AUTONOMOUS BUGGY PROJECT

PROJECT REPORT

Engineering Design –III
(UTA011)
(Second Year)

SUBMITTED BY:

ABHISHEK BANSAL (101510006)
ADITYA KHANIJOW (10151009)
ANKUR SHARMA (101510016)
DEVANSHU NARULA (101510022)

Group: CML-1

SUPERVISOR:
Dr. Rana Pratap Yadav
Assistant Professor
Electronics & Communication Engineering
Thapar University, Patiala.

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT,
COMPUTER SCIENCE ENGINEERING DEPARTMENT,
THAPAR UNIVERSITY, PATIALA-147004, PUNJAB
INDIA

May 2017.

CONTENTS

INTRODUCTION

The objective of the course was, using the engineering process of problem solving, developing practical experimental skills in electronic circuit testing and in software system testing. Recognise issues to be addressed in a combined hardware and software system design and finally implement a project for remotely supervised autonomous Buggies and code version control.

To achieve these objectives, we were first introduced to the Arduino board and its functioning. Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. We were taught how to interface various sensors like IR, Ultrasonic, etc. with Arduino and then subsequently using the L293D motor driver IC to run motors and control their speeds. NVIS 3302ARD Robo Car is an electro-mechanical platform with the capability of sensing the environment, processing the data, and acting according to the pre-programmed sequence. It is a miniature prototype car powered by batteries whose motion is controlled by microcontroller. Various Sensors can be interfaced like IR (Infrared) Sensor, Ultrasonic Sensor, Analog Sensor, and many more. Acquiring the required concepts of interfacing hardware and sensors with the Arduino we were then allowed to work on the final project.

For the final project we were required to create the hardware interface and working code for the given problem of making a buggy autonomous to the extent that it can traverse a given track without any human intervention, detecting obstacles as it goes, making a certain number of passes. On completing the specified number of passes the buggy was supposed to leave the main track and park itself in the position it had started from.

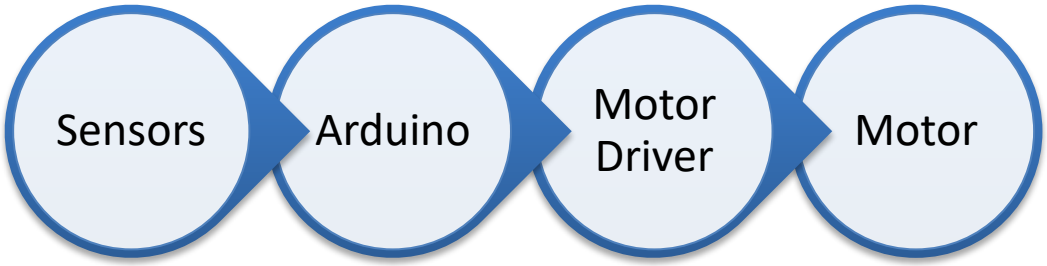
This report on the final project highlights the direction we have taken to achieve the given task, all the hardware used, all the code and logic, contributions of each member and the results achieved thereafter.

2. DESIGN CONSIDERATIONS

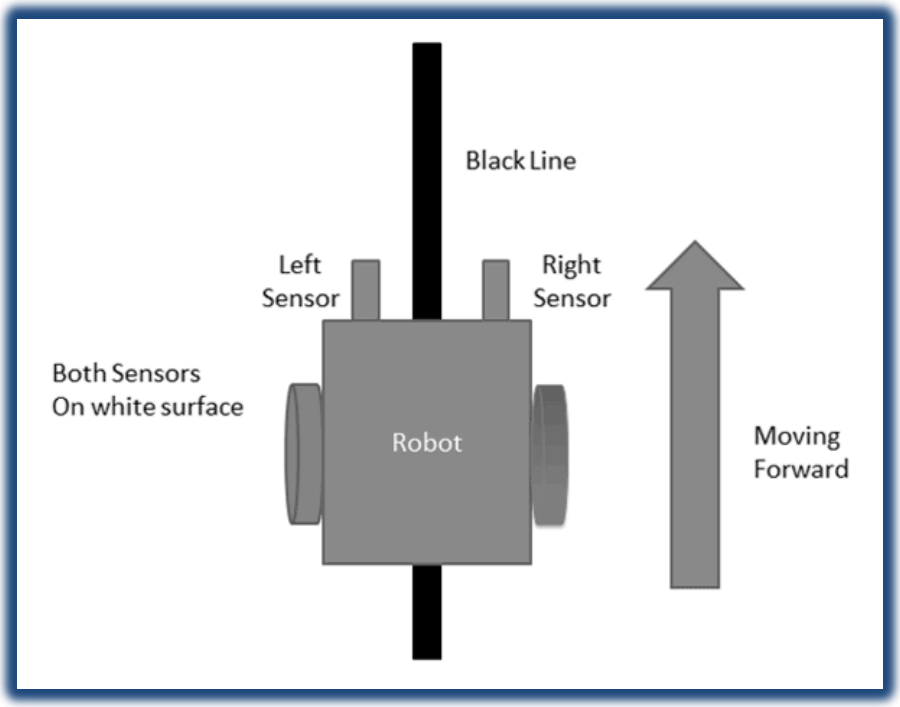
Given the task at hand, certain design considerations were made to make the code and the overall design of the buggy more general in nature. The task was to pass through three gates while traversing the track and increase count by one every time a gate is encountered. To achieve this, initially an ultrasonic sensor was mounted on top of the buggy to detect when passing under a gate in addition to the infrared sensor mounted in front of the buggy to detect the track. However, with this configuration it was found that using the ultrasonic sensor in tandem with the IR sensor is not feasible due to slower processing time of the ultrasonic input, during which the IR sensor stops working and the buggy moves off track. To resolve this issue, the ultrasonic sensor was replaced with another IR sensor mounted on the top of the buggy to detect the gate.

Also, to ensure that the code is more general to the track and not specifically for the one presented in the task, the use of delays in the code by measuring time was minimised, and instead the turns were counted on detection and the required operation to be carried out at the detection of the right stimulus.

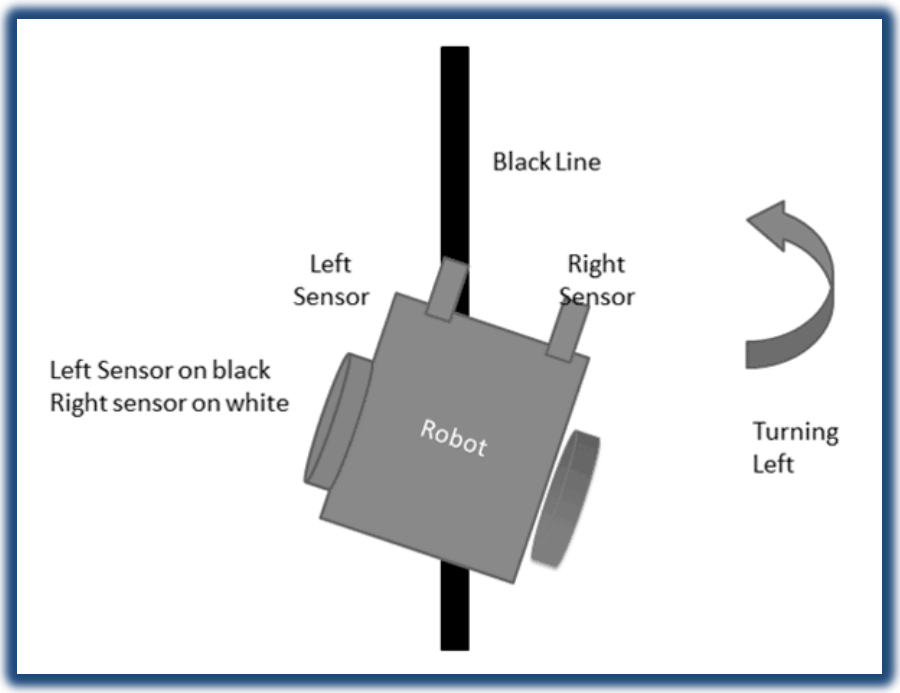
Working of line follower is very interesting. Line follower robot senses black line by using sensor and then sends the signal to Arduino. Then Arduino drives the motor according to sensors' output.



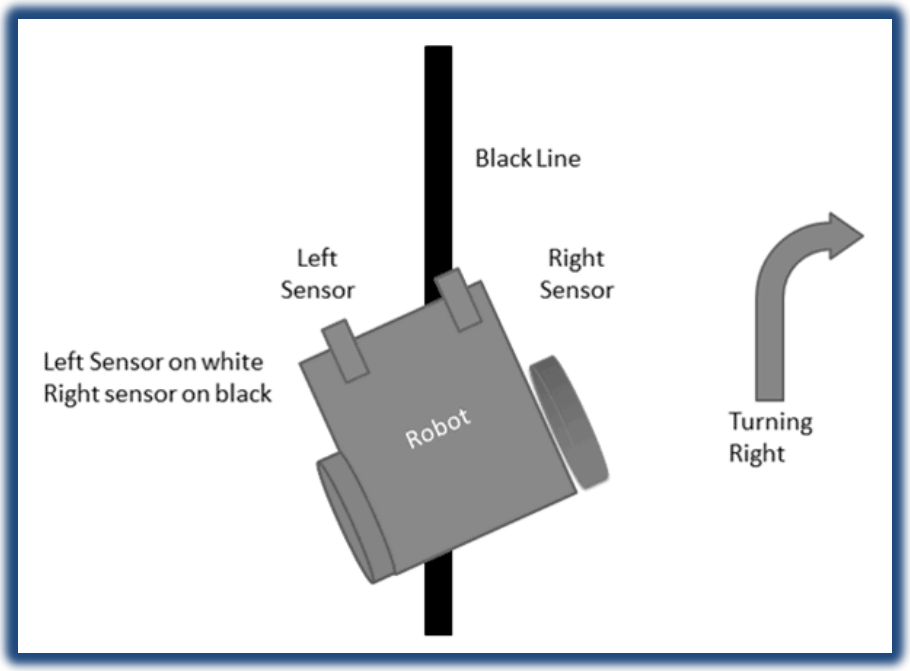
Here in this project we are using two IR sensor modules namely left sensor and right sensor. When both left and right sensor senses white then robot moves forward.



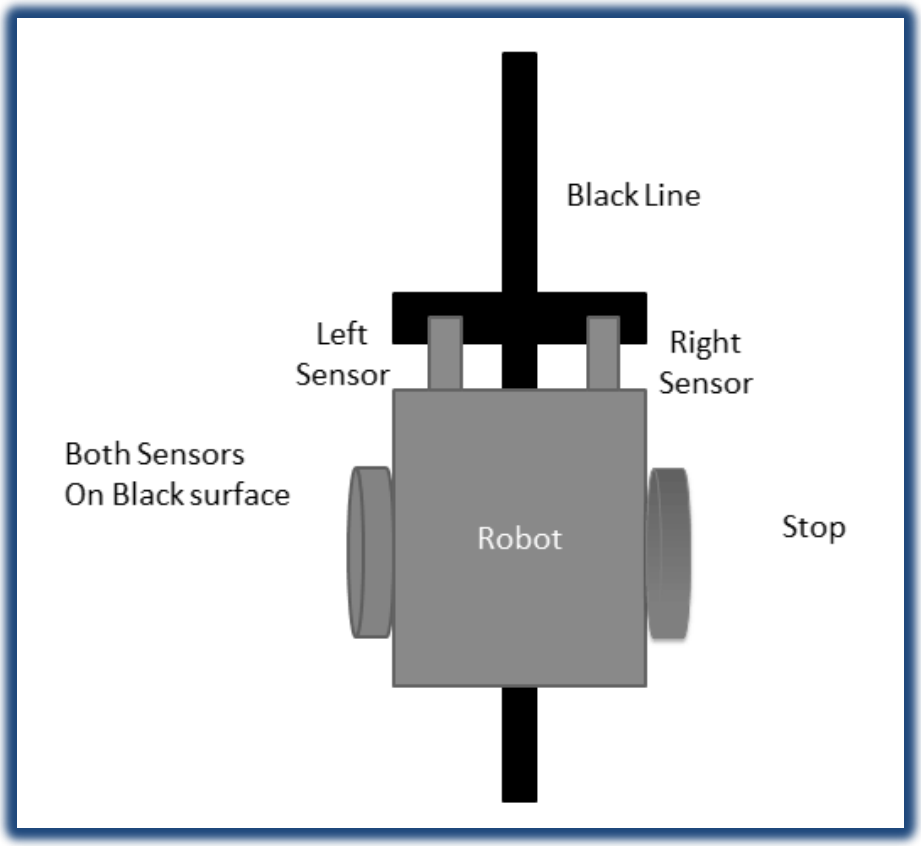
If left sensor comes on black line then robot turn left side.



If right sensor sense black line then robot turn right side until both sensor comes at white surface. When white surface comes robot starts moving on forward again.



If both sensors comes on black line, robot stops.



2.1 Student 1: ABHISHEK BANSAL

2.1.1 GANTT CHART

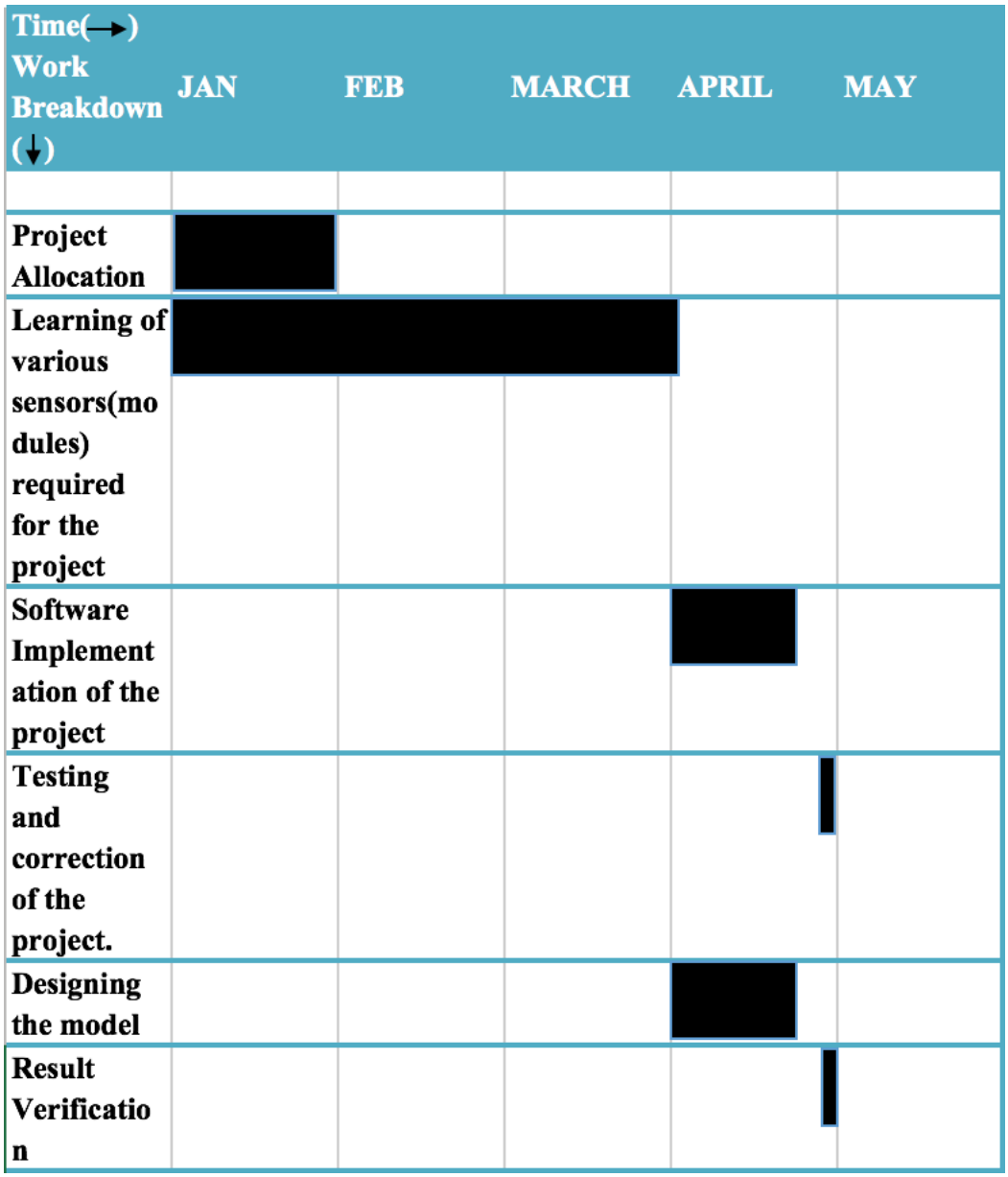
Time(→) Work Breakdown (↓)	JAN	FEB	MARCH	APRIL	MAY
Project Allocation					
Learning of various sensors(modules) required for the project					
Software Implementation of the project					
Testing and correction of the project.					
Designing the model					
Result Verification					

2.1.2 CONTRIBUTION

Student 1 learned the functioning of different sensors along with the rest of the group, helped with the software implementation, suggested provided valuable insights when problems arose, gave unique and useful ideas to solve the problem at hand.

2.2 Student 2: ADITYA KHANIJOW

2.2.1 GANTT CHART



2.2.2 CONTRIBUTION

Student 2 learned the functioning of different sensors along with the rest of the group, helped with the software implementation, suggested provided valuable insights when problems arose, helped out with the code, gave unique and useful ideas to solve the problem at hand.

2.3 Student 3: ANKUR SHARMA

2.3.1 GANTT CHART

Time(→) Work Breakdown (↓)	JAN	FEB	MARCH	APRIL	MAY
Project Allocation					
Learning of various sensors(modules) required for the project					
Software Implementation of the project					
Testing and correction of the project.					
Designing the model					
Result Verification					

2.3.2 CONTRIBUTION

Student 3 learned the functioning of different sensors along with the rest of the group, helped with the software implementation, suggested provided valuable insights when problems arose, helped out with the code, gave unique and useful ideas to solve the problem at hand.

2.4 Student 4: DEVANSHU NARULA

2.4.1 GANTT CHART

Time(→) Work Breakdown (↓)	JAN	FEB	MARCH	APRIL	MAY
Project Allocation					
Learning of various sensors(modules) required for the project					
Software Implementation of the project					
Testing and correction of the project.					
Designing the model					
Result Verification					

2.4.2 CONTRIBUTION

Student 4 learned the functioning of different sensors along with the rest of the group, helped with the software implementation, suggested provided valuable insights when problems arose, helped out with the code, gave unique and useful ideas to solve the problem at hand.

3. COMPONENT DETAILS

L293D IC: L293D is a typical Motor driver or Motor Driver IC which allows motor to drive on either direction. A single L293D chip has two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If anyone of the either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It's like a switch.

Motor is connected to OUTPUT pins. Output pins vary according to the inputs applied at INPUT pins. Truth table is as follows:

- Pin 2 = Logic 1 and Pin 7 = Logic 0 | Clockwise Direction
- Pin 2 = Logic 0 and Pin 7 = Logic 1 | Anticlockwise Direction
- Pin 2 = Logic 0 and Pin 7 = Logic 0 | Idle [No rotation] [Hi-Impedance state]
- Pin 2 = Logic 1 and Pin 7 = Logic 1 | Idle [No rotation]

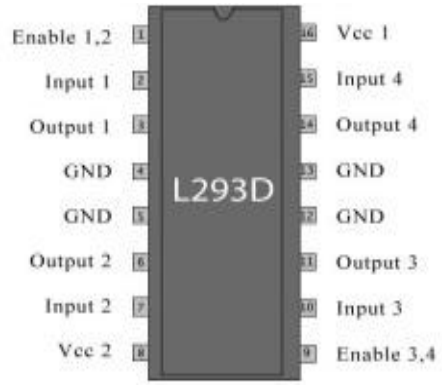
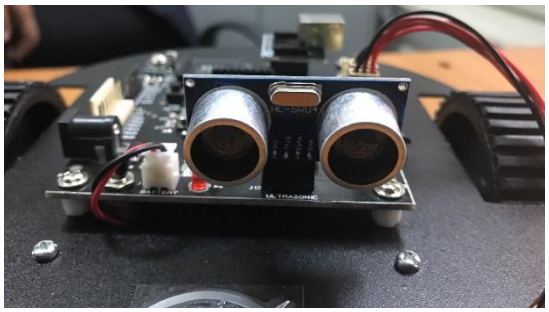
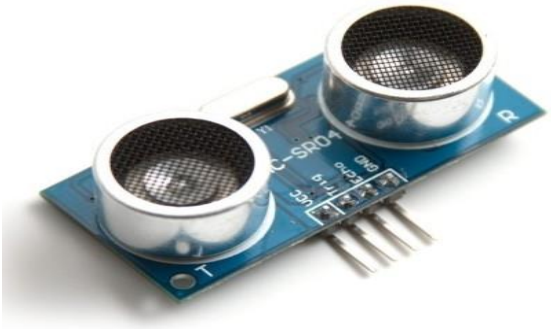


Fig.3. pin out configuration of L293D IC

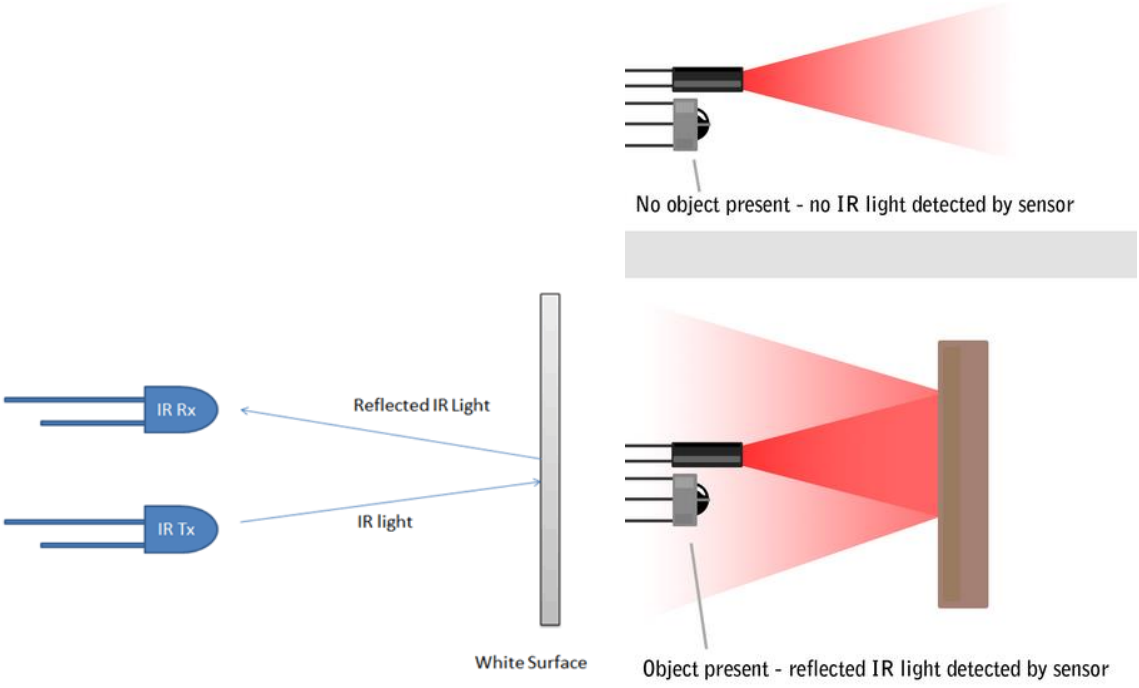
DC MOTOR: A motor is a machine that converts electrical energy into mechanical energy (rotation). The key elements of DC motor are field winding and an armature winding. As electric currents flow through the windings, torque is developed between these two windings. Its applications are fans, blowers and pumps, machine tools, household appliances, power tools etc.

ULTRASONIC SENSOR: The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It comes complete with ultrasonic transmitter and receiver module. Ultrasound has several characteristics which make it so useful. Firstly, it is inaudible to humans and therefore undetectable by the user. Secondly, ultrasound waves can be produced with high directivity. Thirdly, they are a compression vibration of matter (usually air). Finally, they have a lower propagation speed than light or radio waves.





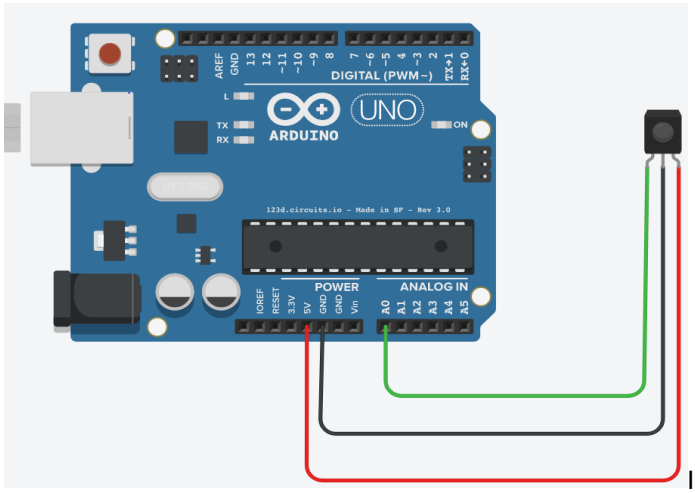
IR SENSOR: An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor. IR sensor works in the near infrared region. IR sensor consists of an IR emitter LED and an IR receiver. IR sensor that is used in this project has 3 pins. GND, SUPPLY and OUTPUT pin. GND is connected to ground, SUPPLY is connected to +5V and the OUTPUT pin give the intensity of light received in terms of voltage.



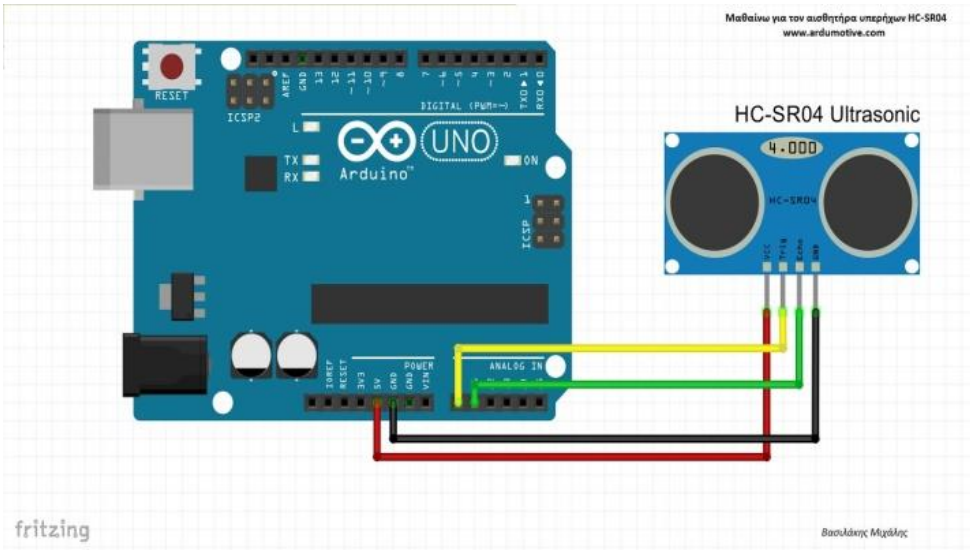
ZIGBEE MODULE: ZigBee S2 RF module is used for long range communication. It has range of 1600 meters in line of sight or 90 meters in indoors or urban area. It is used for embedded solutions providing addressable wireless end - point connectivity to devices. This ZigBee wireless device can be directly connected to the serial port (at 3.3 V level) of your microcontroller. By using a logic level translator it can also be interfaced to 5V logic (TTL) devices having serial interface. This module supports data rates of up to 115kbps.

4. CONNECTION DIAGRAM WITH ARDUINO

IR SENSOR

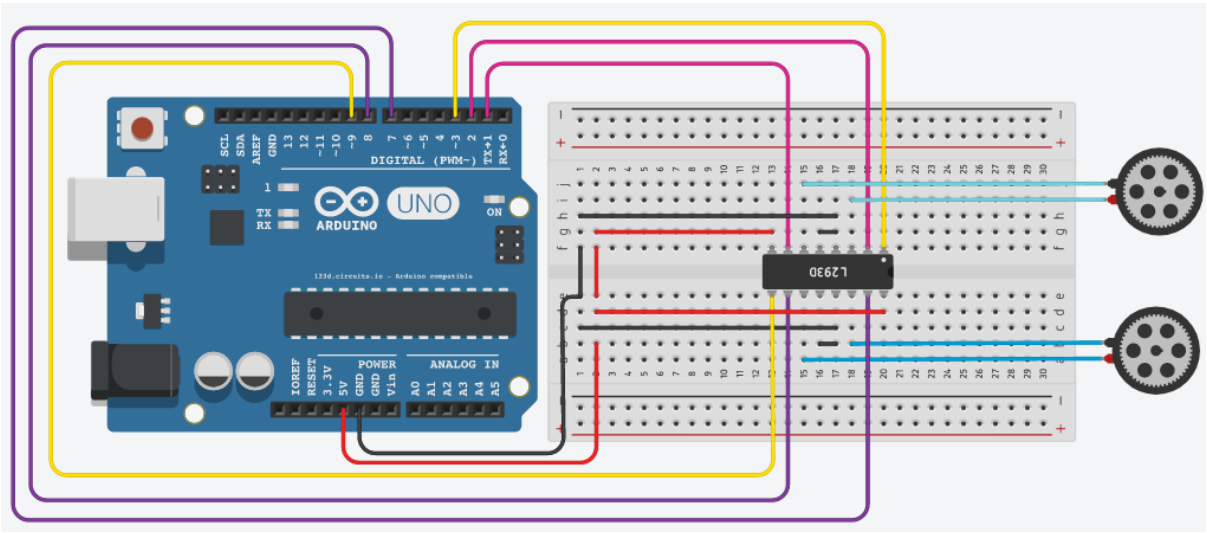


ULTRASONIC SENSOR



DC MOTOR

Input 1 (pin 2)	Input 2 (pin 7)	Effect on motor
0	0	No rotation
0	1	Anticlockwise rotation
1	0	Clockwise rotation
1	1	No rotation



5. PROGRAM CODE

```
int lm1=5;
int lm2=6;
int rm1=7;
int rm2=8;
float duration;
void setup()
{
  pinMode(lm1,OUTPUT);
  pinMode(lm2,OUTPUT);
  pinMode(rm1,OUTPUT);
  pinMode(rm2,OUTPUT);
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
}

void forward(){
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,HIGH);
  delay(5);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
  delay(10);
}
```

```
void left(){
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
  delay(5);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
  delay(10);
}

void right(){
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,HIGH);
  delay(2);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
  delay(5);
}

void brake(){
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
```

```
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
}

float irl(){
  return analogRead(A0);
}

float irr(){
  return analogRead(A1);
}

int gray=400;
int barr=0;
int blacks=0;
void loop()
{
   //forward();
   float irreadl=irl();
   float irreadr=irr();

   if(blacks!=7){
    if(irreadl>gray&&irreadr>gray){
      forward();
    }
    else if(irreadl<gray&&irreadr>gray){
      left();
    }
    else if(irreadl>gray&&irreadr<gray){
```

```
 right();
}
else if(blacks==0&&irreadl<gray&&irreadr<gray){
  blacks++;
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,HIGH);
  delay(20);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
  delay(1000);                //Optimize
}
else if(blacks!=4&&blacks>0&&irreadl<gray&&irreadr<gray){
  blacks++;
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,HIGH);
  delay(100);                 //Optimize
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
  delay(50);                  //Optimize
}
else if(blacks==4&&irreadl<gray&&irreadr<gray){
```

```
        blacks++;
        digitalWrite(lm1,HIGH);
        digitalWrite(lm2,LOW);
        digitalWrite(rm1,LOW);
        digitalWrite(rm2,LOW);
        delay(900);                 //Optimize
      }
    }
    else{
     brake();
    }
}
```

6. STANDARDS USED:

S. No.	Subject	Standards
1.	**Microprocessor and their Applications**	**1. 1754-1994 - IEEE Standard for a 32-bit Microprocessor Architecture** A 32-bit microprocessor architecture, available to a wide variety of manufacturers and users, is defined. The standard includes the definition of the instruction set, register model, data types, instruction op-codes, and coprocessor interface. **2. IEEE 694-1985 - IEEE Standard for Microprocessor Assembly Language** The intent of this standard is to name a common set of instructions used by most general purpose microprocessors, to provide rules for the naming of new instructions and the derivation of new mnemonics, and to establish assembly language conventions and syntax. It is intended to be used as a basis for microprocessor assembler specification. This standard sets forth the functional definitions and corresponding mnemonics for microprocessor instructions. The specific set of instructions differs for each type of processor. Likewise, the number of condition codes and their setting and resetting is processor-dependent, and is not prescribed by this standard. Each use of a particular microprocessor should be thoroughly conversant with its operation and should make no a priori assumptions with regard to the detailed behaviour of its instructions.

2.	**Engineering Design-II (Buggy)**	**1. IEEE 1212-2001 - IEEE Standard for a Control and Status Registers (CSR) Architecture for Microcomputer Buses** A common bus architecture (which includes functional components-modules, nodes, units-and their address space, transaction set, CSRs, and configuration information) suitable for both parallel, serial buses is provided in this standard. Bus bridges are enabled by the architecture, but their details are beyond its scope. Configuration information is self-administered by vendors, organizations based upon IEEE Registration Authority company id. **2. 1451.2-1997 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats** A digital interface for connecting transducers to microprocessors is defined. A TEDS and its data formats are described. An electrical interface, read and write logic functions to access the TEDS and a wide variety of transducers are defined. This standard does not specify signal conditioning, signal conversion, or how the TEDS data is used in applications.

3.	**Microcontrollers and Embedded Systems**	**1. IEEE 1275.1-1994 - IEEE Standard for Boot (Initialization Configuration) Firmware: Instruction Set Architecture (ISA) Supplement for IEEE 1754** Firmware is the read-only-memory (ROM)-based software that controls a computer between the time it is turned on and the time the primary operating system takes control of the machine. Firmware's responsibilities include testing and initializing the hardware, determining the hardware configuration, loading (or booting) the operating system, and providing interactive debugging facilities in case of faulty hardware or software. The core requirements and practices specified by IEEE Std 1275-1994 must be supplemented by system-specific requirements to form a complete specification for the firmware for a particular system. This standard establishes such additional requirements pertaining to the instruction set architecture (ISA) defined by IEEE Std 1754-1944, IEEE Standard for a 32-bit Microprocessor Architecture. **2. 21451-2-2010 - ISO/IEC/IEEE Standard for Information technology -- Smart transducer interface for sensors and actuators -- Transducer to microprocessor communication protocols and Transducer Electronic Data Sheet (TEDS) formats** Adoption of IEEE Std 1451.2-1997. A digital interface for connecting transducers to microprocessors is defined. A Transducer Electronic Data Sheets (TEDS) and its data formats are described. An electrical interface, read and write logic functions to access the TEDS and a wide variety of transducers are defined. This standard does not specify signal conditioning, signal conversion, or how the TEDS data is used in applications.

7. RESULT

The result observed was that with all the hardware interfacing and code, the robot car was able to traverse the track anticlockwise on its own using the infrared sensor, counting each time it passed under a gate. After completing three rounds, the robot car exited the main track and parked itself where it had started.