

Egyptian Russian University
Faculty Of Engineering
Mechatronics Department



Group Project

Obstacle Avoidance on inclined plane challenge

Design Of Mechatronic Systems – ME403

Supervised by:

Dr Tamer Mansour

No.	Name	ID
1	Omar Khaled	171033
2	Mostafa Mohamed	171154
3	Omar Alaa	171385
4	Omar Adel	171112
5	Abdelrahman Kotb	171009
6	Ramez Mohamed	171113
7	Hadeer Mahmoud	171433
8	Marwan Wael	171098

Table of Contents

Introduction	3
Components.....	5
Assembling.....	7
Schematic Diagram	9
Code	10
V-model.....	14
Conclusion.....	17

Introduction

In our world today, robotics is a very interesting research area, which is fast growing as it is the simplest way for modifying modern day technology. Robotics plays a major role in technology advancement, which is why I decided to work on the robotics field and design something intelligent to make human life simpler.

An autonomous robot is one which can move without any external interference in an environment which is unstructured and unknown to the robot. The robot is able to do this because of the software intelligence embedded inside it in order for it to be able to sense the environment, detect any obstacle which is in its path and move round the environment by avoiding these obstacles [1]. In the designing of an autonomous robot, there are many robotic designs that can be used. To make a selection of the design to be used, the main factor to be put into consideration is the physical environment in which the robot will be operated. Examples of autonomous robots: walking robots, drones, robotic cars, and snake robots.

The obstacle avoiding robot has enough intelligence in order for it to cover the maximum area of the space provided and it has an ultrasound sensor which is used to detect any obstacles in the path of the robot, after which it will move in a direction to avoid the obstacle.

The main aim of such technology is that it can play a huge role in today's transportation as it can be used to avoid accidents, which generally happen on congested roads by applying emergency brake. If this technology is used in a car, it will automatically sense any objects (living things or objects) in the path of the car and automatically apply breaks or take a side to the available free space where necessary.

A necessary requirement of every autonomous mobile robot is obstacle avoidance. This obstacle avoidance feature is of high importance in a robot's navigation system in an unknown area so as to prevent collisions during its operation. It is necessary for an autonomous robot to avoid collisions in order to prevent damage to the object or to the robot itself. Application areas where obstacle avoidance is necessary include automatic vacuum cleaners and helicopters. Even in robots which work in a familiar environment and the path of the robot has been adequately defined, some environmental changes could occur and cause the robot to run into an object in its way so it is necessary for the robot to be able to adapt to the change by avoiding any objects in its path. This problem of effective trajectory planning is what has led to the need for a robot that can detect and avoid objects in a pre-computed path, or objects that appear suddenly.




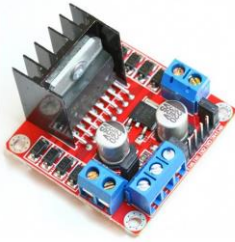



The solution to this trajectory problem involves the use of sensors by the robot to detect objects and avoid them thereby making the robot to be more independent since it would not require external influence.




The Aim of this project is to design and implement a robot car that is able to move round an unknown environment without running into obstacles in its path.

The Objectives of the project are as follows:

- A robot is capable of traversing across an inclined non-smooth surface obstacle course
- The robot must be autonomous
- The weight of the robot should not exceed 1.5 kg (including batteries)
- Only one controller is allowed in the robot design
- Any valid software

Components

No.	Name	No. of pieces	Image
1	Ultrasonic sensor	1	
2	Ultrasonic sensor mounting bracket	1	
3	NodeMCU Based ESP8266	1	
4	Dual H-bridge motor driver using L298N	1	
5	SG90 Servo Motor	1	
6	DC motor	2	
7	Jumper wires	25	

8	1.5v DC battery	4	
9	2WD Robot Car Chassis Kit	1	
10	Bread board mini	1	

All equipment and components were bought from:

(<https://ram-e-shop.com/>)

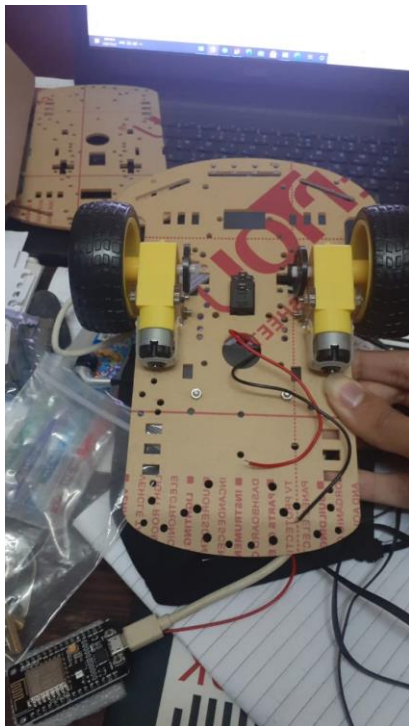
All Details in the link above

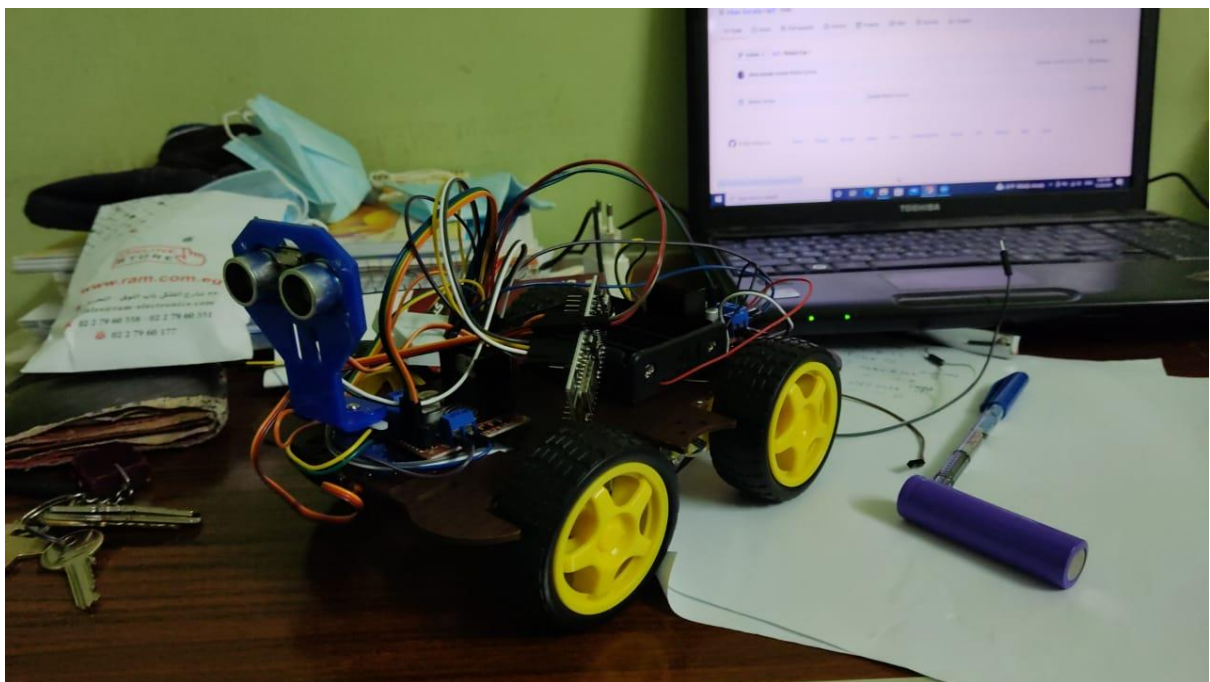
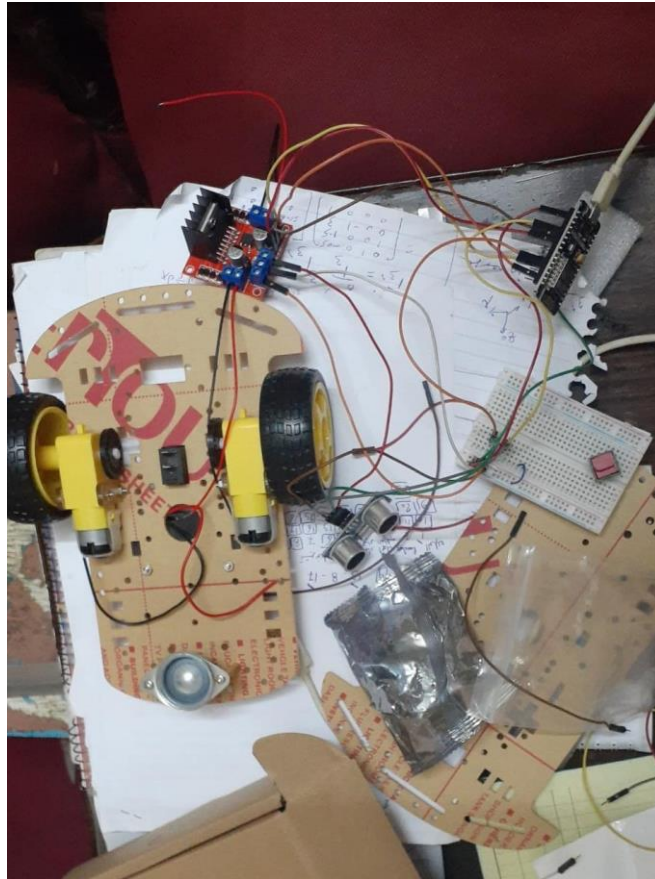
Assembling

Assembling this project is pretty straight forward. First fit the DC motors to chassis using screws. Then attach the wheels to motors and attach the third single wheel. Then attach the battery mount at top.

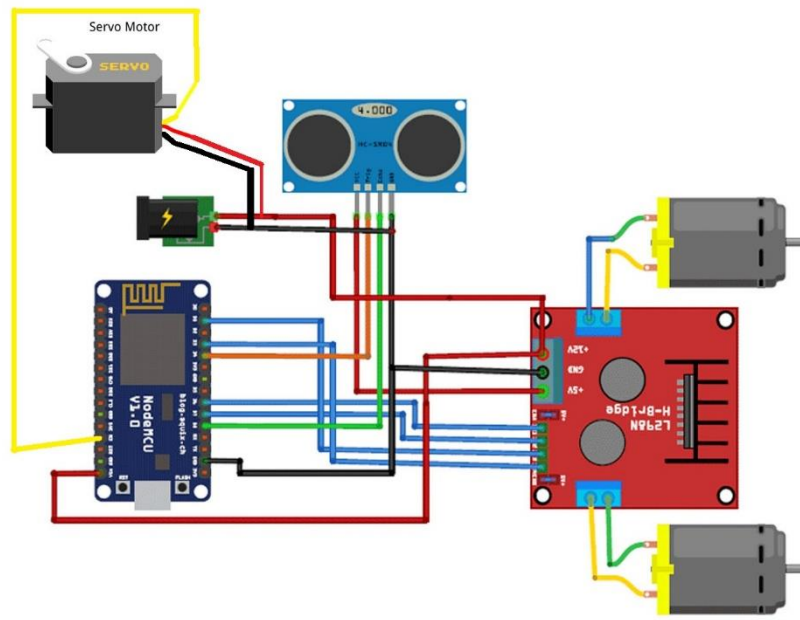


Also, mount ESP8266 on H-bridge then it's very easy to make connections.





Schematic Diagram



Code

To summarize, check distance in front of robot using range sensor and change its direction if obstacle is found in front. Before changing direction again scan on left and right using range sensor and then turn robot accordingly.

One tip I would like to share is how to debug issues. Put robot car on box or something to make it stationary but still wheels are free to rotate. The code has serial prints to show useful messages which can be seen while connected to computer. By checking the log of printed messages, you can see what is going wrong like distance not detected correctly or motor not spinning in the right direction.

```
// #include <AFMotor.h>
#include <Servo.h>

#define enA 0
#define in1 14
#define in2 2
#define enB 5
#define in3 16
#define in4 4
#define servoPin 12

Servo servoLook;

int trig = 13;
int echo = 15;

int distance ;
int maxDist = 150;
int stopDist = 50;

int motorSpeed = 110;
int motorSpeedL= 110;

int turnSpeed = 110;

void setup()
{
    pinMode(enA, OUTPUT);
    //Import library to control motor shield
    //Import library to control the servo

    //Create an object to control the servo

    //Assign the ultrasonic sensor pins

    //Maximum sensing distance (Objects further than this distance are ignored)
    //Minimum distance from an object to stop in cm

    //The maximum motor speed

    //Amount to add to motor speed when turning
```

```

void setup()
{
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);

    servoLook.attach(servoPin);
    pinMode(trig,OUTPUT);
    pinMode(echo,INPUT);

    Serial.begin(115200);
}

void loop()
{
    servoLook.write(90);
    delay(750);
    distance = getDistance();
    if(distance >= stopDist)
    {
        moveForward();
        Serial.println("moveForward()");
    }
    while(distance >= stopDist)
    {
        distance = getDistance();
        delay(250);
    }
    stopMove();
    Serial.println("stopMove()");
    int turnDir = checkDirection();
    Serial.println(turnDir);
    switch (turnDir)
    {
        case 0:
            turnLeft (950);
            Serial.println("turnLeft (850)");
            break;
        case 1:
            turnLeft (1650);
            Serial.println("return (1550)");
            break;
        case 2:
            turnRight (800);
            Serial.println("turnRight (700)");
            break;
    }
}

void moveForward()
{
    analogWrite(enA,motorSpeed);
    analogWrite(enB,motorSpeedL);

    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);

    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
}

void stopMove()

```

```

void stopMove()                                     //Set all motors to stop
{
    analogWrite(enA,0);
    analogWrite(enB,0);

    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);

    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}

void turnLeft(int duration)                         //Set motors to turn left for the specified duration then stop
{
    analogWrite(enA,turnSpeed);
    analogWrite(enB,turnSpeed);

    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);

    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);

    delay(duration);

    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);

    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}

void turnRight(int duration)                       //Set motors to turn right for the specified duration then stop
{
    analogWrite(enA,turnSpeed);
    analogWrite(enB,turnSpeed);

    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);

    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);

    delay(duration);
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);

    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}

int getDistance()                                  //Measure the distance to an object
{
    unsigned long pulseTime;                       //Create a variable to store the pulse travel time
    int distance;                                   //Create a variable to store the calculated distance
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);                      //Generate a 10 microsecond pulse
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
    pulseTime = pulseIn(echo, HIGH);
    distance = (float)pulseTime * 340 / 2 / 10000;  //Calculate the object distance based on the pulse time
    Serial.print("Distance: ");                   //For Debuging
    Serial.print(distance);
    Serial.println(" cm");
    return distance;
}

```

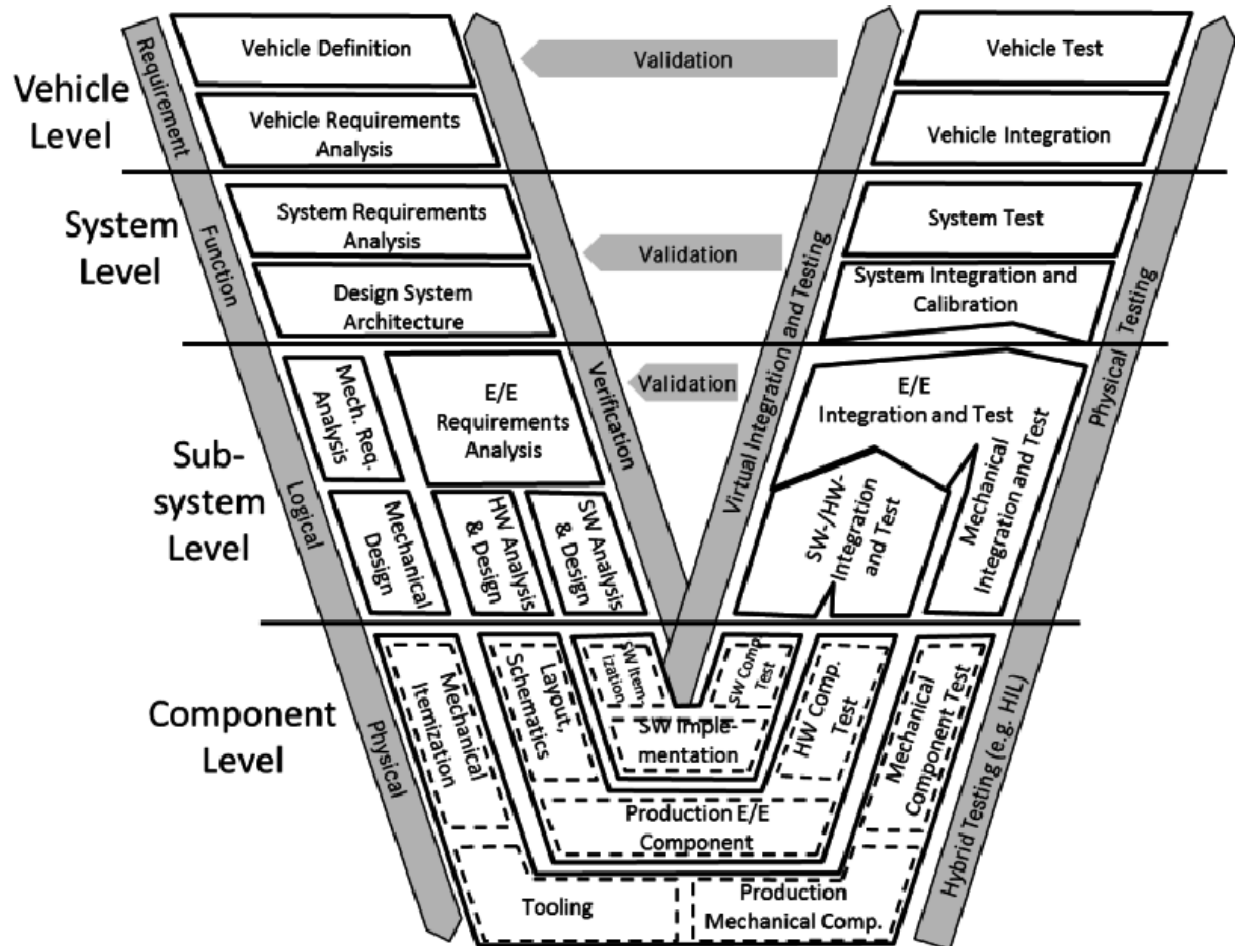
```

}

int checkDirection()                                     //Check the left and right directions and decide which way to turn
{
    int distances [2] = {0,0};                          //Left and right distances
    int turnDir = 1;                                     //Direction to turn, 0 left, 1 reverse, 2 right
    servoLook.write(180);                                //Turn servo to look left
    delay(500);
    distances [0] = getDistance();                       //Get the left object distance
    servoLook.write(0);                                  //Turn servo to look right
    delay(1000);
    distances [1] = getDistance();                       //Get the right object distance
    if (distances[0]>=200 && distances[1]>=200)           //If both directions are clear, turn left
        turnDir = 0;
    else if (distances[0]<=stopDist && distances[1]<=stopDist) //If both directions are blocked, turn around
        turnDir = 1;
    else if (distances[0]>=distances[1])                //If left has more space, turn left
        turnDir = 0;
    else if (distances[0]<distances[1])                  //If right has more space, turn right
        turnDir = 2;
    return turnDir;
}

```

V-model



1) Functional Requirement:

Make an autonomous robot that avoid obstacles on an incline plane.

2) System to Subsystem:

This system consists of 3 Subsystems

A) Mechanical System

B) Electrical System

C) Information Technology (IT)

3) Subsystem to Elements

A - Mechanical System:

The body is made from plastic as it's lighter than wood and has a better appearance overall.

Wheels are 6cms in diameter each to be able to go upwards on the incline plane.

Steering Wheel: a wheel in the first half of the robot to give it equilibrium and direction.

B- Electrical System:

DC Motors: can work at 90 rpm, no load, operating voltage of 5 volt and 12UDC with maximum torque of 800 gf. cm. Gear ratio 48: 1.

Ultra-Sonic Sensor (HC-SR04): a sensor that has waves transmitter & receiver. The distance to be measured mainly depends on the speed of Ultra-Sonic waves in Space or air which is constant.

Servomotor: to rotate the Ultrasonic and give it wide range of vision.

Controller: Node MCU Based (ESP 8266) it's easy to code with python & C++, it's cheap and available

C- IT (Information Technology): Software.

Arduino IDE: used to make a code and burn it into the micro-Controller (Node MCU)

Proteus: is used to check the electric wiring.

Programming Language: C++ Language (worldwide language easy to code with it).

4) Elements to Subsystem: testing each element individually DC Motors: Connect each motor with DC batteries & check if they work in both directions.

Servo motor: the same as the DC Motor, check if it's working finely, by burning a mini code to test if it's working.

Controller: burn a code and check properly or not.

Ultrasonic Sensor: burn the ultrasonic sensor code to check if it receives & emits waves when an obstacle faces the robot.

5) Subsystem to System: Testing each Subsystem individually.

Mechanical System:

Assemble the wheels to the chassis of the robot and also fix the marble wheel.

Electrical System:

Connect DC Motors, servomotor and ultrasonic to run a small code to check if they work with synchronization.

IT (Information Technology):

Run the code in Arduino IDE & simulate the circuit diagram in proteus.

6) Feedback Loop:

After assembling all subsystems together and checking if there is an error go back and repair or change the source of error.

7) Realized function:

The function of the system applied and achieved maintenance.

Conclusion

We finally ready to run the code and see the robot avoiding obstacles and walls.

