



Ned University Of Engineering And Technology



COMPUTER ARCHITECTUR And
ORGANISATION
CT-252

**HURDLE
DETECTION
CAR**

PROJECT MEMBERS

CT-55 Shaheer Akram

CT-48 Asad Ali

CT-64 Abdul Qadir

CT-41 Ahmed Ali Noori

SUBMITTED TO: MISS SANA FATIMA

DATED: 25/06/19

ABSTRACT:

A Hurdle Detection Car is an application of self-automated robotics. This type of pre-programmed robot can be used in multiple situations and serves as a small prototype and learning tool for self-driving cars like the current Tesla. Arduino UNO is a brilliant and easy to use tool for this purpose. It is affordable and user-friendly using a custom library of C++ to code and simple and clean circuitry. Ultrasonic sensors are used to detect obstacles and ditches by placing one directly in its front and the other on top with a servo-motor scanning the surroundings. The code is designed to calculate the distance and then decide its path and move accordingly.

This project is the first step towards learning intermediary level robotics and did certainly help us as students to expand our minds and explore the logical world of a small level robotic implementation.

TABLE OF CONTENTS:

- ❖ Introduction
- ❖ Background
- ❖ Working Principle
- ❖ Hardware and Components
- ❖ Software's
- ❖ Coding Description
- ❖ Application
- ❖ Conclusion

INTRODUCTION:

Automation has always been a fascinating concept for humans. We are constantly finding ways to reduce human effort, be it industrialization, household chores or leisure. Robotics and AI has been the focus of various researchers for the past decade or two. There are many microcontrollers in the market consisting of various types of capability from basic input output to high end microcontroller. These various types of microcontroller are purpose-made for general application. The most basic implementation and practice of this field starts with hardware programming microcontrollers such as Arduino and Raspberry Pi. Arduino is a microcontroller board which functions as a tiny computer; it is a platform where creation and development of interacting objects is possible with required programming software. The Arduino software IDE (Integrated Development Environment) provides space to write codes in the language (programming languages C, C++) that Arduino board understands and responds to. Inexpensiveness, easy-to-use design and flexibility for advance modifications are some features of the microcontroller-based Arduino hardware and software that are making its range of use wider. One of the most important factors that affects its increasing range of use is its freedom of use. Both the Arduino hardware and the software are open source. Which means that one can easily use the ideas generated by others in their work and modify them without anyone's authorization. It can be used by anyone to do anything they want to do with it. Arduino boards are designed in such a way that one without prior knowledge of electronics or previous experience of programming can use information from other people's work and build their own interactive object that can sense the environment and control it.

BACKGROUND:

Ever since Isaac Asimov introduced the three laws of Robotics in 1941 the world quickly familiarized itself with it. Countless researchers spent years of their lives pursuing self-automated robots and perfecting them. The first Lunar Rover was sent to the Moon in 1971 and 1972 during the Apollo Space Program. The world for the first time, saw a robot controlled from such a far distant. Robots have been the frontrunners of technological advances for the better part of the past century. Rescue robots, Space Rovers, maze solving robots and even as simple as Roombas (room cleaning robots).

Our project is somewhere along the lines of self-automated cars. It is a "Hurdle Detector Car", a basic prototype for self-driving cars or automated robot helpers. 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. We used Arduino UNO as a microcontroller to program our robot to detect obstacles in its path and around itself using a rotating sensor. It also senses depth and any ditches that it may come across.

WORKING PRINCIPLE:

The hurdle detection robot car uses ultrasonic sensors for its movements. A microcontroller of 8051 family is used to achieve the desired operation. The motors are connected through motor driver IC to microcontroller. 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 microcontroller. The microcontroller 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).

HARDWARE AND COMPONENTS:

- ARDUINO

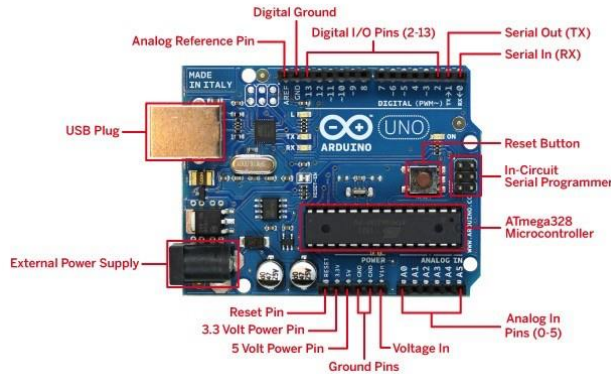


The heart part of the building monitoring system; the Arduino is defined in Wikipedia as

“An open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world”

In other words, it can also be defined as a single board microcontroller for building digital objects and interactive devices. Arduino is designed to sense the environment and/or surrounding by receiving input signal through sensors and communicates with its surrounding through actuators. An actuator could be a simple LED (light emitting diode), a motor or sensors, ethernet or some other electronics depending on the project. The Arduino hardware are available in many format and design enabling different features. The programming is based on hardware wiring. The Arduino software can be run on Windows, Linux or Mac OS (Sandhu, 2016). The Arduino can be programmed to work stand-alone, with computer or other electronic devices; which can be done with Arduino software which generally termed as IDE (Integrated Development Environment). Since the Arduino hardware and software is an open source, there are already many clones of Arduino hardware available with many exciting features, the board used in this thesis project is Arduino Uno which is shown in Figure 1. Arduino Uno board is a microcontroller board based on Atmel Atmega328 8-bit microprocessor. There are 14 digital input and output pins; 6 of

which can be used as pulse-width (PWM) outputs. It has 6 Analog inputs and a 16 MHz quartz crystal or oscillator. Arduino Uno board has USB (universal serial bus) cable to connect to a computer, a power jack, an ICSP (In Circuit Serial Programming) header and a reset button (Anon., 2017). In Italian “Uno” means one and the board was named so because the Arduino Uno board is the first in a series of USB Arduino boards.



- **ULTRASONIC SENSORS:**

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board..



- **DC MOTORS:**

A **servomotor** is a [rotary actuator](#) or [linear actuator](#) that allows for precise control of angular or linear position, velocity and acceleration.^[1] It consists of a suitable motor

coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.



- **SERVO MOTOR:**

A **DC motor** is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.



- **BATTERIES:**

Batteries provide the source of current for the robot.

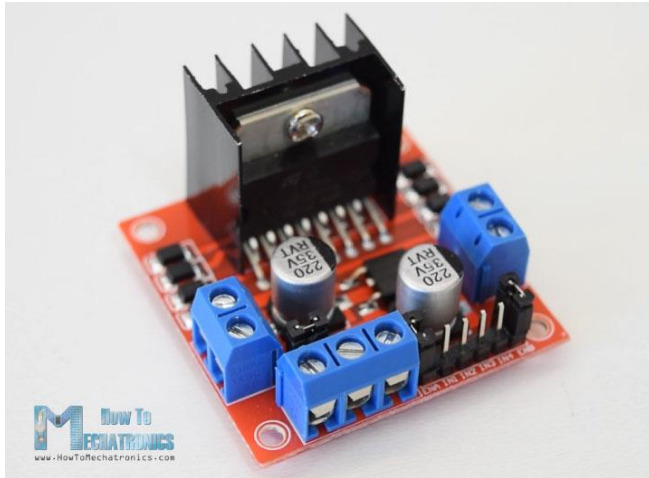
- **SWITCH:**

The switch turns the source current on and off when pressed for the motor.

- **MOTOR CONTROLLER:**

PWM, or pulse width modulation is a technique which allows us to adjust the average value of the voltage that's going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of

time the signal is ON versus the amount of time the signal is OFF in a single period of time.



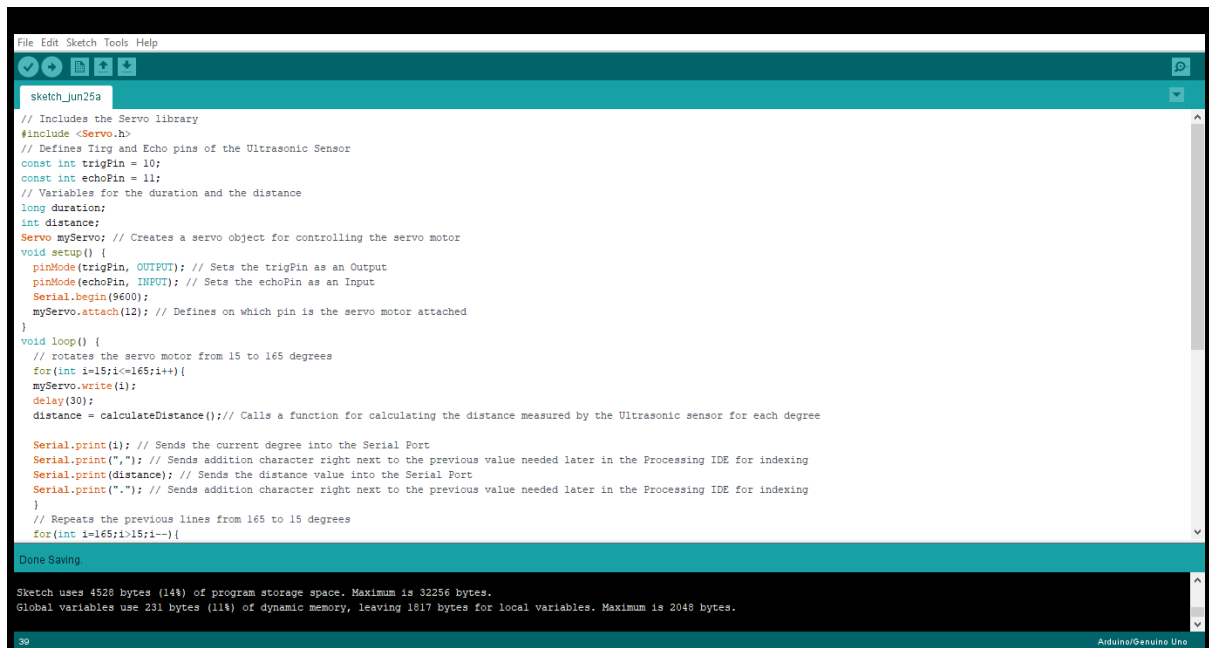
SOFTWARES:

SERVO MOTER LIBRARY:

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

CODING DESCRIPTION:



The screenshot shows the Arduino IDE interface with a sketch named 'sketch_jun25a'. The code includes the Servo library and defines pins for the ultrasonic sensor (trigPin = 10, echoPin = 11) and a servo motor (pin 12). The setup function configures the pins and attaches the servo. The loop function rotates the servo from 15 to 165 degrees, calculates the distance using the ultrasonic sensor, and prints the current degree and distance to the Serial Port. The code is as follows:

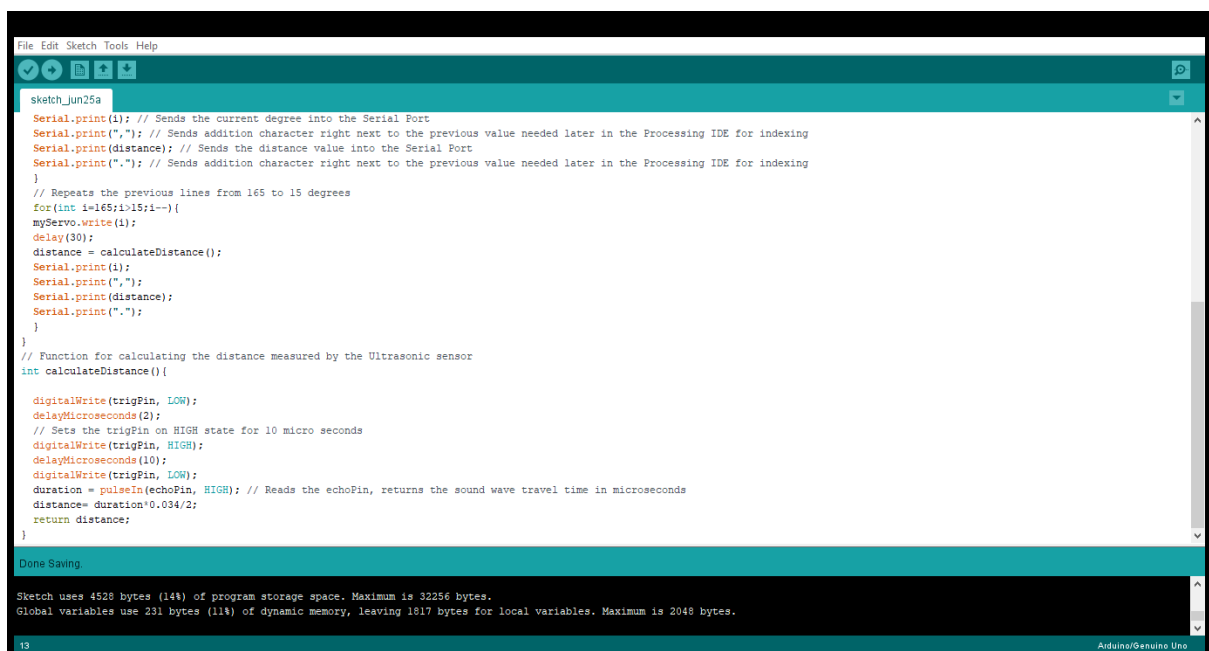
```
// Includes the Servo library
#include <Servo.h>
// Defines Trig and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
    myServo.write(i);
    delay(30);
    distance = calculateDistance(); // Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

    Serial.print(i); // Sends the current degree into the Serial Port
    Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
    Serial.print(distance); // Sends the distance value into the Serial Port
    Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
```

Done Saving.

Sketch uses 4528 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 231 bytes (11%) of dynamic memory, leaving 1817 bytes for local variables. Maximum is 2048 bytes.

99 Arduino/Genuino Uno



The screenshot shows the continuation of the sketch in the Arduino IDE. It includes the Serial.print statements for the distance and a loop that repeats the previous lines from 165 to 15 degrees. The calculateDistance function is defined, which uses digitalWrite to set the trigPin to LOW and HIGH, delays, and then reads the echoPin to calculate the distance. The code is as follows:

```
Serial.print(i); // Sends the current degree into the Serial Port
Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
Serial.print(distance); // Sends the distance value into the Serial Port
Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
}
// Repeats the previous lines from 165 to 15 degrees
for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(",");
}
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
  distance= duration*0.034/2;
  return distance;
}
```

Done Saving.

Sketch uses 4528 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 231 bytes (11%) of dynamic memory, leaving 1817 bytes for local variables. Maximum is 2048 bytes.

19 Arduino/Genuino Uno

```
File Edit Sketch Tools Help
arduino1 $
int motor1_left=4;
int motor1_left=5;
int motor2_right=6;
int motor2_right=7;
int trigpin=8;
int echopin=9;
void setup()
{
  pinMode(motor1_left,OUTPUT);
  pinMode(motor1_left,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(trigpin,OUTPUT);
  pinMode(echopin,INPUT);
  pinMode(motor2_right,OUTPUT);
  pinMode(motor2_right,OUTPUT);
  Serial.begin(9600);
}
long duration,distance;
void loop() {
  digitalWrite(trigpin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  duration = pulseIn(echopin, HIGH);
  distance = (duration*0.0343) /2;
  Serial.println(distance);
  if (distance<15)
  Done compiling.

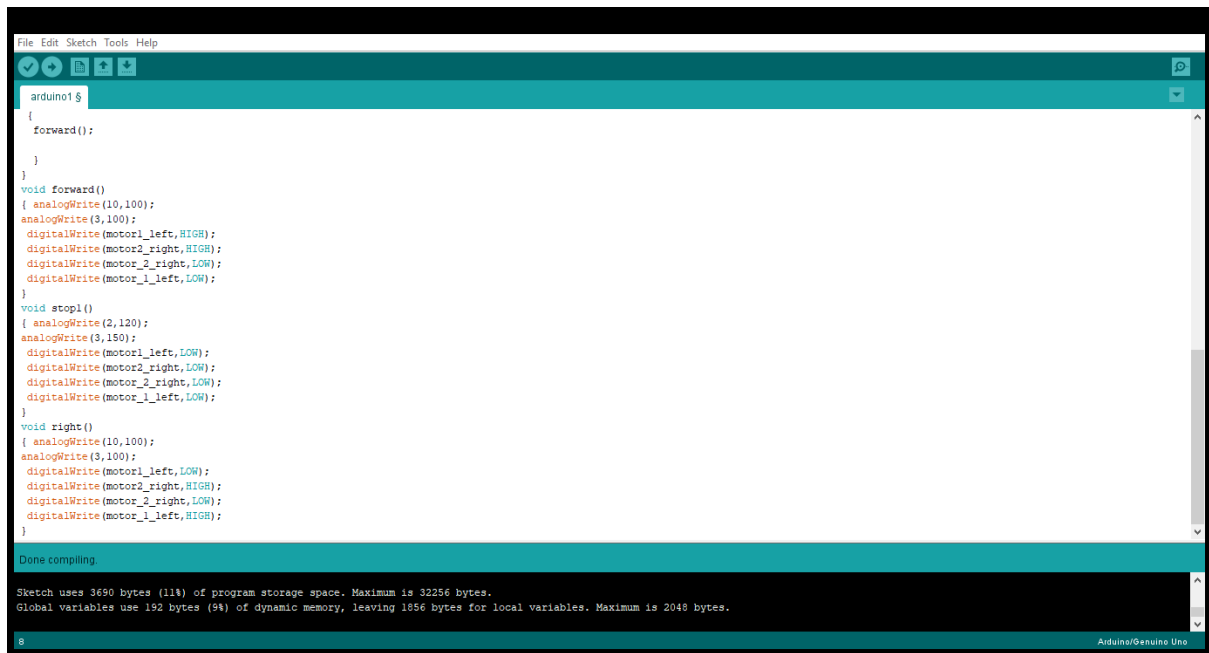
Sketch uses 3690 bytes (11% of program storage space. Maximum is 32256 bytes.
Global variables use 192 bytes (9% of dynamic memory, leaving 1856 bytes for local variables. Maximum is 2048 bytes.

00 Arduino/Genuino Uno
```

```
File Edit Sketch Tools Help
arduino1 $
if (distance<15)
{
  stop1();
  delay(2000);
  right();
  delay(500);
}
else if (distance>15)
{
  forward();
}
}
void forward()
{ analogWrite(10,100);
  analogWrite(3,100);
  digitalWrite(motor1_left,HIGH);
  digitalWrite(motor2_right,HIGH);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor1_left,LOW);
}
void stop1()
{ analogWrite(2,120);
  analogWrite(3,150);
  digitalWrite(motor1_left,LOW);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor1_left,LOW);
}
Done compiling.

Sketch uses 3690 bytes (11% of program storage space. Maximum is 32256 bytes.
Global variables use 192 bytes (9% of dynamic memory, leaving 1856 bytes for local variables. Maximum is 2048 bytes.

8 Arduino/Genuino Uno
```



```
File Edit Sketch Tools Help
arduino1 $
{
  forward();
}
}
void forward()
{ analogWrite(10,100);
  analogWrite(3,100);
  digitalWrite(motor1_left,HIGH);
  digitalWrite(motor2_right,HIGH);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor1_left,LOW);
}
void stop1()
{ analogWrite(2,120);
  analogWrite(3,150);
  digitalWrite(motor1_left,LOW);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor1_left,LOW);
}
void right()
{ analogWrite(10,100);
  analogWrite(3,100);
  digitalWrite(motor1_left,LOW);
  digitalWrite(motor2_right,HIGH);
  digitalWrite(motor2_right,LOW);
  digitalWrite(motor1_left,HIGH);
}
```

Done compiling.

Sketch uses 3690 bytes (11% of program storage space. Maximum is 32256 bytes.
Global variables use 192 bytes (9% of dynamic memory, leaving 1856 bytes for local variables. Maximum is 2048 bytes.

8 Arduino/Genuino Uno

APPLICATIONS:

ROOMBA:

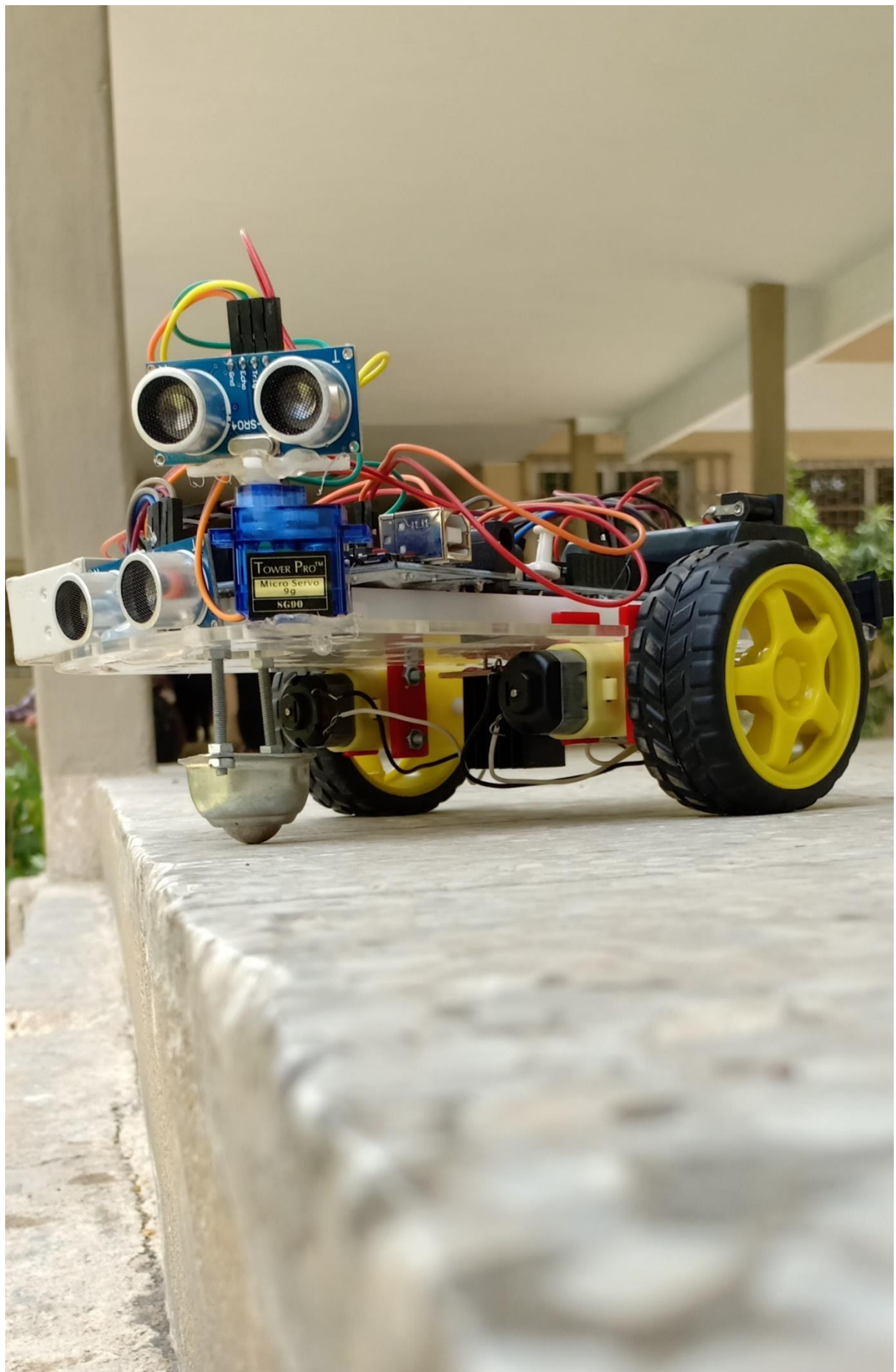
Roomba is a smart room cleaning robot , it can clean the whole house without any effort. A Roomba goes around the entire house and it collect what ever dust or waste that comes in its way and it never strikes a wall or any object it comes in its way, basically it is also uses an hurdle detection robot.

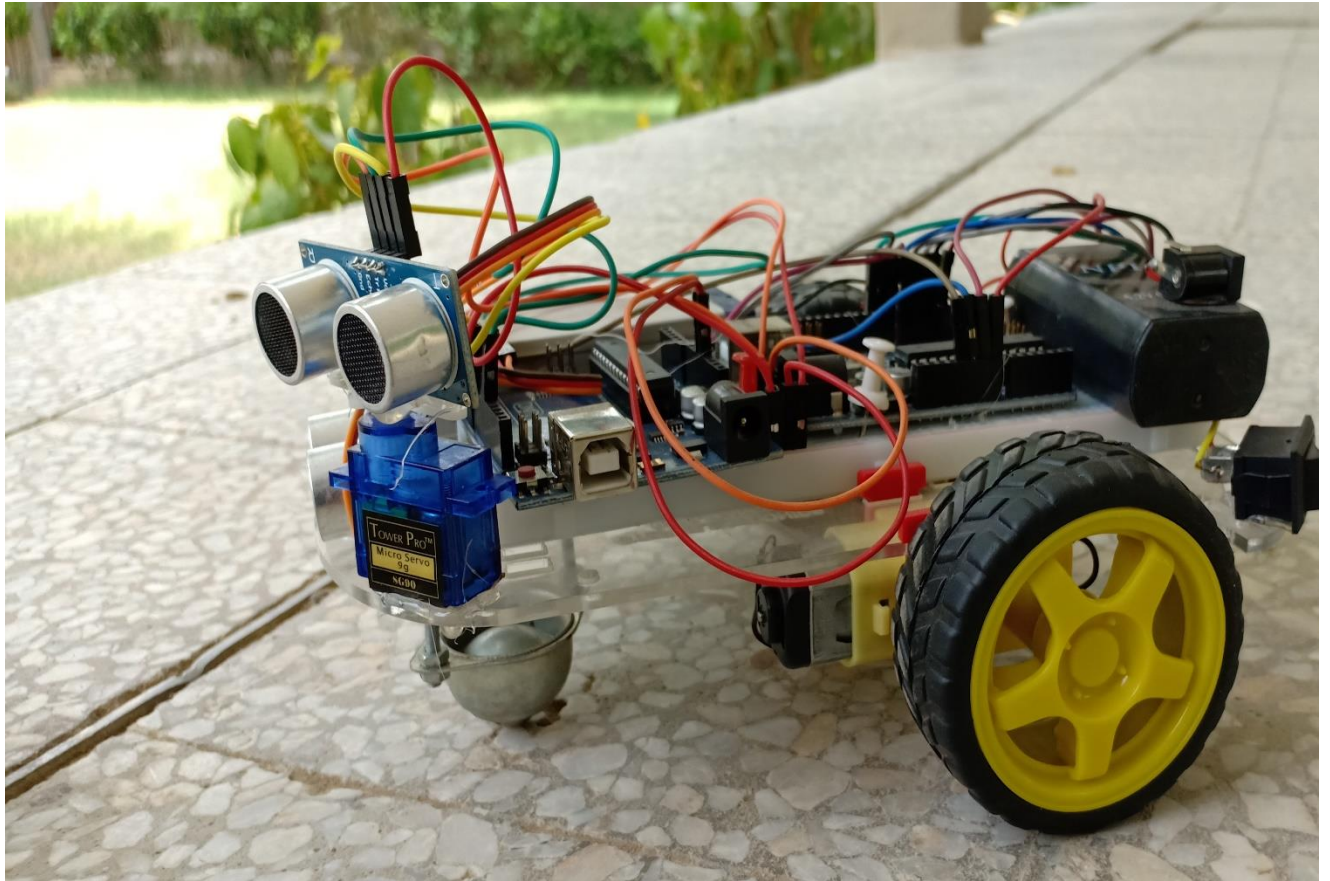


MAZE SOLVING:

We can also create a hurdle detection robot using this robot since it never strikes a wall so it will find its own way so this same robot can be used to solve a hacken maze.

maze follower finds a wall and starts following it until it finds an escape route. But unlike a line follower which has just to follow a predetermined route, a maze follower is designed to find an escape route that is not known beforehand. However, both types of robots are designed to be autonomous, they basically perform different tasks.





CONCLUSION:

The Hurdle Detection Car is a very effective way of prototyping and learning how to implement a full-scale self-driving car. As students it is an extremely helpful tool to know how to use microcontrollers to program different types of robots efficiently. We personally will be using this technology quite often in the future.