OBSTACLES AVOIDING RACE

A project report submitted in partial fulfillment of requirement for the course

On

Smart System Design

By

Kola.Vennela	**(2203A51210)**
Sanka.Anitha	**(2205A41179)**
Mohammad.Zeeshan	**(2205A41174)**
Vengala.Uday Kumar	**(2203A51227)**

Under the guidance of

Mr. Rajeshwar Rao Arabelli

Asst. Prof. & Director, Centre for Embedded Systems and IoT,

Department of ECE

&

Dr. Ch.Rajendra Prasad

Asst. Prof., Department of ECE

Center for Embedded Systems and Internet of things

SR UNIVERSITY

CERTIFICATE

This is to certify that the course project entitled "Obstacles Avoiding Race" is the bonafide work carried out by K.Vennela(2203A51210),S.Anitha(2205A41179), M.Zeeshan (2205A41174),V.Uday Kumar (2203A51227), in the partial fulfillment of the requirement for the award of course Smart System Design during the academic year 20222023 under our guidance and Supervision.

Mr. Rajeshwar Rao Arabelli

Asst. Prof. & Director, Centre for Embedded Systems and IoT,

Department of ECE

CONTENTS

ABSTRACT

Obstacle avoiding robots play a vital role in various fields, ranging from industrial automation to home applications. This abstract provides a comprehensive overview and analysis of obstacle avoiding robots, focusing on their design, functionality, sensing mechanisms, and navigation algorithms. The aim is to understand the underlying principles and advancements in this domain.

The abstract begins by introducing the concept of obstacle avoiding robots and their significance in modern robotics. It highlights the need for these robots in navigating dynamic environments autonomously while avoiding collisions with obstacles. Various types of sensors used in obstacle detection and ranging are discussed, including ultrasonic, infrared, laser, and vision-based systems. The strengths and limitations of each sensing method are evaluated, considering factors such as accuracy, range, response time, and environmental conditions.

CHAPTER 1
INTRODUCTION

INTRODUCTION:

A project on Obstacle Avoiding Robot is designed here. Robotics is an interesting and fast growing field. Being a branch of engineering, the applications of robotics are increasing with the advancement of technology. The concept of Mobile Robot is fast evolving and the number of mobile robots and their complexities are increasing with different applications. There are many types of mobile robot navigation techniques like path planning, self – localization and map interpreting. An Obstacle Avoiding Robot is a type of autonomous mobile robot that avoids collision with unexpected obstacles. 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 is used for detecting obstacles and avoiding the collision. This is an autonomous robot. The design of the obstacle avoidance robot requires the integration of many sensors according to their task.

OBJECTIVES:

- The objective of this project is to obstacle detection and avoiding it. The robot gets the information from the surrounding area through mounted sensors on the robot.
- The ultrasonic sensor is most suitable for obstacle detection and it is of low cost and has a high ranging capability.

CHAPTER 2
PROJECT DESCRIPTION

BLOCK DIAGRAM OF THE PROJECT

The block diagram of the project is shown in fig. 2.1. The block diagram is consists of two sensors and a fan and a buzzer. The two sensors are temperature sensor and a gas sensor that is mq2. These sensors are connected to arduino. Whenever the temperature sensor detects the rise in temperature then immediately it sends an information to arduino, then arduino switches on the fan. And the other sensor is mq2. If any gas leaks then it immediately sends an information to arduino and then it alerts us.

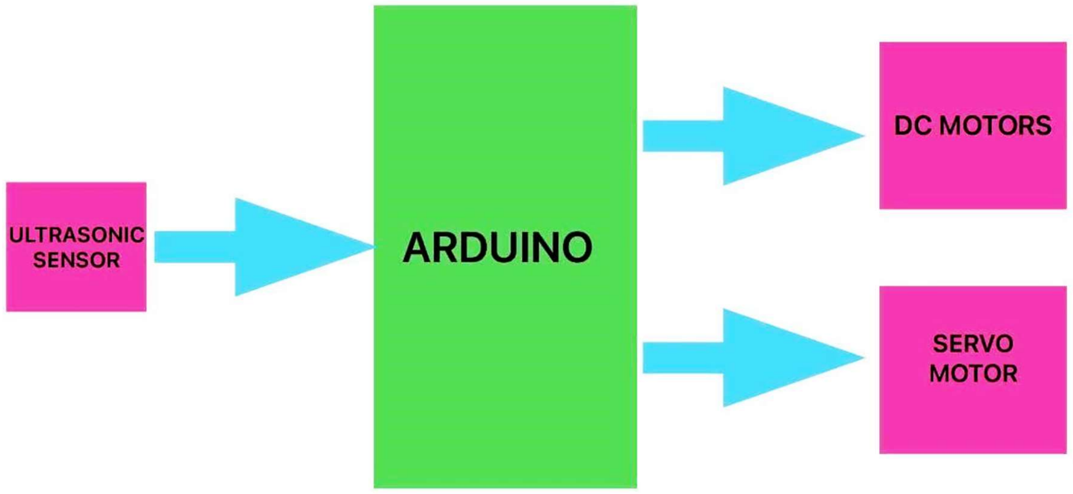


Fig.2.1 Block Diagram

HARDWARE DESCRIPTION

Arduino UNO:

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet) is shown in fig. 2.2. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode. Revision 3 of the board has the following new features: 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes. Stronger RESET circuit. Atmega 16U2 replace the 8U2. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

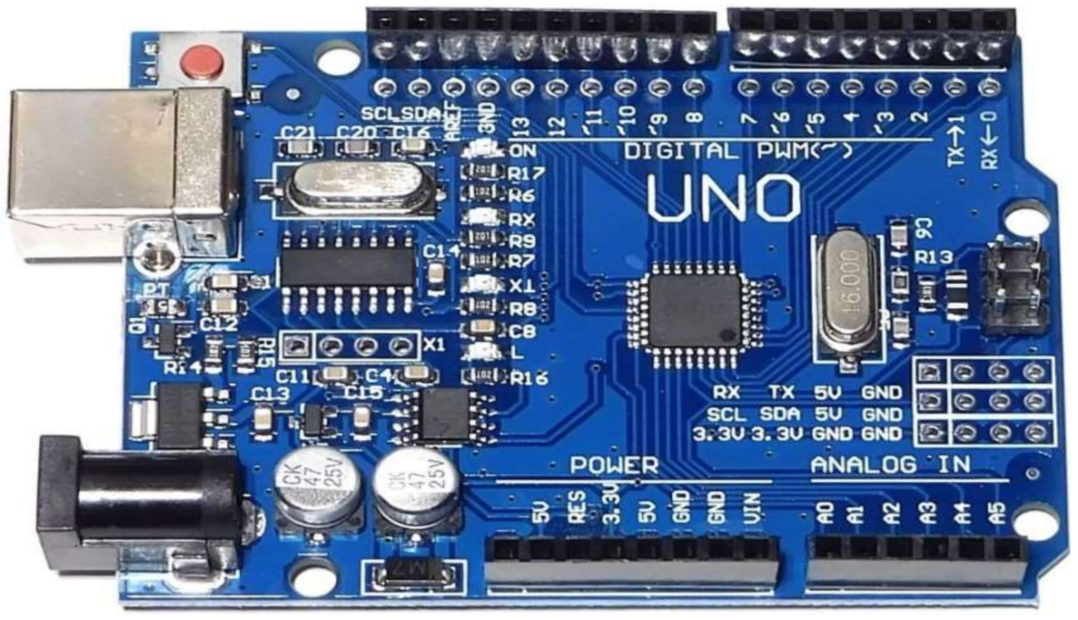


Fig. 2.2 Arduino Uno

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)

Applications:

- Xoscillo, an open-source oscilloscope
- Arduinome, a MIDI controller device that mimics the Monome
- OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars
- Gameduino, an Arduino shield to create retro 2D video games
- ArduinoPhone, a do-it-yourself cellphone
- Water quality testing platform
- Automatic titration system based on Arduino and stepper motor
- Low cost data glove for virtual reality applications
- Impedance sensor system to detect bovine milk adulteration
- Homemade CNC using Arduino and DC motors with close loop control by Homofaciens
- DC motor control using Arduino and H-Bridge

Ultrasonic sensor:

An ultrasonic sensor shown in the Fig. 2.3 is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High-frequency sound waves reflect from boundaries to produce distinct echo patterns. Ultrasonic sensing is one of the best ways to sense proximity and detect levels with high reliability. Ultrasonic sensing is one of the best ways to sense proximity and detect levels with high reliability.

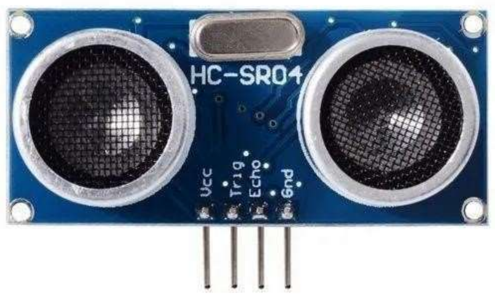


Fig. 2.3Ultrasonic Sensor(HC-SR04)

Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensor, like many others, uses a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. The working principle of this module is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated.

Ultrasonic sensors are a great solution for the detection of clear objects. For liquid level measurement, applications that use infrared sensors, for instance, struggle with this particular use case because of target translucence. For presence detection, ultrasonic sensors detect objects regardless of the colour, surface, or material (unless the material is very soft like wool, as it would absorb sound.) To detect transparent and other items where optical technologies may fail, ultrasonic sensors are a reliable choice.

Our ultrasonic distance, level, and proximity sensors are commonly used with microcontroller platforms like Raspberry Pi, ARM, PIC, Arduino, Beagle Board, and more. Ultrasonic sensors transmit sound waves toward a target and will determine its distance by measuring the time it took for the reflected waves to return to the receiver. This sensor is an electronic device that will measure the distance of a target by transmitting ultrasonic sound waves, and then will convert the reflected sound into an electrical signal. Our sensors are often used as proximity sensors. Ultrasonic sensors are also used in obstacle avoidance systems, as well as in manufacturing. Our Short-range offer the opportunity for closer range detection where you may need a sensor that ranges objects as close to 2cm. These are also built with very low power requirements in mind, as well as environments where noise rejection is necessary.

NOTE: In some cases, the target object is so small that the reflected ultrasonic signal is insufficient for detection, and the distance cannot be measured correctly.

Features:

- Distance: 2cm – 400cm.
- Stable and long life
- Fast response and High sensitivity ✓ Accuracy: 0.3cm.
- Beam Angle: Max 15 degrees.
- Power supply needs: 5V
- Interface type: Analog
- Pin Definition: 1-VCC 2-TRIGGER 3-ECHO 4-GNG

L293D Motor Driver Shield:

The Arduino Motor Shield is based on the L293D, which is a Half-bridge driver designed to drive inductive loads such as relays, solenoids, DC and stepping motors. It lets you drive two DC motors with your Robomart Arduino board, controlling the speed and direction of each one independently. You can also measure the motor current absorption of each motor, among other features. The shield is also compatible with DTMF module, which means you can quickly create projects by plugging DTMF modules to the board. The 74HC595 shift register, on the other hand, extends the Arduino's four digital pins to the eight direction control pins of two L293D chips. The L293D shield is a driver board based on the L293 IC that can simultaneously drive four DC motors and two stepper or servo motors. This module has a maximum current of 1.2A per channel and will not operate if the voltage is greater than 25v or less than 4.5v.

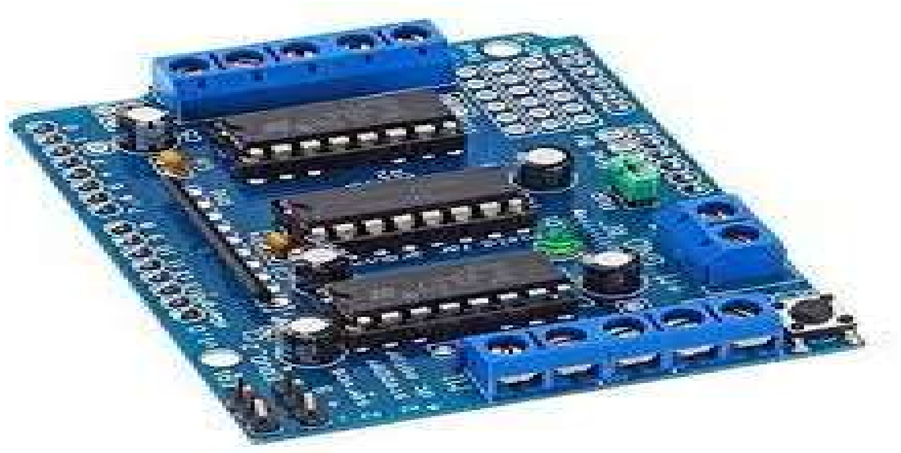


Fig.2.4 L293D Motor Driver Shield

Specification of L293D Motor Driver Shield:

- Operating Voltage: 5V to 12V.
- Motor controller: L293D, Drives 2 DC motors or 1 stepper motor.
- Max current: 600mA per channel.
- Peak Output Current: 1.2 Amp.
- Reset button of arduino is brought to top.

Applications of L293D Motor Driver Shield:

- It is used by arduino users.
- Multiple DIY Projects.

Servo motor:

The servo motor shown in th Fig. 2.5 is the motor used for servo system. The so-called servo system is a control system that acts according to the instructions. It can compare the actual state of the system with the corresponding state of the instructions, and use the comparison result for further control. Servo motors include direct current motors (with brush and brushless) and alternating current motors (synchronous and asynchronous).

Fig.2.5 Servo Motor

The function of the servo motor is to convert the control signal of the controller into the rotational angular displacement or angular velocity of the motor output shaft. Servo motor is used to drive the joints. Generally, servo motors used in robots should have the following characteristics: fast response speed, high starting torque, wide speed range, etc. When used in a collaborative robot, the servo motor should have the characteristics of small size, lightweight, and hollow structure to achieve safe human–robot collaboration.

Servo Motor Working Principle:

A servo shown in the Fig. 2.5 consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier.

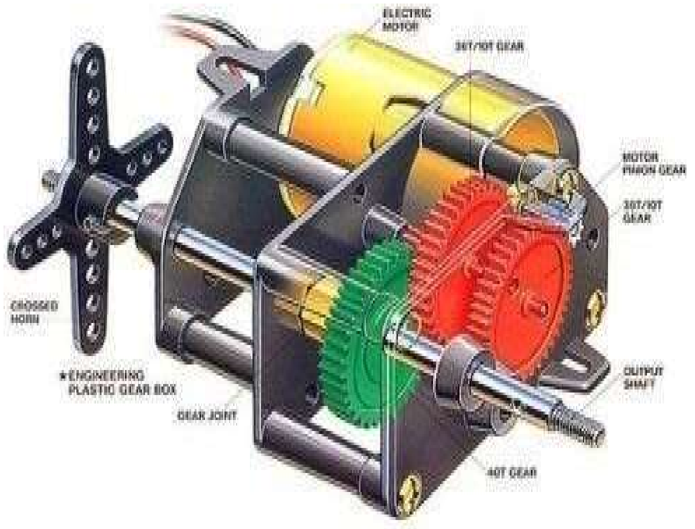


Fig. 2.5 Construction of Servo Motor

Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be

provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor
shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

Special features:

- Low inertia: For this, servo motors normally have light-in-weight construction with large length-to-diameter ratio for rotors. Servo motors, therefore, are distinguished by relatively small-diameter rotors for their frame size.
- High speed of response: This results from high torque-to-weight ratio of a servo motor.
- Hence, these motors can be started, stopped, and reversed very quickly.
- Linear torque speed characteristic i.e., Torque for any control voltage decreases at a definite uniform rate with speed.
- The output torque of the servo motor at any speed is roughly proportional to the applied control voltage. This is true more particularly in respect of starting torque (also called stall torque or locked rotor torque).
- The servo motor operates stably i.e. it does not oscillate or overshoot.

HC-05 Sensor:

The HC-05 is a popular module which can add two-way (full-duplex) wireless functionality to your projects. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality makes this process a lot easier. The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART. We can also configure the default values of the module by using the command mode. So if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you. However do not expect this module to transfer multimedia like photos or songs; you might have to look into the CSR8645 module for that. The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default like a Phone or Laptop. There are many android applications that are already available which device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description. It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU.

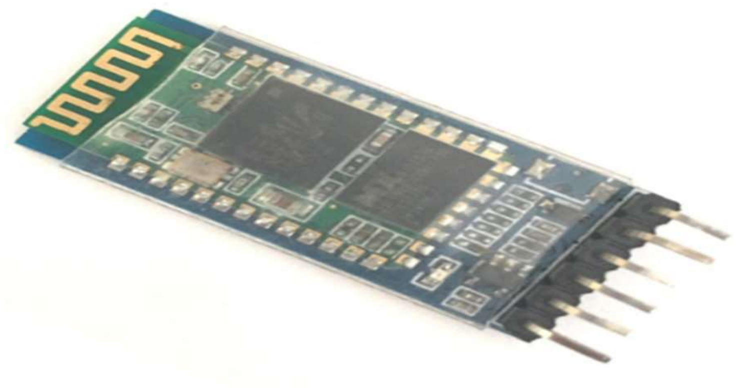


Fig. 2.6 HC-05 sensor

Applications:

- Wireless communication between two microcontrollers
- Communicate with Laptop, Desktops and mobile phones
- Data Logging application
- Consumer applications
- Wireless Robots
- Home Automation

HC-05 Technical Specifications:

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600, 19200, 38400, 57600, 115200, 230400, 460800

SOFTWARE DESCRIPTION:

The software used here is ARDUINO SOFTWARE:

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches:

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB:

Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the ino extension on save.

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save

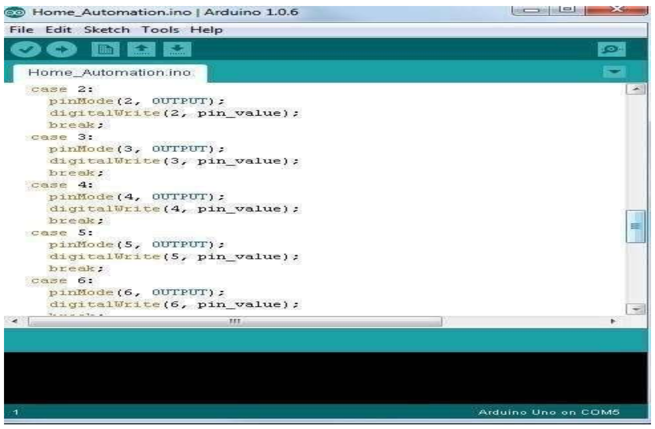
Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools,and help.

Programming on Arduino uno:


Home_Automation.ino | Arduino 1.0.6
File Edit Sketch Tools Help
Home_Automation.ino
1
Arduino Uno on COM5

```
  case 2:
    pinMode(2, OUTPUT);
    digitalWrite(2, pin_value);
    break;
  case 3:
    pinMode(3, OUTPUT);
    digitalWrite(3, pin_value);
    break;
  case 4:
    pinMode(4, OUTPUT);
    digitalWrite(4, pin_value);
    break;
  case 5:
    pinMode(5, OUTPUT);
    digitalWrite(5, pin_value);
    break;
  case 6:
    pinMode(6, OUTPUT);
    digitalWrite(6, pin_value);
```

Fig.2.6 Software IDE

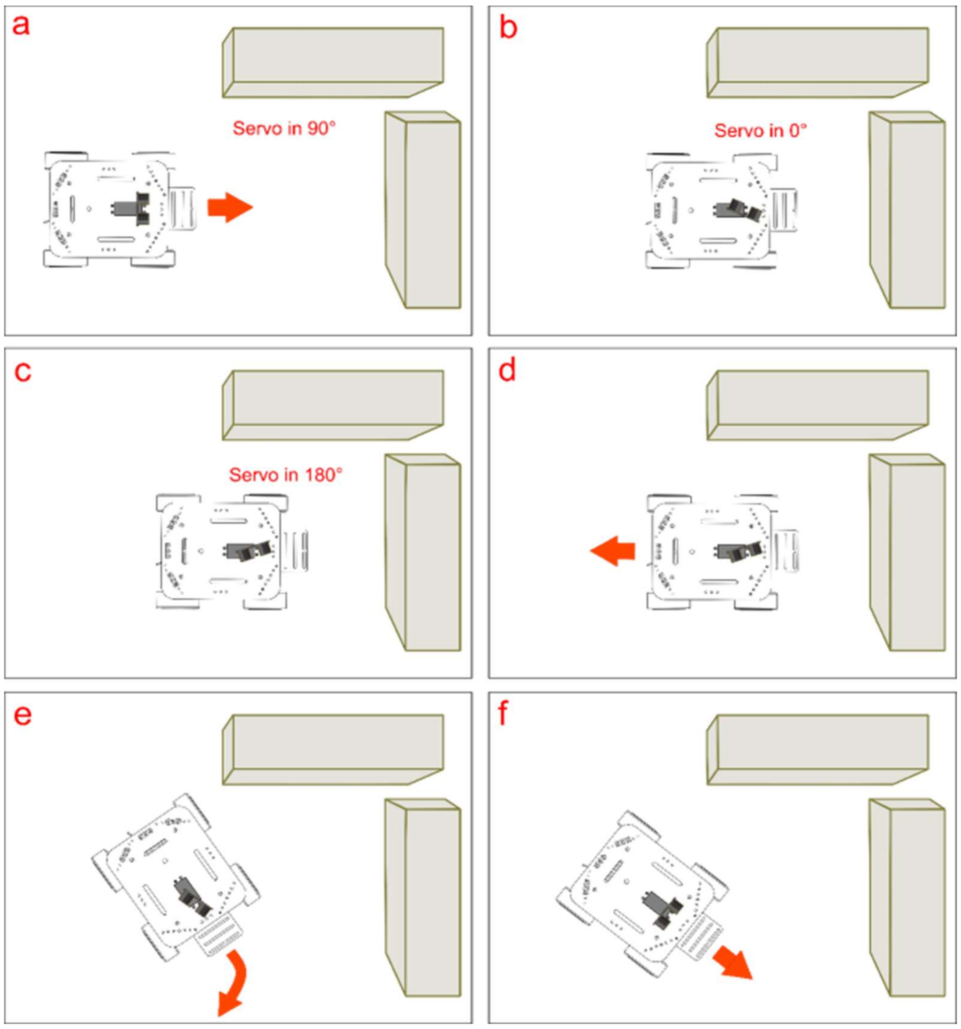
In order for the Arduino-Uno board to be able to interact with the application used in this project certain program (code) needs to be uploaded to the Arduino-Uno.

Arduino Company provides user friendly software which allows writing any code for any function wanted to be performed by the Arduino-Uno and upload it to the board.Refer to appendix A for the full source code of the Arduino-Uno board.

CHAPTER 3
CIRCUIT DIAGRAM AND DESCRIPTION

Working:

The following six diagrams (a,b,c,d,e,f) in Fig. 3.1 briefly illustrate how the three sensors work to avoid obstacles. At first, the car moves forward uninhibited (diagram A). Once it detects obstacle, the sensors command the car to stop; the servo will shift to 0°, survey its surroundings (diagram B), then shift to 180° and survey its surroundings once more (diagram C). The sensors will process the readings from the 0° and 180° angles to determine which direction the nearest obstacle is in — in the case of the below diagrams, that would be the obstacle in the 0° reading (diagram B).

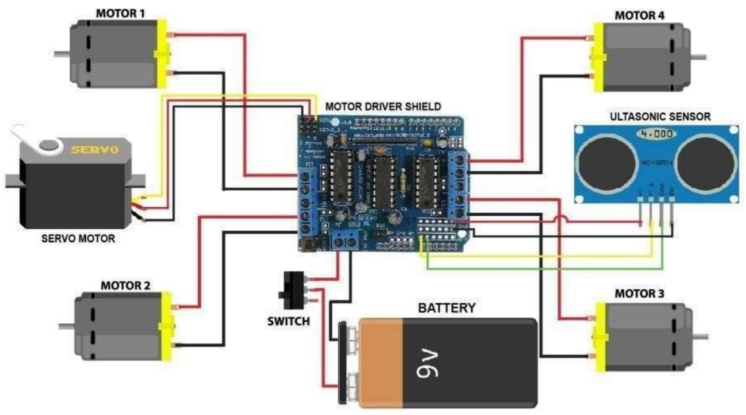


.

Fig.3.1 working process

Circuit connections:

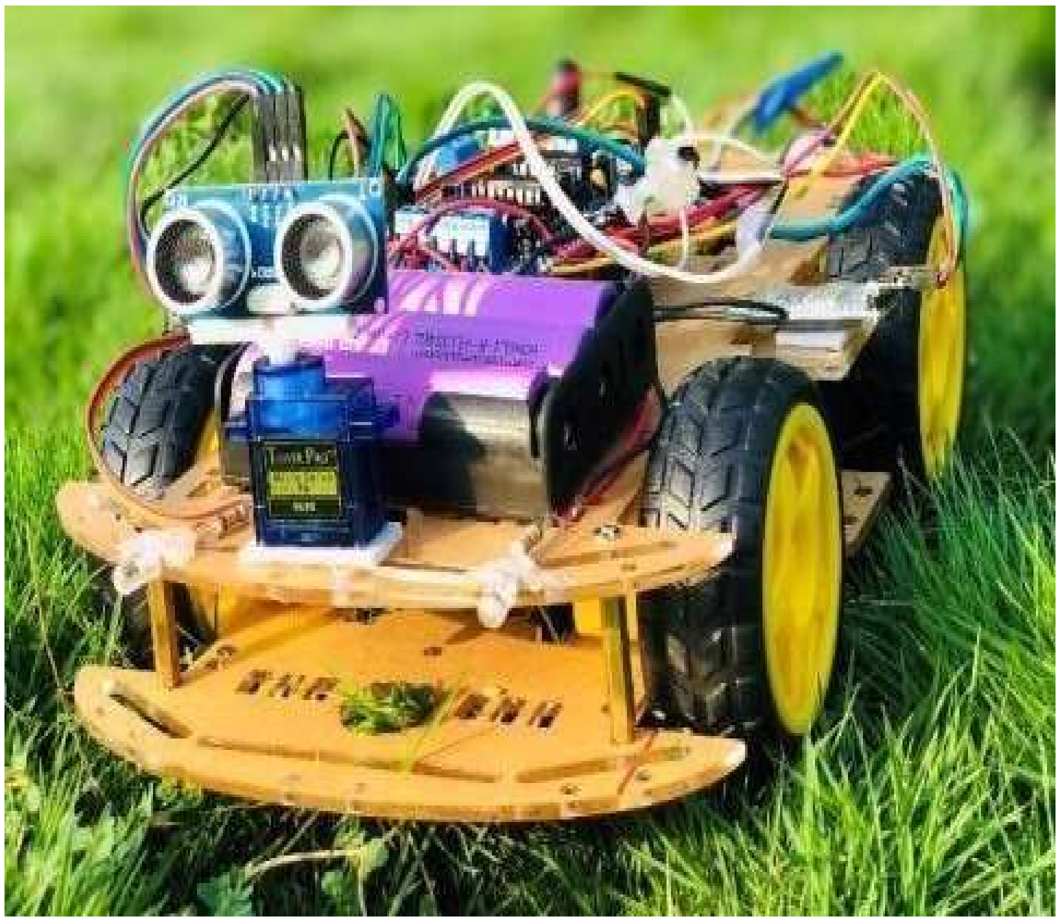
The Circuit connections is as shown in below fig 3.2



3.2 Circuit connections

Result:

The experimental result is as shown in below fig 3.3



3.3 Experimental result

In conclusion we made successfully to apply the OBSTACLE AVOIDING ROBOT USING ARDUINO and it was user friendly and cost effective. User friendly as in anyone can use just a click of the ON & OFF switch. And it is cost effective as in it will cost exactly as the project requires (optimum price).

APPLICATIONS:

- Automated lawn mover.
- Smart room cleaner etc.
- Obstacle avoiding robots can be used in almost all mobile robot navigation systems.
- They can also be used in dangerous environments, where human penetration could be fatal.
- Unmanned vehicle driving.
- Mining Vehicle that uses Obstacle Detection.

ADVANTAGES:

- Very quick to compute
- Simple detection of edges and their orientation.
- Improved signal to noise ratio with a better detection with noise Detect object at distance

DISADVANTAGES:

- May be subject to false positives and false negatives.
- Sensitive to noise.
- May be inaccurate.
- Complex and time-consuming computation false zero crossing time.Accuracy affected by sunlight

CHAPTER 4
CONCLUSION

CONCLUSION:

Based on all the systems this presents the features to be possessed by an ideal vehicle for obstacle avoiding robot. An ideal vehicle should be available from all over the world to a user and in real time homes as automated lawn mower, automated vacuum cleaner and while earth quakes life save robots. This project is absolutely future changing one.

FUTURE SCOPE:

As we know our project is obstacle avoiding robot which means it avoids the obstacles and tries to move forward by overcoming it. Now if we extend this idea to the large scale manufacturing, we can make cars which are auto piloted. This reduces driving task to the human being and also saves driver’s expenses.

BIBLIOGRAPHY

Source:

1. h.ps://www.elprocus.com/obstacle-avoidance-roboCc-vehicle/
2. h.ps://www.maxboCx.com/arCcles/how-ultrasonic-sensors-work.html
3. h.ps://www.robomart.com/l293d-motor-driver-shield
4. h.ps://www.sciencedirect.com/topics/engineering/servo-motor
5. h.ps://www.dfrobot.com/blog-559.html
6. h.ps://www.electronicshub.org/obstacle-avoiding-robot-arduino/

APPENDIX

```
#include <Servo.h>
#include <AFMotor.h>
#define Echo A0
#define Trig A1
#define motor 10
#define Speed 120
#define spoint 103
#define horn 13
#define flight 2
#define blight 11
char value;
int distance;
int Left;
int Right;
int L = 0;
int R = 0;
int L1 = 0;
int R1 = 0;
Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);
void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(horn, OUTPUT);
  pinMode(blight, OUTPUT);
  pinMode(flight,OUTPUT);
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
void loop() {
  Obstacle();
  Bluetoothcontrol();
  voicecontrol();
}
void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
  } else if (value == 'B') {
    backward();
  } else if (value == 'L') {
    left();
  } else if (value == 'R') {
    right();
  } else if (value == 'S') {
    Stop();
  }
}
void Obstacle() {
  distance = ultrasonic();
  if (distance <= 12) {
    Stop();
    backward();
    delay(100);
    Stop();
    L = leftsee();
    servo.write(spoint);
    delay(800);
    R = rightsee();
    servo.write(spoint);
    if (L < R) {
      right();
      delay(500);
      Stop();
      delay(200);
    } else if (L > R) {
      left();
      delay(500);
      Stop();
      delay(200);
    }
  } else {
    forward();
  }
}
void voicecontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
    if (value == '^') {
      forward();
    } else if (value == '-') {
      backward();
    } else if (value == '<') {
      L = leftsee();
      servo.write(spoint);
      if (L >= 10 ) {
        left();
        delay(500);
        Stop();
      } else if (L < 10) {
        Stop();
      }
    } else if (value == '>') {
      R = rightsee();
      servo.write(spoint);
      if (R >= 10 ) {
        right();
        delay(500);
        Stop();
      } else if (R < 10) {
        Stop();
      }
    } else if (value == '*') {
      Stop();
    }
    else if (value == 'W'){
     digitalWrite(flight,HIGH);
      //digitalWrite(flight,LOW);
    }
    else if (value == 'w'){
      digitalWrite(flight,LOW);
    }
    else if (value == 'U'){
      digitalWrite(blight,HIGH);
    }

    else if (value == 'u'){
      digitalWrite(blight,LOW);
    }
    else if (value == 'V'){
      digitalWrite(horn,HIGH);
      delay(10);
    }
```

```
    else if (value == 'v'){
      digitalWrite(horn,LOW);
    }


    }
  }

// Ultrasonic sensor distance reading function
int ultrasonic() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(4);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  long t = pulseIn(Echo, HIGH);
  long cm = t / 29 / 2; //time convert distance
  return cm;
}
void forward() {
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
  }
void backward() {
  M1.run(BACKWARD);
  M2.run(BACKWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD);

  digitalWrite(horn,HIGH);
  digitalWrite(blight,HIGH);
   delay(500);
   digitalWrite(horn,LOW);
```

```
 digitalWrite(blight,LOW);
}
void right() {
  M1.run(BACKWARD);
  M2.run(BACKWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
}
void left() {
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD);
}
void Stop() {
  M1.run(RELEASE);
  M2.run(RELEASE);
  M3.run(RELEASE);
  M4.run(RELEASE);
}
int rightsee() {
  servo.write(20);
  delay(800);
  Left = ultrasonic();
  return Left;
}
int leftsee() {
  servo.write(180);
  delay(800);
  Right = ultrasonic();
  return Right;
}
```

***