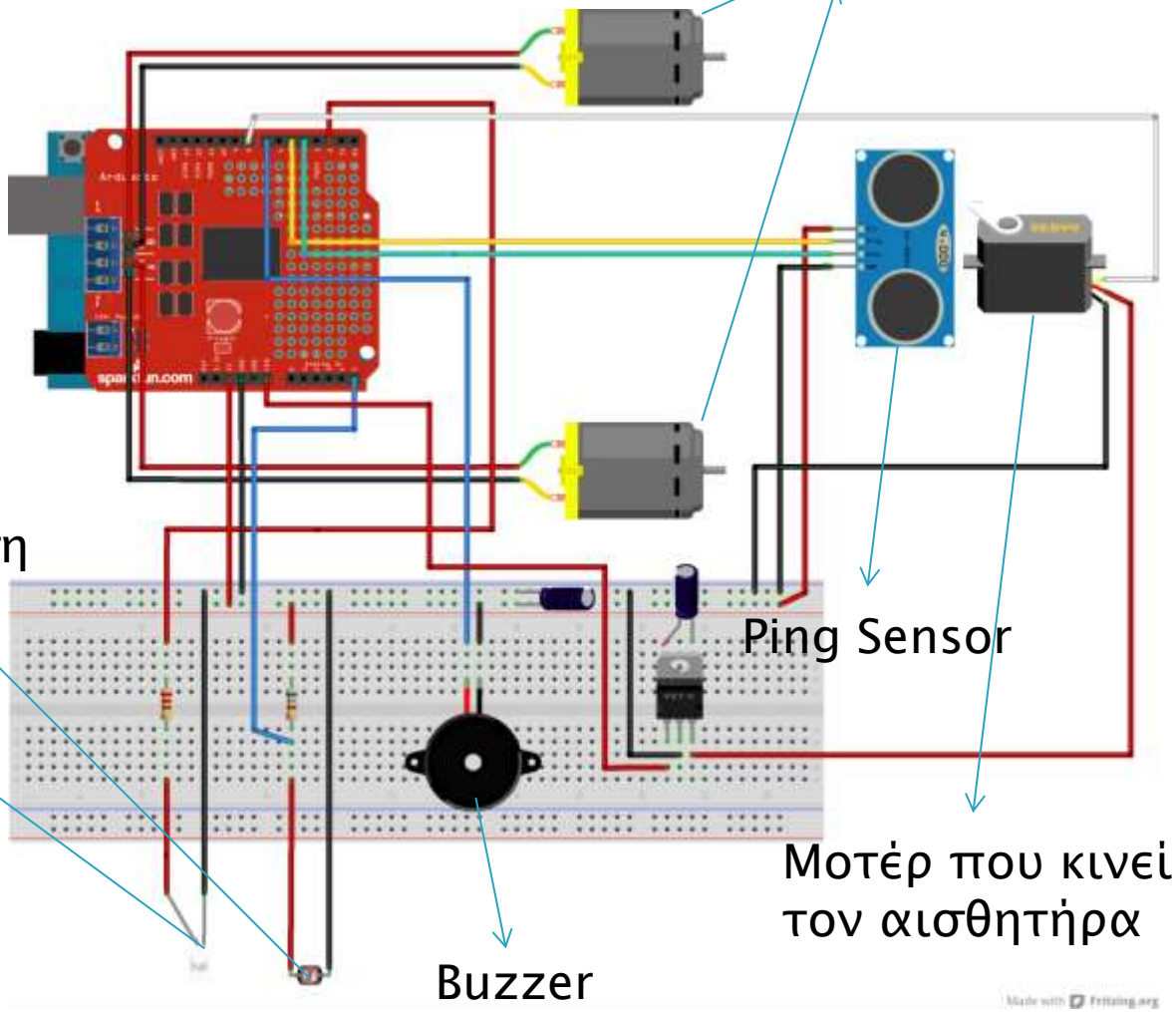
# Μελέτη & Σχεδίαση Αυτόνομου Ρομποτικού Οχήματος


Κωστής Τζεβελεκάκης
Μιλτιάδης Στούρας
Α4 – Β' Λύκειο

# Διάταξη Ηλ. Κυκλώματος



# Ο κώδικάς μας

```
#include <NewPing.h>
#include <Servo.h>

#define DIRECTION_A_PIN  12
#define DIRECTION_B_PIN  13
#define BRAKE_A_PIN  9
#define BRAKE_B_PIN  8
#define SPEED_A_PIN  3
#define SPEED_B_PIN  11

#define BUZZER_PIN   7

#define TRIGGER_PIN  5
#define ECHO_PIN     4
#define MAX_DISTANCE 120

#define SERVO_PIN    6
#define SERVO_START  20
#define SERVO_STEP   10
#define SERVO_END    160

#define TURN_SPEED 255
#define TURN_TIME 3

#define IDLE_SPEED 200
#define IDLE_TIME 200



unsigned long Time0;
unsigned long Time1;

float Distance,Max_Distance,Degrees,Sw,S,Servo_Target;
float Turn,Speed_A,Speed_B;

int Measurments0, times;

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
Servo SensorServo;




void setup()
{
  pinMode(DIRECTION_A_PIN, OUTPUT);
  pinMode(BRAKE_A_PIN, OUTPUT);


  pinMode(DIRECTION_B_PIN, OUTPUT);
  pinMode(BRAKE_B_PIN, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);

  digitalWrite(DIRECTION_A_PIN, HIGH);
  digitalWrite(BRAKE_A_PIN, LOW);
  digitalWrite(DIRECTION_B_PIN, HIGH);
  digitalWrite(BRAKE_B_PIN, LOW);

  SensorServo.attach(SERVO_PIN);

  ThermalBalance();

}



void loop()
{
  Sw = 0;
  S = 0;
  Max_Distance = 0;
  times = 0;
  Measurments0 = 0;

  label:
  for(Degrees = SERVO_START; Degrees <= SERVO_END; Degrees = Degrees +
SERVO_STEP)
      {
        SensorServo.write(Degrees);
        delay(30);
        Distance = sonar.ping();

            if (Distance>Max_Distance && Distance != 0)
                Max_Distance = Distance;


        Sw = Sw + Degrees * Distance;
        S = S + Distance;


            if( Distance == 0 )
               Measurments0 ++;

            if( Distance < 900 && Distance != 0 )
               times ++;

            if( times >= 3)
               {
                  TargetFound();
               }

      }

  if(Measurments0 == 15)
      {
            analogWrite(SPEED_A_PIN, 0);

            analogWrite(SPEED_B_PIN, 0);

            Time1 = millis();

            while ( (millis()-Time1) < TURN_TIME * 140 )
                  {

                    Speed_A = 0;
                    Speed_B = TURN_SPEED;
                    analogWrite(SPEED_A_PIN, Speed_A);
                    analogWrite(SPEED_B_PIN, Speed_B);
                  }

            analogWrite(SPEED_A_PIN, 0);
            analogWrite(SPEED_B_PIN, 0);

            Measurments0 = 0;
            goto label;
  }

  Servo_Target = getServo_Target(  Sw ,  S );

  SensorServo.write(Servo_Target);

  Turn = getTurn( Servo_Target );

  if (Turn>0)
   {
        analogWrite(SPEED_A_PIN, 0);
        analogWrite(SPEED_B_PIN, 0);

        Time0 = millis();

        while ( (millis()-Time0) < TURN_TIME * Turn)
            {
                Speed_A = 0;
                Speed_B = TURN_SPEED;
                analogWrite(SPEED_A_PIN, Speed_A);
                analogWrite(SPEED_B_PIN, Speed_B);
            }

   }

  else
   {
        analogWrite(SPEED_A_PIN, 0);
        analogWrite(SPEED_B_PIN, 0);

        Time0 = millis();

        while ( (millis()-Time0) < TURN_TIME * (-1) * Turn)
           {
              Speed_A = TURN_SPEED;
              Speed_B = 0;
              analogWrite(SPEED_A_PIN, Speed_A);
              analogWrite(SPEED_B_PIN, Speed_B);
           }

   }

  analogWrite(SPEED_A_PIN, IDLE_SPEED);
  analogWrite(SPEED_B_PIN, IDLE_SPEED);

}


void ThermalBalance()
{
        for( int x = 1; x <= 5; x++)
          {
            Distance = sonar.ping();
            delay(40);
          }


   }

   void TargetFound()
   {
     analogWrite(SPEED_A_PIN, 0);
     analogWrite(SPEED_B_PIN, 0);

     digitalWrite(BUZZER_PIN, HIGH);
     delay(1000);
     digitalWrite(BUZZER_PIN, LOW);
     delay(1000000);

   }


   float getTurn (float Servo_Target)
   {
       Turn = Servo_Target - (SERVO_START + SERVO_END)/2;
       return Turn;
   }

   float getServo_Target( float Sw , float S )
   {
       Servo_Target = Sw/S;
       return Servo_Target;
   }
```

```
#include <NewPing.h>
#include <Servo.h>

#define DIRECTION_A_PIN  12
#define DIRECTION_B_PIN  13
#define BRAKE_A_PIN  9
#define BRAKE_B_PIN  8
#define SPEED_A_PIN  3
#define SPEED_B_PIN  11

#define BUZZER_PIN   7

#define TRIGGER_PIN  5
#define ECHO_PIN     4
#define MAX_DISTANCE 120

#define SERVO_PIN    6
#define SERVO_START  20
#define SERVO_STEP   10
#define SERVO_END    160

#define TURN_SPEED 255
#define TURN_TIME 3

#define IDLE_SPEED 200
#define IDLE_TIME 200
```

```
unsigned long Time0;
unsigned long Time1;

float
Distance,Max_Distance,Degrees,Sw,S,Servo_Target;
float Turn,Speed_A,Speed_B;

int Measurments0, times;

NewPing sonar(TRIGGER_PIN, ECHO_PIN,
MAX_DISTANCE);
Servo SensorServo;
```

```
void setup()
{

  pinMode(DIRECTION_A_PIN, OUTPUT);
  pinMode(BRAKE_A_PIN, OUTPUT);


  pinMode(DIRECTION_B_PIN, OUTPUT);
  pinMode(BRAKE_B_PIN, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);

  digitalWrite(DIRECTION_A_PIN, HIGH);
  digitalWrite(BRAKE_A_PIN, LOW);
  digitalWrite(DIRECTION_B_PIN, HIGH);
  digitalWrite(BRAKE_B_PIN, LOW);

  SensorServo.attach(SERVO_PIN);

  ThermalBalance();

}
```

```
void ThermalBalance()
{
        for( int x = 1; x <= 5; x++)
          {
            Distance = sonar.ping();
            delay(40);
          }

}
```

```
void loop()
{

  Sw = 0;
  S = 0;
  Max_Distance = 0;
  times = 0;
  Measurments0 = 0;

  label:
  for(Degrees = SERVO_START; Degrees <= SERVO_END; Degrees = Degrees + SERVO_STEP)
      {
        SensorServo.write(Degrees);
        delay(30);
        Distance = sonar.ping();

            if (Distance>Max_Distance && Distance != 0)
                  Max_Distance = Distance;


        Sw = Sw + Degrees * Distance;
        S = S + Distance;
```

```
if( Distance == 0 )
               Measurments0 ++;

            if( Distance < 900 && Distance != 0 )
               times ++;

            if( times >= 3)
               {
                   TargetFound();
               }

    }
```

```
void TargetFound()
{
  analogWrite(SPEED_A_PIN, 0);
  analogWrite(SPEED_B_PIN, 0);

  digitalWrite(BUZZER_PIN, HIGH);
  delay(1000);
  digitalWrite(BUZZER_PIN, LOW);
  delay(1000000);

}
```

```
if(Measurments0 == 15)
      {
            analogWrite(SPEED_A_PIN, 0);
            analogWrite(SPEED_B_PIN, 0);

            Time1 = millis();

            while ( (millis()-Time1) < TURN_TIME * 140 )
                  {

                    Speed_A = 0;
                    Speed_B = TURN_SPEED;
                    analogWrite(SPEED_A_PIN, Speed_A);
                    analogWrite(SPEED_B_PIN, Speed_B);
                  }

            analogWrite(SPEED_A_PIN, 0);
            analogWrite(SPEED_B_PIN, 0);

            Measurments0 = 0;
            goto label;

  }
```

```
Servo_Target = getServo_Target(  Sw ,  S );

 SensorServo.write(Servo_Target);

 Turn = getTurn( Servo_Target );
```

```
float getServo_Target( float Sw , float S )
{
    Servo_Target = Sw/S;
    return Servo_Target;
}
```

```
float getTurn (float Servo_Target)
{
    Turn = Servo_Target - (SERVO_START + SERVO_END)/2;
    return Turn;
}
```

```
if (Turn>0)
   {
        analogWrite(SPEED_A_PIN, 0);
        analogWrite(SPEED_B_PIN, 0);


         Time0 = millis();

         while ( (millis()-Time0) < TURN_TIME * Turn)
            {

                  Speed_A = 0;
                  Speed_B = TURN_SPEED;
                  analogWrite(SPEED_A_PIN, Speed_A);
                  analogWrite(SPEED_B_PIN, Speed_B);
            }

   }
```

```
else
   {
        analogWrite(SPEED_A_PIN, 0);
        analogWrite(SPEED_B_PIN, 0);

        Time0 = millis();

        while ( (millis()-Time0) < TURN_TIME * (-1) * Turn)
           {
              Speed_A = TURN_SPEED;
              Speed_B = 0;
              analogWrite(SPEED_A_PIN, Speed_A);
              analogWrite(SPEED_B_PIN, Speed_B);
           }

  }

  analogWrite(SPEED_A_PIN, IDLE_SPEED);
  analogWrite(SPEED_B_PIN, IDLE_SPEED);

}
```

# Ευχαριστούμε για την προσοχή σας