A
REPORT
ON

Tracking Objects Using Ultrasonic Sensors
Designing a 360° Ultrasonic Radar

INSTR F312: Transducers and Measurement Techniques

Prepared for
Dr. Puneet Mishra

By

Praroop Garg	**2021A8PS2699P**
Somin Ranjan	**2021A8PS2704P**
Nitin Aravind Birur	**2021A8PS2698P**
Jatin Gupta	**2020B4A81989P**

At

Birla Institute of Technology and Science Pilani (Rajasthan)

November, 2023

Acknowledgment

We would like to express our gratitude to Dr. Puneet Mishra and the entire faculty of the Transducers and Measurement Systems (INSTR F312) course for their invaluable advice during this project. We have gained knowledge and an understanding of how to apply what we have learned to address problems in the actual world through this project. We intend to carry out related initiatives in the future.

Table of Contents

Aim of the project undertaken

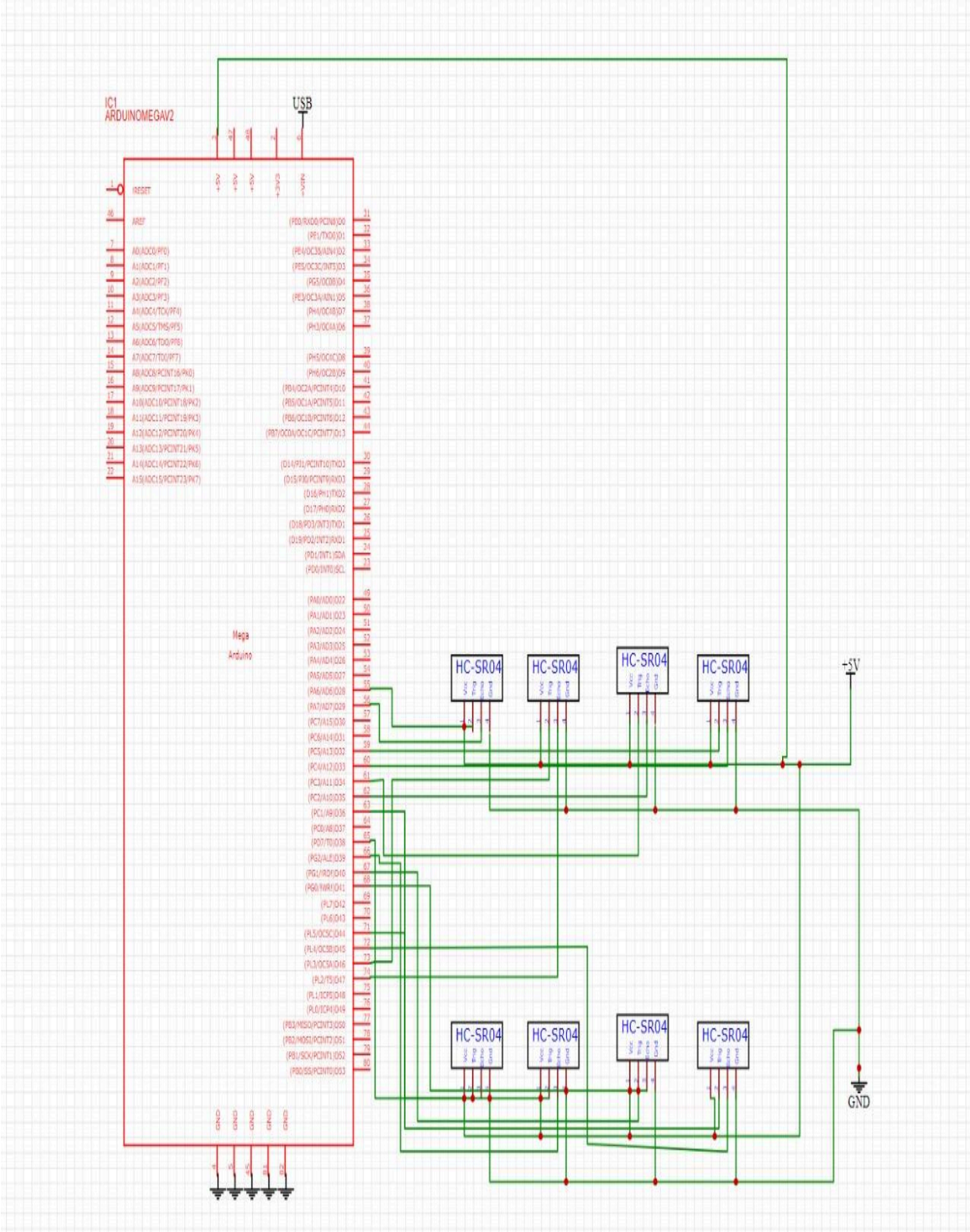
Real time measuring of the objects kept nearby

Components used:

Here is the list of components used in the fulfillment of the project:

Component	Quantity
HC-SR04 ultrasonic sensor	8
Arduino Mega 2560	1
Jumper wires	100

Circuit Schematic:



Purpose and Significance of components in the circuit

This section discusses the functions of the circuit's principal parts.

Ultrasonic Sensor

An ultrasonic sensor is a device that measures an object's distance or presence using ultrasonic sound waves. It operates on the basis of projecting ultrasonic waves and timing how long it takes for the waves to return after colliding with an object.
Here we have used an HC-SR04 ultrasonic sensor.
Specifications of the HC-SR04 ultrasonic sensor are given below.

Operating voltage	5 V DC
Operating current	15 mA
Minimum range	2 cm
Maximum range	400 cm
Measuring angle	< 15°
Accuracy	3 mm
Working frequency	40 KHz
Dimensions	45 x 20 x 15mm
Trigger Input Signal	10uS TTL pulse
Echo Output Signal Input	TTL lever signal and the range in proportion

Arduino Mega 2560

Arduino is an open-source electronics platform is built on user-friendly hardware and software. Arduino Mega 2560 is a microcontroller board based on ATmega2560. Arduino calculates the time taken by the ultrasonic pulse wave to reach the receiver from the sender.
Specifications of Arduino Mega 2560 are given below:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12
Input Voltage (limit)	6-20
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 kB
EEPROM	4 kB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 gm

Jumper wires

We have used male to female jumper wires. In electronic prototype and testing applications, male to female jumper wires are used to connect components quickly and easily without the need for soldering. We have used jumper wires to connect the ultrasonic sensors to the Arduino board.

Working

The project consists of eight HC-SR04 ultrasonic sensors arranged in a ring, equally spaced and connected to an Arduino Mega. The HC-SR04 ultrasonic sensors are used to measure the distance to objects in the surrounding environment by using the principle of echolocation.

Each sensor has a trigger pin and an echo pin. The trigger pin is used to send a high-frequency sound wave (typically 40 kHz) to the environment. The echo pin is used to receive the reflected sound wave from the object. The time between the transmission and reception of the sound wave is proportional to the distance between the sensor and the object.

The Arduino Mega is the main controller of the project. It is responsible for initializing the measurement by sending a 10 µs pulse to the trigger pin of each sensor. It is also responsible for reading the echo pin of each sensor and calculating the distance to the object using the formula:

distance = (speed of sound * time) / 2

where speed of sound is 343 m/s at 20°C and time is the duration of the echo pin being high.

The Arduino Mega can then use the distance values from each sensor to perform various tasks, such as displaying them on a screen, sending them to a computer, or controlling other devices. The project can be used for applications such as obstacle detection, robot navigation, or gesture recognition.

Code in python:

```
import serial
import re
import math
import cv2
import numpy as np
from PIL import Image, ImageDraw

arduino_port = '/dev/cu.usbmodem101'
ser = serial.Serial(arduino_port, 9600, timeout=1)

NUM_SENSORS = 8
```

```
SENSOR_RANGE = 30
THRESHOLD_DISTANCE = 10
NUM_MEASUREMENTS = 10

last_measurements = {i: [] for i in range(NUM_SENSORS)}

def read_distances():
    line = ser.readline().decode('utf-8')
    match = re.match(r'Distances: \[(.*)\]', line)
    if match:
        distances_str = match.group(1)
        distances = [min(int(dist), SENSOR_RANGE) for dist in distances_str.split(', ')]
        return distances
    else:
        return None

def update_last_measurements(sensor_index, distance):
    measurements = last_measurements[sensor_index]
    measurements.append(distance)
    if len(measurements) > NUM_MEASUREMENTS:
        measurements.pop(0)

def get_average_distance(sensor_index):
    measurements = last_measurements[sensor_index]
    if measurements:
        return sum(measurements) / len(measurements)
    else:
        return None

def create_radar_image(distances):
    image_size = 600
    img = Image.new('RGB', (image_size, image_size), (0, 0, 0))
    draw = ImageDraw.Draw(img)
    center = (image_size // 2, image_size // 2)

    for r in range(50, image_size // 2, 50):
        start_angle = 0
        end_angle = 360
        num_points = 100
        for angle in np.linspace(start_angle, end_angle, num_points):
```

```
            x = int(center[0] - r * math.cos(math.radians(angle)))
            y = int(center[1] + r * math.sin(math.radians(angle)))
            draw.point((x, y), fill=(100, 100, 100))

    for i in range(len(distances)):
        angle = -90 + i * (360 / NUM_SENSORS)
        x = int(center[0] - (image_size // 2 - 20) * math.cos(math.radians(angle)))
        y = int(center[1] + (image_size // 2 - 20) * math.sin(math.radians(angle)))
        draw.text((x, y), str(NUM_SENSORS - i), fill=(100, 100, 100))
        draw.point((x, y), fill=(255, 0, 255))

    for i, distance in enumerate(distances):
        normalized_distance = min(distance, SENSOR_RANGE) / SENSOR_RANGE
        angle = -90 + i * (360 / NUM_SENSORS)
        radius = normalized_distance * (image_size // 2 - 50)
        x = int(center[0] - radius * math.cos(math.radians(angle)))
        y = int(center[1] + radius * math.sin(math.radians(angle)))
        x = image_size - x
        line_color = (0, 0, 255) if distance <= THRESHOLD_DISTANCE else (0, 255, 0)

        update_last_measurements(i, distance)
        avg_distance = get_average_distance(i)

        draw.line([center[0], center[1], x, y], fill=line_color, width=3)

        if avg_distance is not None:
            avg_radius = avg_distance / SENSOR_RANGE * (image_size // 2 - 50)
            avg_x = int(center[0] - avg_radius * math.cos(math.radians(angle)))
            avg_y = int(center[1] + avg_radius * math.sin(math.radians(angle)))
            avg_x = image_size - avg_x
            draw.point((avg_x, avg_y), fill=(255, 0, 255))
            draw.text((avg_x, avg_y), f"{int(avg_distance)}", fill=(255, 255, 255))

    rotated_img = img.rotate(90)

    return np.array(rotated_img)

try:
    while True:
        distances = read_distances()
```

```
        if distances is not None:
            print("Distances:", distances)
            radar_img = create_radar_image(distances)
            cv2.imshow('TMT Assignment - 2D SONAR Room Scanner', radar_img)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

except KeyboardInterrupt:
    ser.close()
    print("Serial port closed.")

cv2.destroyAllWindows()
```

Arduino Code:

```
const int numSensors = 8;

const int trigPins[numSensors] = {28,36,32,40,46,44,38,34};
const int echoPins[numSensors] = {29,37,33,41,47,45,39,35};

long durations[numSensors];
int distances[numSensors];

void setup() {
  for (int i = 0; i < numSensors; i++) {
    pinMode(trigPins[i], OUTPUT);
    pinMode(echoPins[i], INPUT);
  }
  Serial.begin(9600);
}
void loop() {
  for (int i = 0; i < numSensors; i++) {
    digitalWrite(trigPins[i], LOW);
    delayMicroseconds(2);
    digitalWrite(trigPins[i], HIGH);
    delayMicroseconds(2);
    digitalWrite(trigPins[i], LOW);
```

```
    durations[i] = pulseIn(echoPins[i], HIGH);

    distances[i] = durations[i] * 0.034 / 2;
  }

  Serial.print("Distances: [");
  for (int i = 0; i < numSensors; i++) {
    Serial.print(distances[i]);
    if (i < numSensors - 1) {
      Serial.print(", ");
    }
  }
  Serial.println("]");
  delay(500);
}
```

Possible practical use of the project:

Real world applications would require more advanced setup. The applications listed below will contain the project as a basic unit.

1. Distance measurement: Ultrasonic sensors can measure the distance to an object by calculating the time it takes for the sound waves to travel to the object and back.
2. Object detection and avoidance: These sensors are used in robotics and automation to detect the presence of objects and avoid collisions.
3. Parking assistance systems: In automotive applications, ultrasonic sensors are often used in parking sensors to detect obstacles around a vehicle.
4. Liquid level measurement: Ultrasonic sensors can be employed to measure the level of liquids in tanks.
5. Presence detection and controlling devices based on the proximity of individuals.

References

1. Specifications of the HC-SR04 ultrasonic sensor: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf?_gl=1*8qgne8*_ga*MjQ5

MTg1NDQ2LjE3MDA1NDEzOTM.*_ga_T369JS7J9N*MTcwMDU0MTM5My4xLjAuMTcw
MDU0MTM5My42MC4wLjA.
2. Specifications of Arduino mega 2560:
https://store.arduino.cc/products/arduino-mega-2560-rev3