OBSTACLE AVOIDING ROBOT

## ABSTRACT

## Obstacle avoidance is a critical task for autonomous robots operating in unstructured environments. This paper presents the design and implementation of an obstacle avoidance robot capable of navigating in a cluttered environment. The robot is equipped with an ultrasonic sensor, which provide accurate distance measurements to surrounding obstacles. A microcontroller-based control system uses these sensor readings to implement a reactive control strategy that guides the robot around obstacles. Experimental results demonstrate that the obstacle avoidance robot is capable of navigating through complex environments and avoiding

**INTRODUCTION**

**AIM of Mini Project:**

The aim of the obstacle avoiding robot is to navigate autonomously in an unstructured environment while avoiding collisions with obstacles. The robot must be able to accurately detect and localize obstacles in its vicinity, and implement a control strategy to safely maneuver around them. The ultimate goal is to develop a robust and efficient robotic system that can perform obstacle avoidance tasks in various scenarios, such as in industrial settings, warehouses, and outdoor environments. Such a system can improve productivity, reduce the risk of accidents, and enhance the overall safety and efficiency of autonomous robotic systems.

**Methodology Used:**

The methodology used for an obstacle avoiding robot involves the following steps:

1. **Sensor Selection:** Selecting appropriate sensors to detect obstacles in the robot's surroundings. Typically, ultrasonic, infrared, or laser sensors are used for this purpose.
2. **Sensor Fusion:** Combining the sensor data from multiple sensors to get an accurate and reliable estimate of the obstacle's location and distance.
3. **Control Strategy:** Developing a control strategy that allows the robot to reactively avoid obstacles in real-time. This may involve implementing a simple reactive control system or a more advanced planning and control strategy.
4. **Testing and Validation:** Testing the robot in various environments to validate its performance and ensure that it can safely navigate through obstacles.
5. **Optimization:** Optimizing the robot's performance by fine-tuning the control parameters, sensor configuration, and overall design.
6. **Integration:** Integrating the obstacle avoidance system into a larger robotic system and interfacing it with other modules such as localization, mapping, and navigation.

By following this methodology, engineers can design and develop effective obstacle avoidance systems that can operate autonomously in various scenarios while ensuring safety and efficiency.

**Tools & Technologies Used**

There are various tools and technologies used for building an obstacle avoiding robot. Some of them are:

1. **Microcontroller:**

**Arduino Uno:** The Arduino Uno is the most popular and widely used board in the Arduino family. It is based on the ATmega328P microcontroller and has 14 digital input/output pins, 6 analog input pins, and a USB interface for programming and power.



Figure 1: An Arduino Uno Board

1. **Ultrasonic Sensor:**

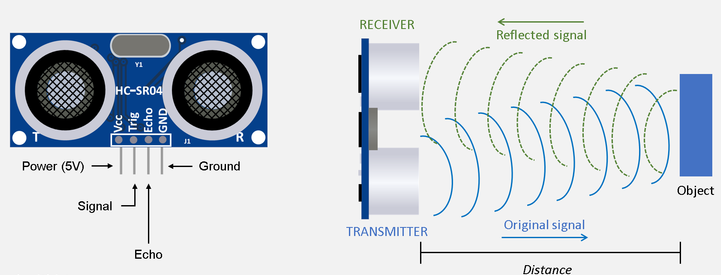
The Ultrasonic Sensor measures distances and detects objects using high-frequency sound waves. It's non-contact and ideal for delicate or hazardous environments. It's widely used in robotics and automation for accurate distance measurement.

Figure 2: An Ultrasonic Sensor

1. **Motor controllers:**

The Motor Driver Shield is a type of expansion board used to control and drive motors in electronics projects. It contains H-bridge circuits and protection circuits for safe and reliable motor control. It simplifies the connection and control of motors and is a popular choice for hobbyists and professionals building robotics and automation systems.

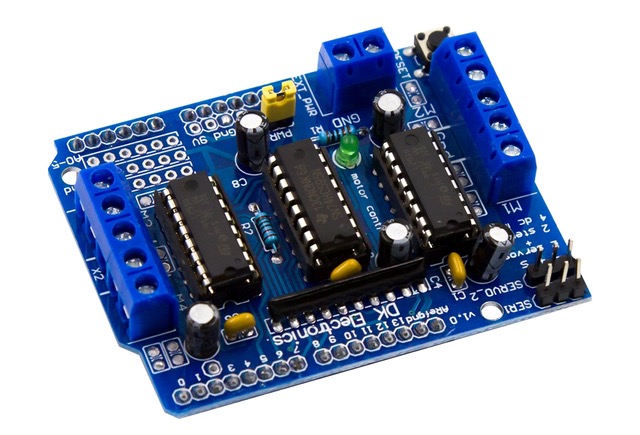


Figure 3: A Motor Driver Shied

1. **Batteries:**

Rechargeable batteries like Li-ion or LiPo are used for powering the robot.



Figure 4: A Li-ion Battery Cell

1. **Mechanical Components:**

Mechanical components like wheels, gear motors and servo motor are used for building the robot's physical structure.

A Gear Motor is a motor with a gear reduction unit for increased torque. A Servo Motor can be precisely controlled to a specific position. Both are used in robotics and automation for specific needs, with Gear Motors providing high torque and Servo Motors providing precise control.



Figure 5: Wheels, Servo Motor and Gear Motors

1. **Programming Languages:**

Programming language like C++ is commonly used for programming the robot's microcontroller.

1. **Simulation software:**

Simulation software like Arduino Integrated Development Environment (IDE), is used for simulating the robot's behavior and testing the control algorithms. The Arduino UNO can be programmed using the Arduino Integrated Development Environment (IDE), which is a software application for writing and uploading code to the board.

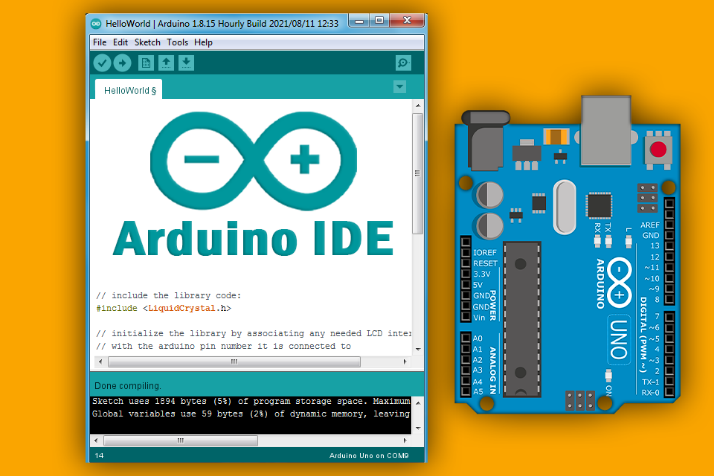


Figure 6: An Arduino IDE Interface

1. **Base Structure:** A metallic plate is used as a base structure for components assembling on in.

By utilizing these tools and technologies, engineers can design, build, and test an effective obstacle avoidance robot that can navigate autonomously in various scenarios while ensuring safety and efficiency.

**History & Features of the Technology Used**

1. **Arduino Boards**:
   1. **Arduino Mega:**

The Arduino Mega is a larger board than the Uno and is based on the ATmega2560 microcontroller. It has 54 digital input/output pins, 16 analog input pins, and a USB interface for programming and power.



Figure 7: An Arduino Mega Board

* 1. **Arduino Nano:** The Arduino Nano is a small and compact board based on the ATmega328 microcontroller. It has 14 digital input/output pins, 8 analog input pins, and a mini USB interface for programming and power.

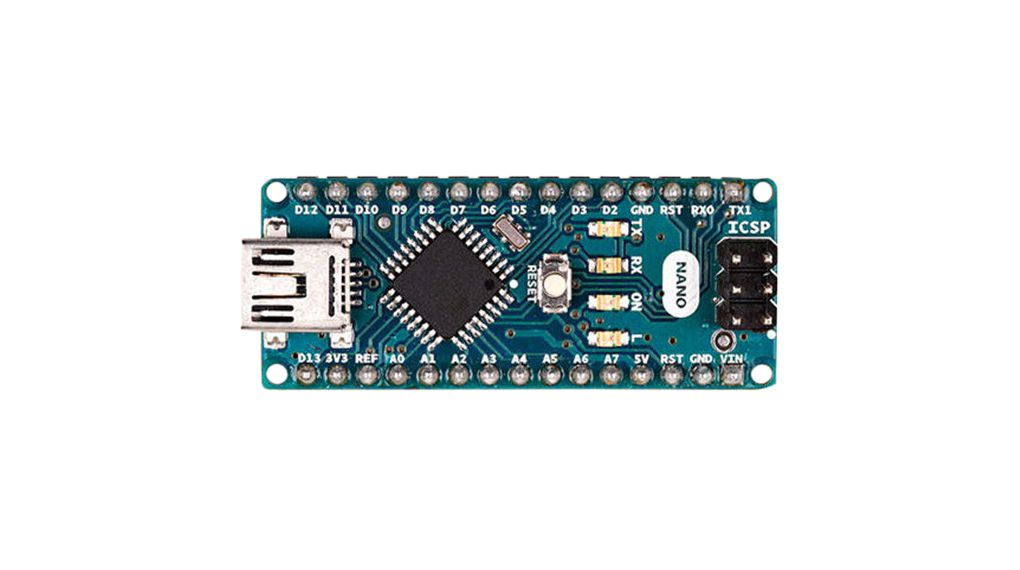


Figure 8: An Arduino Nano Board

* 1. **Arduino Due:** The Arduino Due is a powerful board based on the ARM Cortex-M3 microcontroller. It has 54 digital input/output pins, 12 analog input pins, and a USB interface for programming and power.



Figure 9: An Arduino Due Board

* 1. **Arduino Leonardo:** The Arduino Leonardo is a board based on the ATmega32u4 microcontroller. It has 20 digital input/output pins, 12 analog input pins, and a USB interface for programming and power. The Leonardo also has built-in USB HID capabilities, which allows it to emulate a keyboard, mouse, or other HID devices.



Figure 10: An Arduino Leonardo Board

* 1. **Arduino Pro Mini:** The Arduino Pro Mini is a small and compact board based on the ATmega328 microcontroller. It has 14 digital input/output pins, 8 analog input pins, and can be powered by an external power source.

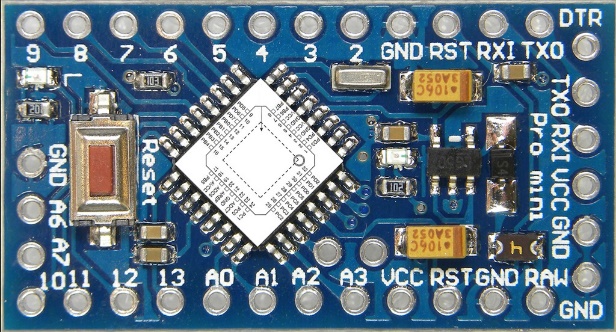


Figure 11: An Arduino Pro Mini Board

* 1. **Arduino MKR:** The Arduino MKR series is a family of boards that are designed for IoT (Internet of Things) applications. They are based on various microcontrollers, including the SAMD21, SAMD51, and ESP32, and have built-in connectivity options such as WiFi, Bluetooth, and LoRa.

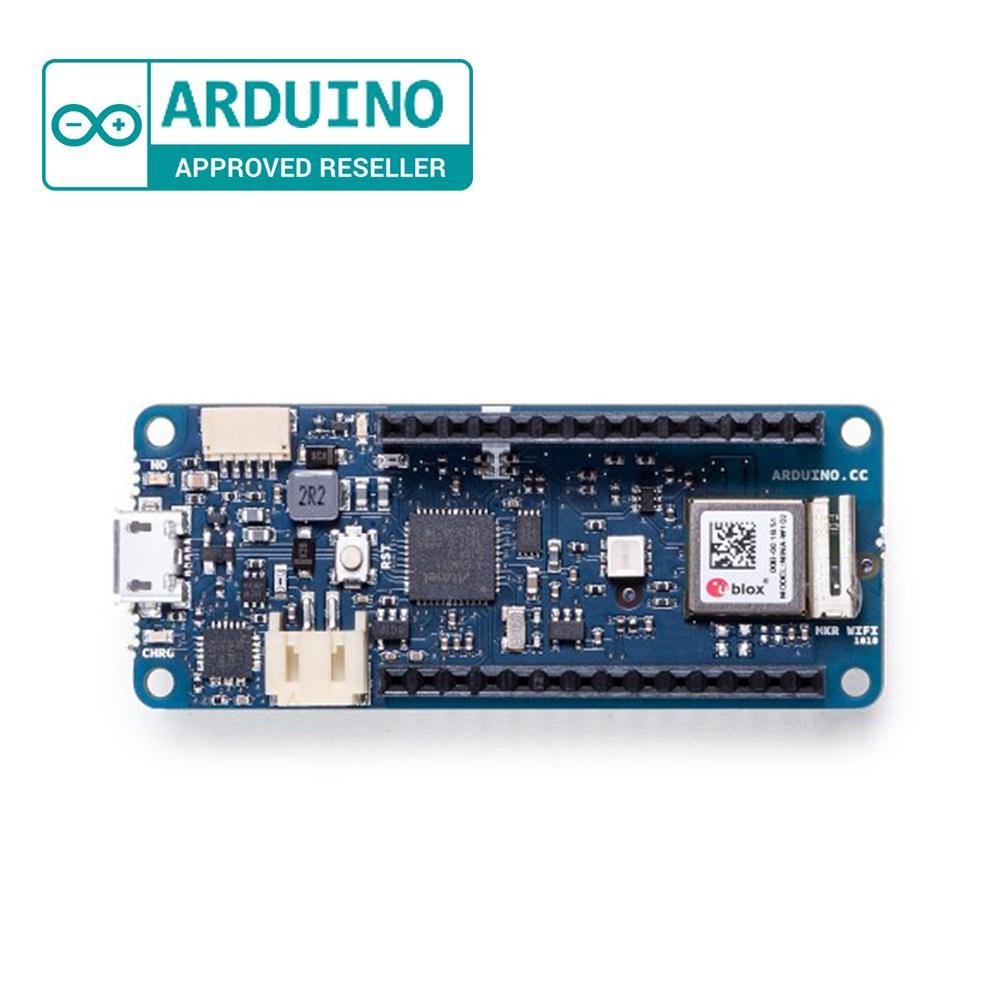


Figure 12: An Arduino MKR Board

These are just some of the popular types of Arduino boards available in the market. Each board has its own unique features and capabilities, making it suitable for different types of projects and applications.

1. **Ultrasonic Sensor:**

Ultrasonic sensors are a popular technology used in obstacle avoidance robots and have the following features:

* 1. **Detection Range:** Ultrasonic sensors can detect objects at a range of up to several meters, depending on the sensor's specifications. This makes them suitable for use in robots that need to operate in large environments.
  2. **Accuracy:** Ultrasonic sensors can detect obstacles with high accuracy, typically within a few centimeters. This allows robots to avoid obstacles and navigate around them without colliding. Non-contact: Ultrasonic sensors do not require physical contact with the object being detected, making them suitable for use in a wide range of applications.
  3. **Low Cost:** Ultrasonic sensors are relatively inexpensive compared to other sensor technologies such as laser sensors, making them a popular choice for hobbyist robots and low-cost commercial applications.
  4. **Wide Angle Detection:** Ultrasonic sensors typically have a wide detection angle, which allows them to detect obstacles in a large area. This is particularly useful in environments with multiple obstacles.
  5. **Reliability:** Ultrasonic sensors are reliable and have a long service life, making them suitable for use in industrial and commercial applications where high reliability is required.
  6. **Easy to Use:** Ultrasonic sensors are easy to use and can be integrated into robots with minimal effort. They typically require only a few connections to the microcontroller and can be programmed using a wide range of programming languages.

1. **Motor Driver Shield**:

A motor driver shield is an electronic module used to control the speed and direction of a motor. Some of its key features include:

* 1. **Easy to use**: Motor driver shields are designed to be easy to use, with simple connections to the microcontroller and motor. This makes them a popular choice for hobbyist and DIY projects. Compatibility: Motor driver shields are compatible with a wide range of microcontrollers, including Arduino and Raspberry Pi.
  2. **Dual H-bridge**: Most motor driver shields feature a dual H-bridge design, which allows the motor to be controlled in both directions. This is useful for applications such as robotics and motorized vehicles.
  3. **Protection:** Motor driver shields often include built-in protection features such as overcurrent protection and thermal shutdown to prevent damage to the motor and motor driver shield.
  4. **Compact design:** Motor driver shields are typically compact and can be easily mounted onto the microcontroller board or other circuit boards, making them suitable for use in small and portable projects.

1. **Batteries:**

Compared with traditional battery technology, lithium-ion batteries charge faster, last longer and have a higher power density for more battery life in a lighter package. When you know a little about how they work, they can work that much better for you.

1. **Motors:**

Gear motors and servo motors are two types of motors commonly used in robotics and automation applications. Here are some of their key features:

**Gear motors:**

* 1. **High torque:** Gear motors have high torque output, which makes them suitable for applications that require a lot of power such as driving wheels and lifting heavy loads.
  2. **Precise speed control:** Gear motors can provide precise speed control and are often used in applications where the speed needs to be regulated.
  3. **Compact size:** Gear motors are available in a range of sizes and are often designed to be compact, making them suitable for use in small and portable devices.
  4. **Low cost:** Gear motors are generally lower in cost compared to other motor types, making them a popular choice for hobbyist and DIY projects.

**Servo motors:**

* 1. **High precision:** Servo motors are known for their high precision, making them suitable for applications that require precise positioning or movement control.
  2. **High speed:** Servo motors can operate at high speeds, making them useful for applications that require fast and accurate movement.
  3. **Closed-loop control:** Servo motors have a closed-loop control system that allows for accurate and repeatable motion control.
  4. **High torque:** Servo motors can provide high torque output, making them suitable for applications that require a lot of power.
  5. **Compact size:** Servo motors are often designed to be compact, making them suitable for use in small and portable devices.

1. **C++ programming languages:**

C++ is a general-purpose programming language that was developed as an enhancement of the C language to include an object-oriented paradigm. It is an imperative and compiled language. C++ has a number of features, including: Object-Oriented Programming, Machine Independent, Simple, High-Level Language, Popular, Case-sensitive, Compiler Based, Dynamic Memory Allocation Memory Management.

1. **Arduino IDE 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 hardware to upload programs and communicate with them.

**Work Done**

The following is a general procedure for the work that needs to be done on an obstacle-avoiding robot:

1. **Design the robot:**

This involves determining the size and shape of the robot, selecting the appropriate sensors, motors, and other components, and creating a circuit diagram and mechanical design.

1. **Assemble the robot:**

Once the design is complete, assemble the robot by connecting the components and wiring them according to the circuit diagram.

1. **Write the code:**

Write the code that will allow the robot to detect obstacles and avoid them. This may involve using sensors such as ultrasonic sensors or infrared sensors to detect obstacles and programming the robot to move around them.

1. **Test the robot:**

Test the robot to ensure that it is functioning correctly. Test the obstacle avoidance feature by placing obstacles in front of the robot and verifying that it can successfully detect and avoid them.

1. **Fine-tune the robot:**

Once the robot has been tested, fine-tune the code and make any necessary adjustments to the robot's design to optimize its performance.

1. **Deploy the robot:**

Once the robot has been fully tested and optimized, deploy it for use in the desired application, such as in a manufacturing facility, a warehouse, or a research laboratory.

Throughout the process, it is important to document the design, code, and testing procedures to ensure that the robot can be replicated or modified in the future if necessary. Additionally, it is important to follow best practices for safety, such as using proper wiring techniques and ensuring that the robot's movements are not a hazard to people or equipment in the surrounding area.

**Circuit Diagram:**

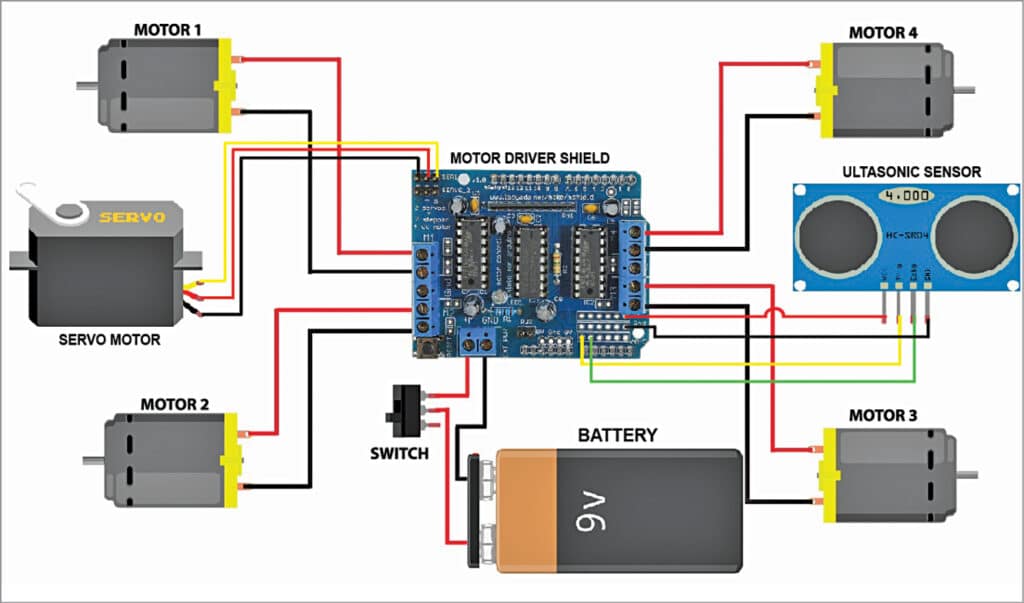


Figure 13: Circuit diagram of motor shield

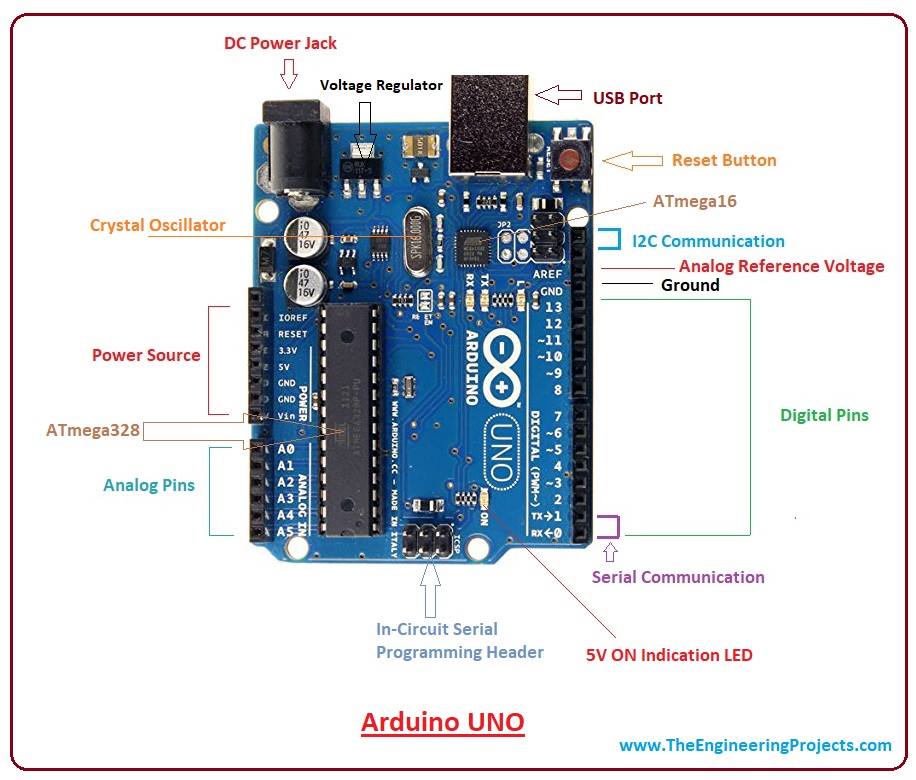
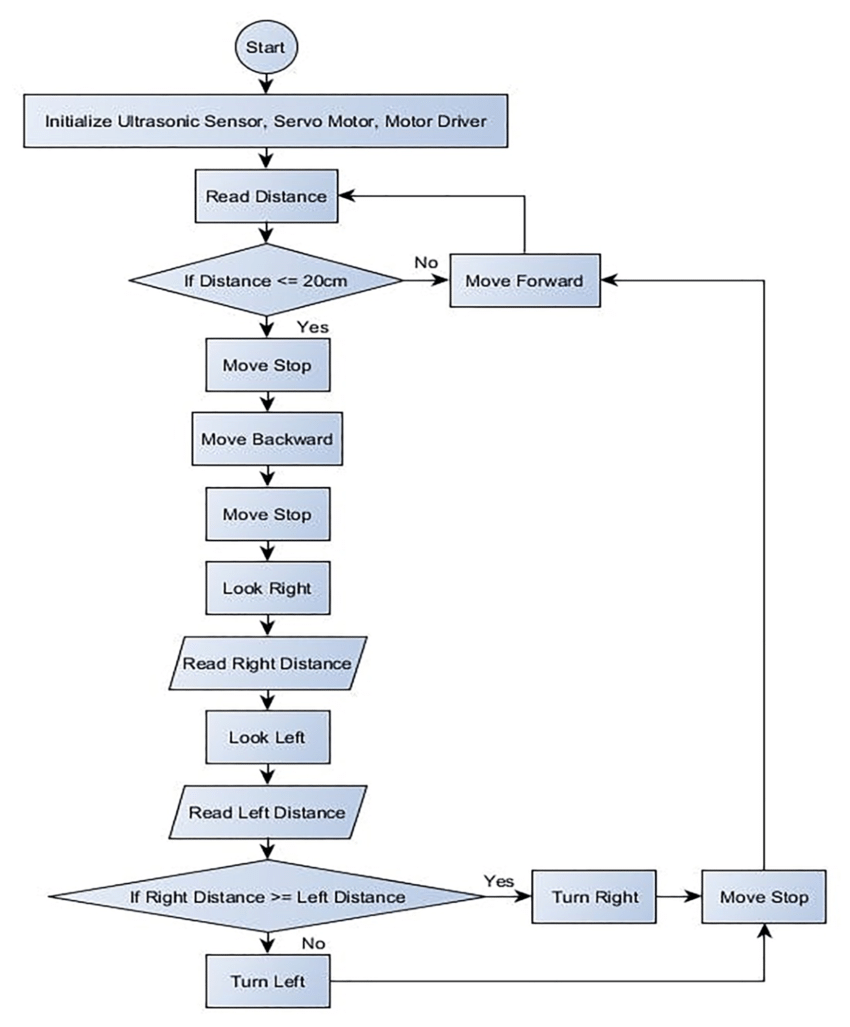


Figure 14: Circuit diagram of Arduino Uno

**Flowchart:**



**Program:**

#include <AFMotor.h>

#include <NewPing.h>

#include <Servo.h>

#define TRIG\_PIN A0

#define ECHO\_PIN A1

#define MAX\_DISTANCE 200

#define MAX\_SPEED 190 // sets speed of DC motors

#define MAX\_SPEED\_OFFSET 20

NewPing sonar(TRIG\_PIN, ECHO\_PIN, MAX\_DISTANCE);

AF\_DCMotor motor1(1, MOTOR12\_1KHZ);

AF\_DCMotor motor2(2, MOTOR12\_1KHZ);

AF\_DCMotor motor3(3, MOTOR34\_1KHZ);

AF\_DCMotor motor4(4, MOTOR34\_1KHZ);

Servo myservo;

boolean goesForward=false;

int distance = 100;

int speedSet = 0;

void setup() {

myservo.attach(10);

myservo.write(115);

delay(2000);

distance = readPing();

delay(100);

distance = readPing();

delay(100);

distance = readPing();

delay(100);

distance = readPing();

delay(100);

}

void loop() {

int distanceR = 0;

int distanceL = 0;

delay(40);

if(distance<=15)

{

moveStop();

delay(100);

moveBackward();

delay(300);

moveStop();

delay(200);

distanceR = lookRight();

delay(200);

distanceL = lookLeft();

delay(200);

if(distanceR>=distanceL)

{

turnRight();

moveStop();

}else

{

turnLeft();

moveStop();

}

}else

{

moveForward();

}

distance = readPing();

}

int lookRight()

{

myservo.write(50);

delay(500);

int distance = readPing();

delay(100);

myservo.write(115);

return distance;

}

int lookLeft()

{

myservo.write(170);

delay(500);

int distance = readPing();

delay(100);

myservo.write(115);

return distance;

delay(100);

}

int readPing() {

delay(70);

int cm = sonar.ping\_cm();

if(cm==0)

{

cm = 250;

}

return cm;

}

void moveStop() {

motor1.run(RELEASE);

motor2.run(RELEASE);

motor3.run(RELEASE);

motor4.run(RELEASE);

}

void moveForward() {

if(!goesForward)

{

goesForward=true;

motor1.run(BACKWARD);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(FORWARD);

for (speedSet = 0; speedSet < MAX\_SPEED; speedSet +=2) // slowly bring the speed up to avoid loading down the batteries too quickly

{

motor1.setSpeed(speedSet);

motor2.setSpeed(speedSet);

motor3.setSpeed(speedSet);

motor4.setSpeed(speedSet);

delay(5);

}

}

}

void moveBackward() {

goesForward=false;

motor1.run(FORWARD);

motor2.run(FORWARD);

motor3.run(FORWARD);

motor4.run(BACKWARD);

for (speedSet = 0; speedSet < MAX\_SPEED; speedSet +=2) // slowly bring the speed up to avoid loading down the batteries too quickly

{

motor1.setSpeed(speedSet);

motor2.setSpeed(speedSet);

motor3.setSpeed(speedSet);

motor4.setSpeed(speedSet);

delay(5);

}

}

void turnRight() {

motor1.run(BACKWARD);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(BACKWARD);

delay(500);

motor1.run(BACKWARD);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(FORWARD);

}

void turnLeft() {

motor1.run(FORWARD);

motor2.run(FORWARD);

motor3.run(BACKWARD);

motor4.run(FORWARD);

delay(500);

motor1.run(BACKWARD);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(FORWARD);

}

**CONCLUSION**

In conclusion, an obstacle-avoiding robot is a versatile and useful robot that can navigate through an environment while avoiding obstacles in its path. It is a practical application of robotics technology that has many potential uses in various fields, including manufacturing, logistics, and research.

The robot's ability to sense and avoid obstacles using sensors such as ultrasonic or infrared sensors makes it an efficient and safe solution for performing tasks that involve movement in dynamic environments. It can navigate through narrow spaces, avoid collisions with other objects, and move smoothly around obstacles.

The development of an obstacle-avoiding robot involves designing the robot, assembling it, writing code to control its movements and sensors, testing and fine-tuning the robot, and deploying it for use in the desired application. By following best practices for safety and documentation, the robot can be replicated or modified in the future if necessary.

Overall, obstacle-avoiding robots are an exciting and promising area of robotics technology that will likely continue to see rapid development and application in a wide range of industries and fields.