This is ULTRA-ROBO!

ULTRA-ROBO is an Arduino-based robot which can act like a remote control car with 2 enhanced features: following targets and the ability to avoid obstacles. The robot is built on an Arduino UNO (R3) development board and uses a mobile phone app for the movement commands. The communication between the smartphone and the robot is done via bluetooth technology.

Our idea was to build more than a remote controlled bluetooth car, a robot that, with the additional capabilities added through the ultrasonic sensor can scan the environment that surrounds it and take decisions by itself.

The project can evolve into a prototype which can move without external help/on its own with the help of its environment scanning sensors (in our case an ultrasonic sensor), having the potential of performing several tasks for saving human lives without endangering other human lives.

In spite of its simplistic appearance, it is a fairly complex robot because of the functionalities of all the components that are part of the circuit (which is an ample one).

When building such a project, it is very important to combine skills from different areas of science and engineering. In our case, the 3 main fields are computer science, physics and mathematics. The computer science part consists of embedded programming and robotics, physics knowledge is required for building electrical circuits, and the calculations of response and movement time is a collection of mathematical calculations.

Now that we covered the basics of the project, it is time to dive a little deeper into the actual components and technologies used. Initially, ULTRA-ROBO was designed in a very inefficient way, with a wooden chassis and with all the components being on the superior part of the chassis, which resulted in a quick deterioration of the board because of the applied forces when any change of cables between components was needed. In addition, the wheels, being placed on the upper part, were blocking the ultrasonic sensor range of view. However, the current version of the robot is built on an aluminum board wrapped in an insulating material which offers us more flexibility because of the exact cuts that can be made due to it being a rougher material that came in a thinner shape. (For example, in the building process we changed analog to digital pins ad vice-versa to test in which way the ultra-sonic sensor works better. We also need to unplug the cables connected to the RX and TX pins on the arduino board when we want to upload a new program because they also play a role in data-transmitting).

ULTRA-ROBO represents, in fact, the base concept behind many robots developed in the military and not only, which make use of various environment analysis sensors and embedded systems built especially for them. These robots are used for accomplishing different tasks which could otherwise be dangerous for humans. For instance, these kind of robots are made to be sent on a battlefield in order to retrieve wounded soldiers or operated by humans in order to enter a buildings that are on fire and do similar tasks.

La baza robotului ULTRA-ROBO se află, după cum am mai precizat anterior, placa de dezvoltare [Arduino UNO (R3)](https://cleste.ro/arduino-uno-r3-atmega328p.html), ce are ca „nucleu” microcontroller-ul (circuitul integrat) [Atmega328P](https://ro.wikipedia.org/wiki/Atmega328). Tehnologia Arduino ne-a permis să controlăm robotul, placa de dezvoltare [Arduino UNO (R3)](https://cleste.ro/arduino-uno-r3-atmega328p.html) reprezentând locul de unde pleacă comenzile către componente și unde are loc procesarea tuturor informațiilor de la și pentru componente. Programul pe care placa de dezvoltare [Arduino UNO (R3)](https://cleste.ro/arduino-uno-r3-atmega328p.html) îl rulează pentru a realiza aceste procese a fost scris cu ajutorul softului [Arduino IDE](https://www.arduino.cc/en/software) de către noi.

Am utilizat acest senzorul ultrasonic HC-SR04 pentru a calcula distanța dintre robot și obiectele din jur. În urma timpului dintre transmiterea unor ultrasunete la o frecvență de 40 kHz, reflectării acestora pe suprafața unui obiect și receptării acestora de către senzor, putem calcula distanța cu următoarea formulă : . Datele transmise de către senzor fiind analog, în urma convertirii semnalului primit de către placa Arduino de la senzor, vom lucra cu microsecunde (ținând cont că viteza sunetului în aer o putem considera 340 m/s, adică 0.034 cm/µs), în acest mod, distanța va fi măsurată în centimetri, timpul în microsecunde, iar viteza în centimetri/microsecunde, adică, în cele din urmă formula folosită pentru a calcula distanța în program este .

Modulul L298H ne permite să controlăm viteza și sensul motoarelor. Astfel, acest modul este conectat la placa Arduino prin intermediul a 6 pini. În funcție de voltajul transmis fiecărui pin (comanda dată de către placa Arduino), modulul L298H setează voltajul (controlând viteza de rotație) și polaritatea (controlând sensul de rotație) curentului electric transmis către motorașe.

#include <Servo.h>

#define TrigPIN A1   // Assign PIN A1 as TrigPIN (Connect Arduino UNO "A1" PIN with Ultrasonic   Sonar Sensor "Trig" PIN).

#define EchoPIN A2   // Assign PIN A2 as EchoPIN (Connect   Arduino UNO "A2" PIN with Ultrasonic Sonar Sensor "Trig" PIN).

#define SERVO\_PIN 7  //Assign digital PIN 7 as Servo\_Pin

Servo myservo;

int pos = 0;

long time;

int i;

int RightDistance, LeftDistance;  // Distances on either side.

float Distance = 0.00;            // Float type variable declaration.

unsigned long startMilis;

unsigned long currentMillis;

const unsigned long period = 100;

const unsigned long period1 = 300;

char t;

int enA = 8;

int enB = 13;

void setup() {

  startMilis = millis();

  pinMode(enA, OUTPUT);

  pinMode(enB, OUTPUT);

  pinMode(9, OUTPUT);   //left motors  forward

  pinMode(10, OUTPUT);  //left motors reverse

  pinMode(11, OUTPUT);  //right  motors forward

  pinMode(12, OUTPUT);  //right motors reverse

  Serial.begin(9600);

  myservo.attach(SERVO\_PIN);

  // Initialize servo and ultrasonic sensor

  myservo.write(90);

  pinMode(TrigPIN, OUTPUT);

  pinMode(EchoPIN, INPUT);

  pinMode(SERVO\_PIN, OUTPUT);

}

int Search(void)  //   Integer type variable declaration.

{

  float Duration = 0.0;               // Float   type variable declaration.

  float CM = 0.0;                     // Float type   variable declaration.

  digitalWrite(TrigPIN, LOW);         // TrigPIN output   as 0V (Logic low level).

  delayMicroseconds(2);               // Delay for 2us,   Send 10 us high pulse to Ultrasonic Sonar Sensor "TrigPIN".

  digitalWrite(TrigPIN, HIGH);        // TrigPIN output as 5V (Logic high level).

  delayMicroseconds(10);              // Delay for 10us.

  digitalWrite(TrigPIN, LOW);         //   TrigPIN output as 0V (Logic low level).

  Duration = pulseIn(EchoPIN, HIGH);  // Start counting time, Upto again EchoPIN back to logic "High Level" and puting   the "Time" into variable called "Duration".

  CM = (Duration / 58.8);             //   Convert Distance into CM.

  return CM;                          // Return to   CM.

}

void ChangePath()  // Path Change   loop.

{

  digitalWrite(12, LOW);

  digitalWrite(11, LOW);

  digitalWrite(10, LOW);

  digitalWrite(9, LOW);

  analogWrite(enA, 0);

  analogWrite(enB, 0);  // Robot Stop.

  digitalWrite(12, LOW);

  digitalWrite(10, LOW);

  digitalWrite(11, HIGH);

  digitalWrite(9, HIGH);

  analogWrite(enA, 200);

  analogWrite(enB, 200);  //   Robot run Backward direction.

  digitalWrite(12, LOW);

  digitalWrite(11, LOW);

  digitalWrite(10, LOW);

  digitalWrite(9, LOW);

  analogWrite(enA, 0);

  analogWrite(enB, 0);       // Robot Stop.

  myservo.write(12);         // Check Distance to the Right.

  delay(500);                //   Delay for 0.5s.

  RightDistance = Search();  // Set Right Distance.

  delay(500);                // Delay for 0.5s.

  myservo.write(160);        // Check   Distance to the Left.

  delay(1000);               // Delay for 1s.

  LeftDistance = Search();   // Set Left Distance.

  delay(500);                // Delay for   0.5s.

  myservo.write(80);         // Return to center.

  delay(500);                //   Delay for 0.5s.

  CompareDistance();         // Find the longest distance.

}

void ChangePath1()  // Path Change   loop.

{

  digitalWrite(12, LOW);

  digitalWrite(11, LOW);

  digitalWrite(10, LOW);

  digitalWrite(9, LOW);

  analogWrite(enA, 0);

  analogWrite(enB, 0);  // Robot Stop.

  digitalWrite(12, LOW);

  digitalWrite(10, LOW);

  digitalWrite(11, HIGH);

  digitalWrite(9, HIGH);

  analogWrite(enA, 200);

  analogWrite(enB, 200);  //   Robot run Backward direction.

  digitalWrite(12, LOW);

  digitalWrite(11, LOW);

  digitalWrite(10, LOW);

  digitalWrite(9, LOW);

  analogWrite(enA, 0);

  analogWrite(enB, 0);       // Robot Stop.

  myservo.write(12);         // Check Distance to the Right.

  delay(500);                //   Delay for 0.5s.

  RightDistance = Search();  // Set Right Distance.

  delay(500);                // Delay for 0.5s.

  myservo.write(160);        // Check   Distance to the Left.

  delay(1000);               // Delay for 1s.

  LeftDistance = Search();   // Set Left Distance.

  delay(500);                // Delay for   0.5s.

  myservo.write(80);         // Return to center.

  delay(500);                //   Delay for 0.5s.

  CompareDistance1();        // Find the longest distance.

}

void CompareDistance1()  // Distance Compare loop.

{

  if (RightDistance < LeftDistance)  // If Right is less obstructed.

  {

    digitalWrite(12, HIGH);

    digitalWrite(10, LOW);

    digitalWrite(11, LOW);

    digitalWrite(9, HIGH);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    delay(300);                             //   Robot Turn into Right direction.

  } else if (LeftDistance < RightDistance)  // If Left is less obstructed.

  {

    digitalWrite(12, LOW);

    digitalWrite(10, HIGH);

    digitalWrite(11, HIGH);

    digitalWrite(9, LOW);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    delay(300);  //   Robot Turn into Left direction.

  } else         //   If both are equally obstructed.

  {

    digitalWrite(12, LOW);

    digitalWrite(10, HIGH);

    digitalWrite(11, HIGH);

    digitalWrite(9, LOW);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    delay(700);  //   Robot Turn Around.

  }

}

void CompareDistance()  // Distance Compare loop.

{

  if (RightDistance > LeftDistance)  // If Right is less obstructed.

  {

    digitalWrite(12, HIGH);

    digitalWrite(10, LOW);

    digitalWrite(11, LOW);

    digitalWrite(9, HIGH);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    delay(300);                             //   Robot Turn into Right direction.

  } else if (LeftDistance > RightDistance)  // If Left is less obstructed.

  {

    digitalWrite(12, LOW);

    digitalWrite(10, HIGH);

    digitalWrite(11, HIGH);

    digitalWrite(9, LOW);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    delay(300);  //   Robot Turn into Left direction.

  } else         //   If both are equally obstructed.

  {

    digitalWrite(12, LOW);

    digitalWrite(10, HIGH);

    digitalWrite(11, HIGH);

    digitalWrite(9, LOW);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    delay(700);  //   Robot Turn Around.

  }

}

void loop() {

  if (Serial.available()) {

    t = Serial.read();

    if (t != 'S') { Serial.println(t); }

  }

  if (t == 'F')  // move forward (all motors rotate in forward direction)

  {

    digitalWrite(12, HIGH);

    digitalWrite(10, HIGH);

    digitalWrite(11, LOW);

    digitalWrite(9, LOW);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    // Serial.print("F");

  } else if (t == 'B')  // move backward (all motors rotate in reverse direction)

  {

    digitalWrite(12, LOW);

    digitalWrite(10, LOW);

    digitalWrite(11, HIGH);

    digitalWrite(9, HIGH);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    Serial.print("B");

  } else if (t == 'L')  // turn left (left side motors rotate in reverse direction, right side motors rotate in forward direction)

  {

    digitalWrite(12, LOW);

    digitalWrite(10, HIGH);

    digitalWrite(11, HIGH);

    digitalWrite(9, LOW);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    // Serial.print("L");

  } else if (t == 'R')  // turn right (left side motors rotate in forward direction, right side motors rotate in reverse direction)

  {

    digitalWrite(12, HIGH);

    digitalWrite(10, LOW);

    digitalWrite(11, LOW);

    digitalWrite(9, HIGH);

    analogWrite(enA, 200);

    analogWrite(enB, 200);

    // Serial.print("R");

  }

  else if (t == 'W') {

    //Obstacle Avoidance

    //   Main loop.

    do {

      startMilis = millis();

      myservo.write(80);    // Tells the Servo   to position at 80 degrees (Facing forward).

      delay(100);           //   Delay for 0.1s.

      Distance = Search();  // Measuring   the Distance in CM.

      if (Distance < 30) {

        startMilis = millis();

        while (millis() - startMilis < period1) {

          // If obstacle   found in 30cm.

          analogWrite(enA, 200);

          analogWrite(enB, 200);  // Speed down.

          ChangePath();           // If forward is blocked Change   direction.

        }

      }

      else if ((Distance >= 30) && (Distance < 60)) {

        startMilis = millis();

        while (millis() - startMilis < period1) {

          {

            // If obstacle   found between 30cm to 60cm.

            digitalWrite(12, HIGH);

            digitalWrite(10, HIGH);

            digitalWrite(11, LOW);

            digitalWrite(9, LOW);

            analogWrite(enA, 200);

            analogWrite(enB, 200);

          }

        }

      } else if ((Distance >= 60) && (Distance < 90)) {

        startMilis = millis();

        while (millis() - startMilis < period1) {

          digitalWrite(12, HIGH);

          digitalWrite(10, HIGH);

          digitalWrite(11, LOW);

          digitalWrite(9, LOW);

          analogWrite(enA, 200);

          analogWrite(enB, 200);

        }

      }

      else {

        startMilis = millis();

        while (millis() - startMilis < period1) {

          // If obstacle cannot be   found in 90cm.

          digitalWrite(12, HIGH);

          digitalWrite(10, HIGH);

          digitalWrite(11, LOW);

          digitalWrite(9, LOW);

          analogWrite(enA, 250);

          analogWrite(enB, 250);

        }

      }

    } while (millis() - startMilis < 3000);

  }

  else if (t == 'w') {

    digitalWrite(12, LOW);

    digitalWrite(11, LOW);

    digitalWrite(10, LOW);

    digitalWrite(9, LOW);

    analogWrite(enA, 0);

    analogWrite(enB, 0);

  } else if (t == 'U') {

    do {

      startMilis = millis();

      //Obstacle Avoidance

      //   Main loop.

      myservo.write(80);    // Tells the Servo   to position at 80 degrees (Facing forward).

      delay(100);           //   Delay for 0.1s.

      Distance = Search();  // Measuring   the Distance in CM.

      if (Distance > 15) {

        startMilis = millis();

        while (millis() - startMilis < period) {

          // If obstacle   found in 30cm.

          analogWrite(enA, 0);

          analogWrite(enB, 0);  // Speed down.

          ChangePath1();        // If forward is blocked Change   direction.

        }

      }

      else if ((Distance >= 0) && (Distance < 15)) {

        startMilis = millis();

        while (millis() - startMilis < period) {

          digitalWrite(12, HIGH);

          digitalWrite(10, HIGH);

          digitalWrite(11, LOW);

          digitalWrite(9, LOW);

          analogWrite(enA, 200);

          analogWrite(enB, 200);

        }

      }

    } while (millis() - startMilis < 3000);

  }

  else if (t == 'u') {

    digitalWrite(12, LOW);

    digitalWrite(11, LOW);

    digitalWrite(10, LOW);

    digitalWrite(9, LOW);

    analogWrite(enA, 0);

    analogWrite(enB, 0);

  } else if (t == 'S')  //STOP (all motors stop)

  {

    digitalWrite(12, LOW);

    digitalWrite(11, LOW);

    digitalWrite(10, LOW);

    digitalWrite(9, LOW);

    analogWrite(enA, 0);

    analogWrite(enB, 0);

  }

  delay(10);

}