A
MST Practical Activity Report
Submitted for

ENGINEERING DESIGN-II (UTA024)

Submitted by:

(102103219) Susnigdha Kheda

(102103348) Sannidhya Jain

(102103349) Sneha Tuli

(102103350) Mayank Aggarwal

(102103351) Gaurvi Sood

BE Second Year

2CO13

Submitted to-

Govind Ram Chimmpa

CSED

Computer Science and Engineering Department

TIET, Patiala

Jan-June 2023

ABSTRACT

The Engineering Design Project II is a one of its kind project. In a curriculum of theoretical courses, this course provides an opportunity to implement theoretical knowledge into real-life applications. The application of Programming and Circuits in real-time applications is the highlight of this course.

This course is based on the basic fundamental concepts of electronics such as basic electronic devices, circuit-design and transmitter and receiver and also includes basic C and Arduino programming. It also introduces the concepts of sensors such as Infra-Red (IR) and Ultrasonic.

This project file illustrates the various steps involved in the construction of an Arduino Uno based “Buggy” robot which has various sensors attached to it, complimented with exploiting various functions available in Arduino IDE, to better understand how the board works and communicates with the user and the buggy. The Arduino Uno circuit can be simulated using Breadboard, IR Sensors, Ultrasonic sensors and various other components.

This file delves into, particularly, the programming embedded into the ICs that power the buggy and explores various logic-based codes (in C language), the command blinking/dimming of specific LEDs with delays inputted by the user, programming the two wheels of the buggy to move/rotate it in desired fashion, and the usage of built-in functions within the Arduino IDE to program an ultrasonic sensor to display the distance of an obstacle from itself.

DECLARATION

We declare that the project work entitled “Buggy” for the CS Lab department project report is carried out under the supervision of Govind Ram Chimmpa, CSED, whose guidance helped us realise the results of our work. It is assured that the conclusions drawn from this report are based on original research and efforts and are devoid of any plagiarism. This report has not been submitted to any university or institution except Thapar Institute of Engineering and Technology, Patiala.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INDEX

Experiment 1

OBJECTIVE: Introduction to Arduino Microcontroller

SOFTWARE USED: Arduino IDE

HARDWARE USED: Arduino UNO board

THEORY:

A microcontroller is a small, low-power computer-on-a-chip that is designed to control various devices and perform various functions. It typically includes a processor, memory, input/output (I/O) interfaces, and other components, all integrated onto a single chip. Microcontrollers are widely used in a variety of applications, from simple toys and appliances to complex industrial systems and automobiles.

Arduino is an open-source electronics platform based on microcontroller boards. It provides a simple and easy-to-use environment for creating and programming microcontroller projects. The Arduino board includes a microcontroller and a variety of input/output (I/O) interfaces, such as digital and analog inputs and outputs, serial communication, and pulse-width modulation (PWM). The Arduino software development environment (IDE) is used to write and upload code to the board. The code is written in the Arduino programming language, which is based on the C++ programming language.

One of the main benefits of using Arduino is its simplicity. The Arduino community has also developed a large number of libraries and resources that make it easier to add new functions and capabilities to your projects.

In summary, Arduino is a platform that combines hardware and software to make it easier to create microcontroller projects. Whether you're a beginner or an experienced engineer, Arduino provides a flexible and powerful tool for creating a wide range of projects.

Arduino Uno:

Arduino Uno is a microcontroller board based on the ATmega328P. It is one of the most popular boards in the Arduino lineup and is widely used for a variety of projects, from simple hobby projects to complex industrial applications.

The Arduino Uno features 14-digital input/output (DIO) pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, and a power jack. It also includes a reset button and ICSP header for programming the board. The Arduino Uno runs at 5V and has a maximum current rating of 20 mA per I/O pin.

CIRCUIT DIAGRAM:

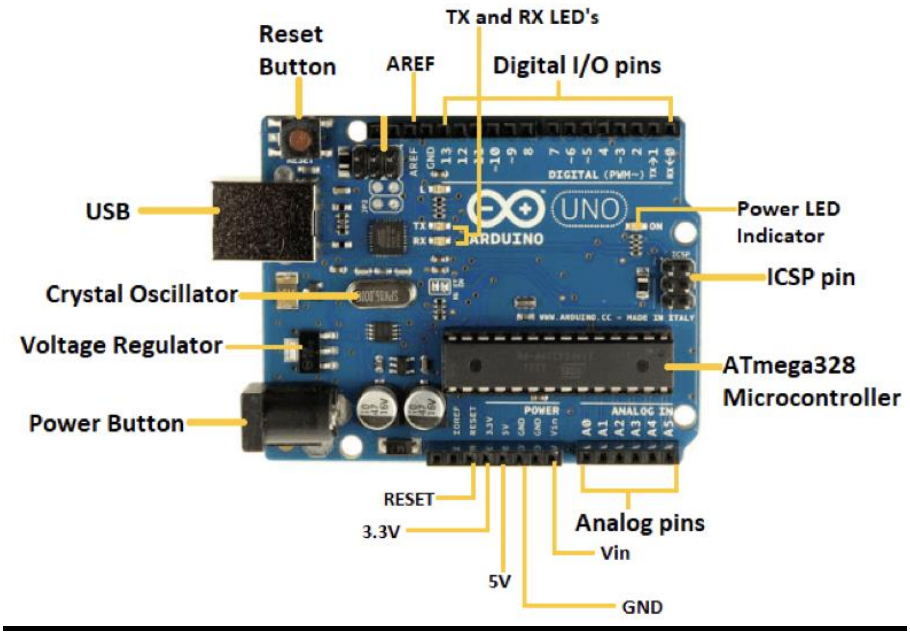


Fig 1.1 Arduino Uno Microcontroller

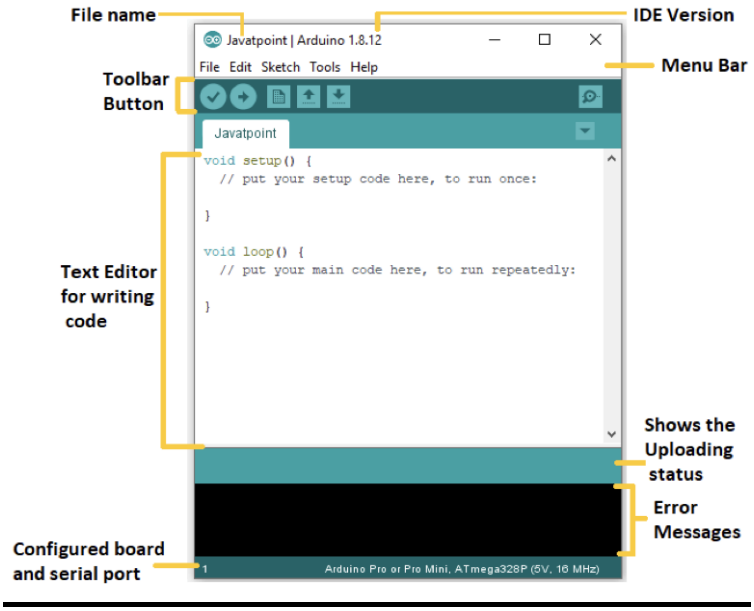


Fig 1.2 Arduino IDE

CONCLUSION:

This Arduino microcontroller will help us to communicate with the buggy (Nvis 3302ARD RoboCar). Thus, it will enable us to code into Robocar and is the most important part of our project.

Signature of Faculty member

Experiment 2(a)

OBJECTIVE: To Blink an LED

SOFTWARE USED: Arduino IDE

HARDWARE USED: Arduino IDE along with listed hardware.

Table 2.1 Hardware to blink LEDs

NAME OF COMPONENT	QUANTITY
White LEDs	1
Breadboard	1
Arduino UNO	1
Jumper Wire	2
Resistors	1

THEORY:

1. **LEDs**: Light Emitting Diode, a semiconductor device that emits light when an electric current is passed through it. They are widely used in a variety of applications, including lighting, displays, and signaling. They are energy efficient and have a long lifespan.
2. **Breadboard**: A breadboard is a type of reusable circuit board that allows for the easy prototyping of electronic circuits. It has a grid of holes that are used to connect components, such as resistors, capacitors, and transistors, with jumper wires. Breadboards are commonly used by engineers and students for circuit design and experimentation.
3. **Arduino UNO**: Arduino UNO is an open-source microcontroller board based on the ATmega328P microcontroller. It is widely used in the prototyping and development of interactive objects, and also in the development of IoT, home automation and other applications.
4. **Jumper Wire**: Jumper wires are short cables used to connect two points in an electronic circuit. They have connectors that can be inserted into a breadboard or other circuit board. They are used to connect different parts of a circuit for prototyping and experimentation.

5. **Resistors**: A resistor is a passive component that resists electrical current, measured in ohms. They are used to regulate current, divide voltage and create a potential difference in a circuit. They are widely used in electronic circuits for multiple purposes.

In this experiment, a C program was written to blink an LED with a delay of 1000ms between each blink.

Arduino IDE basic commands:

1. void setup(): The setup() function is called once when the sketch starts running. It is used to initialize variables, pin modes, start libraries, and perform any other actions that need to be done once at the start of the program

2. void loop(): The loop() function is called repeatedly after the setup() function has finished executing. It contains the main logic of the program and is executed repeatedly until the board is powered off or reset

3. pinMode: This function sets a pin as either an input or an output.

4. digitalWrite: This function writes a digital value (HIGH or LOW) to a specified pin. For example, to set pin 13 to a high level, you would use the following code:

```
digitalWrite(13, HIGH);
```

5. digitalRead: This function reads the digital value of a specified pin. For example, to read the value of pin 2, you would use the following code:

```
int value = digitalRead(2);
```

6. analogRead: This function reads the analog value of a specified pin. The analog input pins can be used to read sensors that produce an analog signal, such as a potentiometer or a temperature sensor.

```
int value = analogRead(A0);
```

7. analogWrite: This function writes an analog value (PWM) to a specified pin. The PWM pins can be used to control the brightness of LEDs or the speed of motors. For example, to set the brightness of pin 9 to 50%, you would use the following code:

```
analogWrite(9, 128);
```

8. <u>delay:</u> This function pauses the program for a specified number of milliseconds. For example, to delay the program for 1000 milliseconds (1 second), you would use the following code:

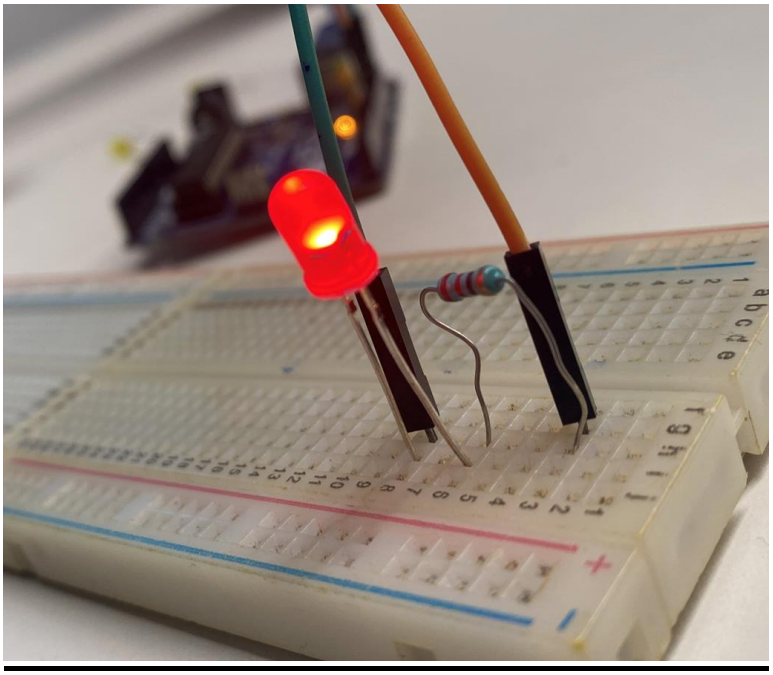
delay(1000);

In this experiment, we need to write a program so that the LED should switch ON/OFF after a delay of 1000ms.

CODE:

```
void setup(){
  pinMode(5, OUTPUT);
}
void loop(){
  digitalWrite(5, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(5, LOW);
  delay(1000); // Wait for 1000 millisecond(s)}
}
```

CIRCUIT DIAGRAM:



<u>Fig 2.1 Blinking of LEDs</u>

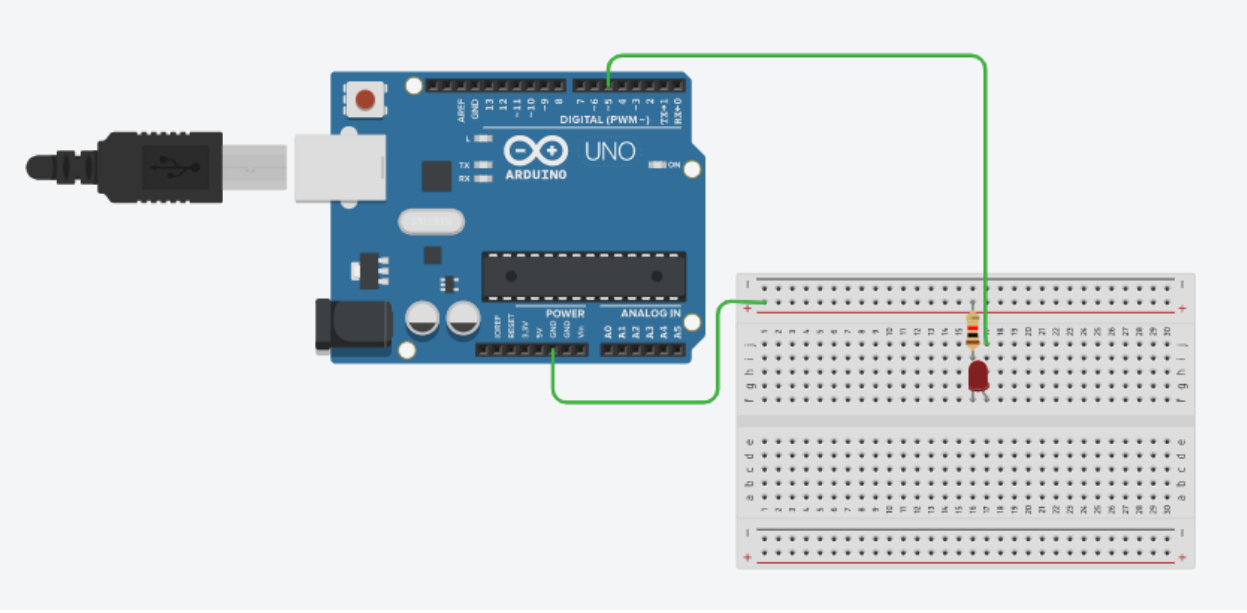


Fig 2.2 Blinking of LEDs TinkerCad Circuit

CONCLUSION:

The Arduino UNO board was successfully programmed to blink an LED. The code was written in C language as mentioned above. The Use of various pins on the board was successfully demonstrated.

Signature of Faculty member

Experiment 2(b)

OBJECTIVE: To blink multiple LEDs.

SOFTWARE USED: Arduino IDE

COMPONENTS USED: Arduino IDE along with listed hardware.

Table 2.2 Hardware to blink LEDs

NAME OF COMPONENT	QUANTITY
LEDs	**5**
Breadboard	**1**
Arduino UNO	**1**
Jumper Wire	**6**
Resistors	**5**

THEORY:

In this experiment, 5 LEDs were programmed to blink in pattern one by one starting from first till last LED and then in reverse pattern. Emphasis was laid to first set pins to OUTPUT mode and then blink them one by one using for loop in loop() function, in the program to successively switch them ON/OFF, with a delay of 1000ms bring provided between each blink.

CODE:

```
int my_delay = 1000;
int a[5] = {2,3,4,5,6};
void setup(){
i=0
while(i<5){
  pinMode(a[i],OUTPUT);
i++;
  };
}
void loop(){while(i<5){
    digitalWrite(a[i],HIGH);
    delay(my_delay);
    digitalWrite(a[i],LOW);
    delay(my_delay);
    i++;
};}
```

CIRCUIT DIAGRAM:

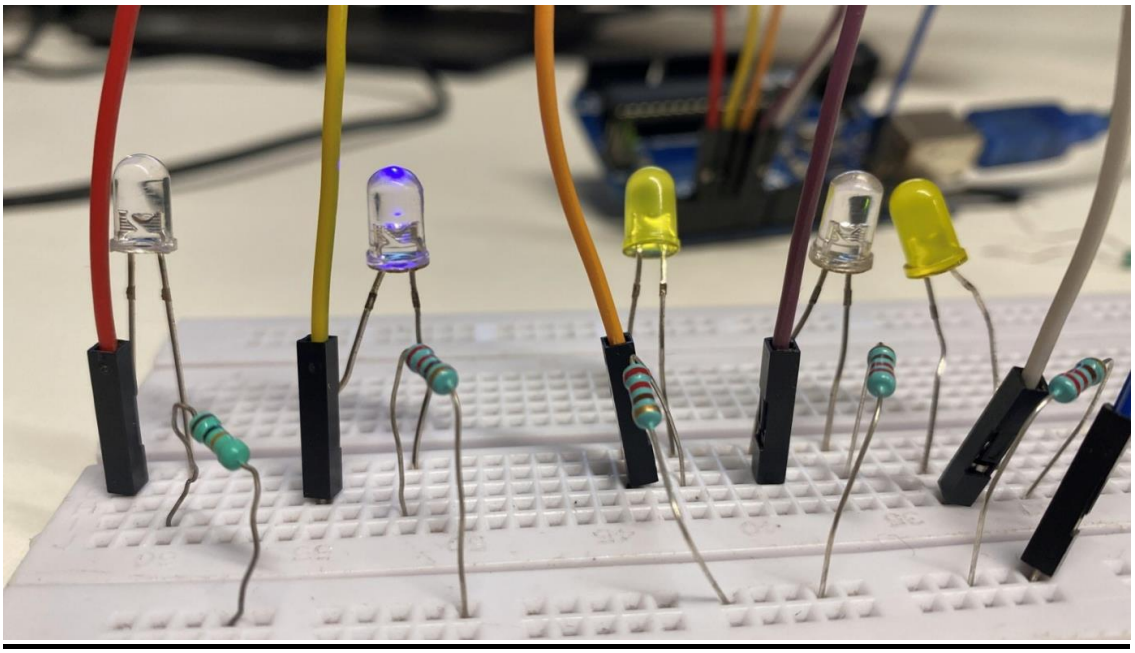


Fig 2.3 Blinking of multiple LEDs

CONCLUSION:

The Arduino UNO board was successfully programmed to blink 5 LEDs. The code was written in Arduino IDE as mentioned above and Concept of arrays, for loop and delay was successfully applied to switch the LEDs ON/OFF in patterns as specified in the objective.

Signature of Faculty member

Experiment 3

OBJECTIVE: To write a program to design a pattern from sequence of multiple LEDs using for loop in Arduino

SOFTWARE USED: Arduino IDE

COMPONENTS USED: Arduino IDE along with listed hardware.

Table 3.1 Hardware to blink LEDs

NAME OF COMPONENT	QUANTITY
LEDs	**5**
Breadboard	**1**
Arduino UNO	**1**
Jumper Wire	**6**
Resistors	**5**

THEORY:

In this experiment, 5 LEDs were programmed to blink in pattern one by one starting from first till last LED and then in reverse pattern. Emphasis was laid on the usage of Arrays and for loop to first set pins to OUTPUT mode and then blink them one by one using for loop in loop() function, in the program to successively switch them ON/OFF, with a delay of 1000ms bring provided between each blink.

CODE:

```
int my_delay = 1000;
int a[5] = {2,3,4,5,6};
void setup(){
for(int i=0;i<5;i++){
  pinMode(a[i],OUTPUT);
  }
}
void loop(){for(int i=0;i<5;i++){
    digitalWrite(a[i],HIGH);
    delay(my_delay);
    digitalWrite(a[i],LOW);
    delay(my_delay);
}}
```

CIRCUIT DIAGRAM:

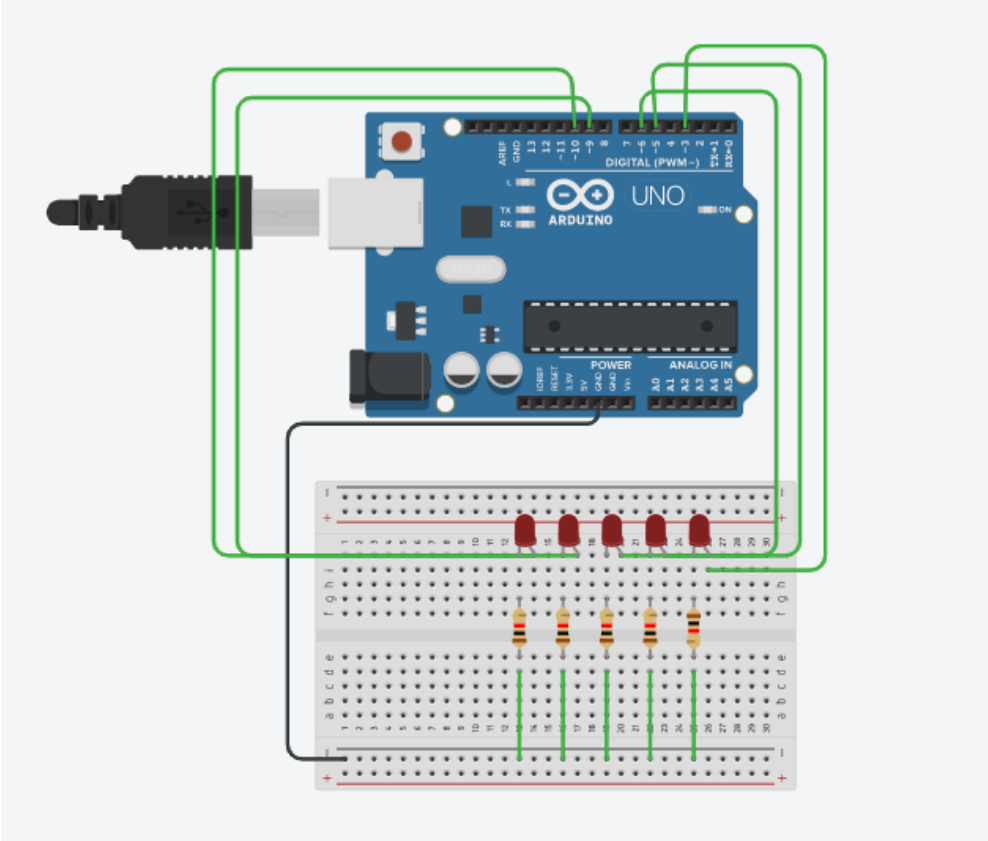


Fig 3.1 Blinking of multiple LEDs (for Loop) TinkerCad Circuit

CONCLUSION:

The Arduino UNO board was successfully programmed to blink 5 LEDs. The code was written in Arduino IDE as mentioned above and Concept of arrays, for loop and delay was successfully applied to switch the LEDs ON/OFF in patterns as specified in the objective.

Signature of Faculty member

Experiment 4

OBJECTIVE: Program to demonstrate sending data from computer to Arduino Board and control brightness of LED

SOFTWARE USED: Arduino IDE

COMPONENTS USED: Arduino IDE along with listed hardware.

Table 4.1 Controlling brightness of LED

NAME OF COMPONENT	QUANTITY
White LEDs	**3**
Yellow LEDs	**2**
Breadboard	**1**
Arduino UNO	**1**
Jumper Wire	**6**
Resistors	**5**

THEORY:

(i) We turn an LED on or off based on the values received from the serial connection. It continuously checks if there is any data available on the serial connection. If there is data, the program reads an integer value using the **parseInt** function and stores it in the **state** variable. If the value of **state** is greater than 0, the LED connected to pin 11 is turned on using the **analogWrite** function. The brightness of the LED is proportional to the input value, where 0 corresponds to 0% brightness and 255 corresponds to 100% brightness. If the value of **state** is less than or equal to 0, the LED will not turn on.

(ii) We have to execute an Arduino program that is used to control the brightness of five LEDs connected to pins 3, 5, 6, 9, and 10 on the board. The program initializes the serial communication with a baud rate of 9600 and sets the specified pins to output mode.

In the **loop** function, the program continuously reads integer values from the serial connection, and based on the value, it sets the brightness of the corresponding LED using the **analogWrite** function. The brightness is proportional to the input value, where 0 corresponds to 0% brightness, and 255 corresponds to 100% brightness. The program waits for 2 seconds before turning off the LED by setting its brightness to 0.

This code can be used to control the brightness of multiple LEDs by sending serial commands to the Arduino board.

This code checks the value of i, which is an integer read from the serial connection, and sets the brightness of the corresponding LED based on the value of i. The code checks the value of i against several ranges, and the corresponding LED is selected based on the range in which i falls.

CODE:

1. Code for performing Dimming Effect on single LED:

```
void setup(){
Serial.begin(9600);
pinMode(11, OUTPUT);
void loop() {
if (Serial.available())
{
int state = Serial.parseInt();
if (state > 0){
analogWrite(11, state);}
}
//or digitalWrite(11);
}
```

2.Code for performing Dimming Effect on 5 LED's :

```
 void setup() {
  Serial.begin(9600);

  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
 }
 void loop() {
  int i;
  i=Serial.parseInt();

 if(i>=0 && i<=50){
    analogWrite(3,i);
    delay(2000);
    analogWrite(3,0);  }
  else if(i>=50 && i<100){
    analogWrite(5,i);
    delay(2000);
    analogWrite(5,0);  }
  else if(i>=100 && i<150){
    analogWrite(6,i);
    delay(2000);
    analogWrite(6,0);  }
  else if(i>=150 && i<200){
    analogWrite(9,i);
    delay(2000);
    analogWrite(9,0);  }
```

```
else{
  analogWrite(10,i);
  delay(2000);
  analogWrite(10,0);}}
```

CIRCUIT DIAGRAM:

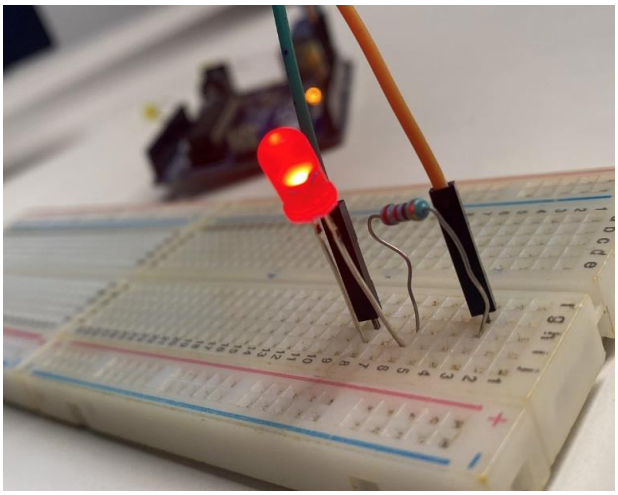


Fig 4.1 Output of single LED Dimming effect

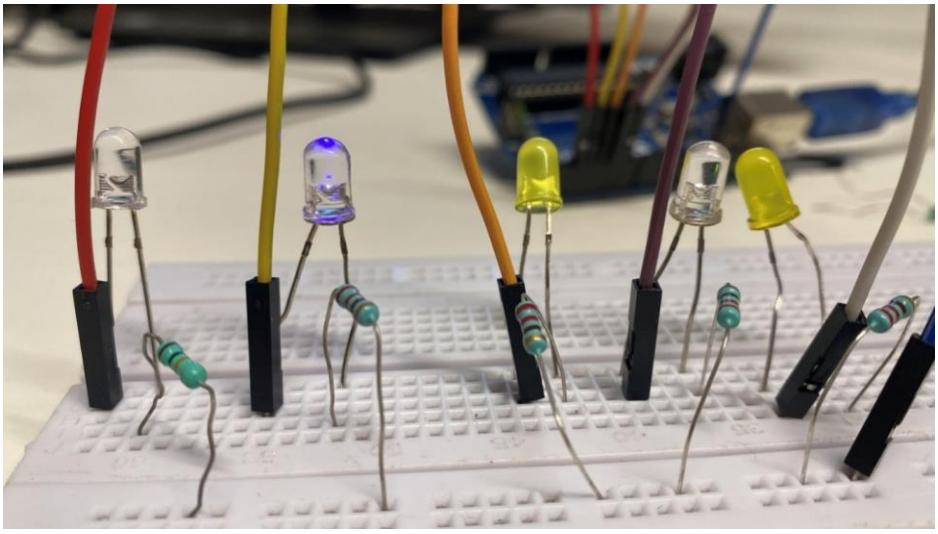


Fig 4.2 Output of 5 LEDs Dimming effect

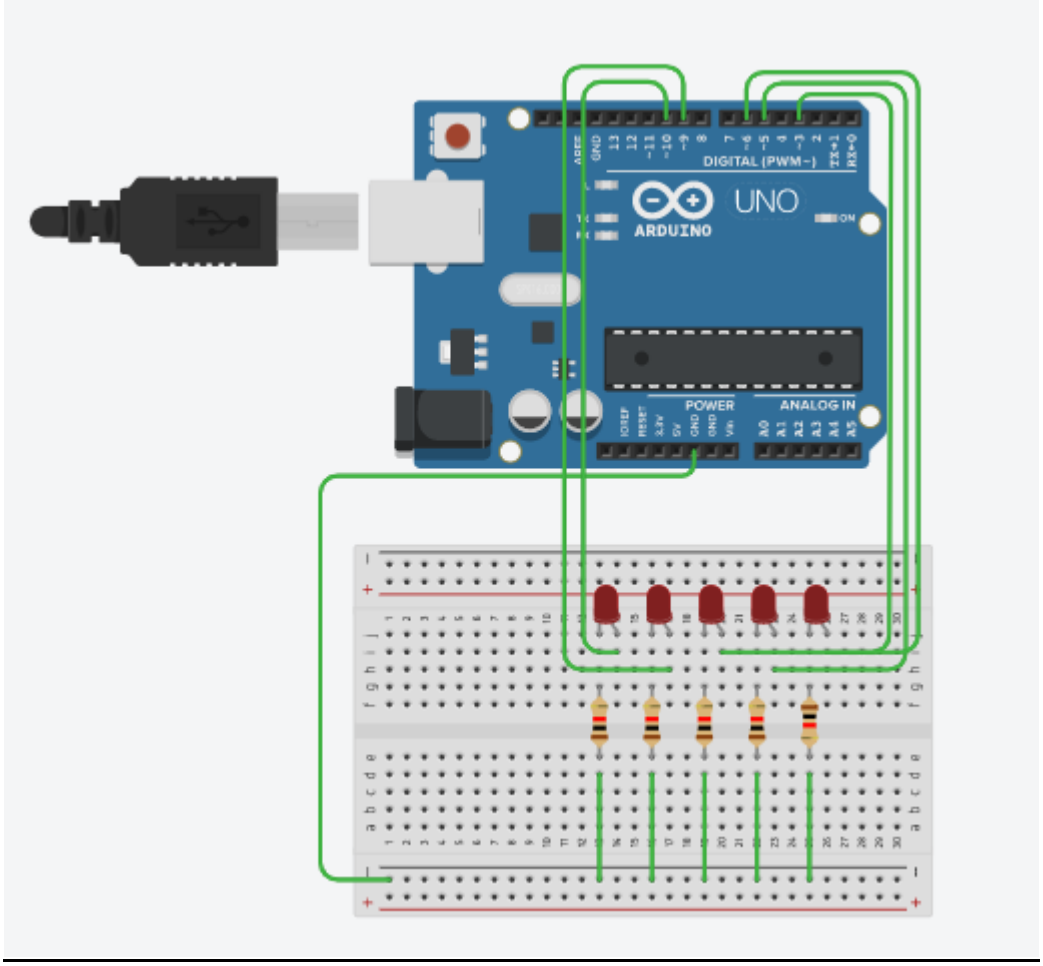


Fig 4.3 TinkerCad Circuit for 5 LEDs Dimming effect

CONCLUSION:

The Arduino UNO board was successfully programmed to manage the intensity of LEDs. First, we used a single LED and changed the intensity of the LED with the input given to it using digitalWrite or analogWrite functions. Then we use five LEDs to glow the LED for which intensity lies within the specified range. Code was written for the same as mentioned above and successfully implemented. Concept of Dimming Effect, was successfully applied to change the intensity of the LEDs in patterns as specified in the objective.

Signature of Faculty member

Experiment 5

OBJECTIVE:

(i) To learn about basic Serial Communication functions of Arduino

- Serial.begin(9600);
- Serial.print();
- Serial.println()
- Serial.read()
- Serial.write()

(ii) WAP for following pattern using for loop:

****

Roll_No._____

****

Name:_____

****

Branch:_____

****

SOFTWARE USED: Arduino IDE

THEORY:

Serial.begin(9600) is used to initialize the serial communication and set the baud rate, which is the number of bits per second that are transmitted. In this case, the baud rate is set to 9600 bits per second.

Serial.print() is used to send data to the serial port as human-readable ASCII text. It can be used to send numbers, characters, and strings.

Serial.println() is similar to **Serial.print()** but adds a newline character at the end of the string being sent. This can be useful when printing data in multiple lines.

Serial.read() is used to receive data from the serial port. It returns the first byte of incoming serial data available.

Serial.write() is used to send raw binary data over the serial port. It can be used to send data more efficiently than using **Serial.print()** or **Serial.println()** since it doesn't require conversion to human-readable ASCII text.

We use the serial commands Serial.print() and Serial.println() to print the above code on Serial Monitor.

CODE:

```
void setup() {
Serial.begin(9600);
  for(int i=0;i<=44;i++)
  {
    Serial.print("*");
    }
  Serial.println();
  Serial.println("Roll No. : 102103350");
  for(int i=0;i<=31;i++){
    Serial.print("*");
    }
    Serial.println();
  Serial.println("Name : Mayank Aggarwal");
  for(int i=0;i<=41;i++){
    Serial.print("*");
    }
  Serial.print("\n");
  Serial.println("Branch : Computer Engineering");
  for(int i=0;i<=34;i++){
    Serial.print("*");
    }}
void loop() {
}
```

CIRCUIT DIAGRAM:

Fig 5.1 Output of Serial Monitor

CONCLUSION:

In this experiment, we learnt about commands of Arduino related to Serial Communication and executed a code to implement them.

Signature of Faculty member

Experiment 6

OBJECTIVE: Program for dimmer using digitalRead, analogRead

SOFTWARE: TinkerCad

HARDWARE: (within Tinkercad)

Table 6.1 Hardware for digitalRead, analogRead

NAME OF COMPONENT	QUANTITY
LEDs	1
Breadboard	1
Arduino UNO	1
Jumper Wire	4
Resistors	1

THEORY:

analogRead() reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input into integer values between 0 and 255.

digitalRead() reads the value from specified pin, mapping the input to 0 or 1.

CODE:

```
void setup()
{
   pinMode(6, OUTPUT);
}
void loop()
{
```

```
  //digitalWrite(6,LOW);
//digitalWrite(6.HIGH);
    delay(1000);
  int checkdigi=digitalRead(6);
    if(checkdigi==0){
  analogWrite(6, 255);
  delay(1000); // Wait for 1000 millisecond(s)
  int checkana=analogRead(6);
  analogWrite(6,checkana-150);
  delay(1000);
  }}
```

CIRCUIT DIAGRAM

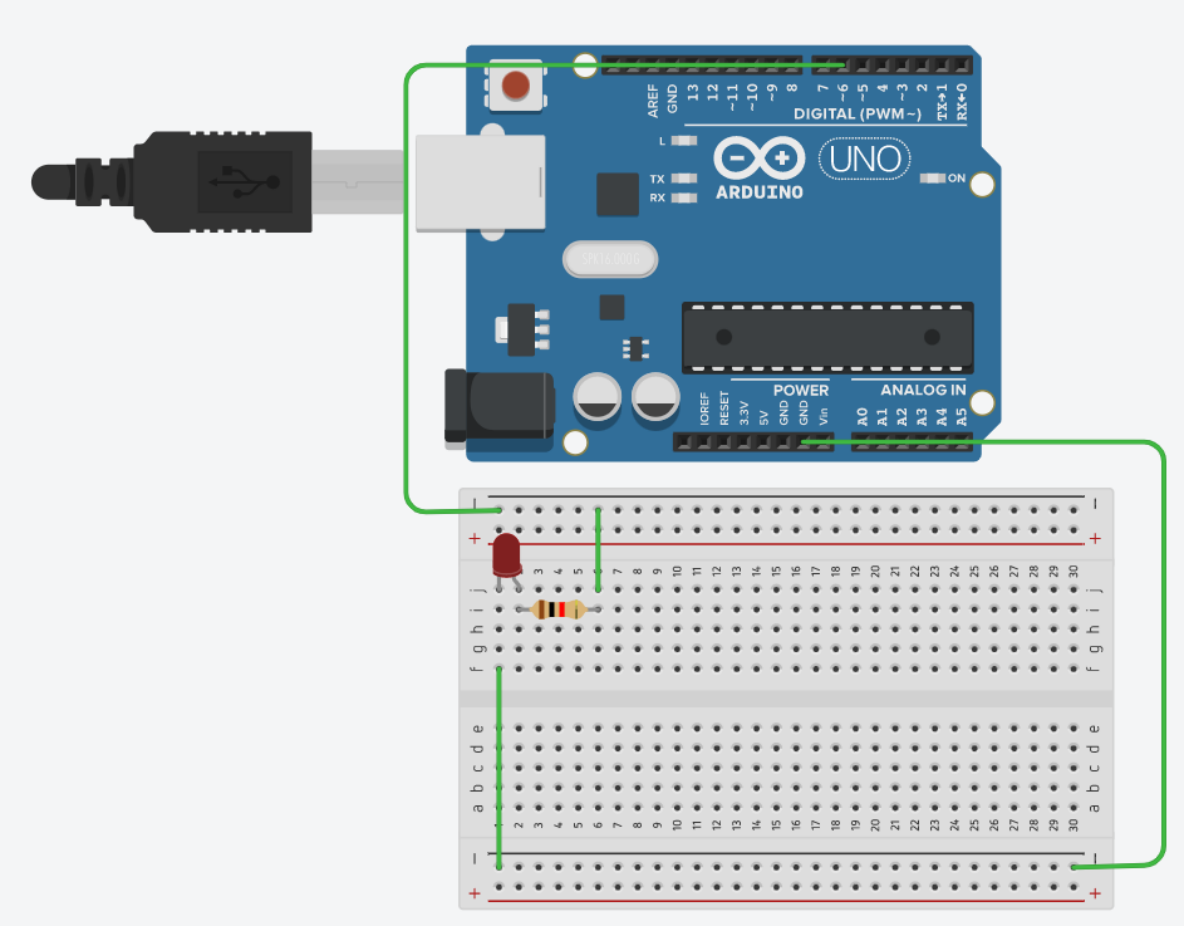


Fig 6.1 Tinkercad Circuit For dimming LED

CONCLUSION:

The functions digitalRead() and analogRead() were successfully used to store the values at pin 6 into variables and and if() statement was used to program LED to glow and dimmer if digitalWrite() was set to low.

Signature of Faculty member

Experiment 7

OBJECTIVE: Program to change intensity of given LEDs for sequence 35214 in forward and reverse direction

SOFTWARE: TinkerCad

HARDWARE: (within Tinkercad)

Table 7.1 Hardware for Intensity Change

NAME OF COMPONENT	QUANTITY
LEDs	**5**
Breadboard	**1**
Arduino UNO	**1**
Jumper Wire	**11**
Resistors	**5**

THEORY:

In this experiment, 5 LEDs were programmed to blink in pattern one by one starting from first till last LED and then in reverse pattern. Emphasis was laid on the usage of Arrays and for loop to first set pins to OUTPUT mode and then blink them one by one using for loop in loop() function, in the program to successively switch them ON/OFF, with a delay of 1000ms bring provided between each blink.

CODE:

```
int a[5]={10,9,6,5,3};
int r[5]={3,5,2,1,4};
void setup()
{
  for(int i=0;i<5;i++){
```

```
  pinMode(a[i], OUTPUT);


}}
void loop()
{
  for(int i=0;i<5;i++){
    analogWrite(a[r[i]-1],50);
      delay(1000);
  }
  for(int i=4;i>=0;i--){
    analogWrite(a[r[i]-1],255);
    delay(1000);
}}
```

CIRCUIT DIAGRAM

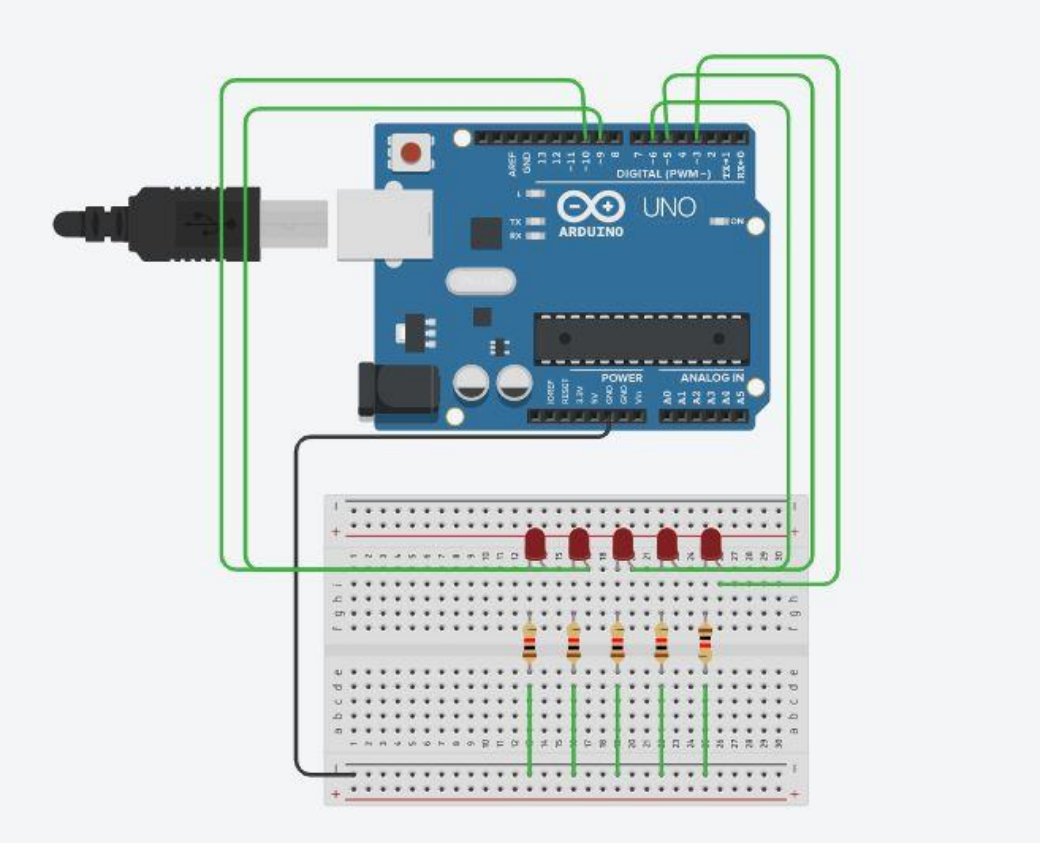


Fig 7.1 Tinkercad Circuit For dimming LED

CONCLUSION:

LEDs were successfully dimmed and glowed in 35412 and reverse pattern within tinkercad software. For loops were used to make an efficient and clean looking code.

Signature of Faculty member

Experiment 8

OBJECTIVE: To implement following actions on RoboCar using Arduino Programming:

1. Move in Forward Direction
2. Move in Backward Direction
3. Turn in left direction
4. Turn in right direction
5. Rotate in ClockWise Direction
6. Rotate in Anti-Clockwise Direction
7. Stop the RoboCar

SOFTWARE USED: Arduino IDE

COMPONENTS USED: Arduino IDE along with listed hardware.

Table 8.1 Hardware for implementing actions on RoboCar

NAME OF COMPONENT	QUANTITY
RoboCar Nvis 3302ARD	**3**
Arduino UNO	**1**

THEORY:

Nvis 3302ARD is capable of sensing the environment using various sensor modules and acts accordingly. Nvis RoboCar is a ready assembled unit consisting of strong chassis wheels with different Sensor modules mounted on it. The Nvis 3302ARD board is an Arduino-compatible microcontroller board that features a powerful 32-bit microcontroller, a range of digital and analog inputs and outputs, and support for serial communication and USB connectivity. The machine is driven by DC motors which are powered by rechargeable batteries. This Nvis 3302ARD is Atmega328P Microcontroller RoboCar, is designed for users to start developing smart robots which are capable of accelerometer balancing, and Gyroscope angular velocity sensing. There is Zigbee for wireless control of your smart RoboCar. The board can receive commands from an

external source, such as a computer, and use these commands to control the speed and direction of the motors, as well as the state of other components such as lights, sensors, and other actuators.

In this experiment, the vehicle's movement is controlled by setting the state of 4 digital output pins (5, 6, 7, and 8). The following movements are controlled:

"go_forward": sets pin 5 to HIGH and pin 8 to HIGH, which makes the vehicle move forward.

"go_backward": sets pin 6 to HIGH and pin 7 to HIGH, which makes the vehicle move backward.

“turn_left”: set pin 5 to HIGH and all others to LOW, which makes the vehicle turn left.

“turn_right”: set pin 8 to HIGH and all others to LOW, which makes the vehicle to turn left

"move_clockwise": sets pin 5,7 to HIGH and pin 6,8 to LOW, which makes the vehicle turn clockwise.

"move_anti_clockwise": sets pin 6,8 to HIGH and pin 5,7 to LOW, which makes the vehicle turn counterclockwise.

"do_stop": sets all pins to LOW, which makes the vehicle stop moving.

PIN CONFIGURATION TABLE:

Table 8.2 Pin Configuration for RoboCar

Configuration Table of Pins:

J13-Phonex Pin Number	Port pin	Arduino Pin	Description
1	PD5	Digital 5	Right Motor Plus
2	PD6	Digital 6	Right Motor Minus
3	PD7	Digital 7	Left Motor Plus
4	PB0	Digital 8	Left Motor Minus

CODE:

```
void setup()
{
  Serial.begin(9600);
  for(int i=5;i<=8;i++){
  pinMode(i, OUTPUT);
  }}
void go_forward(){
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);}
void go_backward(){
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);}
void turn_left(){
    digitalWrite(5,HIGH);
    digitalWrite(6,LOW);
    digitalWrite(7,LOW);
    digitalWrite(8,LOW);
    delay(1200);
    do_stop();
    go_forward();
    delay(1500);
    do_stop();}
void turn_right(){
    digitalWrite(8,HIGH);
```

```
    digitalWrite(7,LOW);
    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
    delay(1200);
    do_stop();
    go_forward();
    delay(1500);
    do_stop();}
void move_clockwise(){
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);}
void move_anti_clockwise(){
  digitalWrite(8,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH);
  digitalWrite(7,LOW);}
void do_stop(){
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);}
void loop() {
  go_forward();
  delay(2500);
  do_stop();
  delay(1000);
  move_clockwise();
  delay(2500);
```

```
  do_stop();
  delay(1000);
  go_backward();
  delay(2500);
  do_stop();
  delay(1000);
  move_anti_clockwise();
  delay(2500);
  do_stop();
}
```

CIRCUIT DIAGRAM:

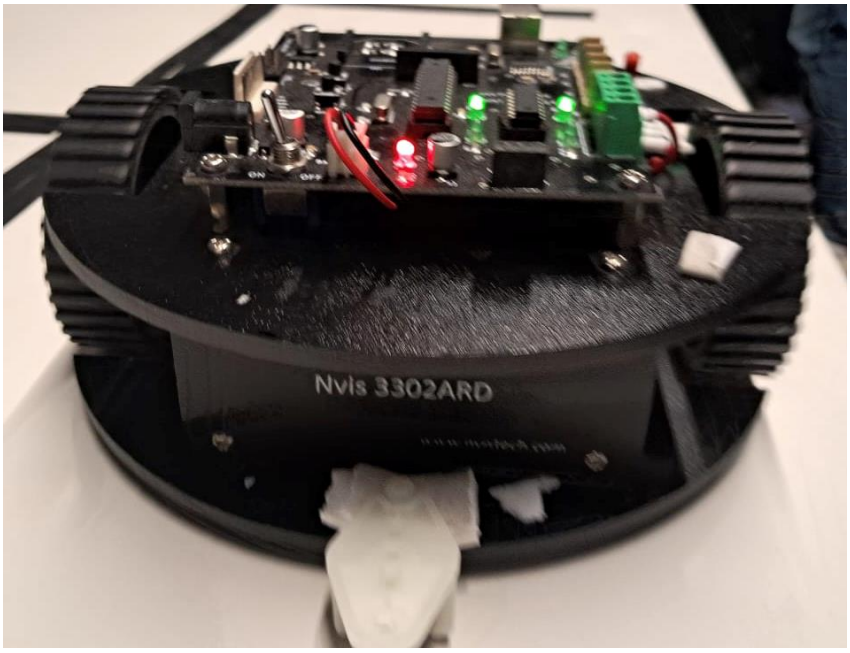


Fig 8.1 Buggy Nvis 3302ARD

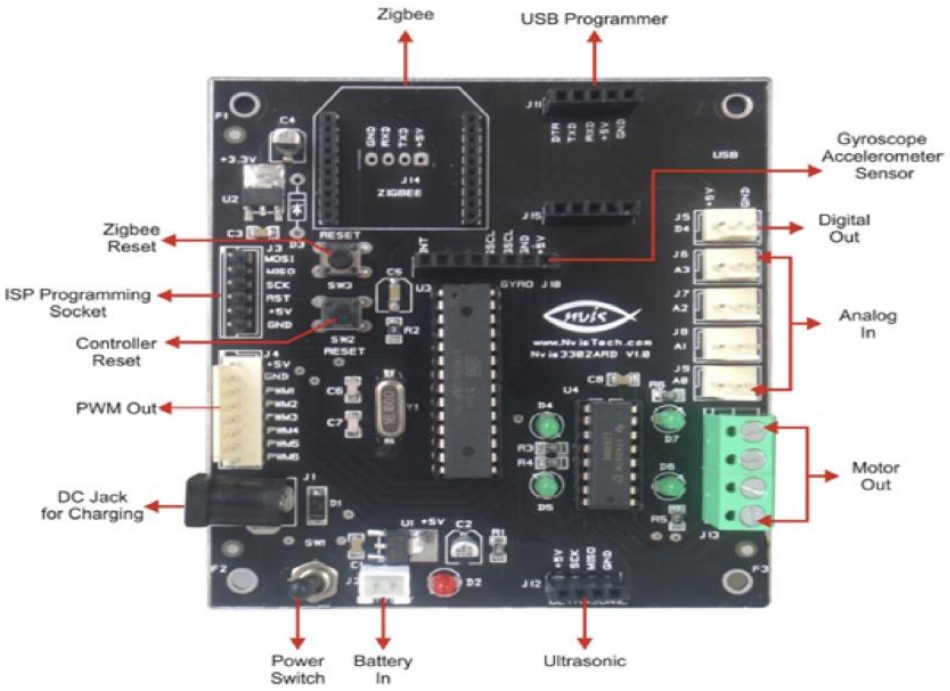


Fig 8.2 Board Configuration and part name

CONCLUSION:

In this experiment, we successfully programmed RoboCar to perform basic functions that are Forward, Backward, Left, Right, Stop, Clockwise and Anti-Clockwise directions. We learned how to implement code in RoboCar and learned the basics of functioning and circuit of RoboCar.

.

Signature of Faculty member

Experiment 9

OBJECTIVE: To demonstrate the use of ultrasonic sensor by integrating line follower RoboCar with obstacle avoidance capability

SOFTWARE USED: Arduino IDE

COMPONENTS USED: Arduino IDE along with listed hardware.

Table 9.1 Hardware to demonstrate ultrasonic sensor

NAME OF COMPONENT	QUANTITY
Ultrasonic sensor	**1**
Breadboard	**1**
Arduino UNO	**1**
Jumper Wire	**4**

THEORY:

The theory behind integrating an ultrasonic sensor with a line follower robot car to add obstacle avoidance capabilities involves using the ultrasonic sensor to measure the distance to nearby obstacles and then using this information to adjust the robot's movement accordingly.

The ultrasonic sensor works by emitting a high-frequency sound wave and then measuring the time it takes for the sound wave to bounce back after hitting an object. Based on this time, the sensor can calculate the distance to the object. The ultrasonic sensor is capable of detecting obstacles within a certain range, typically between 2 and 400 centimeters.

The line following robot, on the other hand, uses sensors to detect a line on the ground and adjust its movement accordingly to stay on the line.

By integrating the ultrasonic sensor with the line following robot, the robot can use the distance readings from the ultrasonic sensor to determine if it is about to collide with an obstacle. If an obstacle is detected, the robot can adjust its direction to avoid the obstacle while still attempting to stay on the line.

The implementation of this integration involves writing a program for the microcontroller that controls the robot. The program should implement an obstacle avoidance algorithm that uses the distance readings from the ultrasonic sensor to determine the presence and location of obstacles. Based on this information, the algorithm should adjust the robot's movement to avoid the obstacle while still attempting to stay on the line.

CODE:

```
int trigPin=11;
int echoPin=9;
int Time;
float Distance;
void setup()
{
  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(trigPin,HIGH);
  delay(125);
  digitalWrite(trigPin,LOW);
  Time=pulseIn(echoPin,HIGH);
  Distance=0.0185*(Time);
  Serial.println(Distance);

}
```

CIRCUIT DIAGRAM:

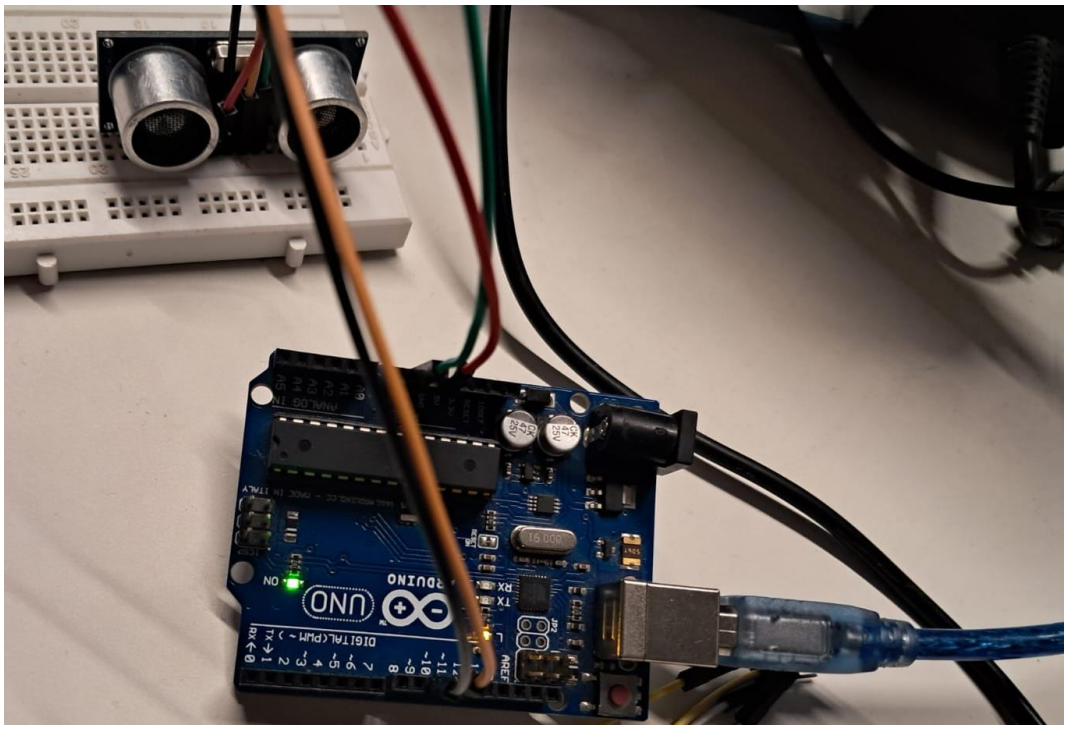


Fig 9.1 Circuit for UltraSonic Sensor

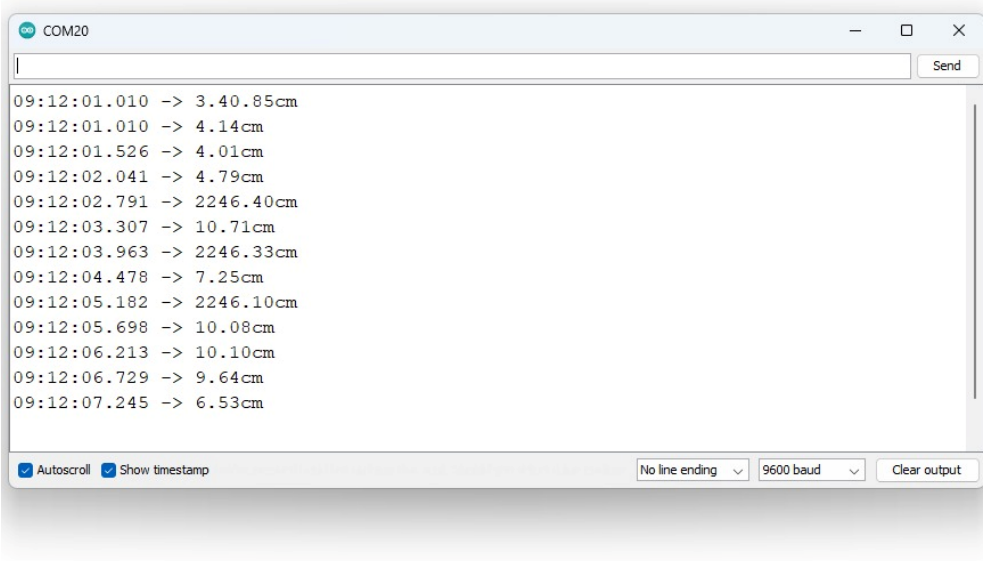


Fig 9.2 Output of Distance Measurement from Ultrasonic Sensor

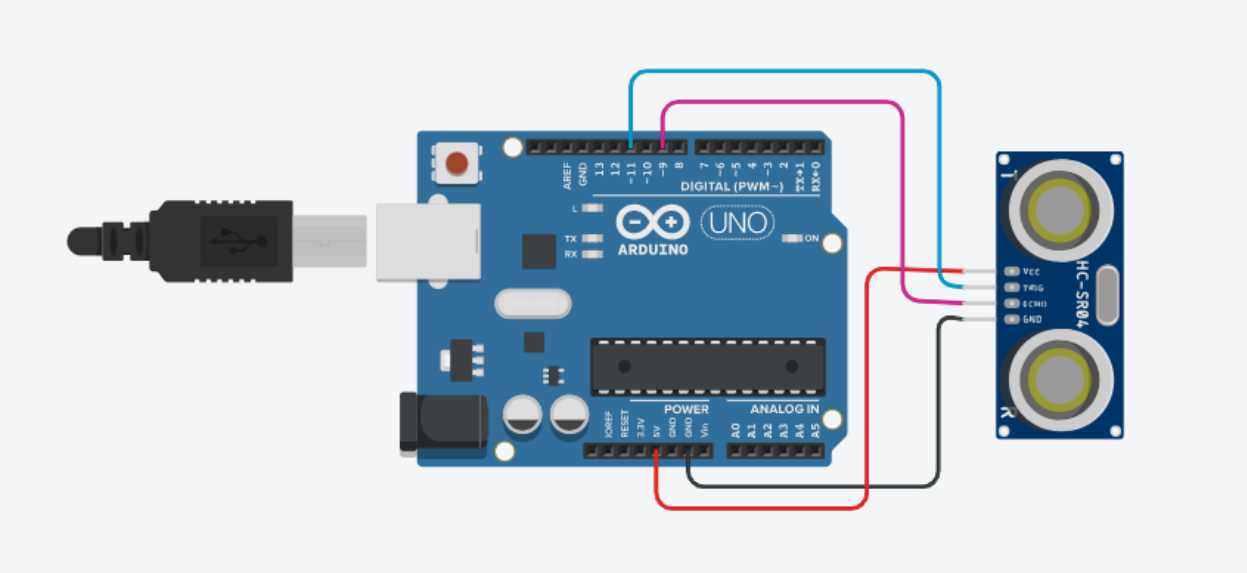


Fig 9.3 TinkerCad Circuit for UltraSonic Sensor

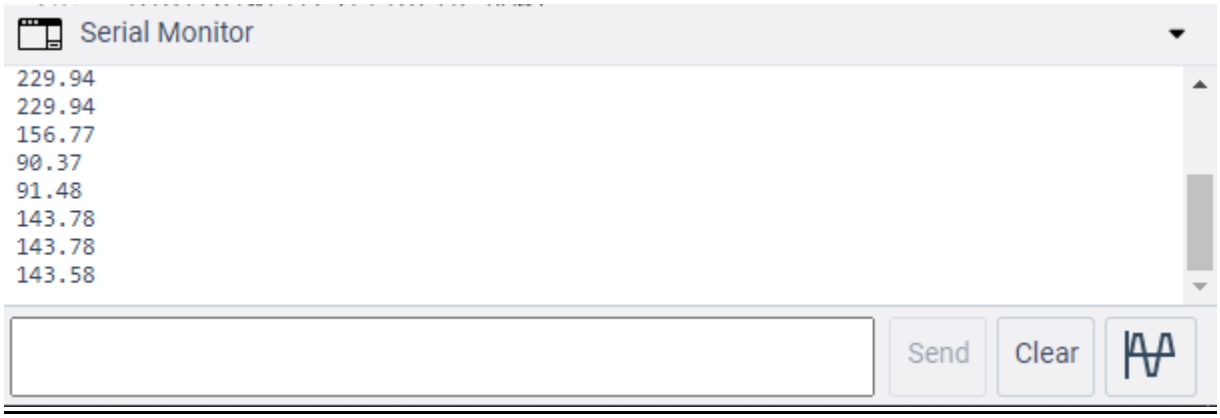


Fig 9.4 Output of TinkerCad Ultrasonic Sensor

CONCLUSION:

The use of an ultrasonic sensor was successfully demonstrated. The sensor was programmed to trigger 8 ultrasonic pulses in 1 second and the resulting 'HIGH' of echoPin was stored in a variable via pulseIn function, giving us twice the time required for the pulse to travel from source to target. Distance was calculated using d=vt/2 formula and displayed using serial.printIn function within Arduino IDE.

Signature of Faculty member