Theory

Movement and Rotation

The tank was programmed to move autonomously with the use of C programming language via the Arduino Software and then the code was uploaded to the Arduino Mega 2560 microcontroller board. The tank works by autonomously moving forward, backward, right and left using the specific functions written in the Arduino program. The program for the movement and rotation of the tank has two main functions. One function controls the forward and backward movement and another function controls the right and left turn of the tank. The user has to only specify and input the direction, duration and the speed in which the tanks needs to make within the logic function of the program and the tank will make the move accordingly. The Arduino is connected to two DC motors, which are responsible for the rotation of the tank. However, the Arduino doesn't supply enough current to activate the DC motors. Therefore the L298N, which is a H-bridge driver motor, is used to control the polarity of the tank and to boost the current from the Arduino. This boost of current by the H-bridge will help to activate the two DC motors. The two output ports from the left motor are connected to IN1 and IN2 and the two output ports from the right motor are connected to the IN3 and IN4 of the H-bridge driver. These four pins are connected from H-bridge to 4 digital pins of the Arduino and work according to their high or low state. Different states that control the motion of the motors are shown in *Table 1.1*. The two ENA and ENB pins of the H-bridge that are connected to two PWM digital pins of the Arduino are used to control the speed of the motors. Enable A controls speed of the right motor and Enable B controls speed of the left one.

	HIGH/LOW State			
Pins	**IN1**	**IN2**	**IN3**	**IN4**
Forward	HIGH	LOW	HIGH	LOW
Backward	LOW	HIGH	LOW	HIGH
Left	HIGH	LOW	LOW	HIGH
Right	LOW	HIGH	HIGH	LOW

Table 1.1

Sensors and Obstacle Avoiding

In order to make the tank move autonomously, a servo motor and an ultrasonic sensor were installed in the front section of the chassis. An ultrasonic sensor is a type of sensor that uses sound wave to measure a distance to an object. The sensor transmits a sound wave signal at a certain frequency via the Trigger pin to an object and waits for the signal to bounce back. The Echo pin of the sensor receives the signal that is bounced back. Therefore the distance to the object is calculated using the elapsed time between the trigger and the echo signals. The ultrasonic sensors can only detect objects within a specific range. They cannot detect objects that are too close to them because the time needed to travel the distance is too short for the sensor to detect the signal. The trigger and the echo pins of the sensor are connected to two digital pins of the Arduino Mega board. The Arduino program is used to control the transmission and the receiver signals of the ultrasonic sensor. The code is written in a way in which the sensor sends a sound

wave signal for 10 Microseconds and waits for the pulse to echo back. The waveform for the signal is shown in *figure 1.1*. Then the code uses the duration needed for the pulse to be bounced back and calculates the distance from the sensor to the object. The ultrasonic sensor is installed on a mini servomotor that rotates from 0 degree to 180 degree accordingly. The servo has a short delay at 0 degree angle to check for obstacles to its right, 90 degree angle to check for obstacles ahead and 180 degree angle for obstacles to its left. In this way, the tank knows if there are any obstacles on the way and makes the correct decision to avoid them. With the use of appropriate functions and logic within the program, the tank is able to move autonomously throughout the maze and avoid any obstacles on its way.

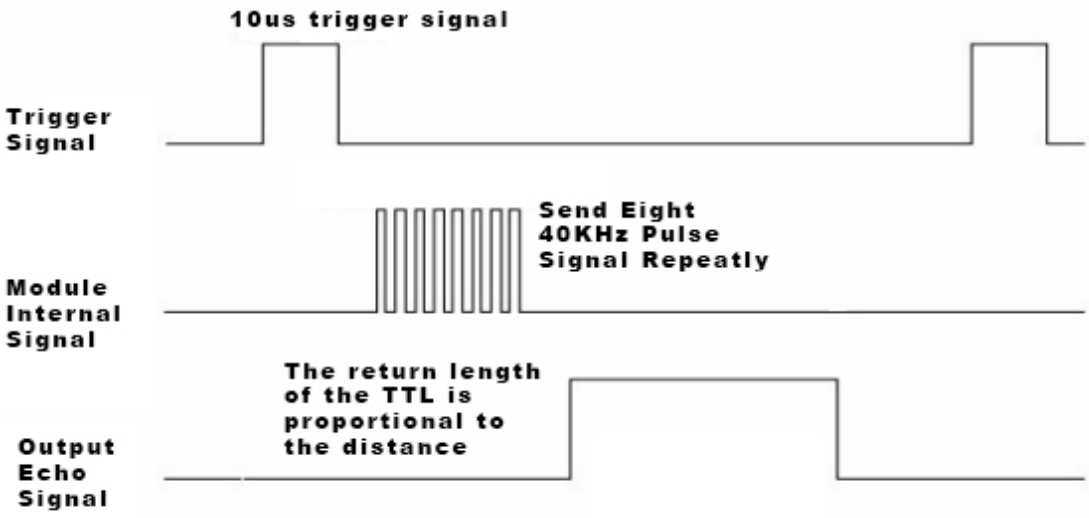


Figure 1.1

Scope

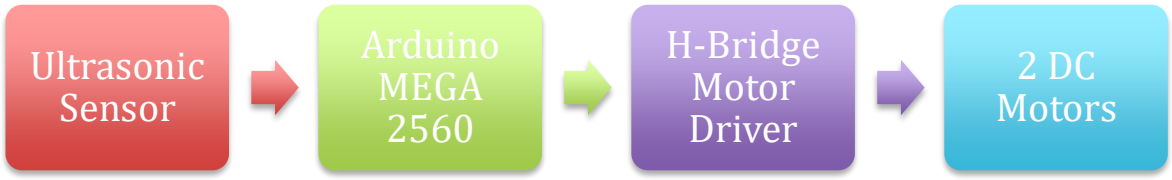
1. As it is shown in the program below, the duration needed for the tank to make a precise 90-degree right or left turn is measured to be 1325 milliseconds.
2. The speed in which the motors move was set to be 150 PWM.
3. 10 PWM was added to Enable-A, which controls the right motor because it had a slight lagging from the left motor that caused some errors in the movement of the tank.
4. The Servomotor is set to begin at 90-degree angle facing the forward direction.
5. The Servo makes 0-degree angle to face the right direction and makes 180-degree angle to face the left direction.
6. The ultrasonic sensor sends the trigger pulse for 10 microseconds and counts the duration of the pulse that was echoed back.
7. The echo pulse duration can range between 250 microseconds to 25 milliseconds. If the sensor detects a duration greater than 25 milliseconds then it assumes that there is no objects ahead and continues to move forward.
8. The given Equation:

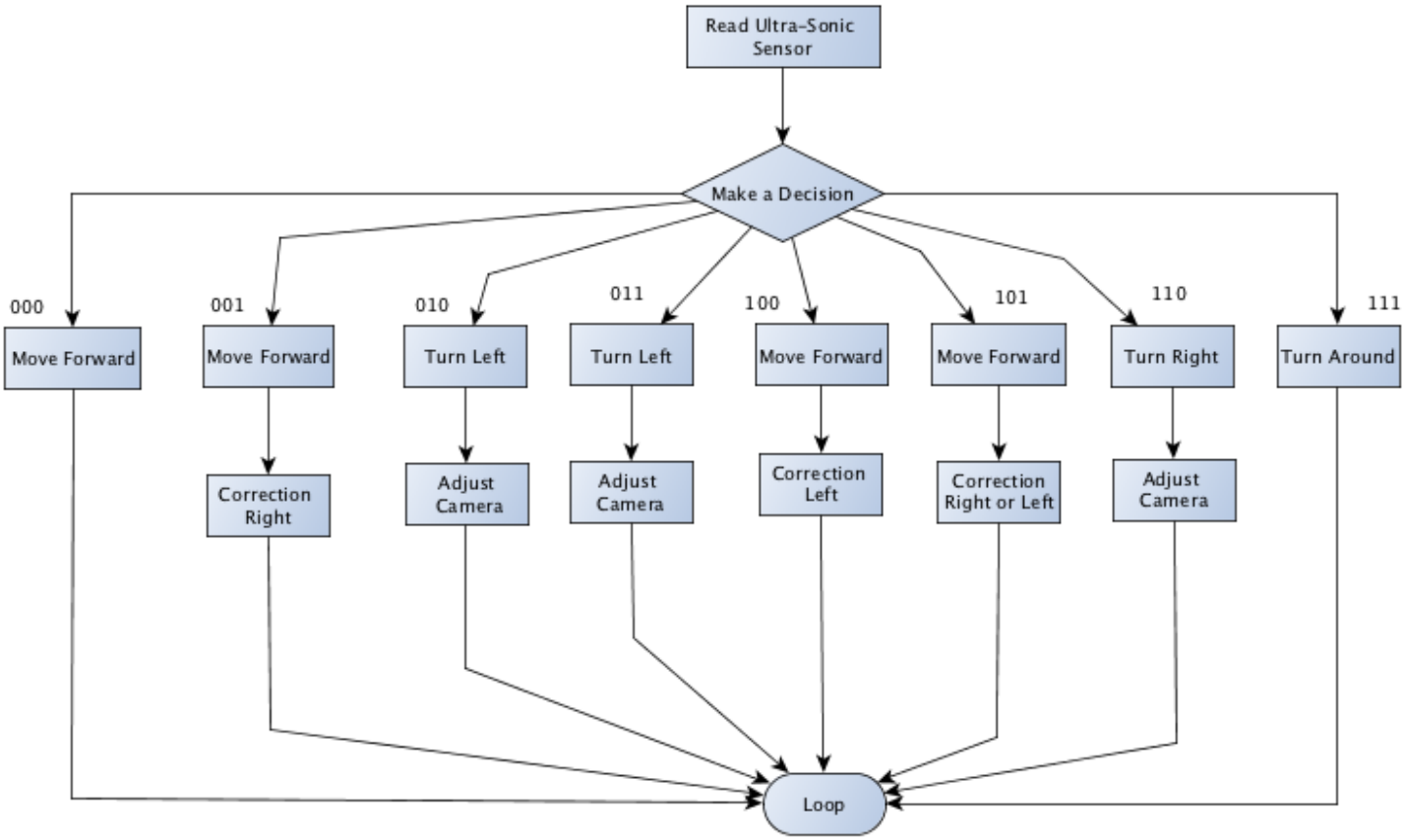
$$Distance\ (cm) = \frac{Echo\ Pulse\ Duration}{58}$$

   Is used to calculate the distance in centimeters from the ultrasonic sensors to the object at 10 microseconds triggering signal.
9. 10.6 centimeters, which is the distance from the sensor to the outer edge of the right track, was measured and subtracted from the distance between the object and the sensor at 0-degree angle.

10. 11.6 centimeters, which is the distance from the sensor to the outer edge of the left track, was measured and subtracted from the distance between the object and the sensor at 180-degree angle.
11. The tank is programmed to avoid obstacles within the range of 30 centimeters to its right and left.
12. The tank is programmed to avoid obstacles within the range of 15 centimeters ahead of it.
13. Generally, ultrasonic sensors can detect any object within the range of 2 cm to 500 cm.
14. Ultrasonic sensors detect objects within 30-degree angle (15-degree to right, 15-degree to left) ahead of them.

Breakdown and Flowchart of Tank Autonomous movement





Correction

In order to make the movement of the tank more precise, two extra ultrasonic sensors were installed at the back of the tank chassis. One of the sensors was connect to the right back corner and the other was connected to the left back corner of the tank. Now the sensors would detect the distance from the tank to any obstacle to the right or left of the tank and use that distance to help the tank to move in a straight line. This method helped to avoid the tank to keep hitting the walls of the maze. The program would consider reading only left and back left sensors for every right turn and consider reading only right and right back sensors for every left turn. Next, depending on the last turn that was made by the tank and the wall that was being detected, the distance error between the distance from the front sensor to the wall and the distance from the back sensor to the wall was calculated. In this way, the program would know by how much the tank is slanted to the sidewalls. Then depending on the size of the error, the program would decide whether to make a slightly big turn, a slightly small turn or ignore to make any slight turns at all. The tank would make slight big turn at 250 milliseconds if the absolute value of the error were bigger tan 25 cm. It would make slight small turn at 125 milliseconds if the absolute value of the error were between 15 cm to 25 cm; and it would ignore any error less than 15 cm and would continue moving forward. The program would also take into account whether the tank is slanted toward or away from the wall by checking if the error calculated is negative or positive. This would determine in what direction the tank has to make the correction. The correction function is only used for when the tank is moving in the forward direction and needs to avoid hitting the sidewalls.

Code

Testing_____3

```
int In1 = 22;
int In2 = 23;
int In3 = 24;
int In4 = 25;
int ena = 4;
int enb = 5;

int trig = 26;
int echo = 27;
int trig1 = 30;
int echo1 = 31;
int trig2 = 28;
int echo2 = 29;


double distSensorRight = 0;
double distSensorLeft = 0;
double distSensorForward = 0;
double distSensorRightBack = 0;
double distSensorLeftBack = 0;

double errorRight=0;
double errorLeft=0;

const int left=1;
const int right =2;
const int forward=3;
const int backward =4;

boolean LT=0;
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
boolean LT=0;


#include <Servo.h>
Servo myservo;


void setup() {
  pinMode(In3, OUTPUT);
  pinMode(In1, OUTPUT);
  pinMode(In2, OUTPUT);
  pinMode(In4, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);

  Serial.begin (9600);
  myservo.attach(38);
  myservo.write(90);

}

void loop()
{

SolveMaze();
}
 void mov(int dir, int duration, int pwm){
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
void mov(int dir, int duration, int pwm){


   if(dir==forward){

   digitalWrite(In1, HIGH);
    digitalWrite(In2, LOW);
   digitalWrite(In3, HIGH);
   digitalWrite(In4, LOW);

   analogWrite(ena, pwm+10);
   analogWrite(enb, pwm);

   delay(duration);

   analogWrite(ena, 0);
   analogWrite(enb, 0);
 }

   if(dir==backward){



   digitalWrite(In1, LOW);
   digitalWrite(In2, HIGH);
   digitalWrite(In3, LOW);
   digitalWrite(In4, HIGH);
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
    analogWrite(ena, pwm+10);
    analogWrite(enb, pwm);

    delay(duration);

    analogWrite(ena, 0);
    analogWrite(enb, 0);

}
 }


 void turn(int dir, int duration, int pwm){

  if(dir==right){

  LT=0;
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);



  analogWrite(ena, pwm+10);
  analogWrite(enb, pwm);

  delay(duration);
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
  analogWrite(ena, 0);
  analogWrite(enb, 0);

 }


 if(dir==left){


  LT=1;
  digitalWrite(In1, LOW);
  digitalWrite(In2, HIGH);
  digitalWrite(In3, HIGH);
  digitalWrite(In4, LOW);


  analogWrite(ena, pwm);
  analogWrite(enb, pwm);

  delay(duration);

  analogWrite(ena, 0);
  analogWrite(enb, 0);
 }

 }
  void sensorRight(){
  myservo.write(0);
  delay(1000);
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
  delay(1000);
  double pulseDur = 0;

  digitalWrite(trig,LOW);

  delayMicroseconds(5);
  digitalWrite(trig, HIGH);

  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  pulseDur = pulseIn(echo, HIGH);
  distSensorRight= ((pulseDur/58)-10.16); //cm


 }

  void sensorLeft(){
  myservo.write(180);
  delay(1000);
  double pulseDur = 0;

  digitalWrite(trig,LOW);
  delayMicroseconds(5);

  digitalWrite(trig, HIGH);
  delayMicroseconds(10);

  digitalWrite(trig, LOW);
  pulseDur = pulseIn(echo, HIGH);
  distSensorLeft= ((pulseDur/58)-11.16); //cm
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
  distSensorLeft= ((pulseDur/58)-11.16); //cm


 }
 void sensorForward(){
  myservo.write(90);
  delay(1000);
  unsigned long pulseDur = 0;

  digitalWrite(trig,LOW);
  delayMicroseconds(5);

  digitalWrite(trig, HIGH);
  delayMicroseconds(10);

  digitalWrite(trig, LOW);
  pulseDur = pulseIn(echo, HIGH);
  distSensorForward= long(pulseDur/58); //cm


 }
//*****************************************************
void readSensors(){

sensorForward();
sensorLeft();
sensorRight();

}
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
void SolveMaze()
{
  sensorForward();

if(distSensorForward>15)

{

     mov(forward,1000,150);
    Correction(LT);
}


else if(distSensorForward<15) {

  delay(500);
  readSensors();

 if (distSensorLeft>30 && distSensorForward <=15 && distSensorRight>30)       // 0 1 0 turn left
  {
    turn(left,1325,150);

  }

 else if (distSensorLeft>30 && distSensorForward<=15 &&  distSensorRight<30)// 0 1 1 turn left
  {
    turn(left,1325,150);
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
}


  else if (distSensorLeft<30 && distSensorForward<=15 && distSensorRight>30)// 1 1 0 turn right
  {
    turn(right,1325,150);

  }
}
}

void sensorRightBack(){
  delay(1000);
  double pulseDur1 = 0;

  digitalWrite(trig1,LOW);
  delayMicroseconds(5);

  digitalWrite(trig1, HIGH);
  delayMicroseconds(10);

  digitalWrite(trig1, LOW);
  pulseDur1 = pulseIn(echo1, HIGH);
  distSensorRightBack= ((pulseDur1/58)-4); //cm


 }

 void sensorLeftBack(){
  delay(1000);
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
  delay(1000);
  double pulseDur2 = 0;

  digitalWrite(trig2,LOW);
  delayMicroseconds(5);

  digitalWrite(trig2, HIGH);
  delayMicroseconds(10);

  digitalWrite(trig2, LOW);
  pulseDur2 = pulseIn(echo2, HIGH);
  distSensorLeftBack= ((pulseDur2/58)-3.24); //cm


 }
void readCorrectionSensorRight(){
  sensorForward();
  sensorRight();
  sensorRightBack();
 }

 void readCorrectionSensorLeft(){
  sensorForward();
  sensorLeft();
  sensorLeftBack();
 }

void correctionRight(){
   readCorrectionSensorRight();
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
void correctionRight(){
   readCorrectionSensorRight();

   errorRight =  (distSensorRight-distSensorRightBack);

  if(abs(errorRight)>25)
  {
    if(errorRight<0){
    turn(left, 250 ,150);

    }
     else if(errorRight>0){
      turn(right, 250,150);
     }
    }
  else if(15<abs(errorRight)<=25)
  {

       if(errorRight<0){
    turn(left, 125 ,150);

    }
     else if(errorRight>0){
      turn(right, 125,150);
     }
    }
     else if(abs(errorRight)<=15)
  {

  }
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```

Testing_____3

```
void correctionLeft(){
   readCorrectionSensorLeft();

   errorLeft =  (distSensorLeft-distSensorLeftBack);

  if(abs(errorLeft)>25)
  {
    if(errorLeft<0){
    turn(right, 250 ,150);

    }
     else if(errorLeft>0){
      turn(left, 250,150);
     }
    }
  else if(15<abs(errorLeft)<=25)
  {

       if(errorLeft<0){
    turn(right, 125 ,150);

    }
     else if(errorLeft>0){
      turn(left, 125,150);
     }
    }
     else if(abs(errorLeft)<=15)
  {

  }
```

Done compiling.

```
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.
```


Testing_____3
{
if(errorLeft<0){
turn(right, 125 ,150);
}
else if(errorLeft>0){
turn(left, 125,150);
}
}
else if(abs(errorLeft)<=15)
{
}
}
void Correction(boolean LastTurn){
if (LastTurn=1) {
correctionRight();
}
else{
correctionLeft();
}
}
Done compiling.
Sketch uses 6774 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 355 bytes (4%) of dynamic memory, leaving 7837 bytes for local variables. Maximum is 8192 bytes.