Technical Report:

# Obstacle-Avoiding Robot Car Project

ENGR-1201 Introduction to Engineering

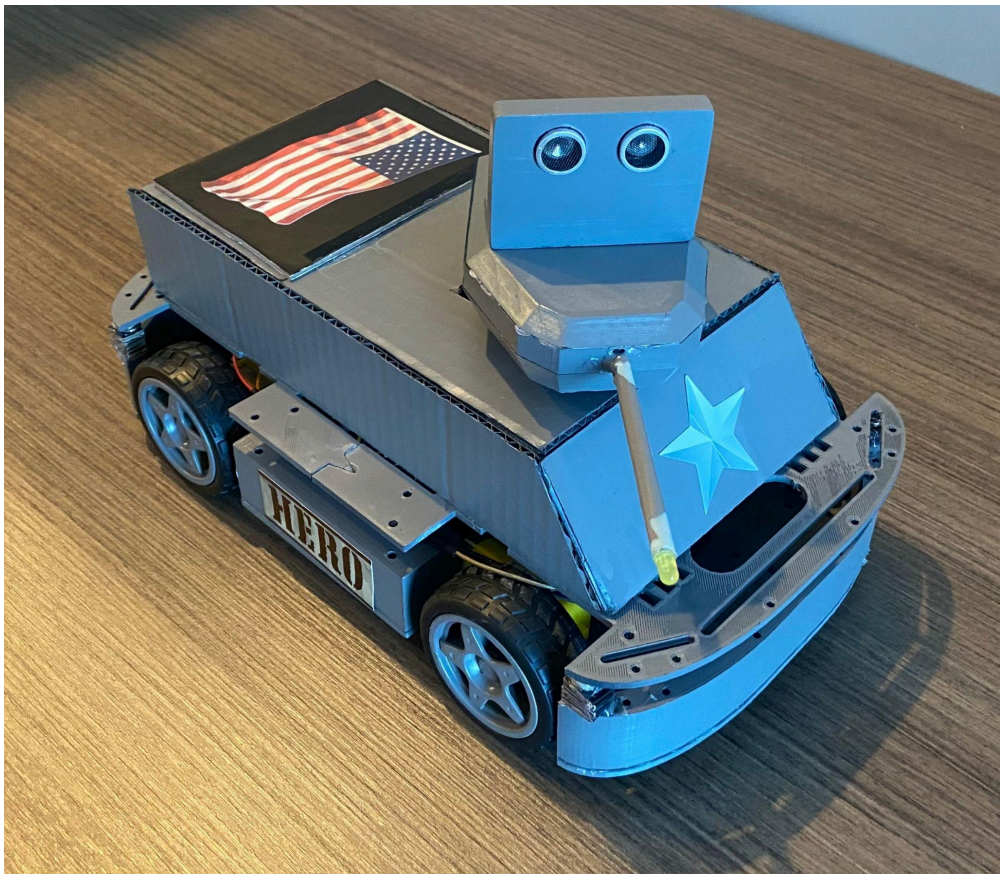
Tasneem Hasem

Justin Pullin

Damian Salinas

Kristina Camacho

10 December 2024

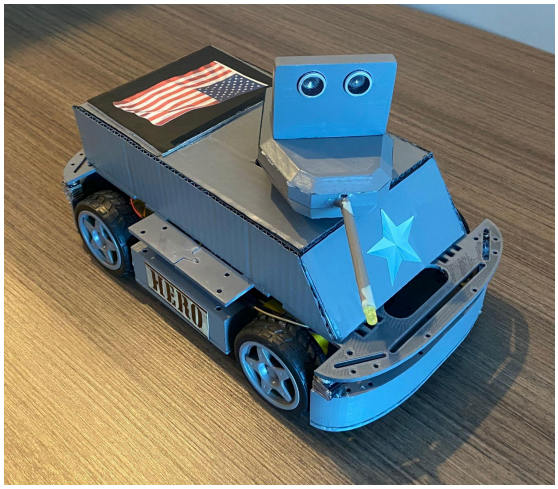


## Summary

The TankBot unfortunately was not successful because the sensor was too high and because the sensor wires kept jamming the servo.

Feasibility was demonstrated for the following features:

- Moved forward smoothly
- Reversed well
- Ultrasonic sensor worked
- Recoil worked
- Tank Hatch and Modular components
- Shooting Sound
- Neutral Turning (Right Turn)
- Turret Swivel




This report concludes with recommendations for improvements to future versions of this robot car.

## Introduction and Product Requirements

We have designed a Tank named H.E.R.O (Heroic Engineering Robotic Object).

In the beginning of the year, after much deliberation our team decided to make a Tank. From there we started to design, program, manage, and finalize the main concept of the vehicle.

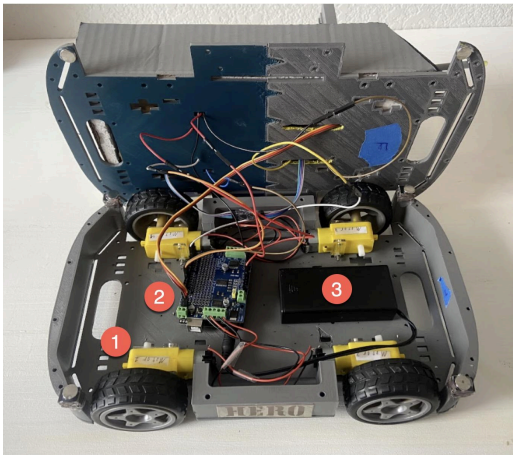
The product requirements were to create a fast moving, durable vehicle that can avoid objects. Our Tank bot includes the following features:

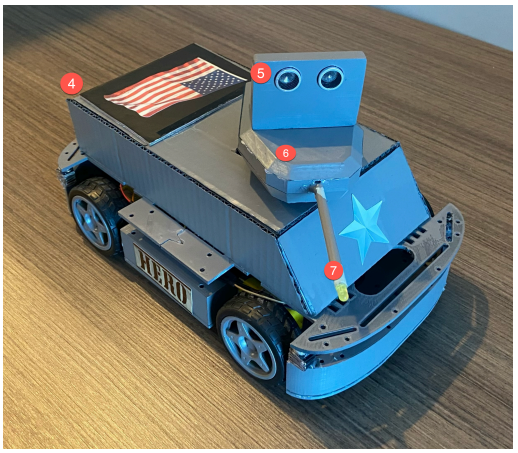
1. Speed
2. Straight Line
3. 90 degree Right turn
4. Durable
5. Turret Swivel
6. Reverse
7. Ultrasonic sensor
8. Recoil
9. Tank Hatch
10. Modular components
11. Shooting Sound
12. LED Light

We also used parts provided from ACC's robot car kits including the Arduino Uno microcontroller and 3D-printed bracket for the ultrasonic sensor.

## Robot Car Components

1. DC Motor
2. Arduino Board
3. Battery/ Power Supply
4. Hatch
5. Ultrasonic sensor
6. Turret
7. Muzzle




<u>Microcontroller</u>

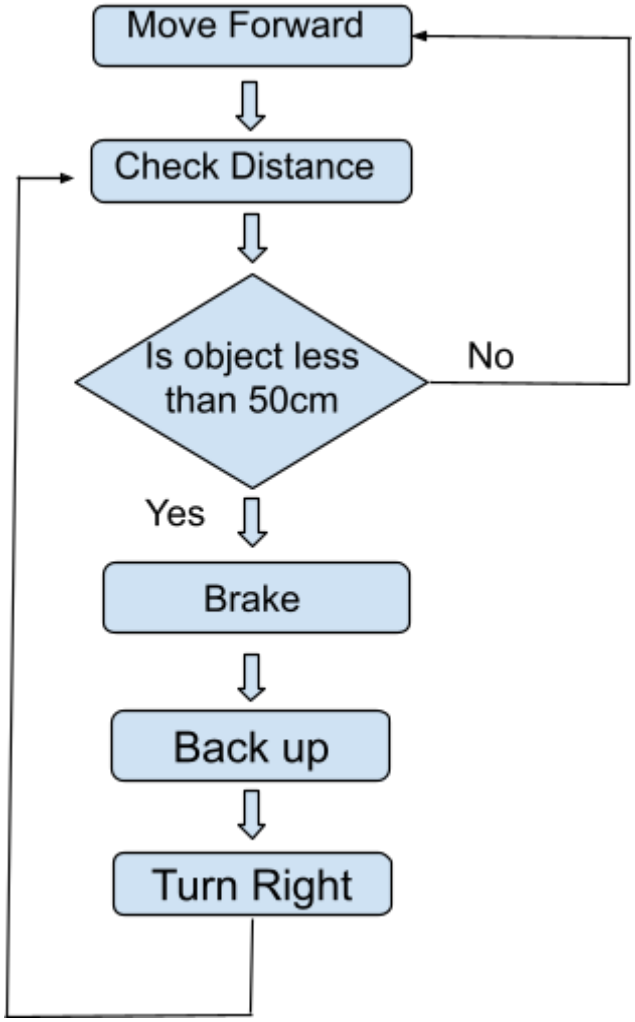
**Board Make & Model:** Arduino UNO R3 ELEGOO

**Memory**: 2KB SRAM, 32KB FLASH, 1KB EEPROM

**Clock Speed:**

- Main Processor: ATmega328P 16 MHz
- USB-Serial Processor: ATmega16U2 16 MHz

**IDE:** Arduino IDE




**Program Flow Chart:**

See appendix A for the complete C++ code

Ultrasonic Sensor

Model number: HC-SR04

The ultrasonic sensor sends out an ultrasonic sound pulse. It measures the distance of an object by multiplying the sound wave's velocity and the time it takes for the sound wave to return to the sensor. It then divides this number by two since the wave traveled that distance two times (once to get there and once to get back). If the object was perpendicular to the sensor, the ultrasonic sensor was very accurate. It struggled when the object was close to the sensor, when the object was soft, or when the object was not perpendicular to the sensor. When the object was close (a couple of centimeters away) the sensor read it as 2-3 cm further than it was. When the object was soft, the sensor could not give an accurate reading at all. WE suspect this is because the soft object absorbed the sound wave. When the sensor was at an angle, the sensor read the distance with a tolerance of ± 3 cm

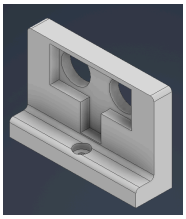
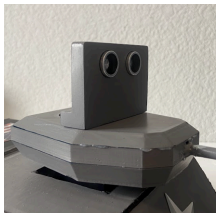
Servo Motor

Model number: MG995

The servo works by turning a specified number of degrees. Since it only turns from 0° to 180°, we set 90° to be straight forward. While moving forward, the servo turned ± 45° from straight forward. Since the range was only from 45°-135°, the car was able to detect objects quicker. The car read the distance at 45°, 90°, and 135°. The larger servo motor took two hundred milliseconds to turn 45°; therefore the car measured the distance every two hundred milliseconds.

Custom-designed bracket

The custom-designed bracket we made was originally made to either screw into any desired location or sit flat on an obstacle. Since we didn't have an exact location for where we wanted to place the ultrasonic sensor we decided to incorporate a hole for a screw and a flat bottom to stick anywhere we chose. With the bracket's dimension, the ultrasonic sensor fits snugly which prevents needing extra parts to keep it intact.

Motor Propulsion

**Make & Model:** DC Motor

**Hardware:** AdaFruit Motor Shield

**Straight Line Speed:** 40 cm/s

**Power:** 3 - 6V

**Continuous No-Load Current:** 150mA +/- 10%

---

| Direction | PWM (Left Wheels/Right Wheels) |
|---|---|
| **Brake** | 0/0 |
| **Forward** | 140/150 |
| **Right Turn** | 140/140 |

**PWM:** 0 - 150

- Set to 0 when stopped
- Set two motors to 140 and two to 150
- This allowed for straight drive precision

---

We also utilized the pulse-width modulation in order to control how the Tank stopped, reversed, turned and drove forward by adjusting the PWM of each wheel. In the table above, the PWM is displayed for each movement ability and the 'speed' or PWM set for each side of the vehicle.

The motors used for propelling the car utilize gears to convert the motor's high speed to low speed with high torque for the Tank wheels.

Our Tank can travel about 40 cm/s and at this speed, the ultrasonic sensor covers a 90 degree sweep (45 degrees looking left and 45 degrees looking right) in 400 milliseconds. This means the Tank covers about 16 cm per ultrasonic sweep.

Power Supply

**Make & Model:** XTAR 18650 2600mAh 4.5A Battery

**Voltage:** 3.7v

**Capacity:** 2600 mAh

**Cars battery life:**

2x Batteries = 5200mAH

Ultrasonic  sensor = 20mA

5mm LED = 20mA

4x DC motors = 160mA

Adafruit motor shield =  600mA

UNO R3 board = 92mA

Active buzzer = 25mA

Servo motor= 550mA

Total run time:

**7 hours**

Other components:

**5mm LED:**

The 5mm LED main purpose was to flash for a quick second at the same time the tank did its recoil movement and shooting noise to represent a fired shot.

**Piezo buzzer:**

The buzzer was used to replicate a firing shot from a tank. It produces a sound by using the reverse piezoelectric effect.

## Testing and Results

| Moving forward | Detecting obstacles | Cars speed |
|---|---|---|
| We tested this by simply measuring how far forward it can move with a meter stick, and making sure it can move in a straight line accurately.<br><br>The results showed that it was not as accurate as we wanted it to be, it kept slightly moving to the left. Resulting in it being slightly obstructed | When the tank is moving, if it detects an obstacle within the distance specified, the tank stops, does a reverse motion, and goes another direction.<br><br>Results were only accurate by a little bit, at the beginning of testing, the tank kept running into things, the reason was that it was jamming the wires that allow it to sense an object. It did not start working till the end. | We tested the car's speed by letting it move in a straight line and seeing how fast it can go past the finish line.<br><br>Results were that the tank was moving 40 cm per second, this shows that it was fast, but not as fast as it could be |
| In the future, we would change the wheels. We think those were one of the reasons why it was not performing as well. And make sure it goes more in a straight line when testing. | Check on the program and make sure it's running smoothly. Change the distance it needs to check at. | Make it faster |

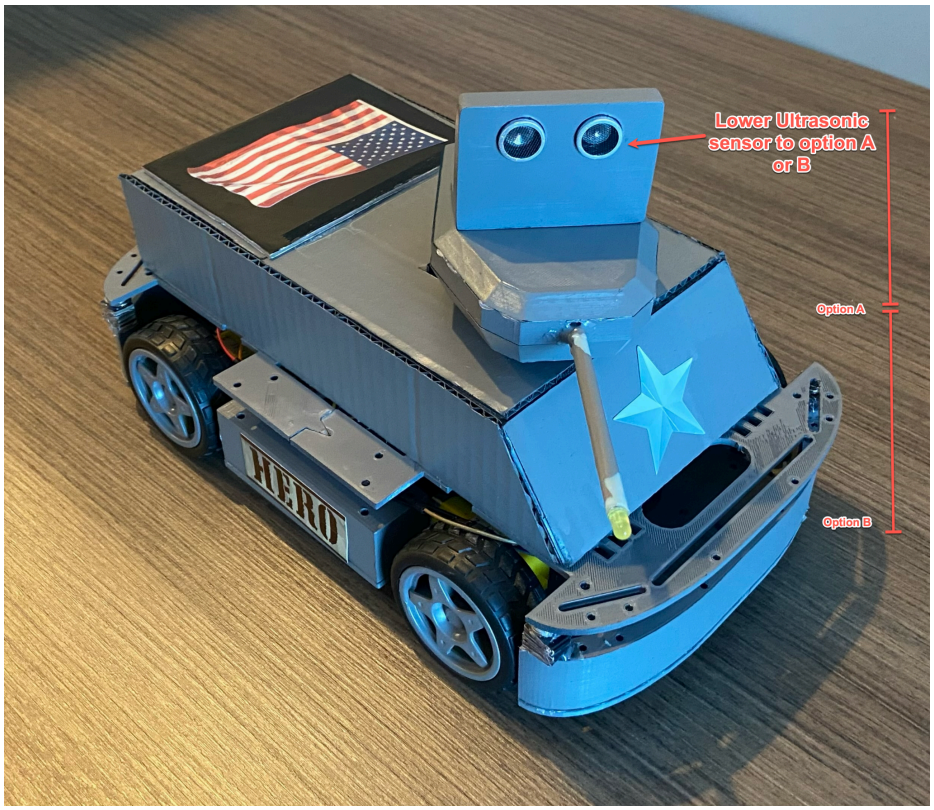
## Conclusions

Throughout the semester, we went through several different designs which would best accomplish our goals. In the end, we decided on a tank. From there, we began programming features and designing parts which would work with ACC's robot car kit. From there we started testing, and seeing what works best. In conclusion, after so much we had an end result just not the one to meet all our expectations. From there, we learn and better ourselves for future projects. This taught us how to better communicate, work collaboratively, and create.

## Recommendations

During the next iteration of Tank Bot H.E.R.O. we recommend adding or considering the following changes:

1. Reduce Sensor Elevation
    a. The original ultrasonic sensor was too high and resulted in missed object detection when objects were below the sensor location
2. Include additional peripheral sensors
    a. Sensor had blind spots in peripherals and lower half of tank
    b. Can be resolved with additional sensors and locations
3. Create a timeline to implement backup plan in case of timing issues
4. Implement backup sooner
    a. Back up plan was implemented late and delayed assembly and testing
    b. Caused last minute delays in troubleshooting unforeseen issues
5. Expect unexpected delays during Holidays
    a. Thanksgiving break caused odd hours for the usage of available 3D printers and training opportunities
6. Consider complexity in relation to time frame available
    a. The Tank Bot build was ambitious for the timeframe and Holidays
    b. Although the build was completed in time the functionality suffered due to lack of time testing after assembly



# Appendix A: C++ code

Below is a listing of the C++ code developed to control the object-avoiding car:

```
#include <Adafruit_MotorShield.h>
// Include additional software 'libraries' to control a servo motor
#include <Servo.h>

Servo myservo;  // create servo object called 'myservo' to control a servo motor

int pos = 0;         // Variable to store the servo position. Allowed range is 0 to 180
int wait_time = 200; // Number of milliseconds to allow servo to reach position
int rotate = 45;

const int trigPin = 11;
const int echoPin = 12;
const int buzzerPin = 3; // Digital pin connected to the Passive Buzzer module

// defines variables
long duration;
int distance;

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *Motor1 = AFMS.getMotor(1);
Adafruit_DCMotor *Motor2 = AFMS.getMotor(2);
Adafruit_DCMotor *Motor3 = AFMS.getMotor(3);
Adafruit_DCMotor *Motor4 = AFMS.getMotor(4);

//SETUP
  void setup() {
  pinMode(trigPin, OUTPUT);// Sets the trigPin (11) as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin (12) as an Input
  Serial.begin(9600); // Starts the serial communication

  // initialize digital pin LED_BUILTIN as an output.
  pinMode(4, OUTPUT);

  if (!AFMS.begin()) {
  }

  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  pinMode (9, OUTPUT);
```

```
  pinMode(buzzerPin, OUTPUT); // Set the BuzzerPin (3) as OUTPUT
  pos=90;
}

//LOOP

void loop()
{
  checkDistance();

  // Call the function to play different sounds or melodies
  //LED Light
  digitalWrite(4, LOW);

  //forward
  Motor1->setSpeed(140);
  Motor2->setSpeed(140);
  Motor3->setSpeed(155);
  Motor4->setSpeed(155);

  Motor1->run(BACKWARD);
  Motor2->run(FORWARD);
  Motor3->run(BACKWARD);
  Motor4->run(FORWARD);

  myservo.write(45);    // tell servo to go to position in variable 'pos'
  delay(wait_time);     // Delay, with 'wait_time' in milliseconds

checkDistance ();

  myservo.write(90);    // tell servo to go to position in variable 'pos'
  delay(wait_time);     // Delay, with 'wait_time' in milliseconds

checkDistance ();

  myservo.write(180);    // tell servo to go to position in variable 'pos'
  delay(wait_time);     // Delay, with 'wait_time' in milliseconds

checkDistance ();

  myservo.write(90);    // tell servo to go to position in variable 'pos'
  delay(wait_time);     // Delay, with 'wait_time' in milliseconds

  if (distance <= 50)
```

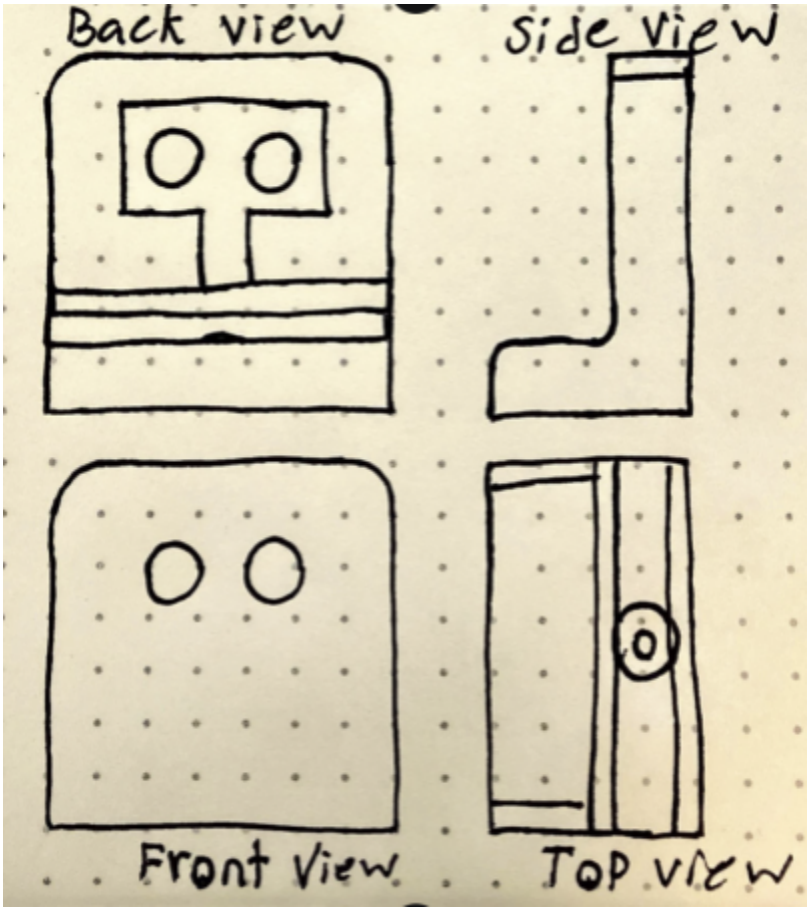
```
{
  //stop
  Motor1->setSpeed(0);
  Motor2->setSpeed(0);
  Motor3->setSpeed(0);
  Motor4->setSpeed(0);

  Motor1->run(BRAKE);
  Motor2->run(BRAKE);
  Motor3->run(BRAKE);
  Motor4->run(BRAKE);
  delay(500);

  // have the tank look around
  pos = 45;
  myservo.write(pos);    // tell servo to go to position in variable 'pos'
  delay(500);

   pos = 180;
  myservo.write(pos);
  delay(900);

  pos = 90;  //  have turret face forward
  myservo.write(pos);    // tell servo to go to position in variable 'pos'
  delay(500);      // Delay, with 'wait_time' in milliseconds

  digitalWrite(4, HIGH);
  delay(10);
  digitalWrite(4, LOW);
  delay(1);

  playSound();
  //delay(420); // Wait for 3 second before playing the sound again

  Motor1->setSpeed(250);
  Motor2->setSpeed(250);
  Motor3->setSpeed(250);
  Motor4->setSpeed(250);

  Motor1->run(FORWARD);
  Motor2->run(BACKWARD);
  Motor3->run(FORWARD);
  Motor4->run(BACKWARD);
  delay(50);
```

```
Motor1->setSpeed(250);
Motor2->setSpeed(250);
Motor3->setSpeed(250);
Motor4->setSpeed(250);

Motor1->run(BACKWARD);
Motor2->run(FORWARD);
Motor3->run(BACKWARD);
Motor4->run(FORWARD);
delay(50);

Motor1->setSpeed(0);
Motor2->setSpeed(0);
Motor3->setSpeed(0);
Motor4->setSpeed(0);

Motor1->run(BRAKE);
Motor2->run(BRAKE);
Motor3->run(BRAKE);
Motor4->run(BRAKE);
delay(1000);
// backs up
Motor1->setSpeed(140);
Motor2->setSpeed(140);
Motor3->setSpeed(155);
Motor4->setSpeed(155);

Motor1->run(FORWARD);
Motor2->run(BACKWARD);
Motor3->run(FORWARD);
Motor4->run(BACKWARD);
delay(600);

Motor1->setSpeed(140);
Motor2->setSpeed(140);
Motor3->setSpeed(140);
Motor4->setSpeed(140);
  //turning right 90deg
Motor1->run(FORWARD);
Motor2->run(BACKWARD);
Motor3->run(BACKWARD);
Motor4->run(FORWARD);
delay(1500);
```

```
  }
}

void playSound() {
  // Little gunshot sounds
  int melody[] = {  // Notes of the melody (pitches)
    500, 4500, 500, 500, 500, 50,
  };

  int noteDuration[] = { // Duration of each note (milliseconds)
    50, 70, 50, 50, 50, 150,
      };

  for (int i = 0; i < 6; i++) {
    if (melody[i] == 0) {
      noTone(buzzerPin); // Turn off the buzzer if there's no note
    } else {
      tone(buzzerPin, melody[i], noteDuration[i]); // Play the note
    }
    delay(noteDuration[i] * .50); // Pause between notes to avoid overlapping
  }
  noTone(buzzerPin); // Ensure the buzzer is off at the end
}

void checkDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  Serial.println(distance);
}
```

## **Appendix B:** Bracket for Ultrasonic Sensor

Orthographic drawings are provided below of the custom-designed bracket for the ultrasonic sensor.



# Endnotes


1. Flow Chart Ideas: Conceptdraw.com
2. DC Motor Specs: https://www.adafruit.com/product/3777
3. Arduino Tech Specs: https://docs.arduino.cc/hardware/uno-rev3/#tech-specs
4. Adafruit Motor Shield Specs: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield-v2-for-arduino.pdf
5. https://www.ledsupply.com/5mm-leds?srsltid=AfmBOorvYJ4NkvydtwsouDKG-AWpe9YrdVEmatnYBM42d0j-3SX0S-Do
6. Ultrasonic sensor spec sheet: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
7. Arduino board power consumption:https://docs.arduino.cc/learn/electronics/power-consumption/
8. Noise maker specs:

   https://www.amazon.com/QMseller-Terminals-Raspberry-Electronic-Continous/dp/B07VRK7ZPF?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&psc=1&smid=A10PG8DP5LW0Q1