**OBSTACLE AVOIDING ROBOT**

**CHAPTER NO. TITLE PAGE No**

**1. PROBLEM STATEMENT 3**

**2. INTRODUCTION 3**

**3. PROBLEM DESCRIPTION 3**

**4. WORKFLOW 4**

**5. IMPLEMENTATION 5**

**5.1 Preparing the Rotating Parts 5**

**5.1.2 Preparing Microcontroller Board 5**

**5.1.3 Preparing the Interface Module 6 5.1.4 Preparing the Proximity Sensor Module 6**

**5.1.4.1 Working Principle: 7**

**5.1.5 Preparing Servo Motor 8**

**6. CONCLUSION 9**

**7. CODE 9**

**1 PROBLEM STATEMENT:**

To develop an obstacle avoidance robot using ultrasonic sensor.

**2.INTRODUCTION:**

Now day’s many industries are using robots due to their high level of performance and reliability and which is a great help for human beings. The [obstacle avoidance robotics](http://www.edgefxkits.com/obstacle-avoidance-robotic-vehicle) is used for detecting obstacles and avoiding the collision. This is an autonomous robot.

The obstacle detection is primary requirement of this autonomous robot. The robot gets the information from surrounding area through mounted sensors on the robot. Some sensing devices used for obstacle detection like bump sensor, infrared sensor, ultrasonic sensor etc. Ultrasonic sensor is most suitable for obstacle detection and it is of low cost and has high ranging capability.

**3.PROBLEM DESCRIPTION:**

The obstacle avoidance [robotic vehicle](http://www.edgefxkits.com/arduino-operated-obstacle-avoidance-robot) uses ultrasonic sensors for its movements. An Arduino UNO is used to achieve the desired operation. The motors are connected through motor driver IC to the Arduino. The ultrasonic sensor is attached in front of the robot.

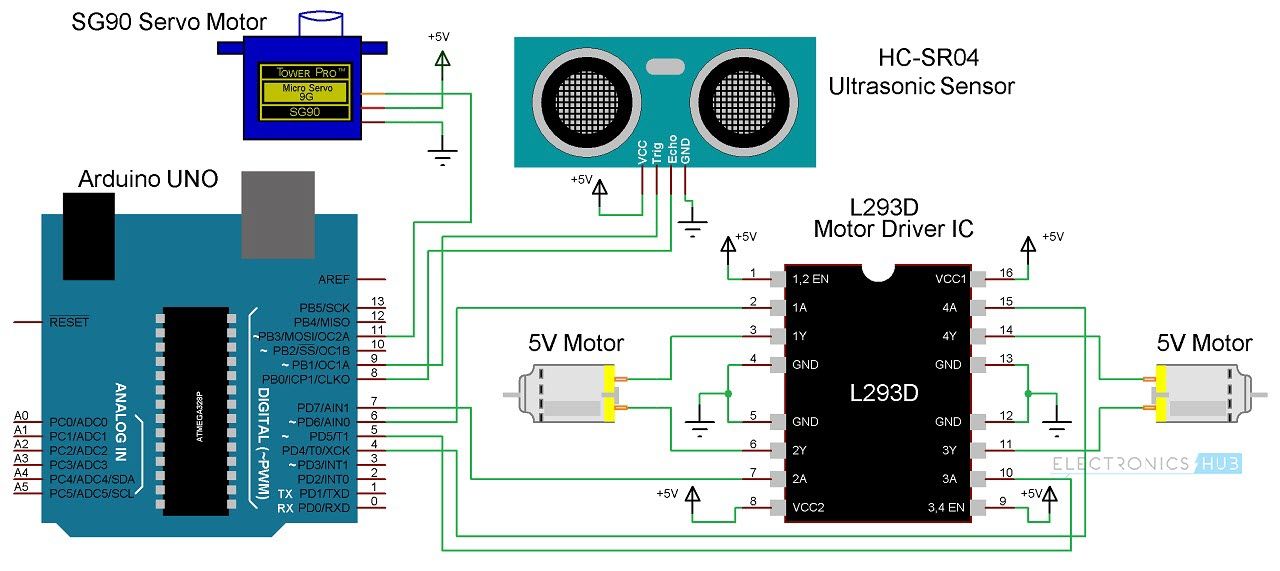
Whenever the robot is going on the desired path the ultrasonic sensor transmits the ultrasonic waves continuously from its sensor head. Whenever an obstacle comes ahead of it the ultrasonic waves are reflected back from an object and that information is passed to the Arduino. The Arduino controls the motors left, right, back, front, based on ultrasonic signals. In order to control the speed of each motor pulse width modulation is used (PWM).

**4.WORKFLOW:**



**4.1 GENERAL OUTLINE OF WORKFLOW**

**5 IMPLEMENTATION:**

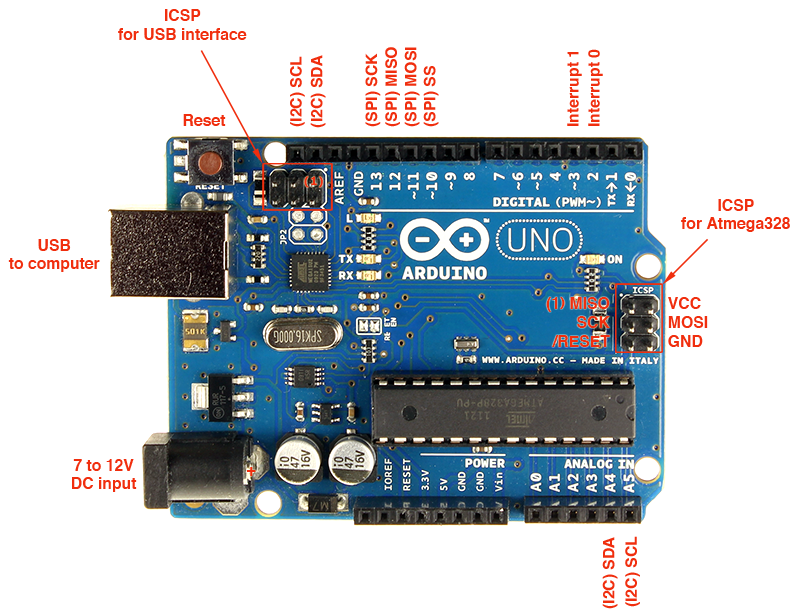


**5.1 CIRCUIT DIAGRAM**

**5.1.1 Preparing the Rotating Parts**

The robot has been designed to be using 2 motors each with its gearbox, 2 wheels -one wheel for each motor gearbox- and 1 passive wheel. The two motorized wheel are placed in the rear while the passive wheel is placed in the front. The passive wheel is basically a ball-shaped bearing than can be directed to any direction by the force produced by the two motorized wheels. In this project the motors are twins, rated at 9vdc.

**5.1.2 Preparing Microcontroller Board**

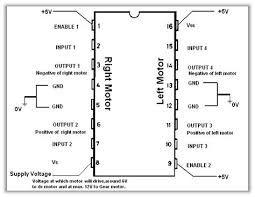
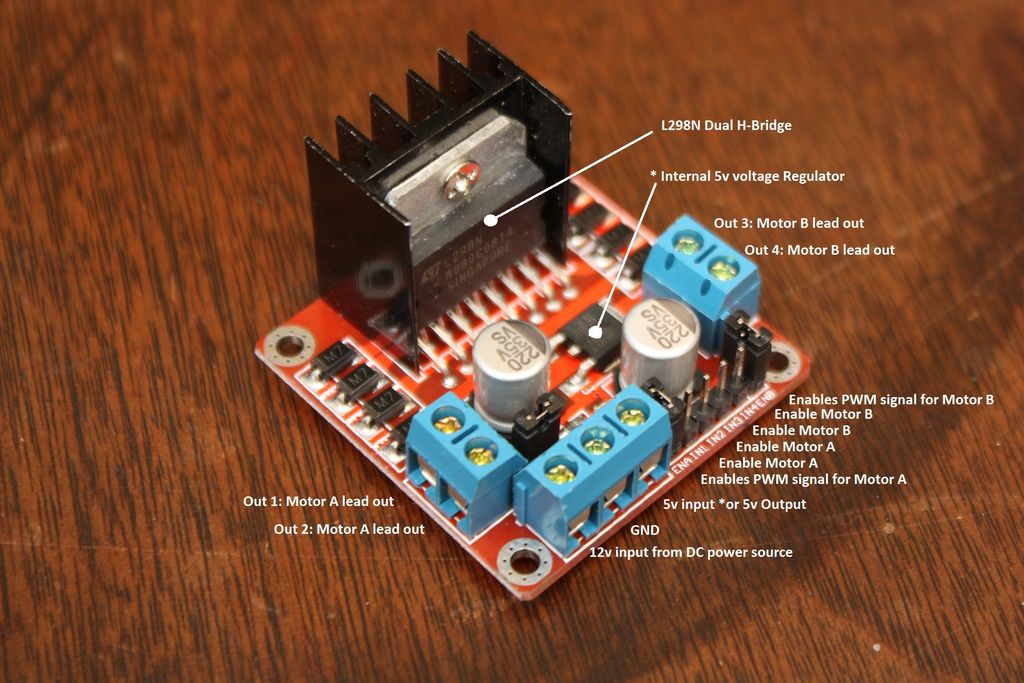
In this activity the Atmel ATmega328P has been used as the microcontroller (uC). This is a 8-bit-microcontroller that has 32 kilo bytes flash memory and able to process instructions at speed of 20 MIPS at the fastest. In particular for this occasion, a ready-to-use Arduino Uno R3 board has been used. 

**5.2 LAYOUT OF ARDUINO**

**5.1.3 Preparing the Interface Module**

The interface module in this context is a part of the robot system that converts logics 1 and 0 of the microcontroller’s pins to applicable and suitable voltage and current to the motors. In this case, the two motors are identical, of 6vdc.

Direction and Speed of motor are determined by the inputs give to the motor controller L239N.



**5.3 L239N Motor Controller 5.4 Layout of L239N**

**5.1.4 Preparing the Proximity Sensor Module**

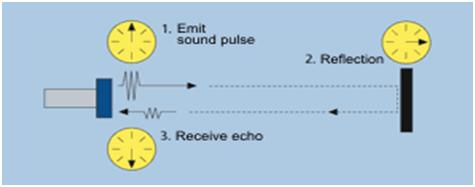
Proximity sensing means detecting the presence or absence of an object. It is also mentioned that the measurement can be done by different type of sensors including hall-effect sensors, inductive sensors, ultrasonic sensors and contact-type sensors. In practice ultrasonic and infra-red sensors are often used for obstacle avoiding robots. Here we use HC-SR04 Ultrasonic Sensor.



**5.5 HC-SR04 Ultrasonic Sensor**

**5.1.4.1 Working Principle:**

The ultrasonic sensor emits the short and high frequency signal. These propagate in the air at the velocity of sound. If they hit any object, then they reflect back echo signal to the sensor. The ultrasonic sensor consists of a multi vibrator, fixed to the base. The multi vibrator is combination of a resonator and vibrator. The resonator delivers ultrasonic wave generated by the vibration.  The ultrasonic sensor actually consists of two parts; the emitter which produces a 40 kHz sound wave and detector detects 40 kHz sound wave and sends electrical signal back to the microcontroller. The ultrasonic sensor enables the robot to virtually see and recognize object, avoid obstacles, measure distance. The operating range of ultrasonic sensor is 10 cm to 30 cm.



**5.6 Working of Ultrasonic Sensor**

When an electrical pulse of high voltage is applied to the ultrasonic transducer it vibrates across a specific spectrum of frequencies and generates a burst of sound waves. Whenever any obstacle comes ahead of the ultrasonic sensor the sound waves will reflect back in the form of echo and generates an electric pulse. It calculates the time taken between sending sound waves and receiving echo. The echo patterns will be compared with the patterns of sound waves to determine detected signal’s condition.

The ultrasonic receiver shall detect signal from the ultrasonic transmitter while the transmit waves hit on the object. The combination of these two sensors will allow the robot to detect the object in its path. The ultrasonic sensor is attached in front of the robot and that sensor will also help the robot navigate through the hall of any building.

**5.1.5 Preparing Servo Motor**

Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the [motor](https://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&freeText=motor&search_type=jamecoall) determines position of the shaft, and based on the duration of the pulse sent via the control wire; the [rotor](https://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=cat_3540&subCategoryName=Electromechanical%20%2F%20Switches%20%2F%20Rotary&category=354055&refine=1&position=1&history=kv7hqebe%7CfreeText~rotor%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV%405hha4bcd%7Ccategory~35%5EcategoryName~category_root%5Eposition~1%5Erefine~1%5EsubCategoryName~Electromechanical%5EprodPage~50%5Epage~SEARCH%252BNAV) will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns.



**5.7 Mini Servo Motor**

**5.1.6 Designing Motion Procedure**

The motion procedure of the robot has been designed as follows:

**Stop**: Both motor do not rotate.

**Move forward:** Both motor rotate forward.

**Turn right:** The left motor rotates forward. The right motor rotates backward.

**Turn left:** The right motor rotates forward. The left motor rotate backward.

**Move backward:** Both motors rotate backward.

**6. CONCLUSION :**

The designed small avoiding obstacle robot has been developed successfully in this project and operated nearly as expected. For this robot design, the robot responded to an obstacle at its best at coming angle of 45°, 90° and 135°. The reason for this, is that at this coming angles one of the sound beam hit the obstacle’s surface perpendicularly. At other coming angles the robot responded to obstacle existence with less sensitivity.

**7. CODE:**

|  |
| --- |
|  |
|  | #include<Servo.h> |
|  | #include <NewPing.h> |
|  | Servo name\_servo; |
|  | const int trigPin = 5; // for ultrasonic sensor |
|  | const int echoPin = 6; |
|  |  |
|  | #define MAX\_DISTANCE\_POSSIBLE 1000 |
|  |  |
|  | NewPing sonar( trigPin, echoPin, MAX\_DISTANCE\_POSSIBLE); // sets up ultrasonic sensor |
|  | long duration; |
|  | int distance; |
|  |  |
|  |  |
|  | int pos = 0; |
|  | int maxDist = 0; |
|  | int maxAngle = 0; |
|  | int maxRight = 0; |
|  | int maxLeft = 0; |
|  | int maxFront = 0; |
|  | int course = 0; |
|  | int curDist = 0; |
|  | String motorSet = ""; |
|  | int speedSet = 0; |
|  | #define COLL\_DIST 20 // min space in front |
|  | #define TURN\_DIST COLL\_DIST+10 // min space on sides |
|  | void setup() { |
|  | name\_servo.attach(2); // Port of servo motor |
|  | name\_servo.write(90); // initial position of servo |
|  | pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output |
|  | pinMode(echoPin, INPUT); // Sets the echoPin as an Input |
|  | Serial.begin(9600); // Starts the serial communication |
|  | pinMode(7, OUTPUT); // input1 of motor1 |
|  | pinMode(8, OUTPUT); // input2 of motor 1 |
|  | pinMode(9, OUTPUT); // Speed control of motor 1 |
|  | pinMode(12, OUTPUT); // input1 of motor2 |
|  | pinMode(13, OUTPUT); // input2 of motor2 |
|  | pinMode(11, OUTPUT); // Speed control of motor 2 |
|  |  |
|  | delay(2000); |
|  | checkPath(); // Check for obstacle |
|  | motorSet = "FORWARD"; |
|  | name\_servo.write(90); |
|  | moveForward(); |
|  |  |
|  | } |
|  |  |
|  | void loop() { |
|  | checkForward(); |
|  | checkPath();   |  | | --- | | } | |  | void checkPath() { | |  | int curLeft = 0; | |  | int curFront = 0; | |  | int curRight = 0; | |  | int curDist = 0; | |  | name\_servo.write(144); // leftmost position of servo | |  | delay(120); | |  | for(pos = 144; pos >= 36; pos-=18) // keep reducing servo angle | |  | { | |  | name\_servo.write(pos); | |  | delay(90); | |  | checkForward(); | |  | curDist = readPing(); // read distance between robot and obstacle | |  | if (curDist < COLL\_DIST) { // if distance< min distance possible | |  | checkCourse(); | |  | break; | |  | } | |  | if (curDist < TURN\_DIST) { // if distance < min side distance | |  | changePath(); | |  | } | |  | if (curDist > curDist) {maxAngle = pos;} // angle for which distance > collision distance | |  | if (pos > 90 && curDist > curLeft) { curLeft = curDist;} | |  | if (pos == 90 && curDist > curFront) {curFront = curDist;} | |  | if (pos < 90 && curDist > curRight) {curRight = curDist;} | |  | } | |  | maxLeft = curLeft; | |  | maxRight = curRight; | |  | maxFront = curFront; | |  | } | |  | void setCourse() { | |  | if (maxAngle < 90) {turnRight();} | |  | if (maxAngle > 90) {turnLeft();} | |  | maxLeft = 0; | |  | maxRight = 0; | |  | maxFront = 0; | |  | } | |  |  | |  | void checkCourse() { | |  | moveBackward(); | |  | delay(500); | |  | moveStop(); | |  | setCourse(); | |  | } | |  | void changePath() { | |  | if (pos < 90) {lookLeft();} | |  | if (pos > 90) {lookRight();} | |  | } | |  | int readPing() { | |  | delay(70); | |  | unsigned int uS = sonar.ping(); | |  | int cm = uS/US\_ROUNDTRIP\_CM; | |  | return cm; | |  | } | |  | void checkForward() { if (motorSet=="FORWARD") {digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); } } | |  |  | |  | void checkBackward() { if (motorSet=="BACKWARD") {digitalWrite(8, LOW); | |  | digitalWrite(7, HIGH); | |  | digitalWrite(9,80); | |  | digitalWrite(13, LOW); | |  | digitalWrite(12, HIGH); | |  | digitalWrite(11,80); } } | |  |  | |  | void moveStop() {digitalWrite(8, LOW); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, LOW); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80);} | |  |  | |  | void moveForward() { | |  | motorSet = "FORWARD"; | |  | digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); | |  |  | |  | } | |  | void moveBackward() { | |  | motorSet = "BACKWARD"; | |  | digitalWrite(8, LOW); | |  | digitalWrite(7, HIGH); | |  | digitalWrite(9,80); | |  | digitalWrite(13, LOW); | |  | digitalWrite(12, HIGH); | |  | digitalWrite(11,80); | |  | } | |  | void turnRight() { | |  | motorSet = "RIGHT"; | |  | digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, LOW); | |  | digitalWrite(12, HIGH); | |  | digitalWrite(11,80); | |  | delay(400); | |  | motorSet = "FORWARD"; | |  | digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); | |  | } | |  | void turnLeft() { | |  | motorSet = "LEFT"; | |  | digitalWrite(8, LOW); | |  | digitalWrite(7, HIGH); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); | |  | delay(400); | |  | motorSet = "FORWARD"; | |  | digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); | |  | } | |  |  | |  | void lookRight() { digitalWrite(8, LOW); | |  | digitalWrite(7, HIGH); | |  | digitalWrite(9,80); delay(400); digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); } | |  | void lookLeft() {digitalWrite(8, LOW); | |  | digitalWrite(7, HIGH); | |  | digitalWrite(9,80); delay(400); digitalWrite(8, HIGH); | |  | digitalWrite(7, LOW); | |  | digitalWrite(9,80); | |  | digitalWrite(13, HIGH); | |  | digitalWrite(12, LOW); | |  | digitalWrite(11,80); } | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |