 EENG Final Report Spring 2020

Simple Robot with Echolocation

A picture containing refrigerator, kitchen

Description automatically generated

**EENG 2301**

**COLLEGE OF MATHEMATICS**

**THE UNIVERSITY OF TEXAS AT TYLER**

REPORT SUBMITTED BY:  Sloke Shrestha

Timothy Bauer

Martin Morales Alvarez

Sameer Sapkota

We certify that the narrative, diagrams, figures, tables,

calculations and analysis in this report are our own work.

DATE REPORT DUE: April 30, 2020

1. **Introduction:**

Echolocation is a technique used to locate objects using infrared signals. It can be used to detect water levels or distance travelled by a tablet along a rail and many more. Echolocation Robot we designed and assembled is a generic robot which detects any object within a defined range using ultrasonic sound. We made a simple robot with the HC – SR04 sensor for echolocation capability. The robot can get the distance between the sensor and the nearest obstacle and try to maneuver out of the obstacle. A microcontroller was used to implement all the logistics of the robot. Although the robot is not robust to the uncertainty of the real world (ex: room full of people walking), it does a decent job in nice settings like room with block obstacles that doesn’t move and help us realize how to use a microcontroller.

Our project comprises of three phases. Designing is the first phase in which we decided with the potential shape, size and functionality of the robot based on requirements of the assignment, our expertise and cost. Coding and Hardware assembly is the second phase of the project, where we did Arduino programming and assembled our hardware parts. It is followed by testing phase, where we tested our prototype and we had to go back for a couple of change in our design and hardware assemble to make a working prototype of our project.

*Keywords: Echolocation, Ultrasonic Sound, Arduino Coding.*

1. **Methodology and Implementation:**
2. ***Materials Used:***

In the component selection process, the factors that had a more prominent influence were price and compatibility with the Arduino microcontroller. Such aspects led to the selection of the L298N H-Bridge motor driver. This IC is commonly used in hobby Arduino projects and simple robotics. The H-bridge allows the ability to spin the DC motors both clockwise and counterclockwise by manipulating the input voltage polarity. This ability was crucial as the robot had to be able to turn and reverse.

The selection of the HC-SR04 ultrasonic sensor was determined in a similar manner. This sensor works by sending out a burst of ultrasonic waves and measuring the time it takes for the echo to bounce back. This type of sensor is cheap and used in DIY projects as well. In respect to the microcontroller, the Arduino Uno is cheap and user-friendly. The operating voltage allows the safe usage of the motors as well. These components allowed the creation of a simple robot with ease to both construction and the wallet. A list of all materials can be seen in Table 1 below.

**Table 1:** Materials Required

|  |  |
| --- | --- |
| Materials | Quantity |
| HC- SR04 Ultrasonic Sound Sensor | 1 |
| Jumper Wires | 20 – 30 |
| Servo Motor | 1 |
| Arduino Uno | 1 |
| Swivel Caster Wheel | 1 |
| 3V-6V DC 1:120 Gear TT Motor and Wheels | 2 |
| L298N H Bridge Motor Driver | 1 |
| AAA Alkaline Batteries | 6 |
| Battery Case | 1 |
| Chassis for Robot | 1 |
| Flip Switch | 1 |
| Bread Board | 1 |
| Digital Multimeter | 1 |

Libraries Used:



**Fig 1:** Libraries used for Arduino Coding

**Table 2:** Pin allocation for the echolocator

|  |  |
| --- | --- |
|  | **Pins in Arduino** |
| Motor Driver digital input 1 | 2 |
| Motor Driver digital input 2 | 4 |
| Motor Driver digital input 3 | 7 |
| Motor Driver digital input 4 | 8 |
| Motor Driver digital Enable A | 3 |
| Motor Driver digital Enable B | 11 |
| SERVO PIN | 13 |
| Sensor Echo PIN | 5 |
| Sensor Trig PIN | 6 |

1. *Flowchart:*

In Fig. 2, the flowchart outlining the basic process the created echolocator took can be seen. To detail what is present in Fig. 2, steps will be explained sequentially. Initially, the echolocator will simply move forward before stopping and scanning its surroundings to its right and left. During this scanning, the echolocator calculates the distance between it and its surroundings based on calculations from code in the NewPing.h library involving the speed of sound. Then, the echolocator simply calculates the distance between it and anything in front of it using the same process detailed before.

Once its detections are complete, the echolocator will prompt itself with a condition on whether it was within less than 15 centimeters of an obstacle. If not, it will again move forward before repeating the processes explained prior. Otherwise, the echolocator will move backwards before again it surroundings to its left and turning left if there is an obstacle within 15 centimeters or turning right to perform the same respective process otherwise. If both left/right checks are false, the echolocator will then simply back up again before turning right and moving forward, then looping back to start the sequence of steps from the beginning. This can be seen in syntactic form in Fig. 3 under section *C. Coding* of this report.

A close up of a map

Description automatically generated**Fig 1:** Flowchart for the working mechanism of code

1. *Coding:*

#include<Servo.h>

#include<NewPing.h>

//motor driver constants

#define INPUT1\_PIN 2

#define INPUT2\_PIN 4

#define INPUT3\_PIN 7

#define INPUT4\_PIN 8

#define ENABLE\_A 3 //pwm (pulse width modulation pins): this can alter the speed of the motor

#define ENABLE\_B 11 //pwm

//servo motor constants

#define SRVO\_PIN 13

#define RIGHT\_ANGLE 60

#define LEFT\_ANGLE 150

//ultra-sonic sensor constants

#define ULTR\_ECHO\_PIN 5

#define ULR\_TRIG\_PIN 6

#define MAX\_DIST 400 //maximum distance measured by the ultrasonic sound sensor

//minimum safe distance in cm

#define TIGGER\_DIST 15

//

NewPing sonar(ULR\_TRIG\_PIN, ULTR\_ECHO\_PIN, MAX\_DIST);

Servo servoMotor;

int distance[] = {0,0,0}; //getting three distances

void setup(){

  setupMotorPins();

  servoMotor.attach(SRVO\_PIN);

  }

void loop(){

  moveForward(500); // move for 300 ms and stop

  //look left and get distance

  servoMotor.write(LEFT\_ANGLE);

  delay(300);

  distance[0] = getDistance();

 //look right and get distance

  servoMotor.write(RIGHT\_ANGLE);

  delay(300);

  distance[2] = getDistance();

  //look straight and get distance

  servoMotor.write(90);

  delay(300);

  distance[1] = getDistance();

  boolean obstacle = (distance[0] < TIGGER\_DIST) or (distance[1] < TIGGER\_DIST)

or (distance[2] < TIGGER\_DIST); // true iff any distance is less than minimum safe distance

  if (obstacle) {

    delay(90);

    moveBackward(400); //move back for 400 ms

    delay(750);

    //Look Left and get distance

    servoMotor.write(175);

    delay(300);

    int leftDistance = getDistance();

    if (leftDistance <= TIGGER\_DIST){

      //Look Right

      servoMotor.write(5);

      delay(300);

      int rightDistance = getDistance();

      if (rightDistance <= TIGGER\_DIST)moveBackward(400);

      turnRight(250);

      }

    else turnLeft(250);

  }

}//end loop

int getDistance(){

  /\*Returns the distance from the sensor to the nearest object in cm

   \* MAX DIST = 400 cm

   \* MIN DIST = 2 cm

   \*/

  int dist = 0;

  dist = sonar.ping\_cm();

  delay(275);

  if(dist == 0) return 20;

  return dist;

  }

//####################MOTOR MOVEMENT METHODS#######################

void setupMotorPins(){

  pinMode(INPUT1\_PIN, OUTPUT);

  pinMode(INPUT2\_PIN, OUTPUT);

  pinMode(INPUT3\_PIN, OUTPUT);

  pinMode(INPUT4\_PIN, OUTPUT);

  pinMode(ENABLE\_A, OUTPUT);

  pinMode(ENABLE\_B, OUTPUT);

  analogWrite(ENABLE\_A,115); //for some reason,right motor is a bit weak

  analogWrite(ENABLE\_B,100);

  }

void rightForward(){

  /\*Moves the Right Motor Fowrward

   \*/

  digitalWrite(INPUT1\_PIN, HIGH);

  digitalWrite(INPUT2\_PIN, LOW);

  }

void leftForward(){

   /\*Moves the Left Motor Fowrward

   \*/

  digitalWrite(INPUT3\_PIN, HIGH);

  digitalWrite(INPUT4\_PIN, LOW);

  }

void rightBackward(){

   /\*Moves the Right Motor Backward

   \*/

  digitalWrite(INPUT1\_PIN, LOW);

  digitalWrite(INPUT2\_PIN, HIGH);

  }

void leftBackward(){

   /\*Moves the Left Motor Backward

   \*/

  digitalWrite(INPUT3\_PIN, LOW);

  digitalWrite(INPUT4\_PIN, HIGH);

  }

void stopMovement(){

  digitalWrite(INPUT1\_PIN, LOW);

  digitalWrite(INPUT2\_PIN, LOW);

  digitalWrite(INPUT3\_PIN, LOW);

  digitalWrite(INPUT4\_PIN, LOW);

  }

void moveForward(int moveTime){

  /\*Moves forward

   \* @args moveTime: move for moveTime miliseconds

   \* @args moveTime: stop for stopTime milliseconds \*/

  leftForward();

  rightForward();

  delay(moveTime);

  stopMovement();

  }

void moveBackward(int moveTime){

    /\*Moves backward

   \* @args moveTime: move for moveTime miliseconds

   \* @args moveTime: stop for stopTime milliseconds \*/

  leftBackward();

  rightBackward();

  delay(moveTime);

  stopMovement();

  }

void turnRight(int moveTime){

    /\*Turn Right

   \* @args moveTime: move for moveTime miliseconds

   \* @args moveTime: stop for stopTime milliseconds \*/

  rightBackward();

  leftForward();

  delay(moveTime);

  stopMovement();

  }

void turnLeft(int moveTime){

    /\*Turn Left

   \* @args moveTime: move for moveTime miliseconds

   \* @args moveTime: stop for stopTime miliseconds

   \*/

  leftBackward();

  rightForward();

  delay(moveTime);

  stopMovement();

  }

**Fig 2:** Code used in the Arduino

1. **Conclusions and Future Prospects:**

The robot performs well in a controlled environment designed specifically. The performance of the robot is constricted, however, by quality of sensors used and restraints of the programming algorithms. Though it does not run into obstacles with big, blocky obstacles it is highly probable that it will fail given, the environment is not nice and predictable. For example, the robot would not be able to echolocate the objects such as chairs with thin legs, or a ball comes rolling toward the robot.

The capacity and scope of robot could be improved by increasing the number and quality of the sensors used and improving the algorithms on which it is based. Introducing algorithms and models like probabilistic models and/or computer vision coupling with better hardware would exponentially increase the scope and prospects of the robot. Sonar devices used by military is one of the many applications of echolocation devices. The ability of tracking objects without codependence on contrast is the most important feature of echolocation devices and if added with additional accessories our robot is applicable for real life usage.