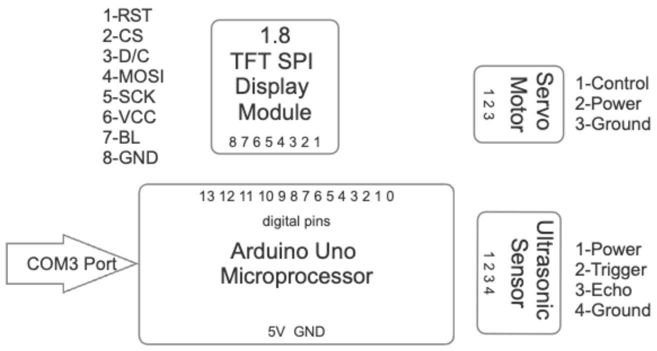

Fig. 1. Schematic Diagram

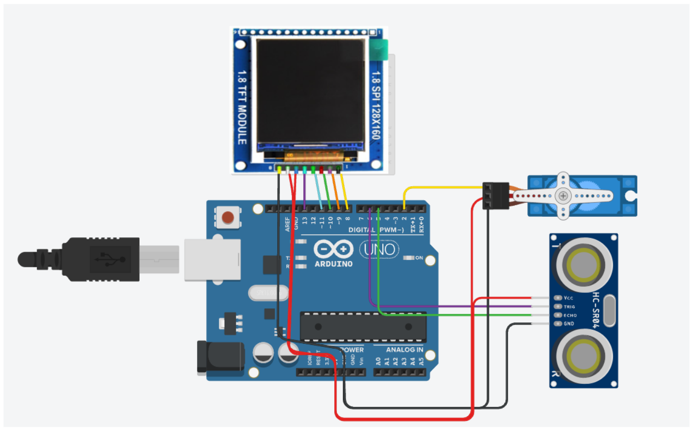

Fig. 2. Circuit Diagram

When our arduino is supplied power, 5V voltage is parallelly supplied to the servo motor, ultrasonic sensor and the tft display from the 5V pin in the arduino's power pin section. Similarly, the ground pin in the arduino is connected to all three components to complete the basis circuit.

There are three connections in the servo motor- power(red), ground(brown) and the control(yellow) pin. We have attached the third pin to the digital pin 2 in the arduino, so that we can control the motion of the servo motor through our code.

In the ultrasonic sensor, there are four pins- power, ground, echo and trigger. The trigger pin in an ultrasonic sensor is used to initiate and transmit signals, while the echo pin is used to receive signals. We have connected the trig pin in the sensor to digital pin 6 in the arduino and echo pin to digital pin number 5. Using these two pins, we can control the ultrasonic sensor to measure the distance by the formula:

$$distance = \frac{duration * speedofsound - wave}{2} \tag{1}$$

Lastly, to display our Radar System in a small display along with our laptop scree, we have connected the 1.8 SPI TFT Arduino Module. This display has 8 pins- Resest, CS, D/C, two SPI pins, two Power and one Ground pin. To make this module function, we have connected the RST, CS and D/C pins in the arduino's digital pins 8, 9 and 10 respectively. We have also connected the two SPI pins- one in digital pin 11 and one in 13 apart from the power connection above.

Once these connections are setup, the Arduino Code can be transferred to the board through the COM3 Serial Port.

D. Software Source Code:

There are two sets of programs in our project, one based on Arduino IDE and the another on Processing IDE.

Arduino Code using C++:

```
#include <Servo.h>
#include <SPI.h>
#include <TFT.h>

const int trigPin = 6;
const int echoPin = 5;
long duration;
int distance, prevDistance=0;
Servo myServo;

#define CS   9
#define DC   10
#define RESET  8

TFT screen = TFT(CS, DC, RESET);
int width=screen.width();
int height=screen.height();

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  myServo.attach(2);

  screen.begin();
  screen.background(0,0,0);
  fix();
}

void loop() {
  int y=0;
  for(int i=0;i<=180;i++){
  myServo.write(i);
  delay(10);
  distance = calculateDistance();
  if(distance<40)
    screen.stroke(0,68,255);
  else
    screen.stroke(0,255,68);
  screen.noFill();
  screen.line(screen.width()/2,0,-200*cos(radians(i)),200*sin(radians(i)));

  if(prevDistance<40)
  screen.stroke(0,0,100);
  else
  screen.stroke(0,100,0);

  screen.line(screen.width()/2,0,-200*cos(radians(y)),200*sin(radians(y)));

  screen.stroke(0,255,68);
  screen.noFill();
  screen.circle(screen.width()/2,0,screen.width()/2-0.5);
  y=i;
  prevDistance=distance;
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }

  fix();
  y=0;
  prevDistance=0;

  for(int i=180;i>0;i--){
   myServo.write(i);
  delay(10);
  distance = calculateDistance();
  if(distance<40)
    screen.stroke(0,68,255);
  else
    screen.stroke(0,255,68);
  screen.noFill();
  screen.line(screen.width()/2,0,-200*cos(radians(i)),200*sin(radians(i)));

  if(prevDistance<40)
  screen.stroke(0,0,100);
  else
  screen.stroke(0,100,0);

  screen.line(screen.width()/2,0,-200*cos(radians(y)),200*sin(radians(y)));

  screen.stroke(0,255,68);
  screen.noFill();
  screen.circle(screen.width()/2,0,screen.width()/2-0.5);
  y=i;
  prevDistance=distance;

  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
  fix();
}

int calculateDistance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(1);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  return distance;
}

void fix(){
  screen.fillRect(0, 0, screen.width(), screen.height(), (0,0,0));
  screen.stroke(0,255,68);
  screen.noFill();
  screen.circle(screen.width()/2,0,screen.width()/2-0.5);
  }
```

Processing Code using Java:

```
import processing.serial.*;
import java.awt.*;
import java.io.IOException;
Serial myPort;

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);
 smooth();
 myPort = new Serial(this,"/dev/cu.usbserial-110", 9600);
 myPort.bufferUntil('.');
}

void draw() {
  fill(98,245,31);
  fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31);
  the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent (Serial myPort) {
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(",");
  angle= data.substring(0, index1);
  distance= data.substring(index1+1, data.length());

  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074);
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  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);
  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);
  strokeWeight(9);
  stroke(255,10,10);
  pixsDistance = iDistance*((height-height*0.1666)*0.025);
  if(iDistance<40){  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);
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle)
  popMatrix();
}
void drawText() {
  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(30);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.285, 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();
}
```

E. Output for our Radar System

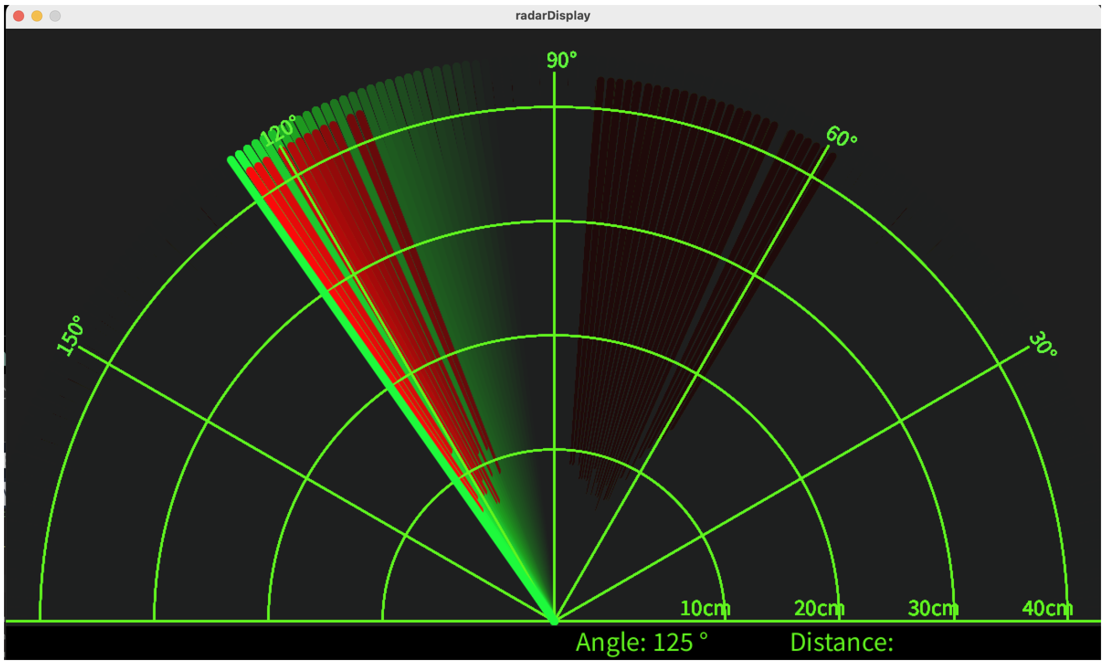

Fig. 3. Output display of Processing file, working with connection to the Radar System via COM3 Serial Port

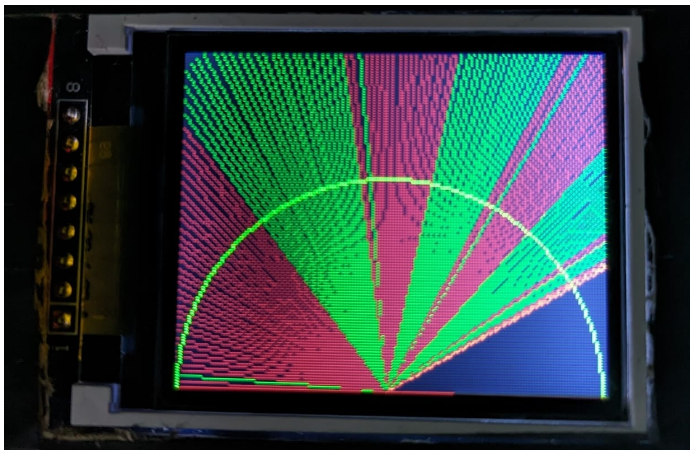

Fig. 4. Output display on Arduino TFT module

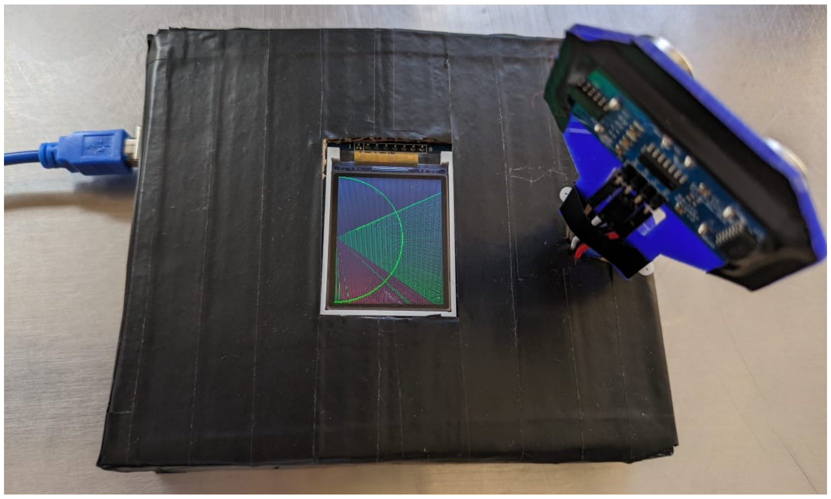

Fig. 5. Assembled Model of our Radar System

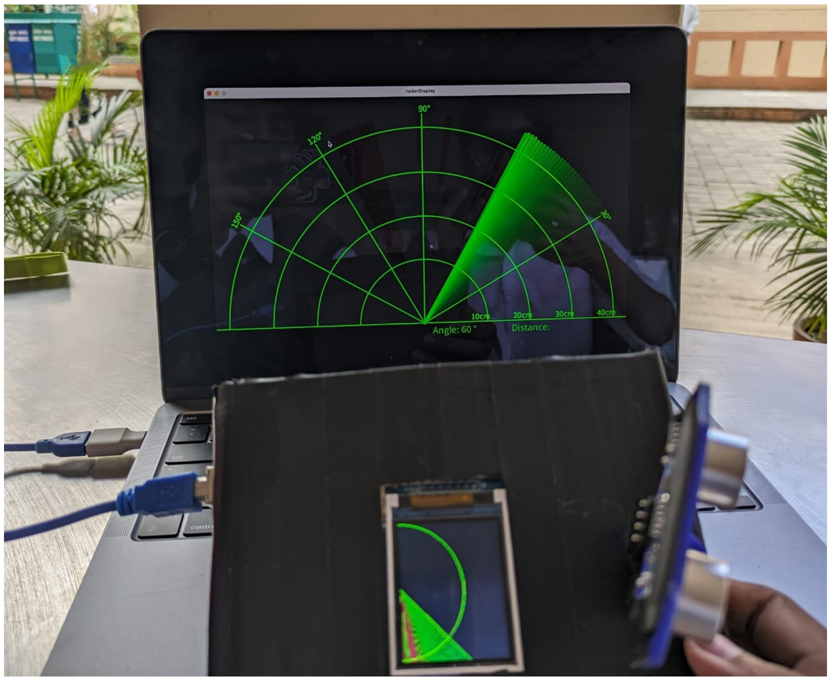

Fig. 6. Our Entire Radar System Project combined with the Integrated Radar and Detailed Radar on Screen

III. Future Scope of our Project

The future scope of a radar system project using Arduino and ultrasonic sensors is promising, with potential advancements and applications in various domains. Here are some potential areas of growth and development:

- **Advanced Signal Processing:** Future projects may focus on implementing more sophisticated signal processing techniques to enhance the accuracy and resolution of radar systems. This could involve using advanced algorithms, machine learning, or artificial intelligence for data analysis.
- **Longer Range and Higher Resolution:** Improving the range and resolution of radar systems built with Arduino and ultrasonic sensors is a logical next step. Advancements in sensor technology and signal processing algorithms may contribute to achieving longer detection ranges and higher resolution imaging.
- **Security and Surveillance Innovations:** Future projects may focus on enhancing the security features of radar systems, including integration with video analytics, facial recognition, or other advanced security technologies for more robust surveillance solutions.
- **Environmental Monitoring:** Radar systems equipped with environmental sensors could be utilized for monitoring and detecting changes in environmental conditions, such as detecting landslides, measuring water levels, or monitoring wildlife.
- **Healthcare Applications:** The use of radar systems in healthcare is an emerging field. Future projects may explore applications such as remote patient monitoring, fall detection for the elderly, or non-contact vital sign monitoring using radar technology.

The future scope of radar system projects using Arduino and ultrasonic sensors is dynamic and will likely evolve as technology advances and new applications are discovered. Continued innovation, collaboration, and integration with other technologies will contribute to the growth and diversification of these projects.

IV. Conclusion

In conclusion, the concept of an Arduino-powered missile system utilizing ultrasound for aiming represents a compelling intersection of technology, innovation, and accessibility. This approach offers a range of benefits and considerations. Arduino's affordability makes this technology accessible to a broad audience, fostering innovation and experimentation in the field of robotics and missile systems. The project has significant educational potential, serving as a hands-on learning tool for individuals interested in electronics, programming, and engineering. It can inspire curiosity and skill development in STEM disciplines. Arduino's open-source nature allows for customization and adaptation, enabling users to modify the system for various applications and learning objectives. The use of ultrasound for aiming provides an alternative to traditional optical systems, potentially offering advantages in environments with visibility challenges.

In summary, an Arduino-powered missile system using ultrasound for aiming holds promise as an educational and innovative project. However, developers and users must navigate safety, regulatory, and ethical considerations, while also acknowledging the inherent limitations of the chosen technology. Responsible development and application are essential for ensuring the project's success and avoiding potential risks and challenges.