Development of a Sonar System using Arduino Mega: SonarDuino

By:

Shantanu Ghodgaonkar	-	sng8399, N11344563
Aditya Abishai Pedapati	-	aap9224, N15101231
Mihir Kshirsagar	-	msk9917, N16144406

New York University, Tandon School of Engineering
Department of Mechanical and Aerospace Engineering
Mechatronics ROB-GY 5103

Abstract

"SonarDuino" is an innovative project that delves into the realm of robotics, specifically focusing on creating a 360-degree radar system for object detection. The project aims to enhance the capabilities of locomotion robots by providing them with the ability to detect the boundaries of their surroundings. This project demonstrates the use of an ultrasonic sensor as a **SOund NAvigation and Ranging** equipment with the use of an Arduino Mega for processing and collecting data. Visualization of objects detected by the system is presented through a GUI (replicating a SONAR scanning system).

Acknowledgements

We extend our heartfelt gratitude to Professor **William Zhiren Peng**, PhD, for his invaluable mentorship and scholarly guidance throughout our academic journey. His expertise and dedication have inspired us, shaping our understanding and passion for the subject.

We also sincerely thank our Teaching Assistants, **Ajey Jagannathan** and **Satyapalsinh Gohil**. Their unwavering support, insightful feedback, and commitment to fostering a conducive learning environment have been instrumental in our academic success.

To our teammates, **Shantanu Ghodgaonkar**, **Mihir Kshirsagar**, and **Aditya Abishai P** we appreciate the collaborative effort and camaraderie that made our academic endeavors even more rewarding.

This journey would not have been possible without the support and encouragement of these exceptional individuals. Thank you for being constant pillars of knowledge and inspiration in our academic endeavors."

Table of Contents

1. Introduction

1.1. Mechatronics

Mechatronics is a domain that provides more efficient systems than Mechanical systems by integrating the use of Electronics and electromechanical systems. Sensors and actuators are mechatronic devices as a result of integrating Mechanical and Electronic systems.

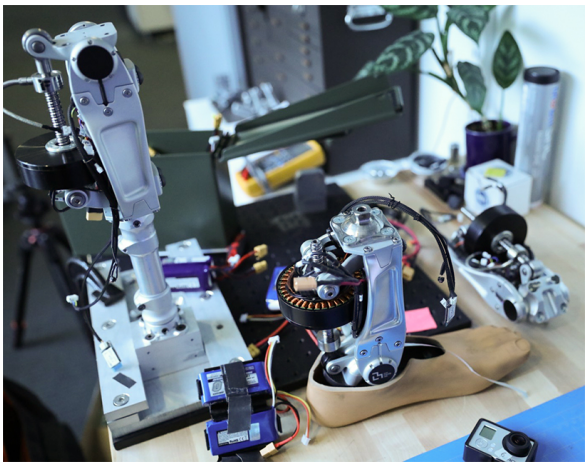

Figure 1: Example of Mechatronic Systems

1.2. Arduino Mega

Microcontroller is an Integrated Circuit that incorporates a processing core, memory and programmable peripherals, that is designed to perform specific tasks, usually used in embedded systems. Arduino is a development board platform wherein the Arduino Mega uses an ATmega2560 microcontroller. It is programmable with up to 54 digital input/output pins.

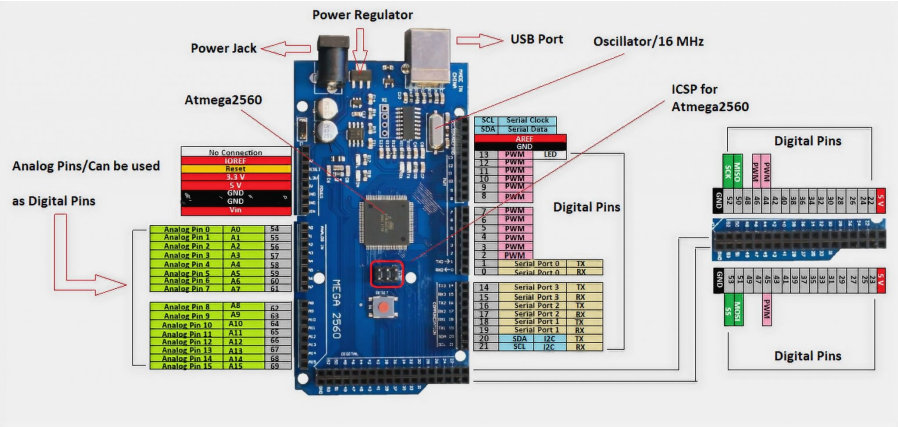

Figure 2: Arduino Mega PinOut Diagram

1.3. SONAR

SONAR, which stands for Sound Navigation and Ranging, is a technology that uses sound waves to navigate, communicate, and detect objects underwater. It plays a crucial role in various applications, including military, commercial shipping, fisheries, and oceanography. In robotics, SONAR (Sound Navigation and Ranging) is a valuable technology used for object detection, mapping, and navigation. Similar to its application in marine environments, SONAR in robotics employs sound waves to sense the surrounding environment. Here are several aspects of how SONAR is utilized in robotics:

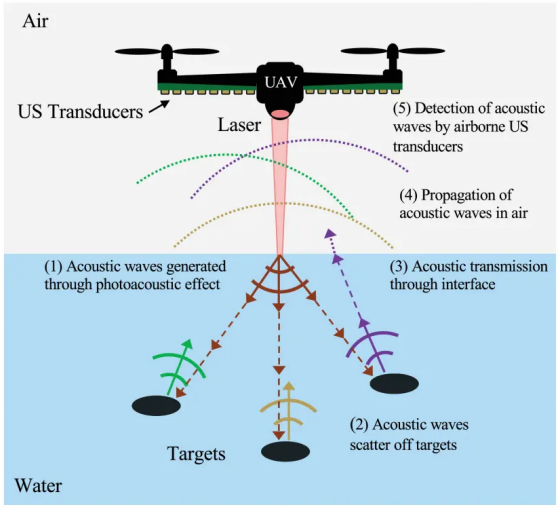

Figure 3: Working of SONAR systems

A. Object Detection and Avoidance:

- *Ultrasonic Sensors:* Many robotic systems use ultrasonic sensors that emit ultrasonic waves and measure the time it "takes for the waves to reflect off an object and return. This information helps robots detect obstacles and avoid collisions.

B. Navigation and Mapping:

- *Simultaneous Localization and Mapping (SLAM):* Some robotic systems use SONAR sensors as part of SLAM algorithms. These algorithms enable a robot to simultaneously create a map of its environment and determine its own location within that environment.

C. Autonomous Vehicles:

- *Robotics Vehicles:* SONAR sensors are often integrated into autonomous vehicles, such as robotic vacuum cleaners or drones, to navigate and avoid obstacles during movement.

D. Underwater Robotics:

- *ROVs and AUVs:* Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs) use SONAR for underwater exploration. SONAR helps them navigate, map the seafloor, and locate underwater objects.

E. Human-Robot Interaction:

- *Proximity Sensing:* SONAR sensors in robots can be used for proximity sensing, enhancing safety in human-robot collaborative environments by detecting the presence of humans or other objects.

F. Assistive Robotics:

- *Navigation for the Visually Impaired:* SONAR technology is sometimes used in assistive robotics to aid visually impaired individuals. It helps detect obstacles and provides feedback to the user about their surroundings.

G. Advantages of SONAR in Robotics:

- *Robustness:* SONAR works well in various environmental conditions, including darkness and low visibility.
- *Low Power Consumption:* Ultrasonic sensors often have lower power requirements compared to some other sensing technologies.
- *Real-time Feedback:* SONAR provides real-time feedback, making it suitable for dynamic and changing environments.

1.4. Ultrasonic Sensor (Parallax PING)

The Parallax PING))) Ultrasonic Sensor is a popular ultrasonic range finder used in robotics and automation applications. It utilizes ultrasonic sound waves to measure distances and is commonly employed for object detection, navigation, and avoidance in various projects. Here are key features and details about the Parallax PING))) Ultrasonic Sensor:

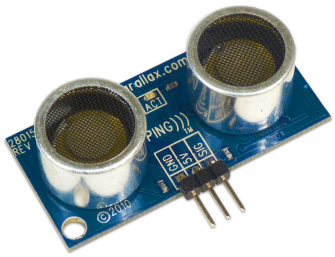

Figure 4: Ultrasonics Sensor (Parallax PING))))

Key Features:

Operating Principle:

- The sensor operates based on the time-of-flight principle. It sends out ultrasonic pulses and measures the time it takes for the pulses to bounce back after hitting an object.

Distance Measurement:

- It can measure distances in the range of a few centimeters up to several meters.

Ultrasonic Transducer:

- The sensor consists of an ultrasonic transducer, which both emits and receives ultrasonic waves.

Resolution:

- The resolution of the PING sensor is typically around 1 inch (2.54 cm), providing relatively fine distance measurements.

Interface:

- It typically interfaces with microcontrollers or single-board computers using a simple digital or analog interface.

Output:

- The sensor outputs distance measurements as pulse-width-modulated (PWM) signals or analog voltage signals, depending on the model.

Power Supply:

- It is powered using a standard voltage supply (usually 5V DC) and consumes low power.

Field of View:

- The PING sensor has a wide-angle field of view, allowing it to detect objects within a broad area.

Application Range:

- Commonly used in robotics for applications such as obstacle avoidance, distance measurement, and navigation.

1.5 Working of Ultrasonic Sensor

Triggering:

- The sensor is triggered by sending a short pulse (typically 2 microseconds) to its trigger pin.

Ultrasonic Pulse Emission:

- Upon receiving the trigger pulse, the sensor emits a burst of ultrasonic pulses.

Echo Reception:

- The sensor listens for the echo of the emitted pulses. The echo signal is received on the echo pin.

Time Measurement:

- The microcontroller measures the time it takes for the ultrasonic pulses to travel to the object and back.

Distance Calculation:

- Using the speed of sound in air, the distance to the object is calculated based on the time-of-flight of the ultrasonic pulses.

2. Methodology

1. Hardware Components:

The foundational hardware components of the Sonarduino project encompass the Arduino Mega microcontroller, a servo motor, and an ultrasonic sensor. The Arduino Mega serves as the central processing unit, executing the programmed logic and orchestrating the interactions between the servo motor and ultrasonic sensor. The servo motor, a key actuator in the system, is responsible for executing a 180-degree rotation, enabling the ultrasonic sensor (which is attached to the servo motor) to scan the surrounding environment. The ultrasonic sensor, functioning as the primary sensor component, emits ultrasonic pulses and measures the time taken for the echoes to return, thereby determining the distance to objects in its path by the Time-of-Flight method. Single Pole Single Throw buttons are used to change the speed of rotation of the servo motor and also emergency shut down its movement. Appropriate resistors are connected to prevent inappropriate current.

2. Software Implementation:

The programming logic for the Arduino Mega is developed using the Arduino Integrated Development Environment (IDE), which facilitates the creation, editing, and uploading of code to the microcontroller. The code encompasses routines for servo motor control, ultrasonic sensor interfacing, and the coordination of their activities. The servo motor is programmed to travel 180 degrees and reverse the direction of rotation. The time required for the sound wave of the ultrasonic sensor to be emit, reflect and return to the receiver is recorded and the distance between the sensor and the object is calculated by (speed of sound = 343 m/s) * (Total time taken for sound waves to emit, reflect and receive). Objects detected further than 40 cm away were disregarded for accuracy purposes. When the emergency stop button is pressed, an interrupt signal is sent to the interrupt pin on the Arduino which ceases the servo motor operation. To change the speed of rotation of the servo motor, two buttons are provided to increase or decrease the speed. The speed increment button decreases the delay constant in the servo motor’s FOR loop to increase the speed.

3. Mechanical Setup:

To avoid damage to the Arduino Mega, it is encompassed in a 3D printed enclosure. The mechanical setup involves mounting the servo motor on a 3D printed case attached to the Arduino's case, allowing for a controlled 180-degree sweep and not requiring any direct contact between the Arduino and motor so it can be repurposed. The ultrasonic sensor is fixed to the servo motor, ensuring its synchronized movement during the scanning operation. This setup enables the sonar system to systematically cover its surroundings, capturing distance data at different angles for a comprehensive assessment of the environment. The speed control and emergency stop buttons are placed above the Arduino enclosure for ease of access.

4. Calibration and Testing:

Calibration procedures are conducted to align the servo motor and ultrasonic sensor, ensuring accurate correspondence between the servo's rotational position and the sensor's scanning direction. Additionally, distance measurements are calibrated to account for any discrepancies in the ultrasonic sensor readings.

5. Safety Protocols:

Safety protocols are implemented to mitigate potential risks associated with the hardware components, particularly the moving parts of the servo motor. An Emergency Stop button is implemented in order to stop the movement of the servo motor to prevent it from exceeding its mechanical limits, ensuring the integrity of the system and minimizing the risk of damage. The maximum speed of the servo motor is limited in a safe range.

In summary, the methodology adopted for the Sonarduino project encompasses the selection and integration of hardware components, the development of software logic, the establishment of a mechanical setup, calibration and testing procedures, the implementation of closed-loop control, and the incorporation of safety protocols. This systematic approach forms the basis for the successful execution of the project, allowing for the reliable operation of the sonar system in scanning and mapping its surrounding environment.

3. Bill Of Materials

No.	Components	Unit Cost	Quantity	Total Cost
1	Arduino Mega	21 USD	1	21 USD
2	Ultrasonic Sensor	35 USD	1	35 USD
3	Servo Motor	1.5 USD	1	1.5 USD
5	Pushbuttons	0.1 USD	3	0.3 USD
6	Resistors (10K)	0.1 USD	3	0.3 USD
7	3D Printed Casing	0.02 USD/gram	103 grams	2.06 USD
8	**TOTAL COST**			**60.16 USD**

4. System Design

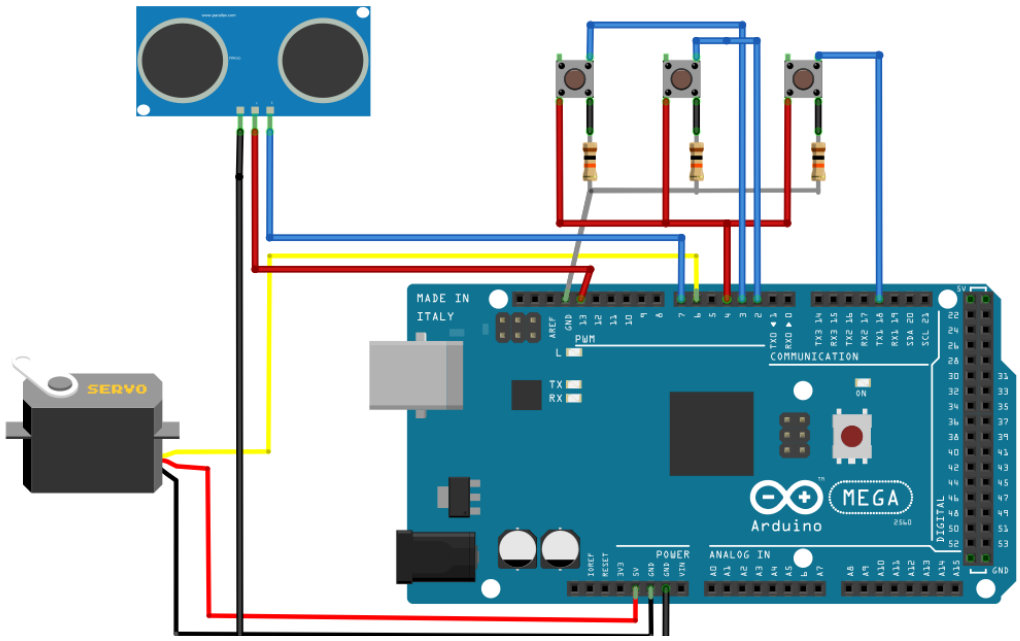

Figure 4: Connections of the components with the Arduino Mega

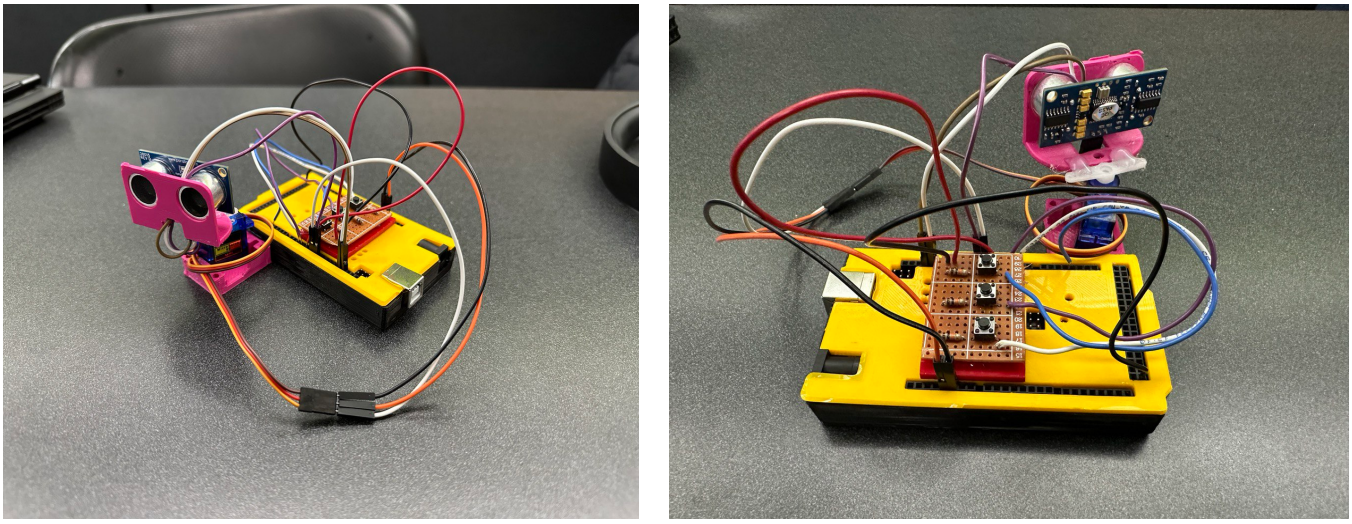

Figure 5: All components assembled with 3D printed holders and wired

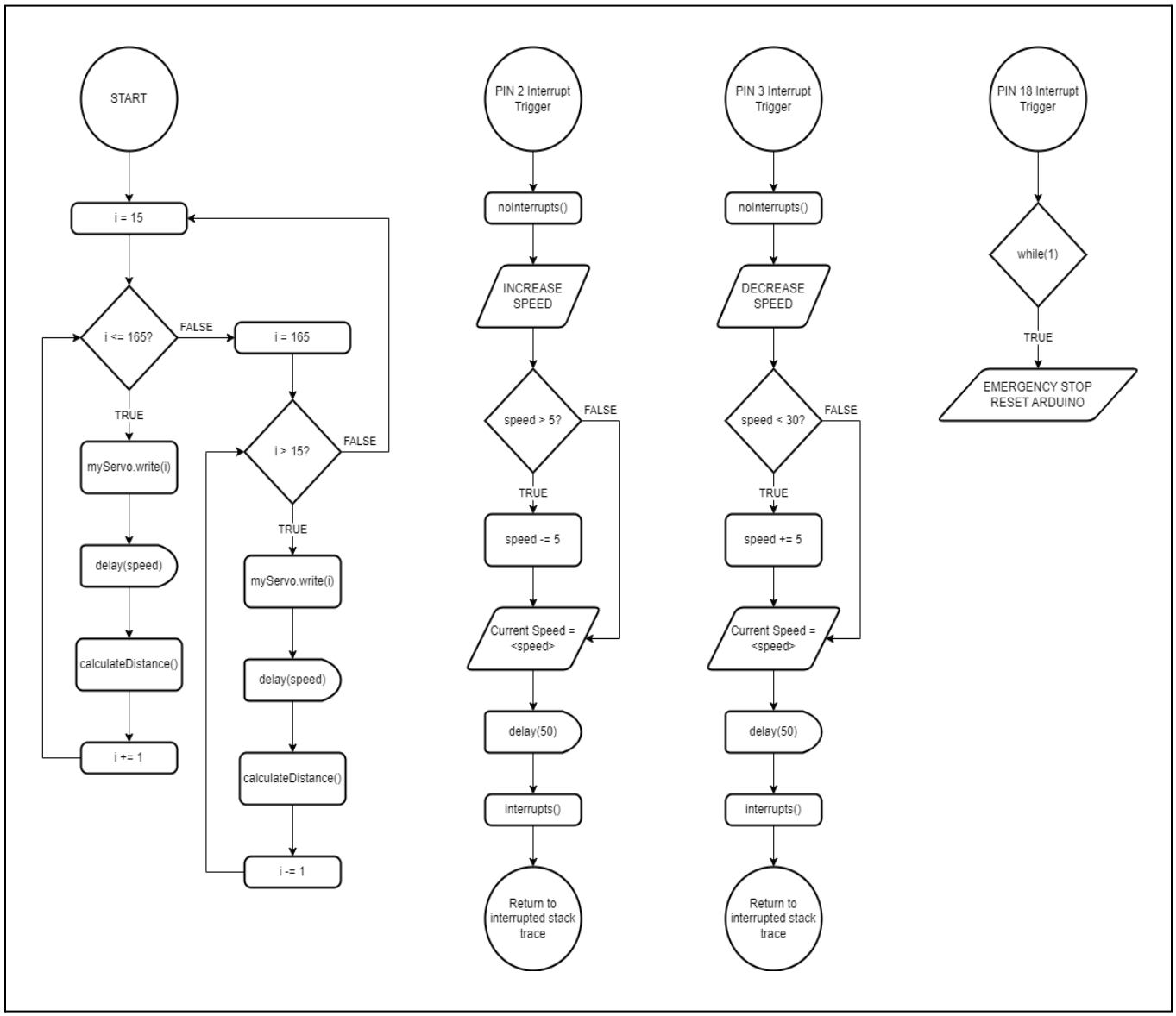

Figure 6: Arduino Program Flowchart

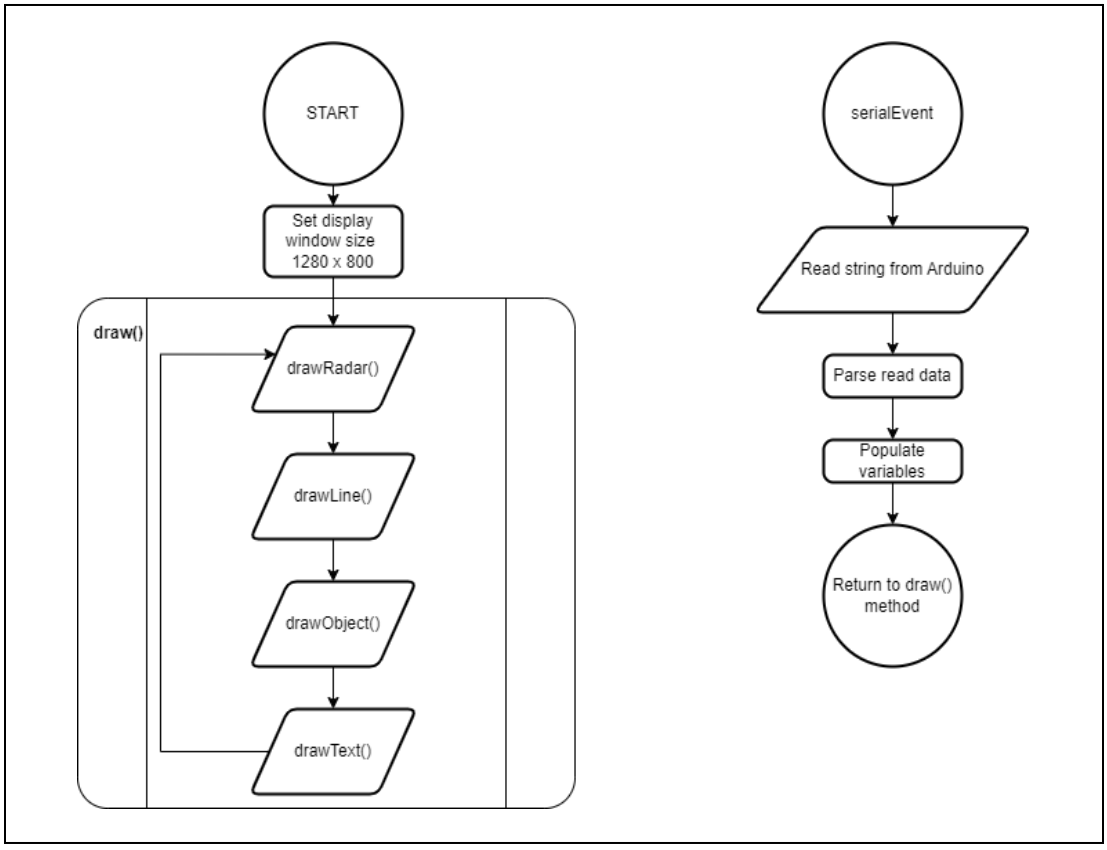

Figure 7: Sonar Display Program Flowchart

5. Operation

1. Switching On:

The Arduino Mega is connected to a laptop which serves as a power source. The connection is with a USB cable capable of transmitting power and data (This needs to be performed only once).

2. Uploading code to Arduino:

Ensure the USB cable is connected and then open the Arduino IDE. Select the correct port and then connect and upload the code to the Arduino Mega. The Arduino can then be disconnected and instead be connected to another power supply appropriate for the system to run the 9V servo motor and the Arduino.

3. Changing Scanning Speed:

The program immediately starts rotating the servo motor to scan its surroundings with the ultrasonic sensor. To change the speed of the servo motor, one of the two buttons is used to increase the speed while the other is used to decrease it. The maximum speed it can reach is within a safe range of operation.

4. Graphical Representation:

Program for visualization has been coded in Java and linked with Arduino IDE using Processing IDE. Green lines indicate absence of an obstacle while red lines indicate that an obstacle is detected. An obstacle is detected within 40 cm from the ultrasonic sensor.

5. Emergency Stop:

This functionality is provided by a push button that stops rotation of the servo motor. The need to use this switch can occur when the ultrasonic sensor is to be tangled, damaged, or if the program is to be stopped.

6. Source Code

6.1. Arduino Source Code -

```
// Link to code hosted on GitHub - click here

#include <Arduino.h>
#include <Servo.h>

#define inc 2 // Sampling Rate Increase Interrupt Controlled Button Pin
#define dec 3 // Sampling Rate Decrease Interrupt Controlled Button Pin
#define stp 18 // Emergency Stop Interrupt Controlled Button Pin
#define pingPin 7 // PING))) Sensor I/O Digital Pin
#define servo 6 // Servo Motor Control Pin
int speed = 15; // Default Sampling rate
// Variables for the duration and the distance
long duration;
int distance;
int i;
Servo myServo;
void setup() {
// Setting I/O Pin Modes
  pinMode(servo, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(inc, INPUT);
  pinMode(dec, INPUT);
  pinMode(stp, INPUT);
// Attaching interrupts to buttons
  attachInterrupt(digitalPinToInterrupt(2), printInc, FALLING);
  attachInterrupt(digitalPinToInterrupt(3), printDec, FALLING);
  attachInterrupt(digitalPinToInterrupt(18), stop, FALLING);
  Serial.begin(9600); // Setting Baud Rate
  myServo.attach(servo); // Attaching servo motor pin
}
void loop() {
// Setting PIN 13 and 4 to HIGH as they act as power supply for PING))) and Buttons
  digitalWrite(13, HIGH);
  digitalWrite(4, HIGH);
```

```
  // rotates the servo motor from 15 to 165 degrees
  for (int i = 15; i <= 165; i++) {
    myServo.write(i);
    delay(speed);
        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
      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
  }
  // Repeats the previous lines from 165 to 15 degrees
  for (int i = 165; i > 15; i--) {
    myServo.write(i);
    delay(speed);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance() {

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH pulse
```

```
  // whose duration is the time (in microseconds) from the sending of the ping
  // to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);
  distance = duration / 29 / 2;
  return distance;
}

// ISR for increasing sampling rate by decreasing the delay in servo motor loop
void printInc() {
  noInterrupts(); // disable all interrupts for debouncing
  Serial.println("INCREASE SPEED");
  if (speed > 5) speed -= 5;
  Serial.print("Current Speed: ");
  Serial.println(speed);
  delay(50);
  interrupts(); // enable all interrupts
}
// ISR for decreasing sampling rate by decreasing the delay in servo motor loop
void printDec() {
  noInterrupts(); // disable all interrupts for debouncing
  Serial.println("DECREASE SPEED");
  if (speed < 30) speed+= 5;
  Serial.print("Current Speed: ");
  Serial.println(speed);
  delay(50);
  interrupts(); // enable all interrupts post execution
}
// ISR for Emergency Stop
void stop() {
  while (1) {
    Serial.println("999,999.");
    delay(1000);
  }
}
```

6.2. Sonar Display Source Code

```
// Link to code hosted on GitHub - click here
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 (1280, 800); // Setting default shell size
 smooth();
 myPort = new Serial(this,"COM3", 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() {  // The “draw” function is akin to the “loop” function in Arduino code
  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 that's the value of the distance

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {// Function to draw the radar background
  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() { // Function to draw the object in red coloured lines upon detection
  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() { // Function to draw the green line sweeping across the radar background
  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();
}
```

7. Results

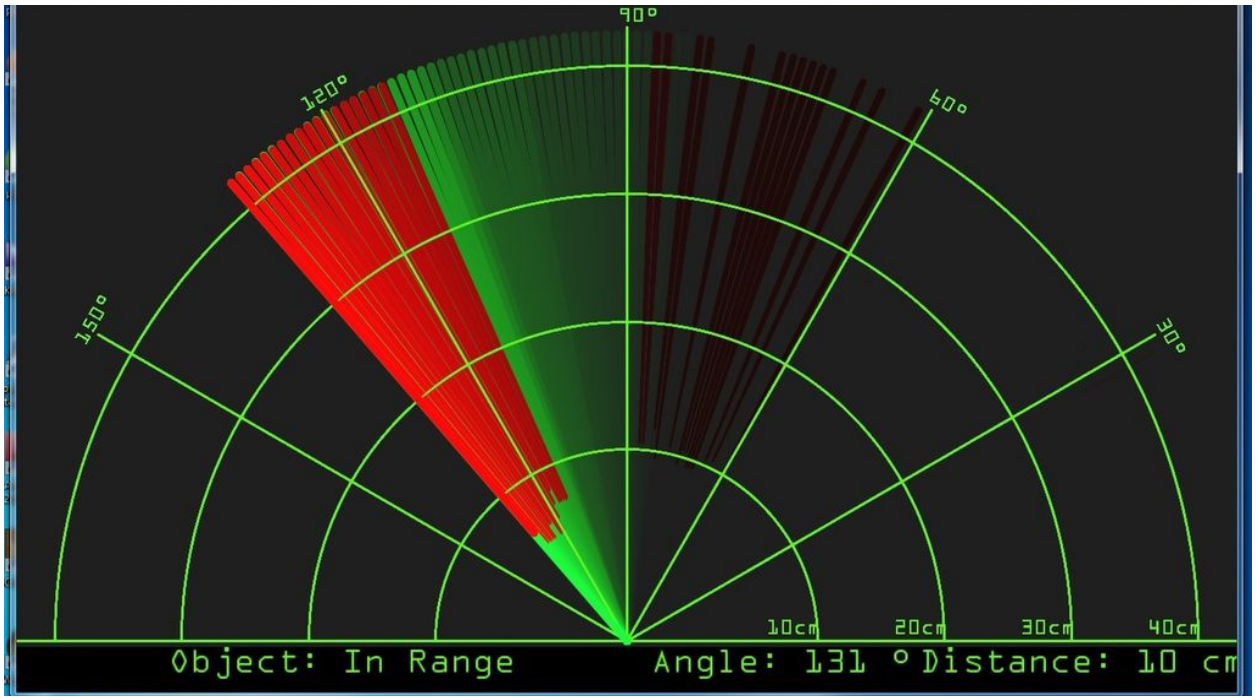

Figure 8: SonarDuino display

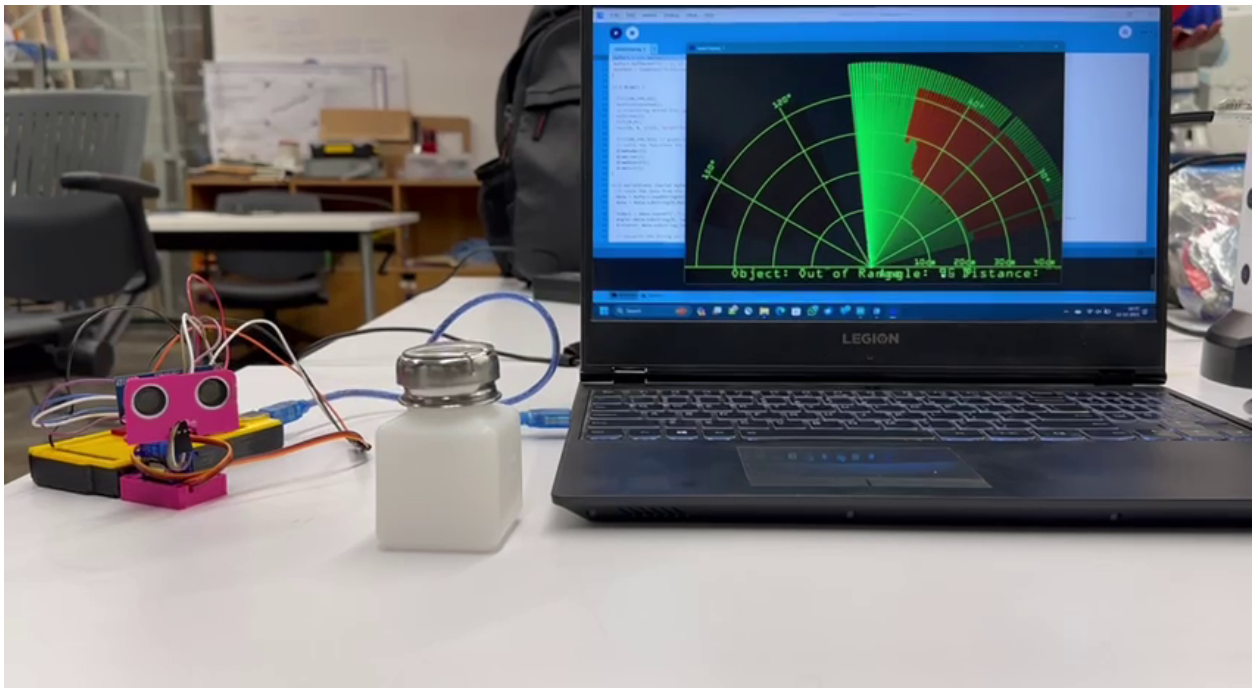

Figure 9: SonarDuino in action

8. Advantages and disadvantages

Advantages

- Sonar systems use ultrasonic sound frequencies that are outside of human hearing. Hence, these systems are not easily detectable by human ears and are more stealthy in operation.
- These systems are cost effective to manufacture on a large scale, and hence cheaper to purchase.
- Main implementation of Sonar is in underwater detection of objects. As the medium of water is thicker than that of air, sound waves propagate longer distances underwater.

Disadvantages

- Ultrasonic sensor's microphone has low directional resolution as increasing it may also introduce more noise.
- Interfering sound waves will produce noise in the system causing high scattering.
- LiDAR systems are costly but will produce more accurate measurements in air.

9. Conclusion

In conclusion, the Sonarduino project exemplifies the successful integration of mechatronics principles in the development of a sonar system. The Arduino Mega microcontroller, serving as the project's brain, orchestrates the coordinated movement of the servo motor and the precise measurements of the ultrasonic sensor.

The project's outcomes contribute valuable insights to the field of mechatronics, showcasing the practical application of object detection and imaging in enhancing the capabilities of robotic systems. The successful implementation of the Sonarduino project lays a foundation for further exploration into the cohesion of various features for expanded functionality.

Future enhancements to the Sonarduino project could explore the incorporation of machine learning algorithms for adaptive control, enabling the system to adapt to dynamic environments and further enhancing its autonomy and range. Additionally, expanding the sonar system's capabilities, such as multi-sensor integration or real-time data visualization, could offer avenues for more sophisticated applications in underwater exploration, robotics, and automation.

In summary, the Sonarduino project successfully demonstrates the fusion of mechatronics and sonar technology, providing a platform for experimentation, learning, and potential advancements in the dynamic field of robotics and automation.

References

1. Arduino Documentation | Ping Ultrasonic Range Finder : https://docs.arduino.cc/built-in-examples/sensors/Ping
2. Arduino Documentation | Servo Motor Basics with Arduino : https://docs.arduino.cc/learn/electronics/servo-motors
3. Arduino Documentation | Reference > Language > Functions > External interrupts > attachInterrupt() : https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
4. Sparkfun Tutorials | Connecting Arduino to Processing : https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing/all
5. Thingiverse | Arduino Mega 2560 Snug Case : https://www.thingiverse.com/thing:1145811
6. Thingiverse | Servo Motor Case : https://www.thingiverse.com/thing:4076030