EST Practical Activity Report

Submitted for

Engineering Design Project-II (UTA024)

Submitted by ***Group 5***

Rehan Bansal	**102203429**
Yashvi Kumar	**102203430**
Kunwar Kataria	**102203431**
Nandini Jain	**102203436**
Geetansh Mohindru	**102203718**

BE Second Year Batch – 2CO10

Submitted to:

Mr. Raghuveer Singh Dhaka

Computer Science and Engineering Department

Thapar Institute of Engineering & Technology, Patiala

Jan-May 2024

DECLARATION

We declare that this project report is based on our own work carried out during the course of our study in our Engineering-design II Computer Lab under the supervision of Mr. Raghuveer Singh Dhaka.

We assert that the statements made and conclusions drawn are an outcome of our own research work.

We further certify that the work contained in this report is original and has been done by us under the general supervision of our supervisor.

We have followed the guidelines provided by the University in writing this report.

We also declare that this project is the outcome of our own effort, that it has not been submitted to any other university for the award of any degree.

INDEX

EXPERIMENT-1

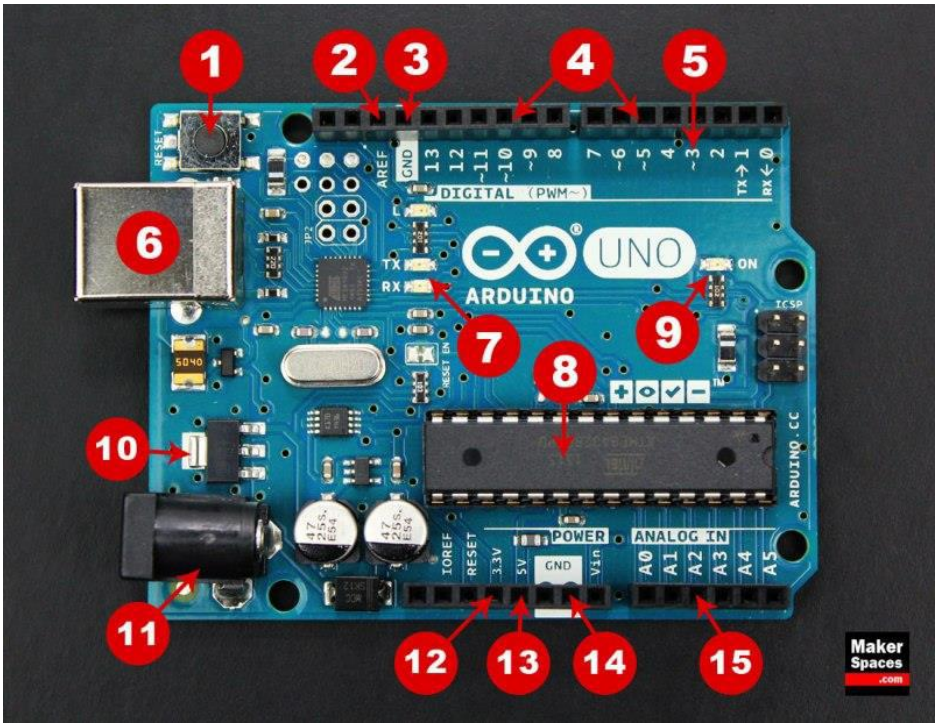
OBJECTIVE: Introduction to Arduino Micro-Controller.

SOFTWARE USED: TinkerCAD Simulator

HARDWARE USED:

Sr. No	Name of Components	Value
1	Arduino Uno Micro-Controller	**1**

LOGIC/CIRCUIT DIAGRAM:



THEORY:

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc and initially released in 2010.

The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts.

1. **Reset Button** – This will restart any code that is loaded to the Arduino board.

2. **AREF** – Stands for “Analog Reference” and is used to set an external reference voltage.

3. **Ground Pin** – There are a few ground pins on the Arduino and they all work the same.

4. **Digital Input/Output** – Pins 0-13 can be used for digital input or output.

5. **PWM** – The pins marked with the (~) symbol can simulate analog output.

6. **USB Connection** – Used for powering up your Arduino and uploading sketches.

7. **TX/RX** – Transmit and receive data indication LEDs.

8. **ATmega Microcontroller** – This is the brains and is where the programs are stored
9. **Power LED Indicator** – This LED lights up anytime the board is plugged in a power source.

10. **Voltage Regulator** – This controls the amount of voltage going into the Arduino board.

11. **DC Power Barrel Jack** – This is used for powering your Arduino with a power supply.

12. **3.3V Pin** – This pin supplies 3.3 volts of power to your projects.

13. **5V Pin** – This pin supplies 5 volts of power to your projects.

14. **Ground Pins** – There are a few ground pins on the Arduino and they all work the same.

15. **Analog Pins** – These pins can read the signal from an analog sensor and convert it to digital.

DISCUSSIONS

In this experiment, we get to know about basics of Arduino Uno Microcontroller and its various functions and components.

Signature of faculty member

EXPERIMENT-2

OBJECTIVE: Write a program to blink a single LED using Arduino and breadboard.

SOFTWARE USED: Tinkercad Simulator.

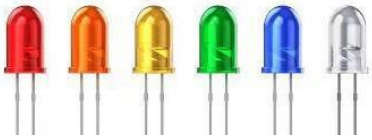
HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1
2.	Breadboard	1
3.	Jumper Wires	2
4.	LED	1
5.	Resistor	220 ohm

THEORY:

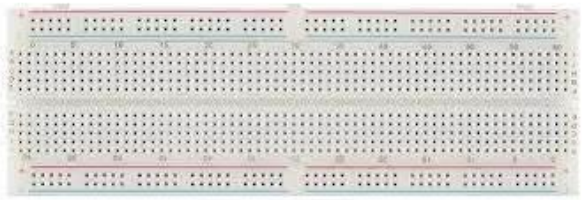
Resistor: Resistors are used in virtually all electronic circuits and many electrical ones. Resistors, as their name indicates resist the flow of electricity and this function is key to the operation most circuits.

LED: A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons (Energy packets).



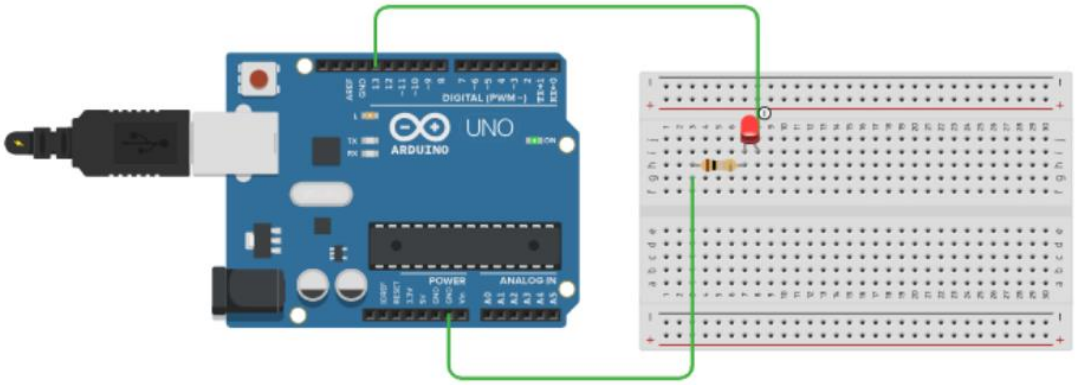
Arduino Uno Board: The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc.

Breadboard: A breadboard is used to place components (resistor, capacitor, LED's etc.) that are wired together. It is used to make temporary circuits.



Jumper Wires: A jumper wire is an electric wire that connects remote electric circuits used for printed circuit boards.

TINKERCAD DIAGRAM:

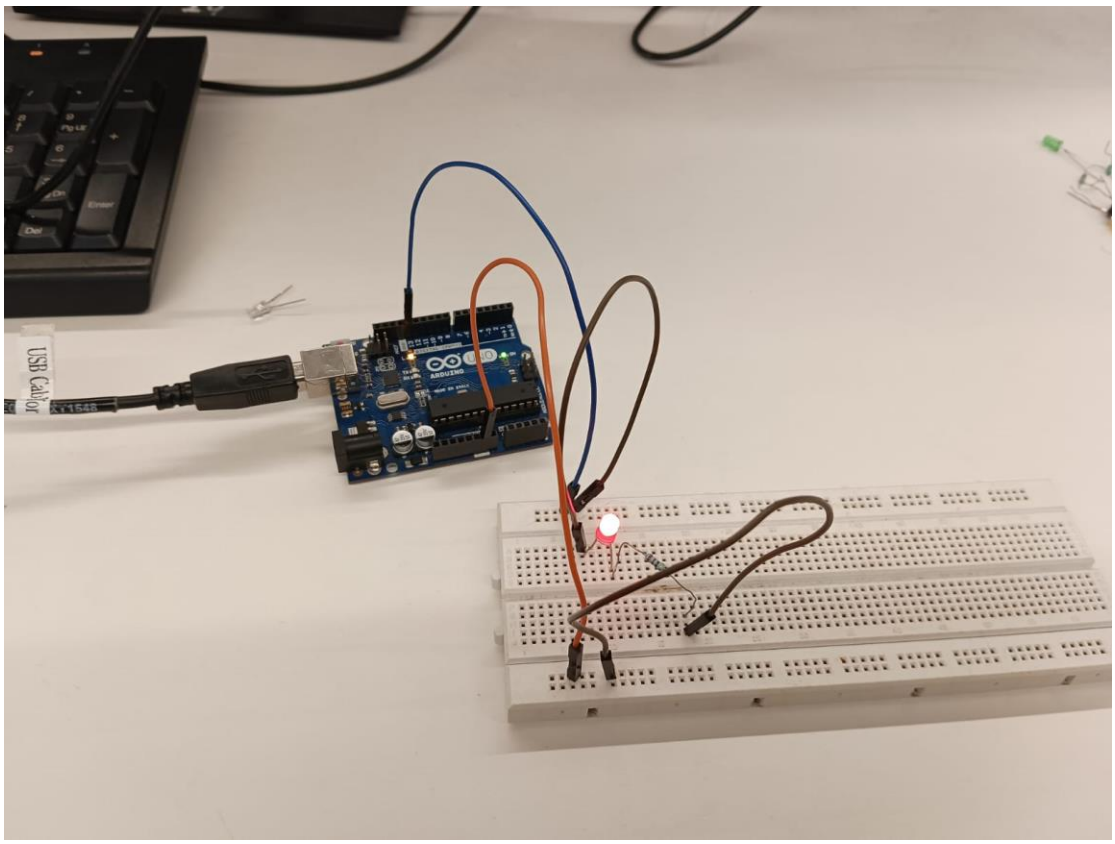


[TinkerCAD Link]

CODE:

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```

RESULTS:
In this experiment, we learnt how to blink an LED using Arduino UNO.



Signature of faculty member

EXPERIMENT-3

OBJECTIVE: To blink multiple LEDs using Arduino Uno and breadboard.

SOFTWARE USED: Tinkercad Simulator.

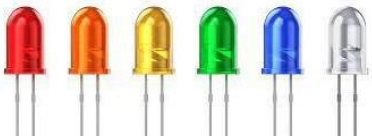
HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1
2.	Breadboard	1
3.	Jumper Wires	9
4.	LED	5
5.	Resistor	220 ohm

THEORY:

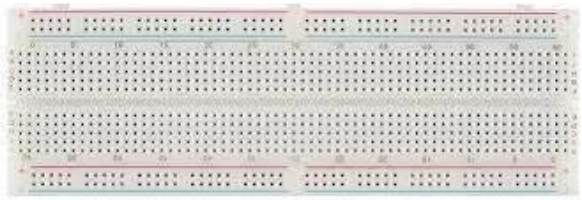
Resistor: Resistors are used in virtually all electronic circuits and many electrical ones. Resistors, as their name indicates resist the flow of electricity and this function is key to the operation most circuits.

LED: A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons (Energy packets).



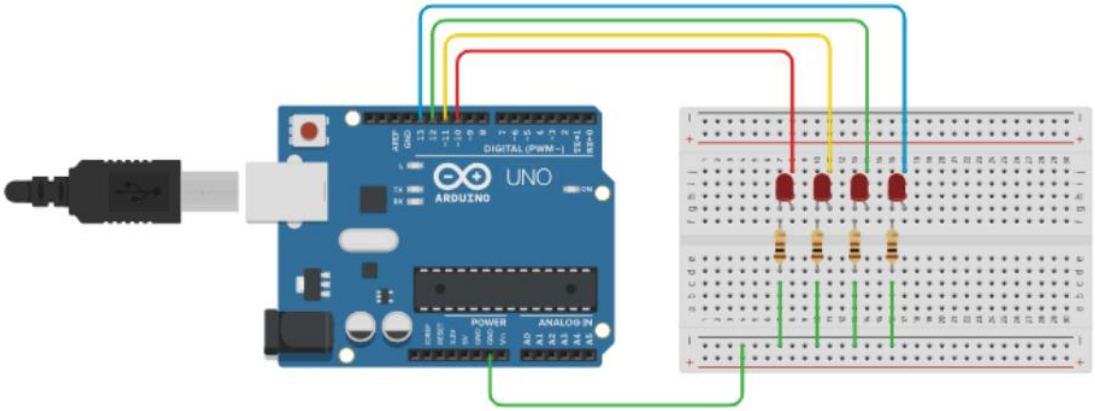
Arduino Uno Board: The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc.

Breadboard: A breadboard is used to place components (resistor, capacitor, LED's etc.) that are wired together. It is used to make temporary circuits.



Jumper Wires: A jumper wire is an electric wire that connects remote electric circuits used for printed circuit boards.

TINKERCAD DIAGRAM:



[TinkerCAD Link]

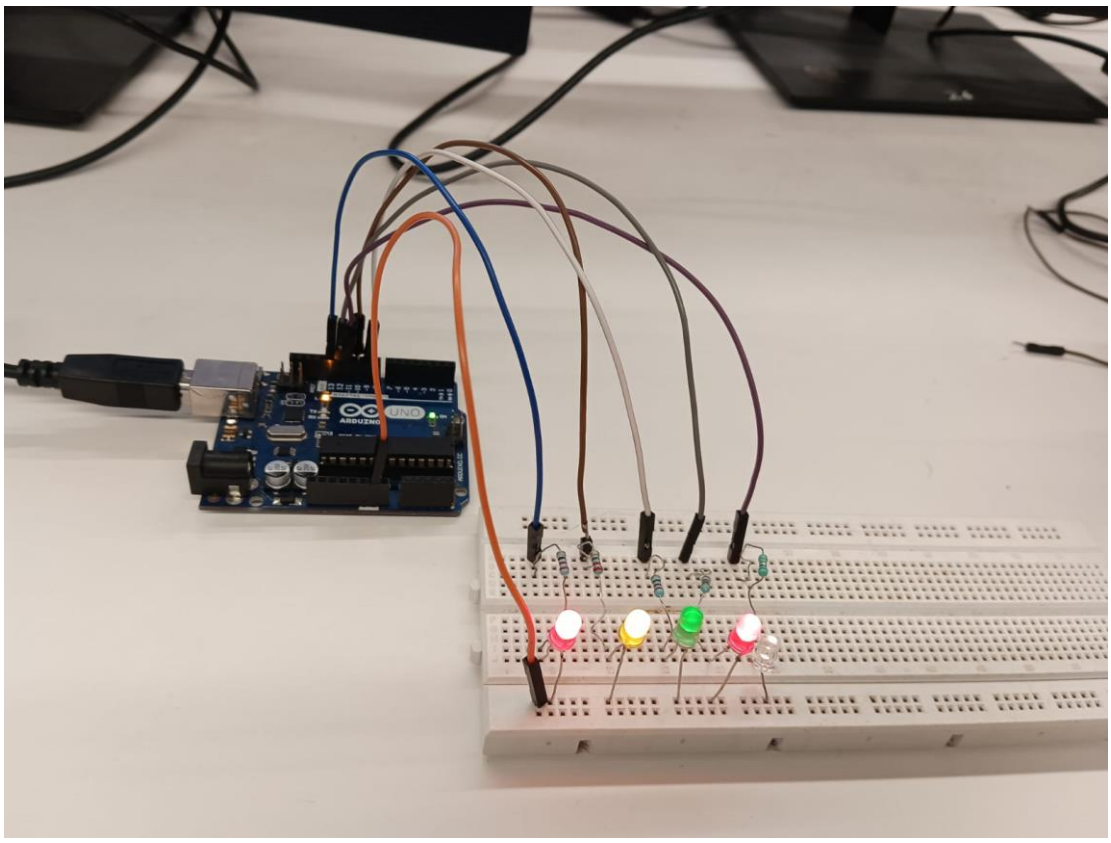
CODE :

```
//Using for Loop
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
```

```
  for(int i=9; i<14; i++)
  {
    digitalWrite(i, HIGH);
     delay(500);
     digitalWrite(i, LOW);
     delay(500);
  }
}
```

RESULTS:

Hence, we successfully blinked multiple LEDs using the above components by making suitable connections and coded using for loop.



Signature of faculty member

EXPERIMENT-4

OBJECTIVE: Write a program to design a pattern of sequence of multiple LED's using for loop in Arduino.

SOFTWARE USED: Tinkercad Simulator.

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1
2.	Breadboard	1
3.	Jumper Wires	9
4.	LED	5
5.	Resistor	220 ohm

THEORY:

In this experiment we use the concept of array,for ,loops and conditonal statements of arduino programming along with digitalWrite() function to design a pattern of sequence of multiple LED's

ARRAY: An array is a collection of variables that are accessed with an index number. Arrays in the C++ programming language Arduino sketches are written in can be complicated, but using simple arrays is relatively straightforward. **Arrays are zero indexed, that is, referring to the array initialization above, the first element of the array is at index 0.**

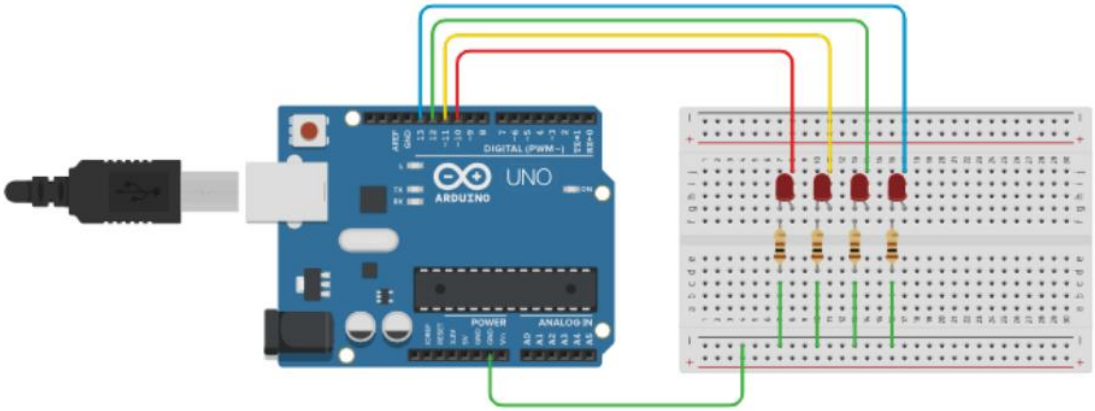
FOR LOOP: The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

CONDITIONAL STATEMENT: The **if()** statement is the most basic of all programming control structures. It allows you to make something happen or not, depending on whether a given condition is true or not.

DigitalWrite(): Write a **HIGH** or a **LOW** value to a digital pin. If the pin has been configured as an **OUTPUT** with **pinMode()**, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for **HIGH**, 0V (ground) for **LOW**.If the pin is configured as an **INPUT**, **digitalWrite()** will enable (**HIGH**) or disable (**LOW**) the internal pullup on the input pin. It is recommended to set the **pinMode()** to **INPUT_PULLUP** to enable the internal pull-up resistor. .If you do not set the **pinMode()** to **OUTPUT**, and connect an LED to a pin, when calling **digitalWrite(HIGH)**, the LED may appear dim. Without explicitly setting **pinMode()**, **digitalWrite()** will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

DELAY(): The delay() function allows you to pause the application of your arduino program fro a specific period.

TINKERCAD DIAGRAM:



TinkerCAD Link

CODE 1:

```
//Using for Loop
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  for(int i=9; i<14; i++)
  {
    digitalWrite(i, HIGH);
      delay(500);
      digitalWrite(i, LOW);
      delay(500);
  }
}
```

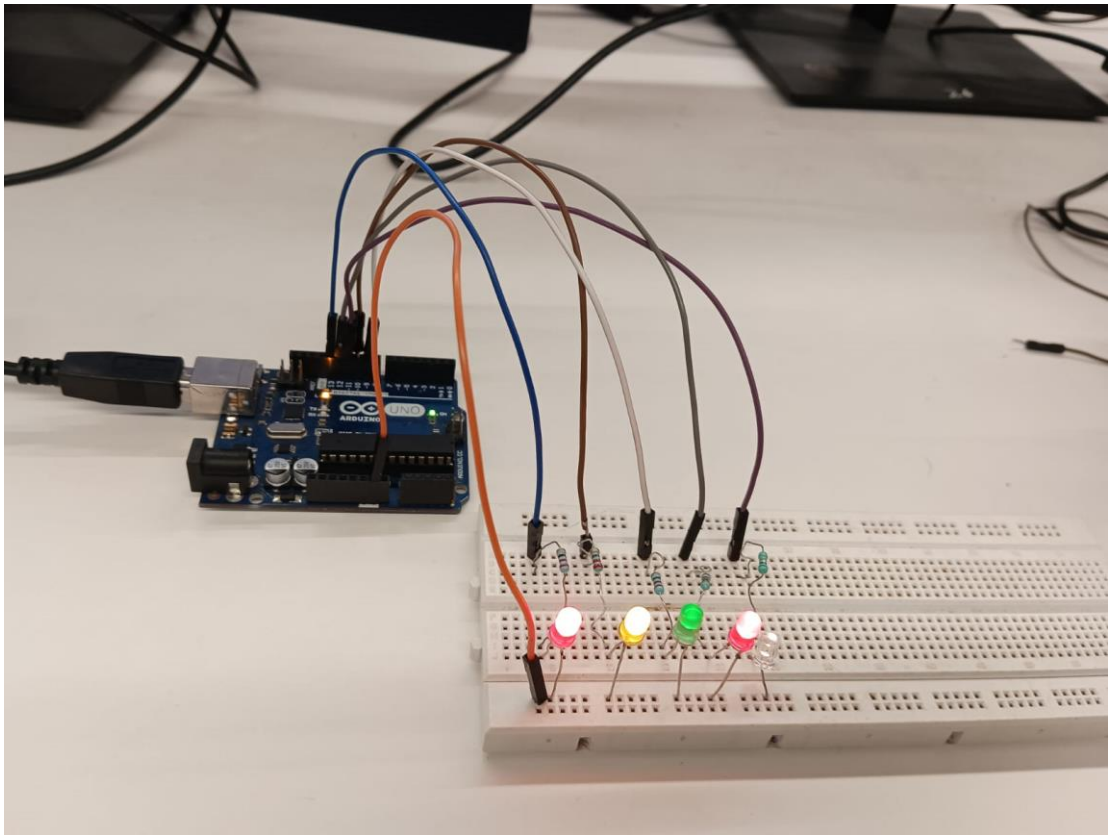
CODE 2:

```
//Using while Loop
void setup()
{
```

```
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  for(int i=9; i<14; i++)
  {
    digitalWrite(i, HIGH);
     delay(500);
     digitalWrite(i, LOW);
     delay(500);
  }
}
```

RESULTS:

In this experiment we made a pattern where the LEDs which were placed at odd positions lit up first then the LEDs which were placed at even positions lit up. They were dimmed in the reverse order.



Signature of faculty member

EXPERIMENT-5

OBJECTIVE: Write a program to demonstrate sending data from the computer to the Arduino board and control brightness of LED.

SOFTWARE USED: Tinkercad Simulator.

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1
2.	Breadboard	1
3.	Jumper Wires	3
4.	LED	1
5.	Resistor	220 ohm

THEORY:
In this experiment we use the concept of array,for loops and conditonal statements of arduino programming along with digitalWrite() AND analogWrite() function to demonstrate sending data from the computer to the Arduino board and control brightness of LED.

ARRAY: An array is a collection of variables that are accessed with an index number. Arrays in the C++ programming language Arduino sketches are written in can be complicated, but using simple arrays is relatively straightforward**. Arrays are zero indexed, that is, referring to the array initialization above, the first element of the array is at index 0.**

SIZEOF(): It returns the size of the given parameter in bytes.

FOR LOOP: The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

CONDITIONAL STATEMENT: The **if()** statement is the most basic of all programming control structures. It allows you to make something happen or not, depending on whether a given condition is true or not.

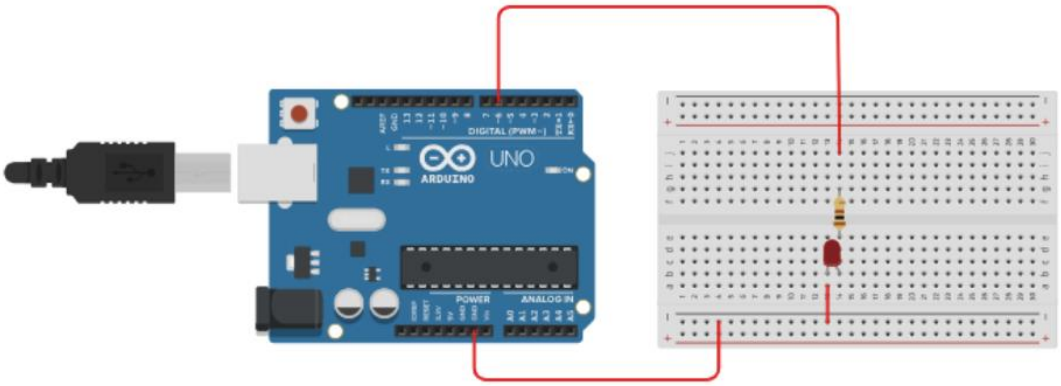
DigitalWrite(): Write a **HIGH** or a **LOW** value to a digital pin. If the pin has been configured as an **OUTPUT** with **pinMode()**, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for **HIGH**, 0V (ground) for **LOW**.If the pin is configured as an **INPUT**, **digitalWrite()** will enable (**HIGH**) or disable (**LOW**) the internal pullup on the input pin. It is recommended to set the **pinMode()** to **INPUT_PULLUP** to enable the internal pull-up resistor. .If you do not set the **pinMode()** to **OUTPUT**, and connect an LED to a pin, when calling **digitalWrite(HIGH)**, the LED

may appear dim. Without explicitly setting **pinMode()**, **digitalWrite()** will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

DELAY(): The delay() function allows you to pause the application of your arduino program fro a specific period.

analogWrite(): Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() on the same pin.

TINKERCAD DIAGRAM:



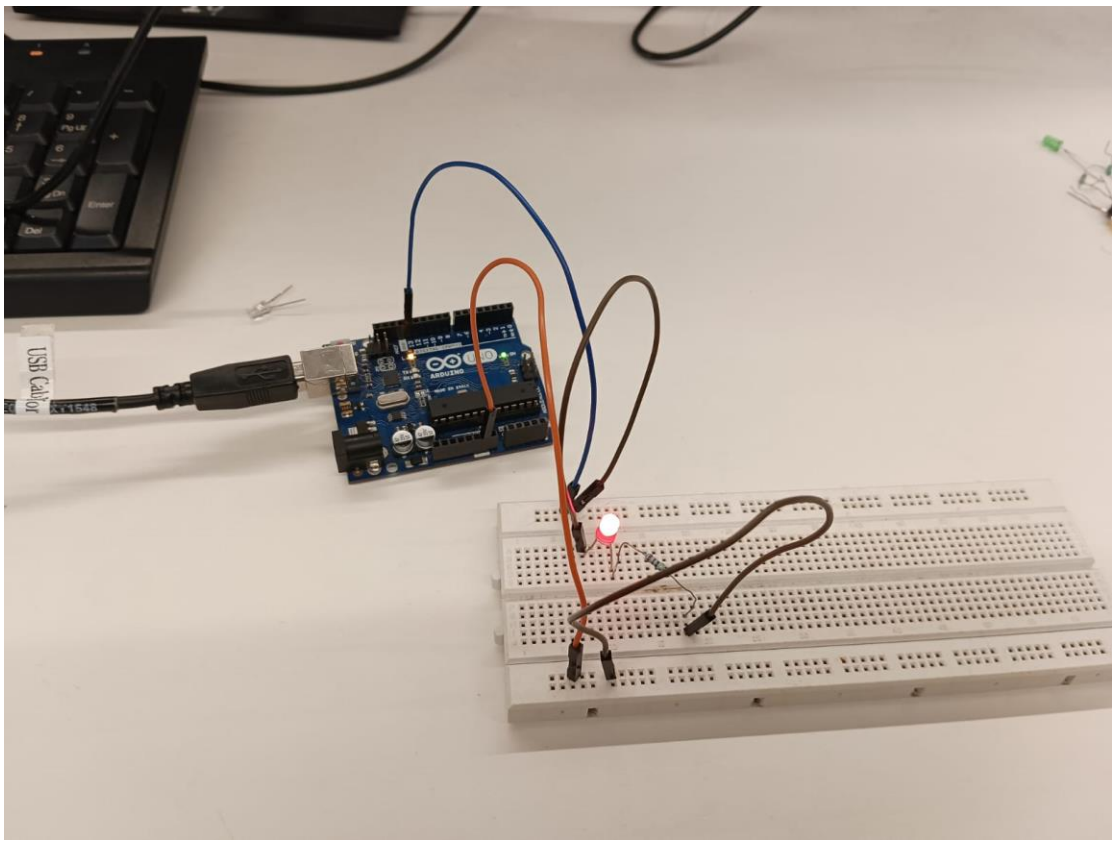
TinkerCAD Link

CODE:

```
int a;
void setup ()
{
  Serial.begin(9600);
  pinMode(6, OUTPUT);
}
void loop ()
{
  Serial.println("Enter Intensity: ");
  a= Serial.parseInt(); analogWrite(6,a);
  delay(500);
  analogWrite(6,0);
  delay(500);
}
```

RESULTS:

In this experiment we successfully learned how to control the brightness of LED using 'analogWrite' function. We also got to know that the brightness can range between 0 (always off) and 255 (always on). If we set the value above 255 then the result we get is Value%255.



Signature of faculty member

EXPERIMENT-6

OBJECTIVE: *Serial Communication:*
WAP to print following pattern using for loop.
****
Roll_No._____
****
Name:_____
****
Branch:_____
****
SOFTWARE USED: Arduino IDE

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1

THEORY:
Serial Communication: Serial is used for communication between the Arduino board and a computer or other devices. Serial data transfer is when we transfer data one bit at a time, one right after the other. When you upload the data to the Arduino, the bits are shoved out one at a time through the USB cable to the Arduino where they are stored in the main chip.

1. Serial.print();
The serial.print ( ) in Arduino prints the data to the serial port. The printed data will be visible in the serial monitor.

2. Serial.read();
Serial.read() is a function of the Serial library. What it does is read out the first available byte
from the serial receive buffer. Say you had sent the phrase “Sub Sandwich” to your Arduino.
This means you had put 12 bytes into your serial receive buffer.

3. Serial.write();
Writes binary data to the serial port. This data is sent as a byte or series of bytes. Let's say you need to send the number 217. The binary (1's and 0's) representation of this number is 11011001. Using the command Serial.write(217) will literally just send 11011001 across the line.

TinkerCAD Link

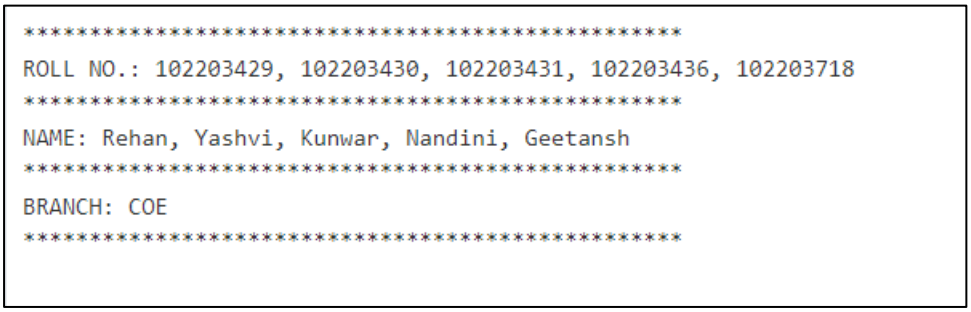
CODE:

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

  for(int i=0; i<50; i++)
  {
    Serial.print("*");
  }
  Serial.println(" ");
```

```
  Serial.println("ROLL NO.: 102203429, 102203430, 102203431,
102203436, 102203718");

  for(int i=0; i<50; i++)
  {
    Serial.print("*");
  }
  Serial.println(" ");

  Serial.println("NAME: Rehan, Yashvi, Kunwar, Nandini,
Geetansh");

  for(int i=0; i<50; i++)
  {
    Serial.print("*");
  }
  Serial.println(" ");

  Serial.println("BRANCH: COE");
   for(int i=0; i<50; i++)
  {
    Serial.print("*");
  }
}

void loop()
{
  //sample code
}
```

RESULT:

We used the 'serial.println' function and for loops to print the pattern to the serial monitor



Signature of faculty member

EXPERIMENT-7

OBJECTIVE: WAP to show dimmer effect where LED 1 should display values between 1-50
LED 2 = 51-100
LED 3 = 101-150
LED 4 = 151-200
LED 5 = 201-255
SOFTWARE USED: Tinkercad Simulator

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1
2.	Breadboard	1
3.	Jumper Wires	11
4.	LED	5
5.	Resistor	220 ohm

THEORY:

In this experiment we use the concept of array,for loops and conditonal statements of arduino programming along with digitalWrite() AND analogWrite() function to show dimmer effect where different LED will display different values between 1-255.

ARRAY: An array is a collection of variables that are accessed with an index number. Arrays in the C++ programming language Arduino sketches are written in can be complicated, but using simple arrays is relatively straightforward. Arrays are zero indexed, that is, referring to the array initialization above, the first element of the array is at index 0.

SIZEOF(): It returns the size of the given parameter in bytes.

FOR LOOP: The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

CONDITIONAL STATEMENT: The **if()** statement is the most basic of all programming control structures. It allows you to make something happen or not, depending on whether a given condition is true or not.

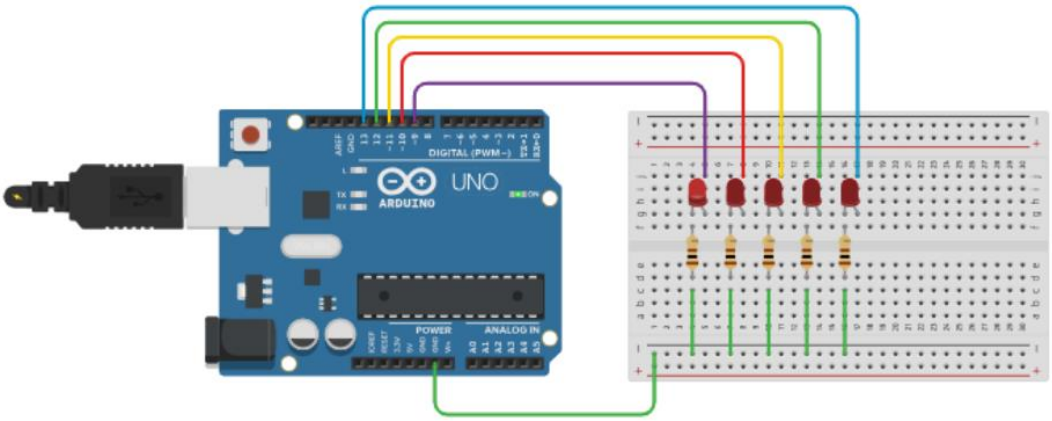
DigitalWrite(): Write a **HIGH** or a **LOW** value to a digital pin.If the pin has been configured as an **OUTPUT** with **pinMode()**, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for **HIGH**, 0V (ground) for **LOW**. If the pin is configured as an **INPUT**, **digitalWrite()** will enable (**HIGH**) or disable (**LOW**) the internal pullup on the input pin. It is recommended to set the **pinMode()** to

INPUT_PULLUP to enable the internal pull-up resistor. .If you do not set the **pinMode()** to **OUTPUT**, and connect an LED to a pin, when calling **digitalWrite(HIGH)**, the LED may appear dim. Without explicitly setting **pinMode()**, **digitalWrite()** will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

DELAY: The delay() function allows you to pause the application of your arduino program fro a specific period.

analogWrite(): Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() on the same pin. Arduino Uno R3 has 6 PWM pins that are **3, 5, 6, 9, 10, and 11**. These pins are marked with the negation sign "~". These pins can generate a pulse as per the given inputs. Arduino supports an 8-bit wide pulse that can have 256 possible levels (0 to 255).
PULSE WIDTH MODULATION (PWM):Pulse Width Modulation is a technique to get variable voltage in terms of Digital Input. PWM device generates ON and OFF pulses according to the control signal, which determines the desired voltage level. PWM is used to control the amount of power delivered to the load. It is commonly used for controlling the brightness of LED, the speed of motors, etc.

TINKERCAD DIAGRAM:



TinkerCAD Link

CODE:

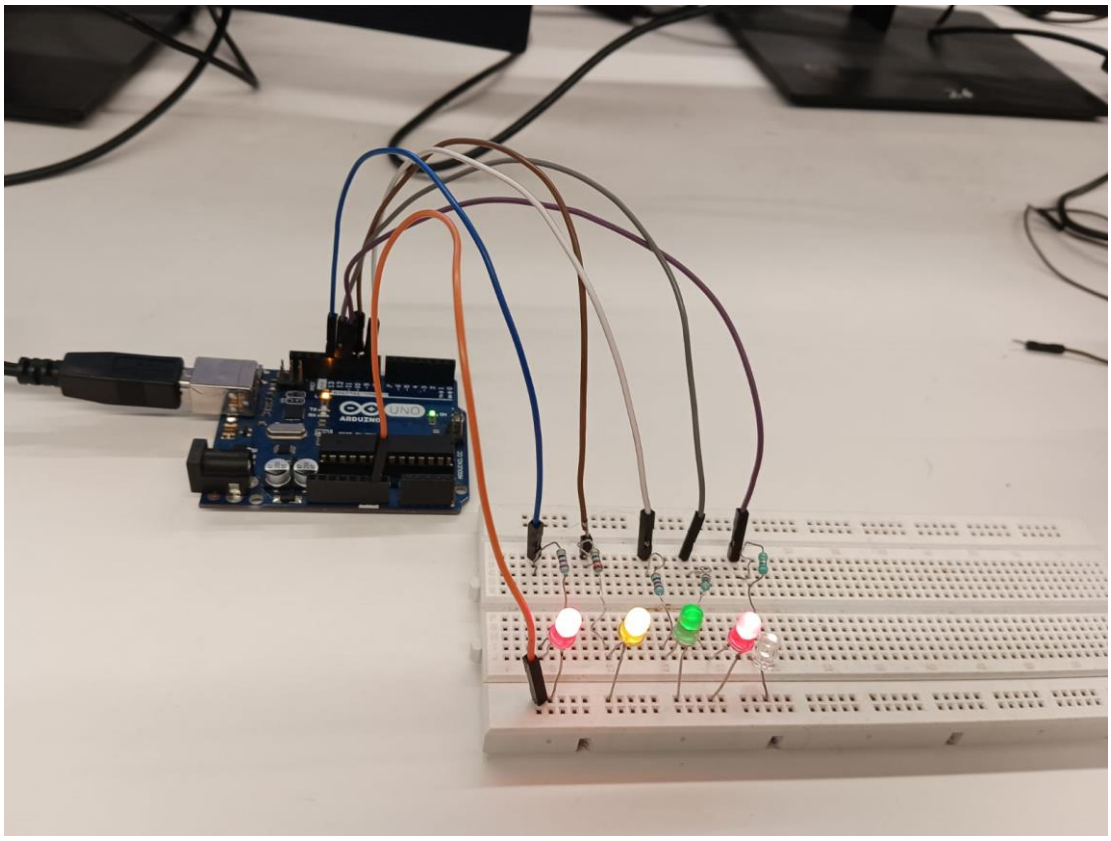
```
void setup()
{
  pinMode(9, OUTPUT);
```

```
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}
void loop()
{
  for(int i=0;i<=50;i++)
  {
    analogWrite(9,i);
    delay(200);
  }
  for(int i=51;i<=100;i++)
  {
    analogWrite(10,i);
    delay(200);
  }
  for(int i=101;i<=150;i++)
  {
    analogWrite(11,i);
    delay(200);
  }
  for(int i=151;i<=200;i++)
  {
    analogWrite(12,i);
    delay(200);
  }
  for(int i=201;i<=255;i++)
  {
```

```
    analogWrite(13,i);
    delay(200);
  }
}
```

RESULT:

We executed dimmer effect by using for loops for each LED and then used ‘analogWrite’ function to manage the brightness according to the question.



Signature of faculty member

EXPERIMENT-8

OBJECTIVE: Write a program to change the intensity of the given LED's for the sequence 35214 in for both forward and reverse order.

SOFTWARE USED: Tinkercad Simulator

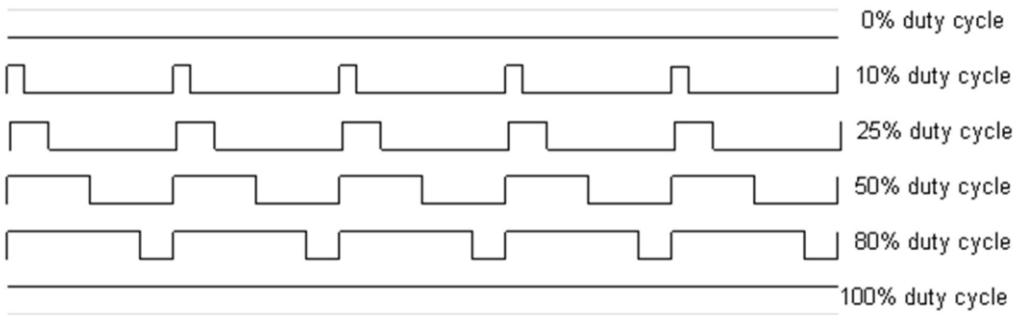
HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Arduino Uno Board	1
2.	Breadboard	1
3.	Jumper Wires	11
4.	LED	5
5.	Resistor	220 ohm

THEORY:

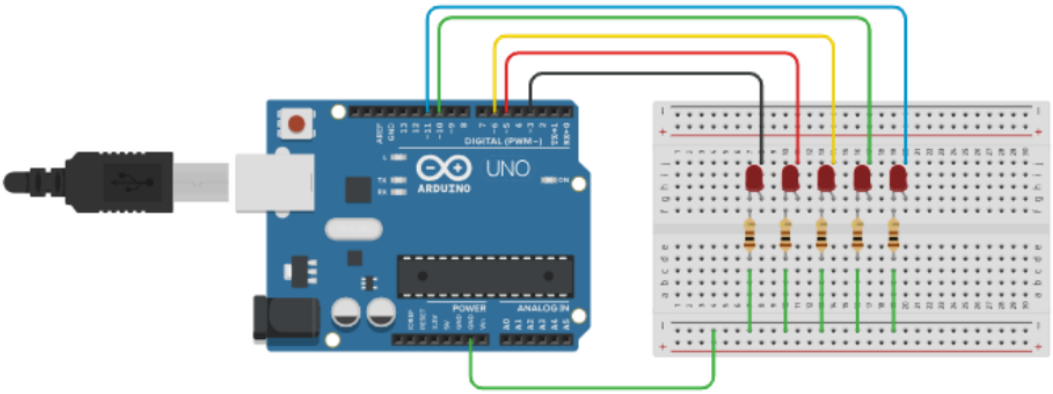
In this experiment, we initialise different LED in the setup() function. Setup() is usedto initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each powerup or reset of the Arduino board.We take pins 3,5,6,9,10 as the output in pinmode() function as these are the PWM pins. And in the loop() function using analogWrite() function change the intensity of the LEDs in the sequence 35214 for both forward and reverse order.

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between the full Vcc of the board (e.g., 5 V on UNO, 3.3 V on a MKR board) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and VCC controlling the brightness of the LED.



The Arduino's programming language makes PWM easy to use; simply call analogWrite(pin, dutyCycle), where dutyCycle is a value from 0 to 255, and pin isone of the PWM pins (3, 5, 6, 9, 10, or 11). The analogWrite() function provides asimple interface to the hardware PWM, but doesn't provide any control over fre- quency.

TINKERCAD DIAGRAM:



TinkerCAD Link

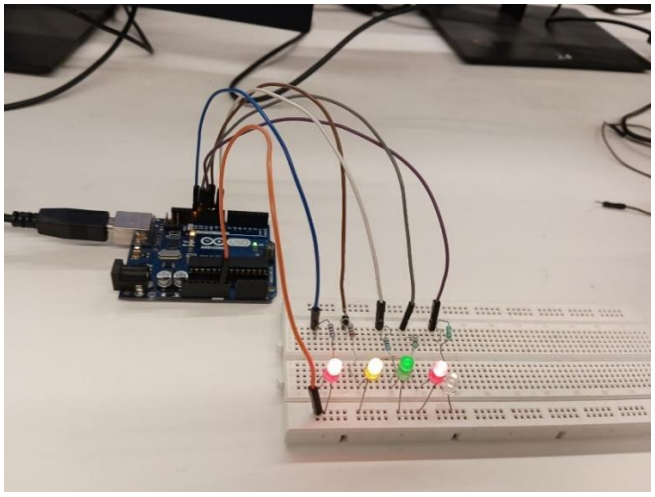
CODE:

```
void setup()
{
pinMode (3, OUTPUT);
pinMode (5, OUTPUT);
pinMode (6, OUTPUT);
pinMode (10, OUTPUT);
pinMode (11, OUTPUT);
}
void loop ()
{
int arr[5]= {6,11,5,3,10};
int intensity=10;
```

```
for ( int i=0 ; i<5; i++)
{
      analogWrite(arr[i], intensity);
      intensity+=30;
       delay(500);
       analogWrite(arr[i],0);
       delay(500);
}
for ( int i=4 ; i>=0; i--)
{
      analogWrite(arr[i], intensity);
      intensity+=30;
      delay(500);
      analogWrite(arr[i],0);
  //delay(500);
}
}
```

RESULT:

In this experiment, we learnt how to change the intensity of the given LED's for thesequence 35214 in for both forward and reverse order.



Signature of faculty member

EXPERIMENT-9

OBJECTIVE: Write a program to demonstrate control of DC Motor using forward, backward, left, right turn motion and clock- wise/anti clock- wise rotation.

SOFTWARE USED: Arduino IDE.

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Nvis 3302ARD RoboCar	1
2.	Arduino UNO	1
3.	USB Cable	1

THEORY:

Nvis 3302ARD is capable of sensing 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 machine is driven by DC motors which are powered by rechargeable batteries. This Nvis 3302ARD is Atmega328P Micro-controller RoboCar. We can design user defined functions in the Arduino IDE to make the buggy move in our own specified directions like left, right, forward, backward, clockwise, and anti-clockwise by setting the pins 5, 6, 7 ,8 on Nvis 3302ARD RoboCar either HIGH/LOW.

Figure: Nvis 3302ARD Model

CODE:

```
void setup()
{
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  Serial.begin(9000);
}
void forward()
{
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
void back()
{
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
}
void left()
{
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
}
void right()
{
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
void CW()
{
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
void ACW()
{
```

```
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
}
void stop()
{
   digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);

}
void loop()
{
  forward();
  delay(3000);
  back();
  delay(3000);
  left();
  delay(3000);
  right();
  delay(3000);
  CW();
  delay(3000);
  ACW();
  delay(3000);
  stop();
  delay(1000000);
}
```

RESULTS:

Hence, we successfully performed the given movements of Robo car (Buggy) using functions from the above components by making suitable connections and code.We saw that the buggy can easily move in forward, then backward, then left, then right, then clockwise, then anticlockwise direction and finally stop in the end after a gap of 2 seconds each.

Signature of faculty member

EXPERIMENT-10

OBJECTIVE: Write a program to read values of IR Sensor using Analog and Digital read and convert buggy into normal line follower robocar.

SOFTWARE USED: Arduino IDE.

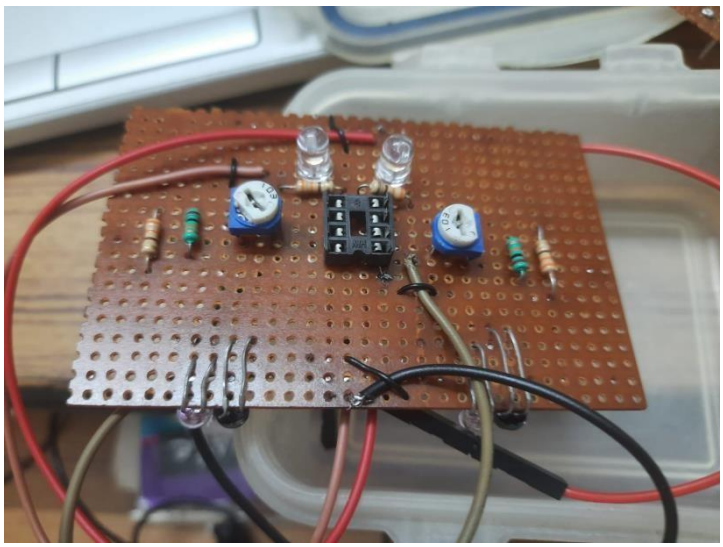
HARDWARE USED:

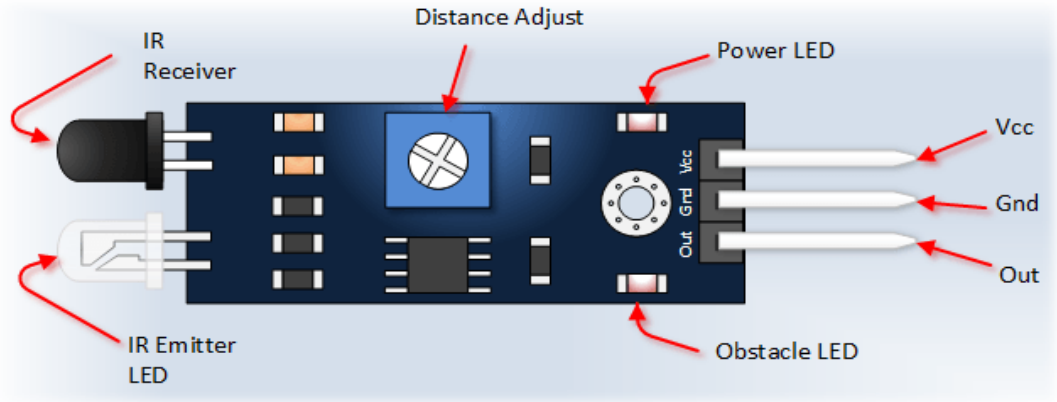
Sr No.	Name of the Component	Value
1.	Nvis 3302ARD RoboCar	1
2.	Arduino UNO	1
3.	USB Cable	1
4.	IR Sensors	2
5.	Jumper Wires	6

THEORY:

IR sensor basically works on intensity of light RGB code for white is (255,255,255) and for Black is (0,0,0)

An IR sensor is a device that emits signals in order to sense some aspects of the surroundings which detects IR radiation falling on it. The emitter is an IR LED (Light Emitting Diode) and the detector is an IR photodiode which is sensitive to IR light. In our buggy, the IR sensor helps it to move only on the black lines of our path defined. We use the pre defined functions from the previous experiment to control the movement of the buggy on the path in the Arduino IDE which in turn gives the instructions to the IR sensor according to values read by the IR sensor on the analog pins..





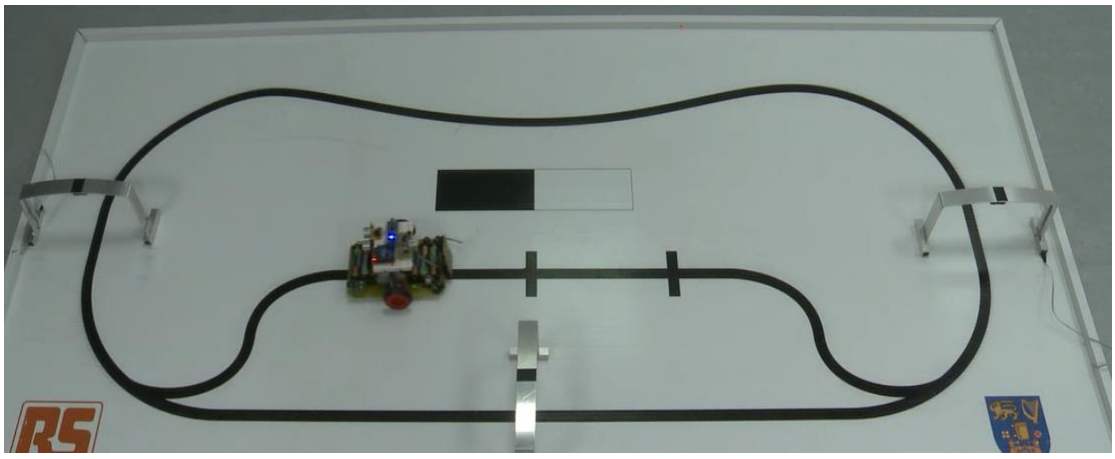
CODE:

```
int l,r;
void setup()
{
  pinMode(A0,OUTPUT);
  pinMode(A1,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  Serial.begin(9000);
}

void forward()
{
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
void left()
{
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
}
void right()
{
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
```

```
void loop()
{
  l=digitalRead(A0);
  r=digitalRead(A1);

  if(l==1 && r==1)
  {
    forward();
  }

  if(l==0 && r==1)
  {
    left();
  }

  if(l==1 && r==0)
  {
    right();
  }

  if(l==0 && r==0)
  {
    forward();
  }
}
```

RESULTS:

In this Experiment we learnt how to take input from IR Sensors using analog pins. And also that how can we make the Buggy follow the black path and execute the proper path follow of the buggy adjusting the sensitivity of the IR Sensor.



Signature of faculty member

EXPERIMENT-11

OBJECTIVE: To demonstrate the use of Ultrasonic Sensor by integrating line-follower Robocar with Obstacle-Avoidance capability.

SOFTWARE USED: Arduino IDE.

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Nvis 3302ARD RoboCar	1
2.	Arduino UNO	1
3.	USB Cable	1
4.	IR Sensors	2
5.	Ultrasonic Sensor	1
6.	Jumper Wires	6

THEORY:

An ultrasonic sensor is an instrument that measures the distance to an object and detects the obstacle using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. Thus, we use the ultrasonic sensor on the buggy to measure the distance between the buggy and obstacle. We stop the buggy as soon as the distance between the buggy and the obstacle is less than 15cm. Hence, we design the code to control the working of the ultrasonic sensor on the robocar.

The HC-SR04 module hosts the ultrasonic transmitter, the receiver and control circuit. The HC-SR04 has four pins namely Vcc, Trigger, Echo, GND and they are explained in detail below:

1) **VCC**: 5V DC supply voltage is connected to this pin.
2) **Trigger**: The trigger signal for starting the transmission is given to this pin. The trigger signal must be a pulse with 10uS high time. When the module receives a valid trigger signal, it issues 8 pulses of 40KHz ultrasonic sound from the transmitter. The echo of this sound is picked by the echo pin.
3) **Echo**: At this pin, the module outputs will be received as a waveform with hightime proportional to the distance.
4) **GND**: Ground is connected to this pin.

CODE:

```
const int trigpin=13;
const int echopin=12;
long duration;
int distance;
void forward()
{
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
}
void stop()
{
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
}
void backward()
{
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
}

void right()
{
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
}
void left()
{
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
}
void setup()
{
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  Serial.begin(9600);
```

```
}

void loop()
{
digitalWrite(trigpin,LOW);
delayMicroseconds(2);
digitalWrite(trigpin,HIGH);
delayMicroseconds(2);
digitalWrite(trigpin,LOW);
duration=pulseIn(echopin,HIGH);
distance=duration*0.0343/2;

if (distance<=20)
{
  stop();
  delay(1000);
}
else
{
  if(Serial.available()>0)
  {
    char s=Serial.read();
    switch(s)
    {
      case 'F':
      {
        forward();
        delay(100);
        break;
      }
       case 'B':
      {
        backward();
        delay(100);
        break;
      }
       case 'L':
      {
        left();
        delay(100);
        break;
      }
       case 'R':
      {
        right();
        delay(100);
        break;
      }
      case 'S':
```

```
      {
        stop();
        delay(100);
        break;
      }
    }
  }
}
}
```

RESULTS:

In this experiment, we learnt how to use the Ultrasonic sensor on the buggy for distance measurement. We also learned to design the code for controlling the motion of the buggy and detecting the obstacles in its path.

Signature of faculty member

EXPERIMENT-12

OBJECTIVE: Write a program
a) To read the pulse width of gantry transmitter and trigger stop_buggy function by detecting individual gantry.

b) To demonstrate Xbee module communication between Buggy and PC.

SOFTWARE USED: Arduino IDE.

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Nvis 3302ARD RoboCar	1
2.	Arduino UNO	1
3.	USB Cable	1
4.	IR Sensors	2
5.	Ultrasonic Sensor	1
6.	XBee Module	1
7.	Jumper Wires	6

THEORY:

XBee module is used for the communication between 2 PC's using the X-CTU software. It is configured first and then attached to the buggy so that it can be controlled from the X-CTU software on its own.
As soon as the receiver on the top of buggy receives signals from the transmitter that isattached on the gantry, our buggy stops.

1) Each network has 1 coordinator
2) Coordinator selects channel and PAN ID
3) Other devices then join the PAN
4) 16-bit address is always 0
5) Assigns 16-bit address for the router and end devices

CODE:

```
int ir1=A0;
int ir2=A1;
int pin=4;
int count=0;
int l,r;
const int TrigPin=13;
const int EchoPin=12;
long duration;
int distance;
void setup()
{
  Serial.begin(9600);
  pinMode(EchoPin,INPUT);
  pinMode(TrigPin,OUTPUT);
  pinMode(ir1,INPUT);
  pinMode(ir2,INPUT);
  pinMode(pin,INPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);

}
void stopbuggy()
{
  digitalWrite(5,LOW);
  digitalWrite(8,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
}
void forward()
{
  digitalWrite(5,HIGH);
  digitalWrite(8,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);

}

void right()
{
  digitalWrite(8,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(5,LOW);
  digitalWrite(7,LOW);
}
```

```
void left()
{
  digitalWrite(5,HIGH);
  digitalWrite(7,LOW);
  digitalWrite(6,LOW);
  digitalWrite(8,LOW);
}

long st=millis(),endt;
int flag=0;

void loop()
{
  digitalWrite(TrigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(TrigPin,HIGH);
  delayMicroseconds(10);

  digitalWrite(TrigPin,LOW);
  duration=pulseIn(EchoPin,HIGH);
  distance=duration*0.0343/2;

  if (Serial.read() =='G' || flag == 1 )
  {

    if(digitalRead(pin)>0)
    {
      int value=pulseIn(pin,HIGH);
      Serial.print("Value = ");
      Serial.println(value);



     if(value>1500 && value<2000)
     {
       Serial.println("Gantry 1 crossed");
       stopbuggy();
       delay(1000);
     }
     if(value>2500 && value<3000)
     {
       Serial.println("Gantry 2 crossed");
       stopbuggy();
       delay(1000);
     }
     if(value>500 && value<1000)
     {
       Serial.println("Gantry 3 crossed");
       stopbuggy();
```

```
    delay(1000);
  }

}

flag=1;

l=digitalRead(ir1);
r=digitalRead(ir2);

if (l==1 && r==1)
{
forward();
if(distance<=20){stopbuggy();
}

if (l==0 && r==1)
left();

if (l==1 && r==0)
right();

if (l==0 && r==0)
{
  endt=millis();

  if(endt-st>1000)
  {
    count++;
    st=millis();
  }

  if(count == 1)
  {
    forward();
    Serial.print(count);
    Serial.println("forward");

  }
  if(count == 2)
  {
    left();
    Serial.print(count);
    Serial.println("left");

  }
  if(count == 3)
  {
    forward();
```

```
        Serial.print(count);
        Serial.println("forward");
      }
      if(count == 4)
      {
        forward();
        Serial.print(count);
        Serial.println("forward");

      }

      if(count == 5)
      {
        left();
        Serial.print(count);
        Serial.println("left");
        delay(500);
      }

      if(count == 6)
      {
        forward();
        Serial.print(count);
        Serial.println("forward");

      }
      if(count > 6)
      {
        stopbuggy();
        Serial.print("Stop");
        flag=0;

      }
    }
  }
}
```

RESULTS:

In this experiment we learnt about XBee Module and the X-CTU application used to configure the XBee. And also to stop the buggy at the gantry and display the distance respectively.

Signature of faculty member

EXPERIMENT-13

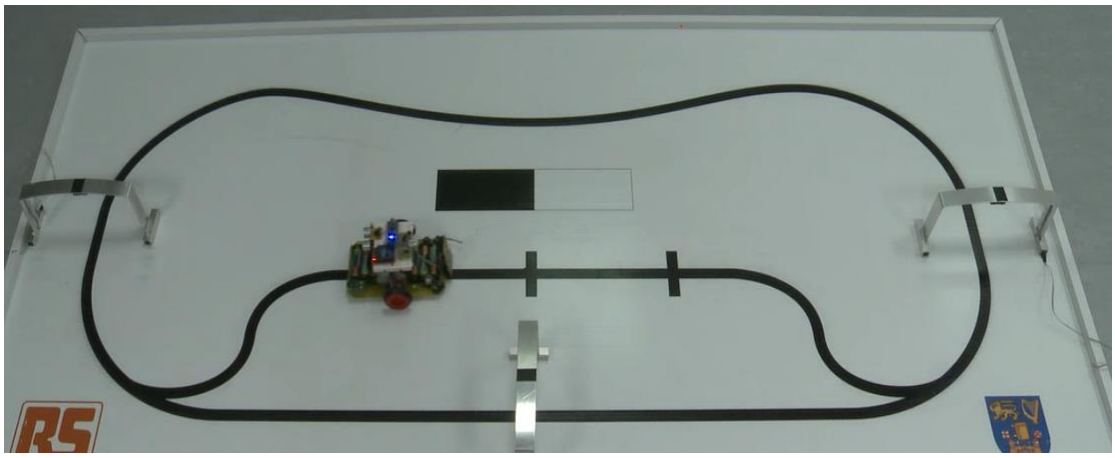
OBJECTIVE: ***Bronze Challenge:***
Single buggy around track twice in clockwise direction, under full supervisory control. Buggy can detect an obstacle, Parks safely. Prints state of the track and buggy at each gantry stop.
SOFTWARE USED: Arduino IDE.

HARDWARE USED:

Sr No.	Name of the Component	Value
1.	Nvis 3302ARD RoboCar	1
2.	Arduino UNO	1
3.	USB Cable	1
4.	IR Sensors	2
5.	Ultrasonic Sensor	1
6.	XBee Module	1
7.	Jumper Wires	6

THEORY:

The bronze challenge i.e. the buggy is required to traverse the whole path twice in clockwise direction while following the path, detect obstacles, park at the right position, stop at each of the three gantry and work with XBee configuration to give initial start to the buggy. This is done with the help of all the circuits prepared earlier and all theexperiments performed. We are using the transmitter, receiver and the IR circuits soldered on our buggy to achieve this challenge.



CODE:

```
int ir1=A0;
int ir2=A1;
int pin=4;
int count=0;
```

```
int l,r;
const int TrigPin=13;
const int EchoPin=12;
long duration;
int distance;
void setup()
{
  Serial.begin(9600);
  pinMode(EchoPin,INPUT);
  pinMode(TrigPin,OUTPUT);
  pinMode(ir1,INPUT);
  pinMode(ir2,INPUT);
  pinMode(pin,INPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);

}
void stopbuggy()
{
  digitalWrite(5,LOW);
  digitalWrite(8,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
}
void forward()
{
  digitalWrite(5,HIGH);
  digitalWrite(8,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);

}

void right()
{
  digitalWrite(8,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(5,LOW);
  digitalWrite(7,LOW);
}

void left()
{
  digitalWrite(5,HIGH);
  digitalWrite(7,LOW);
  digitalWrite(6,LOW);
  digitalWrite(8,LOW);
```

```
}

long st=millis(),endt;
int flag=0;

void loop()
{
  digitalWrite(TrigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(TrigPin,HIGH);
  delayMicroseconds(10);

  digitalWrite(TrigPin,LOW);
  duration=pulseIn(EchoPin,HIGH);
  distance=duration*0.0343/2;

  if (Serial.read() =='G' || flag == 1 )
  {

    if(digitalRead(pin)>0)
    {
      int value=pulseIn(pin,HIGH);
      Serial.print("Value = ");
      Serial.println(value);



     if(value>1500 && value<2000)
     {
       Serial.println("Gantry 1 crossed");
       stopbuggy();
       delay(1000);
     }
     if(value>2500 && value<3000)
     {
       Serial.println("Gantry 2 crossed");
       stopbuggy();
       delay(1000);
     }
     if(value>500 && value<1000)
     {
       Serial.println("Gantry 3 crossed");
       stopbuggy();
       delay(1000);
     }

    }

    flag=1;
```

```
l=digitalRead(ir1);
r=digitalRead(ir2);

if (l==1 && r==1)
{
forward();
if(distance<=20){stopbuggy();
}

if (l==0 && r==1)
left();

if (l==1 && r==0)
right();

if (l==0 && r==0)
{
  endt=millis();

  if(endt-st>1000)
  {
    count++;
    st=millis();
  }

  if(count == 1)
  {
    forward();
    Serial.print(count);
    Serial.println("forward");

  }
  if(count == 2)
  {
    left();
    Serial.print(count);
    Serial.println("left");

  }
  if(count == 3)
  {
    forward();
    Serial.print(count);
    Serial.println("forward");
  }
  if(count == 4)
  {
    forward();
```

```
        Serial.print(count);
        Serial.println("forward");

      }

      if(count == 5)
      {
        left();
        Serial.print(count);
        Serial.println("left");
        delay(500);
      }

      if(count == 6)
      {
        forward();
        Serial.print(count);
        Serial.println("forward");

      }
      if(count > 6)
      {
        stopbuggy();
        Serial.print("Stop");
        flag=0;

      }
    }
  }
}
```

RESULTS:

In this experiment we are performing the bronze challenge i.e. the buggy is required to traverse the whole path twice, detect obstacles, park at the right position, stop at each gantry and work with XBee configuration. We understood the working of the working of the buggy along with the working of various sensors and Zigbee for wireless communication.

Signature of faculty member