# A

**MST Practical Activity Report Submitted for**

# ENGINEERING DESIGN-II (UTA024)

### Submitted by:

|  |  |
| --- | --- |
| **(102103393)** | **Rahul Gurwan** |
| **(102103394)** | **Pia Gupta** |
| **(102103395)** | **Priyanshi Kharbanda** |
| **(102103813)** | **Harshal Kumar Mehra** |
| **(102103815)** | **Radhika Jasra** |

**BE Second Year Batch: 2COE-14(Group-5)**

Submitted to-

Mr Divjot Singh



**Computer Science and Engineering Department TIET, Patiala**

**Jan-May 2023**

## Acknowledgment

We would like to thank our teacher Mr. Divjyot Singh and the professors who gave us a chance to work on this project. We are very grateful that they provided valuable suggestions for the betterment of the project which we greatly appreciate them for doing so.

Furthermore, we want to extend special thanks to our college as well because without their resources then none of what is seen now could have been possible in terms of creative or intellectual development.

## Abstract

This report contains the use of Arduino Uno to perform various experiments. We used Arduino IDE software for writing different codes, which were used in performing various experiments. We used Arduino Uno, breadboard, LEDs, probe/connecting wires, and other various components in order to make multiple circuits and execute the codes to see the results.

## DECLARATION

We, the undersigned solemnly declare that the project report MST PRACTICAL ACTIVITY REPORT is based on our own work carried out during the course of our study under the supervision of Mr. Divjot Singh.

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

* The work contained in the report is original and has been done by us under the general supervision of our supervisor.
* The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University in India or abroad.
* We have followed the guidelines provided by the university in writing the report.
* Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and given their details in the references.

|  |  |
| --- | --- |
| **(102103393)** | **Rahul Gurwan** |
| **(102103394)** | **Pia Gupta** |
| **(102103395)** | **Priyanshi Kharbanda** |
| **(102103813)** | **Harshal Kumar Mehra** |
| **(102103815)** | **Radhika Jasra** |

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.**  **No.** | **Name of Experiments** | **Page No.** | **Remarks** |
| **1.** | Introduction to Arduino Microcontroller. | 9-10 |  |
| **2.** | Write a program (WAP) to blink a single LED. | 11-14 |  |
| **3.** | WAP to blink multiple LEDs using   1. for d) switch case 2. while e) function 3. array | 15-19 |  |
| **4.** | WAP to design an odd-even pattern and then perform the reverse operation also. For example- Sequence = 1 3 5 2 4 ---  4 2 5 3 1 | 20-23 |  |
| **5.** | WAP to show binary inputs upto 2^5 on LEDs. | 24-26 |  |
| **6.** | WAP to blink multiple LEDs for exactly 3 times. (Avoid writing the whole code in void setup). | 27-28 |  |
| **7.** | WAP to send the data using serial communication  a) print b) read c) write | 29-31 |  |
| **8.** | 8. WAP to print the following pattern:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Name-  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Class-  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Department-  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 32-33 |  |
| **9.** | 9. WAP to perform dimmer effect for multiple LEDs:  a) Using for loop b) Using user input | 34-37 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.**  **No.** | **Name of Experiments** | **Page No.** | **Remarks** |
| **10.** | WAP to show the dimmer effect where LED 1 should display values between 0-50 LED 2 = 51- 100  LED3 = 101- 150  LED4 = 151- 200  LED 5 = 201-255 | 38-41 |  |
| **11.** | WAP to perform the following movements of Robo Car  (Buggy) using function:   1. Forward 2. Reverse 3. Left 4. Right 5. Clockwise 6. Anti Clockwise | 41-44 |  |
| **12.** | To read values of IR sensor using analog and digital read and convert buggy into normal line follower robot car. | 45-47 |  |
| **13.** | To demonstrate the use of ultrasonic sensor by integrating line follower robo car with obstacle avoidance capability. | 48-51 |  |
| **14.** | a) Write a program to read the pulse width of gantry transmitter and trigger stop\_buggy function by detecting individual gantry.  b) Write configuration program to demonstrate Xbee module communication between two PC’s using XCTU. | 52-59 |  |
| **15.** | Write a program to demonstrate the full bronze challenge. | 60-68 |  |

# COMPONENT TABLE

|  |  |  |
| --- | --- | --- |
| **Experiment Number** | **Caption** | **Page Number** |
| 2.1 | Hardware used | 13 |
| 3.1 | Hardware used | 17 |
| 4.1 | Hardware used | 22 |
| 5.1 | Hardware used | 26 |
| 6.1 | Hardware used | 29 |
| 7.1 | Hardware used | 31 |
| 8.1 | Hardware used | 34 |
| 9.1 | Hardware used | 36 |
| 10.1 | Hardware used | 40 |
| 11.1 | Hardware used | 44 |
| 12.1 | Hardware used | 47 |
| 13.1 | Hardware used | 51 |
| 14.1 | Hardware used | 59 |
| 15.1 | Hardware used | 66 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Caption** | **Page No.** |
| 1.1 | Arduino uno | 11 |
| 2.1 | Resistors | 14 |
| 2.2 | LED | 14 |
| 2.3 | Arduino UNO | 15 |
| 2.4 | Breadboard | 15 |
| 2.5 | Physical Circuit | 15 |
| 3.1 | Physical Circuit | 18 |
| 3.2 | Physical Circuit | 18 |
| 4.1 | Physical Circuit | 25 |
| 6.1 | Physical Circuit | 29 |
| 6.2 | Physical Circuit | 29 |
| 10.1 | Physical Circuit | 42 |
| 11.1 | NVIS Robocar | 44 |
| 13.1 | Ultrasonic Sensor | 51 |
| 14.1 | Xbee | 58 |

# Experiment: 1

## Objective:

Introduction to Arduino Microcontroller.

## Hardware Used:

Microcontroller model

Csed lab- ATmega328p

Eced lab- ATtiny85

## Software Used:

Arduino Software- (Arduino IDE 2.0.3)

## Theory:

**Definition**- A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system



Fig.1.1 Arduino Uno

### Working-

The inputs and outputs of a microcontroller system are voltages that we can use to determine the state of external devices. The microcontroller reads the voltages from an input device and uses this information to decide on the correct voltage to the output.

### Programming Microcontrollers-

Some microcontrollers like the Arduino have their own integrated development environment (IDE). In the Arduino IDE, you can write and upload the code to the microcontroller via a USB cable.

### ATmega328p-

An ATmega328P-based microcontroller board is the Arduino UNO. It contains 6 analog inputs, a 16 MHz ceramic resonator, a USB port, a power jack, an ICSP header, and a reset button. It also has 14 digital input/output pins, six of which can be used as PWM outputs.

### Purpose of Microcontroller in the Buggy-

Reading the output is simple with a microcontroller like the ATmega328 that has a built-in ADC because it outputs a voltage.

### Discussion:

In this experiment, we came across the concept of Arduino and the functionalities involved in it. We dealt with the concept of pins and the basics of this Arduino Uno board, its different miniature analogies, and how it basically works (surface approach).

## Signature of the Faculty member

**Experiment: 2**

## Objective:

Write a program (WAP) to blink a single LED.

## Software Used:

Arduino Software

## Hardware Component Used:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Resistors | 220 Ω |
| 2. | LEDs | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |
| 5. | Breadboard | NA |

**Theory:**

### Resistor:

A resistor is an element or component which reduces the electrical current and supplies electricity to the electrical or electronic goods in a controlled manner.

It also saves these goods from the damage that may occur due to an excessive supply of electricity.

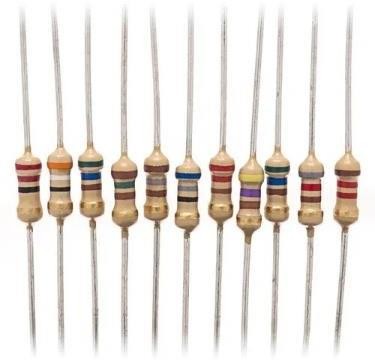


Fig. 2.1 Resistor

**LED-** A light-emitting diode (LED) is a semiconductor device that emits light when an electric current flows through it. When current passes through an LED, the electrons recombine with holes emitting light in the process. LEDs allow the current to flow in the forward direction and blocks the current in the reverse direction.

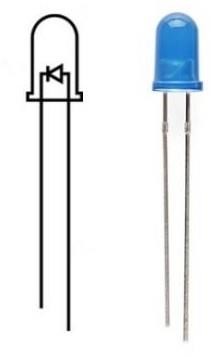


Fig.2.2 LED

**Arduino Uno:** Arduino UNO is a microcontroller board based on the **ATmega328P.** It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.



Fig.2.3 Arduino Uno

**Breadboard:** A breadboard is a solderless device for temporary prototypes with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connects the holes on the top of the board.

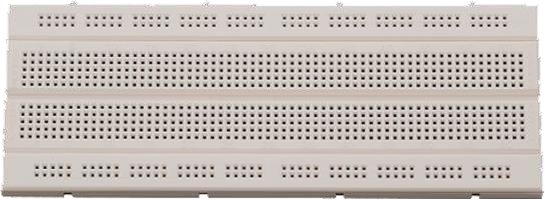


Fig. 2.4 Breadboard

## Logical Circuit diagram :

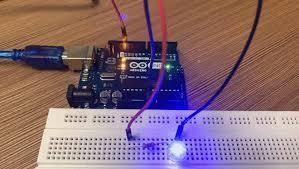


Fig.2.5 Blinking of LED

## Code:

void setup()

{

pinMode(13, OUTPUT)

}

void loop()

{

digitalWrite(13, HIGH) delay(1000) digitalWrite(13, LOW) delay(1000)

}

**Result and Analysis:**

In this experiment, we have learnt how to blink an LED using breadboard and Arduino. First, we connected the positive terminal of resistor with 13th pin and negative terminal of resistor with ground.

Then using USB, we connected the board of Arduino to battery and led started blinking when the code was uploaded in Arduino with the help of software.

## Signature of the Faculty member

**Experiment: 3**

## Objective:

WAP to blink multiple LEDs using

1. for d) switch case
2. while e) function
3. array

**Software Used:**

Arduino Software

## Hardware Component Used:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Resistors (5) | 220 Ω |
| 2. | LEDs (5) | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |
| 5. | Breadboard | NA |

**Logical Circuit diagram :**

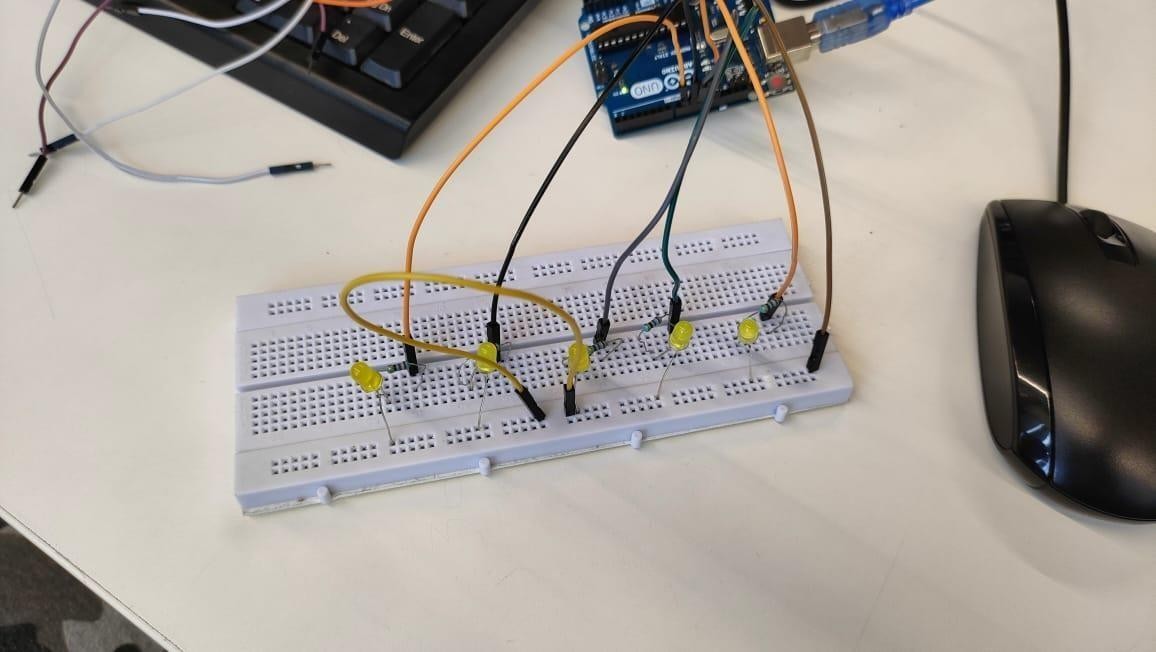


Fig 3.1- Multiple LED physical connection

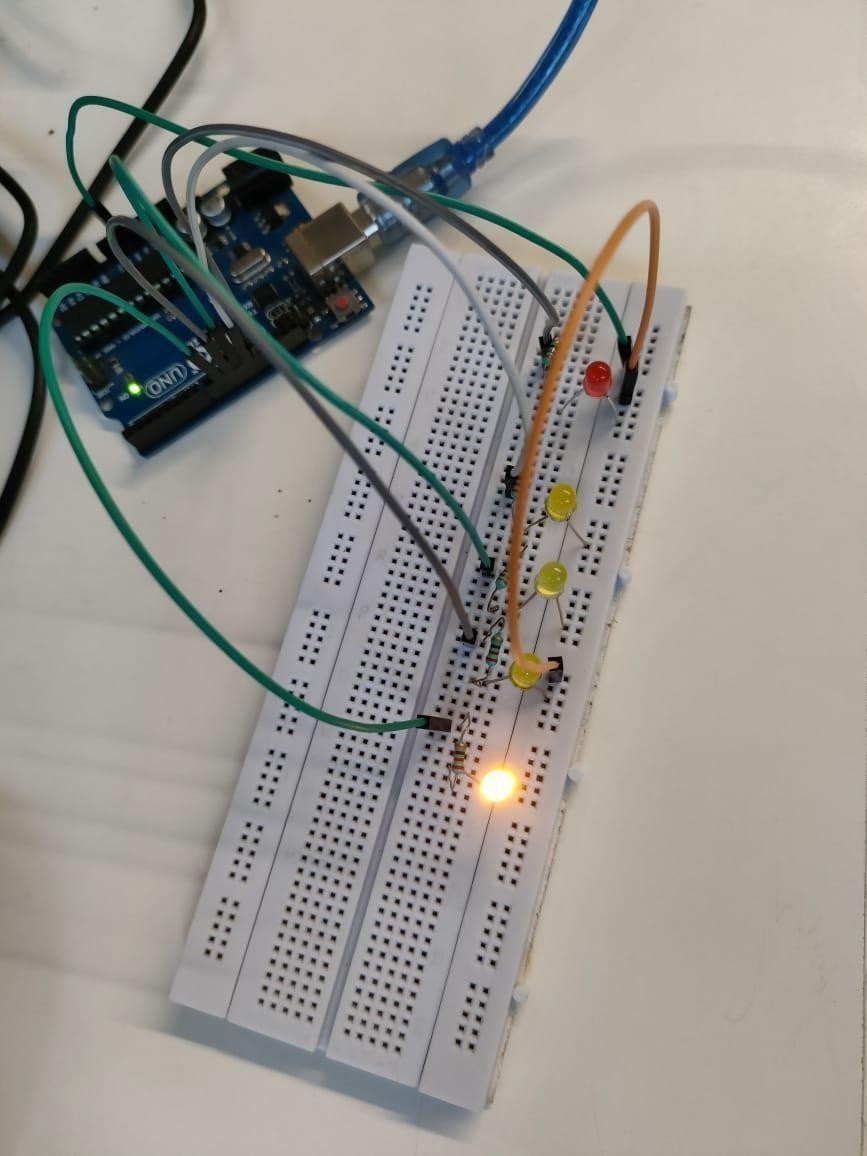


Fig 3.2- Multiple LED physical connection

## Code:

1. **for loop**

void setup()

{

pinMode(6, OUTPUT) pinMode(5, OUTPUT) pinMode(4, OUTPUT) pinMode(3, OUTPUT) pinMode(2, OUTPUT)

}

void loop()

{

for (int i=2

i <= 6 i++)

{

digitalWrite(i, HIGH) delay(1000) digitalWrite(i, LOW)

}

delay(1000) for (int i=6

i >= 2

i--){digitalWrite(i, HIGH) delay(1000) digitalWrite(i, LOW)

}

delay(1000)

}

## while loop

void loop(){ int i=2; while(i<7)

{

digitalWrite(i,HIGH); delay(500); digitalWrite(i,LOW); delay(500);

i++;

}

}

1. **Array**

const int pins[5]={2, 3, 4, 5, 6}; void setup()

{

for(int i=i=0; i < 5; i++) pinMode(Pins[i], OUTPUT);

}

void loop()

{

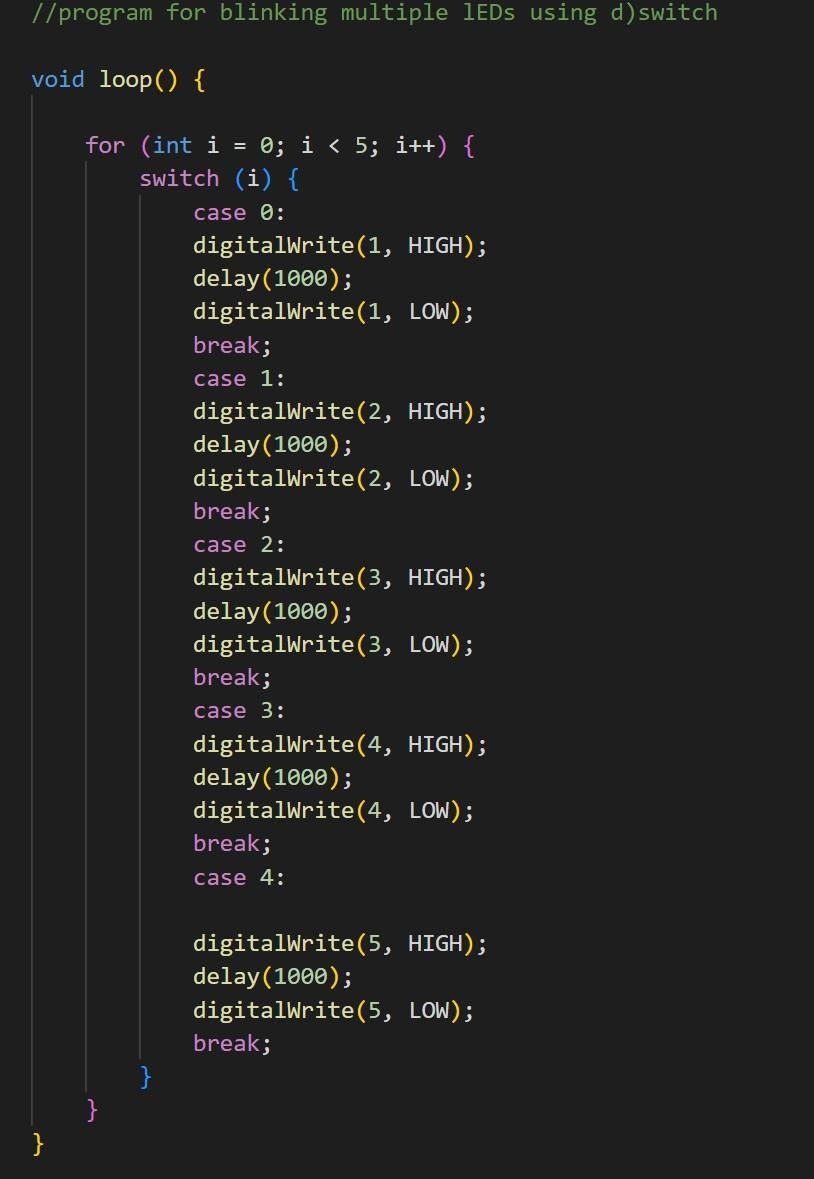
int i;

for(i=0; i < 5; i++)

{

digitalWrite(Pins[i], HIGH); delay(500); digitalWrite(Pins[i], LOW); delay(500);}}

## Switch



//program for blinking multiple LEDs using e) function

void multiple\_blinking()

{

for(int i=1;i<6;i++)

{

digitalWrite(i,HIGH); delay(500);

}

}

void setup ()

{

pinMode(1,OUTPUT); pinMode(2,OUTPUT); pinMode(3,OUTPUT); pinMode(4,OUTPUT); pinMode(5,OUTPUT);

}

void loop ()

{

multiple\_blinking();

}

**Result and Analysis:**

In this experiment, we have learnt how to represent a Sequential series on a circuit using resistors, LEDs, breadboard and connecting this circuit with Arduino uno using jumper wires.

In this experiment, we first connected the positive leg of the LEDs with the resistor as shown in the above circuit snapshot and those resistors are further connected to the ground of Arduino. After this the negative legs of the LEDs are connected to 13th,12th,11th,10th and 9th pins respectively of Arduino using jumper wires. Then the board was connected to the battery using the USB and the respective code was uploaded on Arduino using the same software on computer resulting in the success of the experiment.

## Signature of the Faculty member

**Experiment: 4**

## Objective:

WAP to design an odd-even pattern and then perform the reverse operation also. For example- Sequence

= 1 3 5 2 4 --- 4 2 5 3 1

## Software Used:

Arduino Software

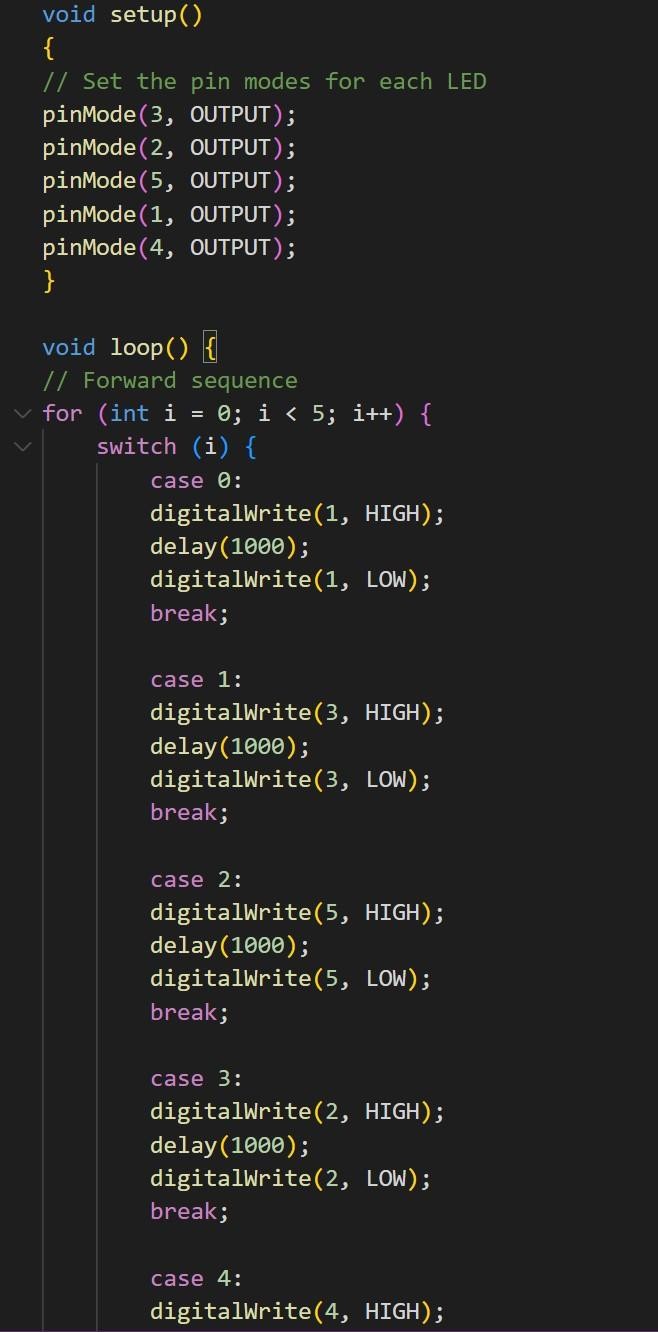
## Hardware Component Used:

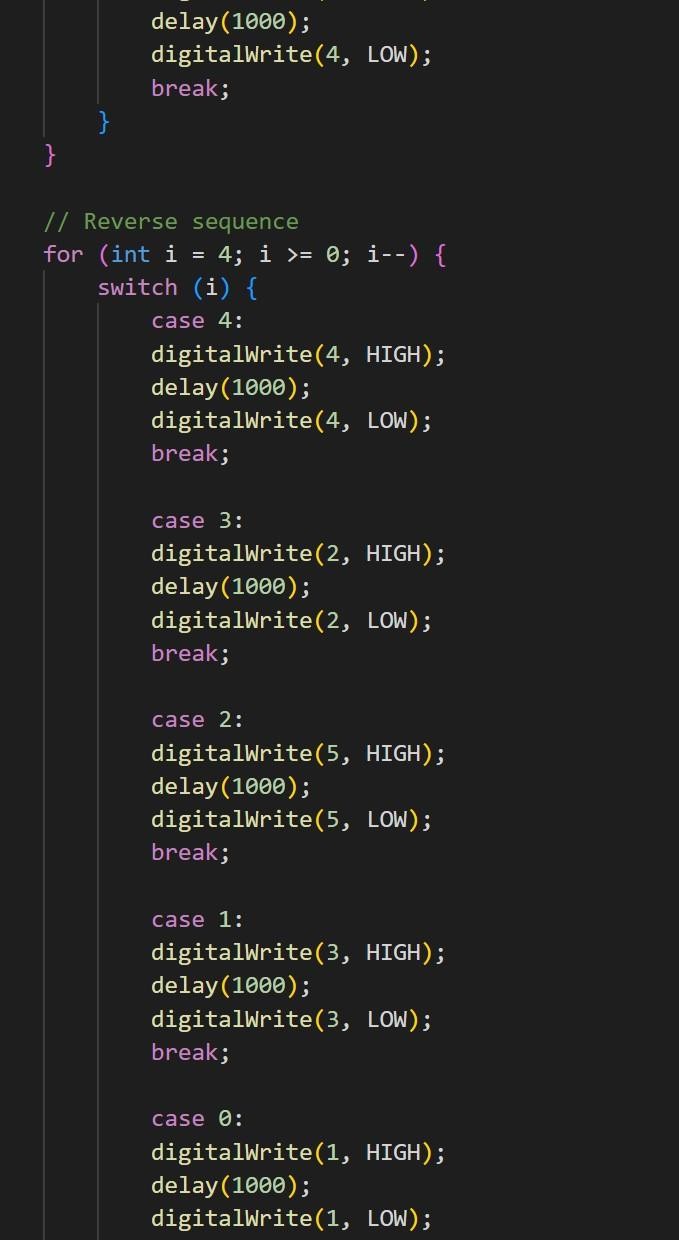
|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Resistors (5) | 220 Ω |
| 2. | LEDs (5) | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |
| 5. | Breadboard | NA |

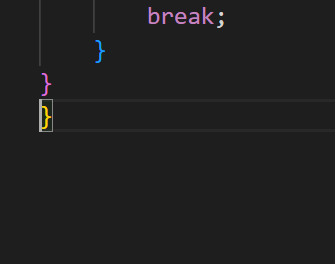
**Theory:**

In this experiment, we use the delay() function and for() loop to simulate the blinking of multi-colored LEDs. But given that there is no fixed pattern of blinking (increment or decrement) we use the Switch : statement to inculcate the desired blinking pattern.

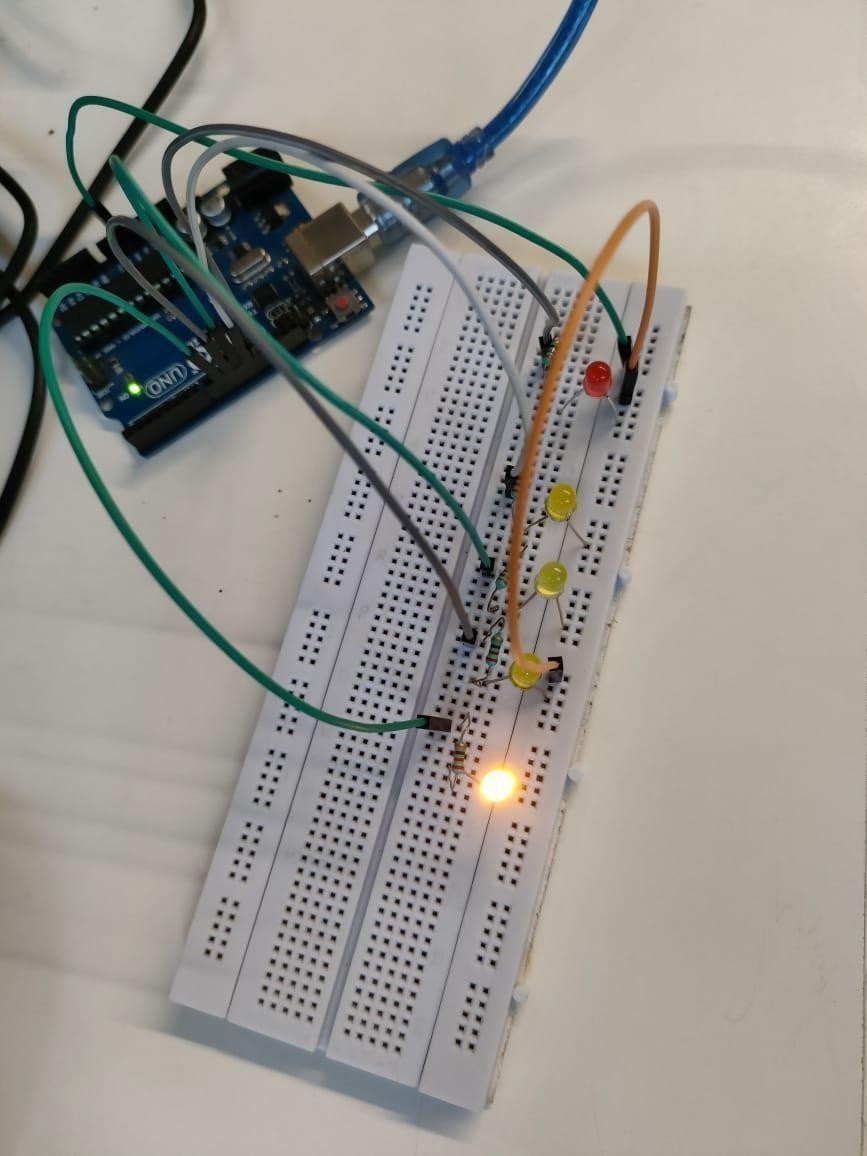
## Code:







**Logical Circuit diagram :**



## Result and Analysis:

Fig 4.1 Physical Connection

In this experiment we wrote an arduino program to design and simulate both forward (1,3,5,2,4)and reverse pattern (4,2,5,3,1) sequences for different color LEDs

using the function.

**Signature of the Faculty member**

# Experiment: 5

## Objective:

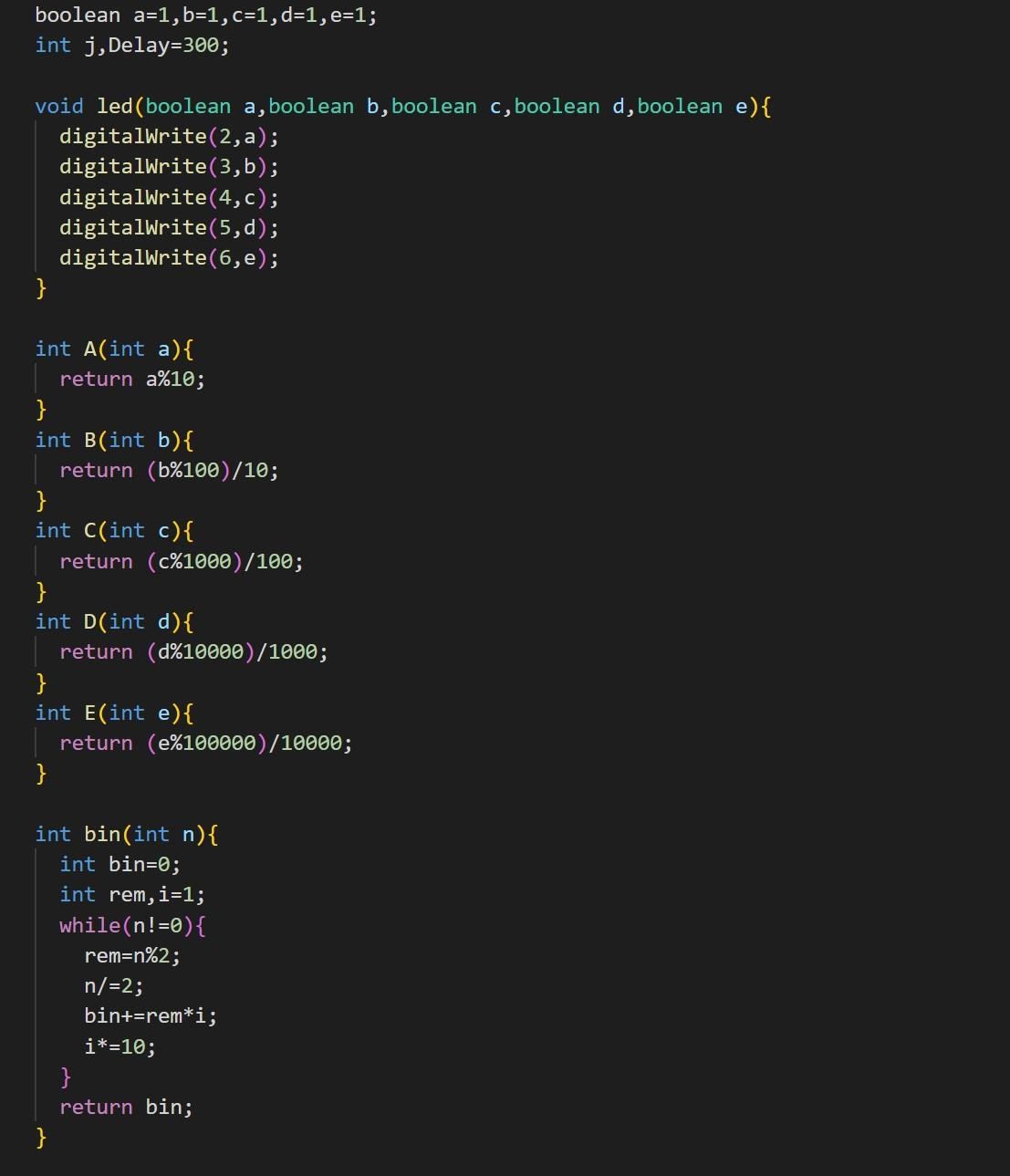
WAP to show binary inputs up to 2^5 on LEDs. **Software Used:**

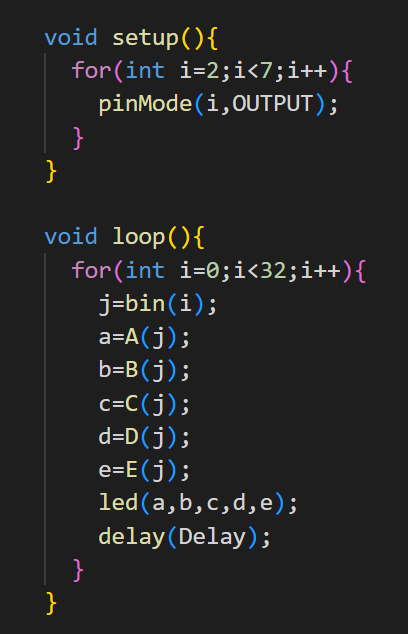
Arduino Software

**Hardware Component Used:**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | BCD (Binary Coded-Decimal) | NA |
| 2. | Breadboard | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |

## Code:





**Result and Analysis:**

We were able to show binary inputs upto 2^5 on LEDs.

## Signature of the Faculty member

**Experiment: 6**

## Objective:

WAP to blink multiple LEDs for exactly 3 times. (Avoid writing the whole code in void setup).

## Software Used:

Arduino IDE Software

## Hardware Component Used:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Resistors | 220 Ω |
| 2. | LEDs | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |
| 5. | Breadboard | NA |

**Logical Circuit diagram :**

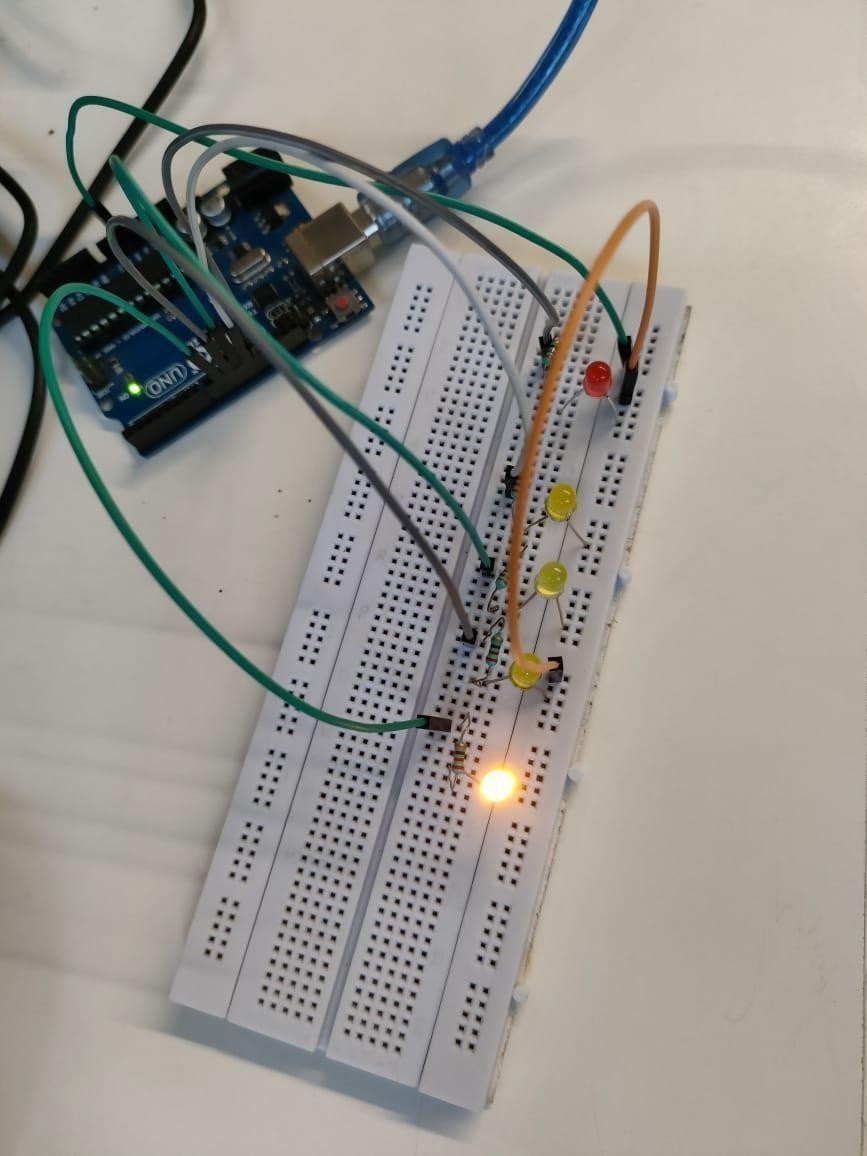


Fig 6.1 blinking LED

## Code:

int switchPin = 6; int switchValue; int counter = 0;

//int counterValue;

void setup()

{

pinMode(1, OUTPUT); pinMode(2, OUTPUT); pinMode(3, OUTPUT); pinMode(4, OUTPUT); pinMode(5, OUTPUT); pinMode(switchPin, INPUT);

digitalWrite(switchPin, HIGH);

}

void loop()

{

int p[]={1,2,3,4,5};

switchValue = digitalRead(switchPin); for(int i=0;i<5;i++){

if ((switchValue == LOW) & & (counter <= 3)) { digitalWrite(i, HIGH);

delay(200); digitalWrite(i, LOW); delay(200); counter++;}

if (switchValue == HIGH)

{

counter = 0; delay(100);

}

}

}

**Results and Analysis:**

The above written code makes the same LED blink thrice. It works on 2 constraints-

* when a switch is turned on, blink an LED 3 times then turn it offafter that
* when a switch is turned off, turn and/or keep the LED off.

## Signature of Faculty member

**Experiment: 7**

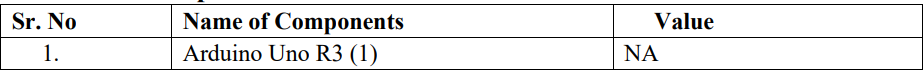
## Objective:

WAP to send the data using serial communication a) print b) read c) write

## Software Used:

Arduino IDE Software

## Hardware Component Used:



**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.

### Serial.print();

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

### Serial.read();

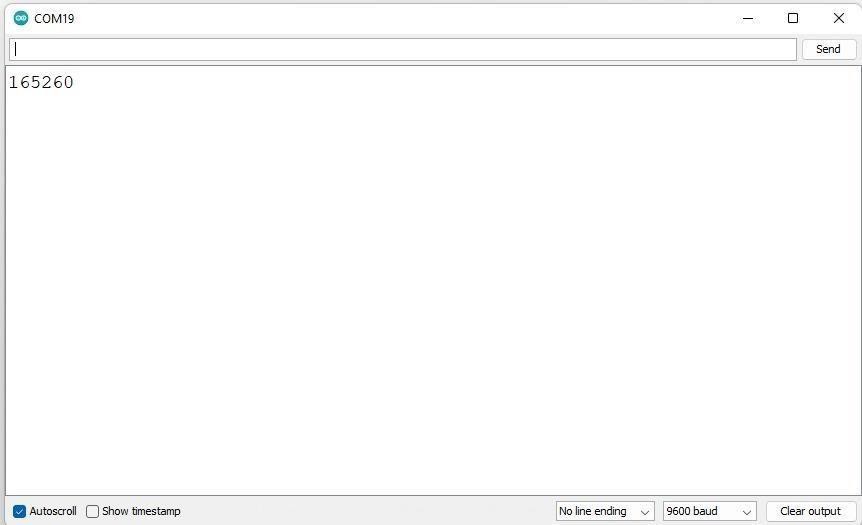
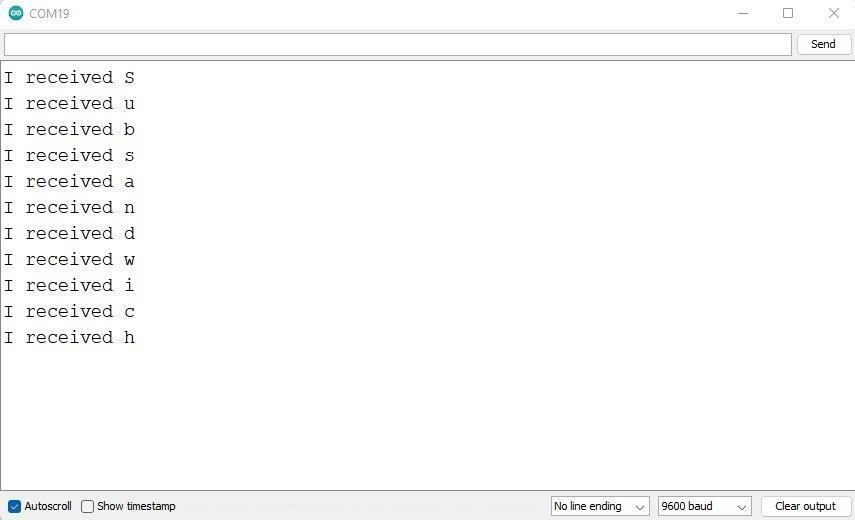
Serial.read() is a function of the Serial library. What it does is read out the first available byte from the serial receive buffer. Say you had sent the phrase “Sub Sandwich” to your Arduino. This means you had put 12 bytes into your serial receive buffer.

### 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.

## Codes:

**SERIAL MONITOR OUTPUT SCREEN:**



## Result and Analysis:

In this experiment, we have learnt how to display different serial() output operations and analogWrite() operations on the Serial Monitor Screen in Arduino IDE. The different operations are performed and output is received on the screen whose snippet is attached above.

## Signature of the Faculty member

**Experiment: 8**

## Objective:

. WAP to print the following pattern:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Name-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Class-

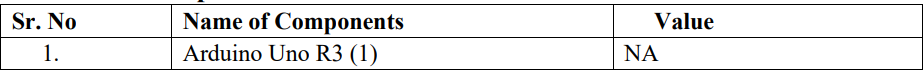
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Department-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Software Used:

Arduino Software

## Hardware Component Used:



**Code:**

void setup() {Serial.begin(9600) for(int i=0

i < 55 i++){

Serial.print("\*")

}

Serial.println(" ")

Serial.println("Roll No. ") for(int i=0

i < 26 i++){

Serial.print("\*")

}

Serial.println(" ") Serial.println("Name: ")

for(int i=0 i < 35 i++){ Serial.print("\*")

}

Serial.println(" ") Serial.println("Branch: ") for(int i=0

i < 26 i++){

Serial.print("\*")

}

}

void loop() {

}

## Result and Analysis:

In this experiment, we have learnt how to use serial commands for basic functions as well. The usage of for loop with them in addition to serial communications to get/print the desired result.

## Signature of the Faculty member

**Experiment: 9**

## Objective:

WAP to perform dimmer effect for multiple LEDs:

* 1. Using for loop
  2. Using user input

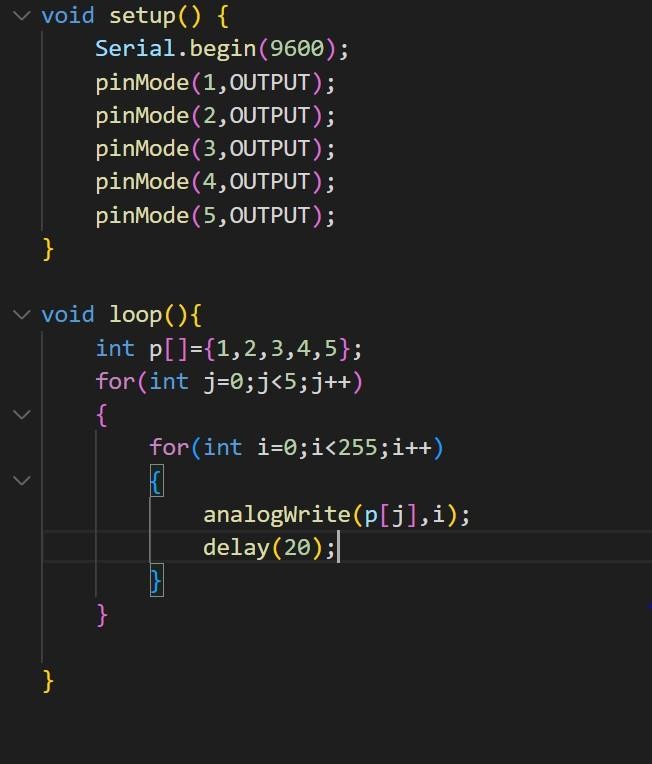
## Software Used:

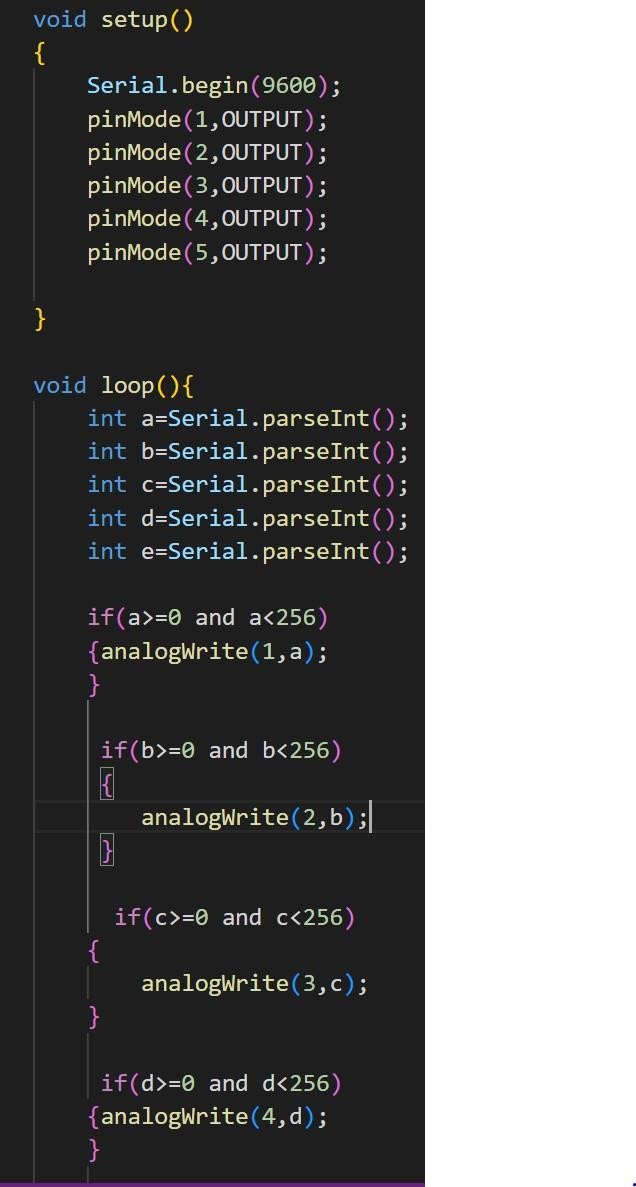
Arduino Software

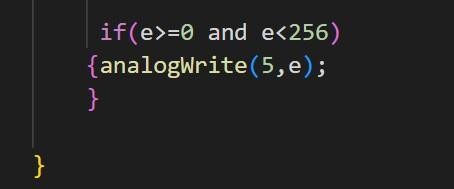
## Hardware Component Used:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Resistor (5) | 220 Ω |
| 2. | LED (5) | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |
| 5. | Breadboard | NA |

**Code:**







## Results and Analysis:

We implemented a circuit wherein we performed dimmer effect of multiple LEDs one by one. We also took input from the user to glow the LED with that intensity.Hence we were able to show dimmer effect both through user input and using for loop.

## Signature of the Faculty member

**Experiment: 10**

## Objective:

WAP to show the dimmer effect where LED 1 should display values between 0-50 LED 2 = 51- 100

LED3 = 101- 150

LED4 = 151- 200

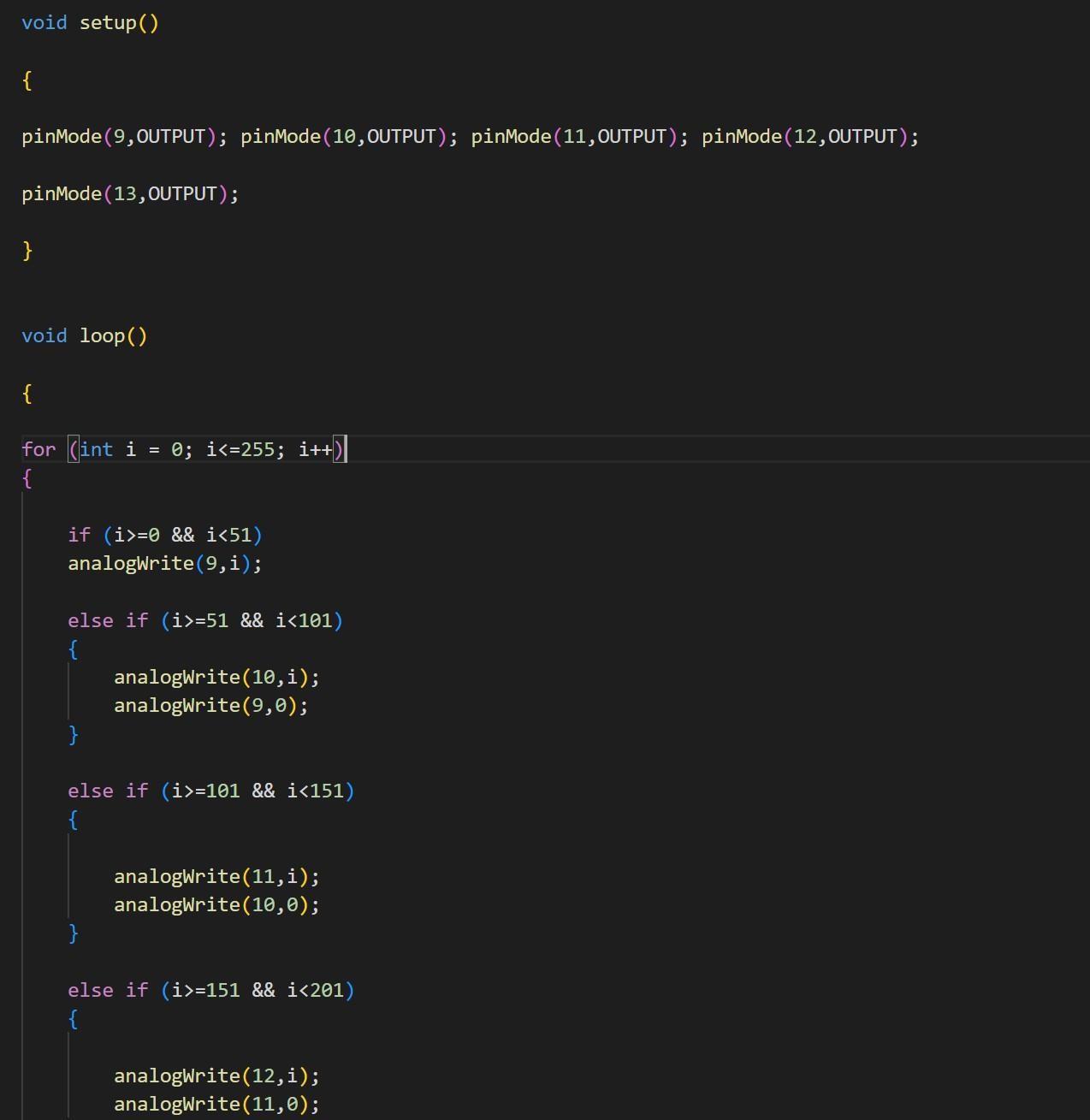
LED 5 = 201-255

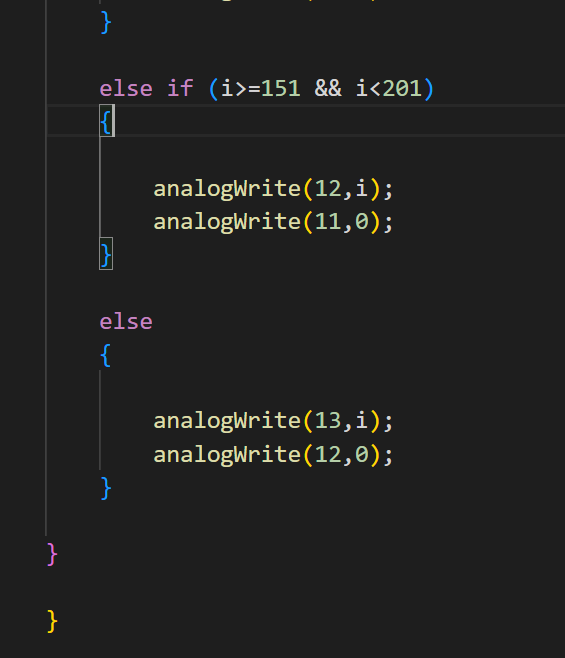
**Software Used:** Arduino Software

## Hardware Component Used:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Resistor (5) | 220 Ω |
| 2. | LED (5) | NA |
| 3. | Jumper wires (as per requirement) | NA |
| 4. | Arduino Uno | NA |
| 5. | Breadboard | NA |

**Code:**





## Logical Circuit diagram :

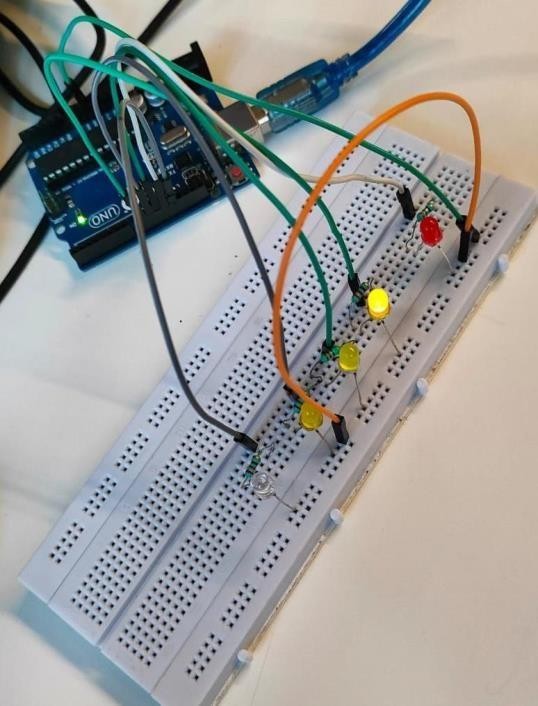
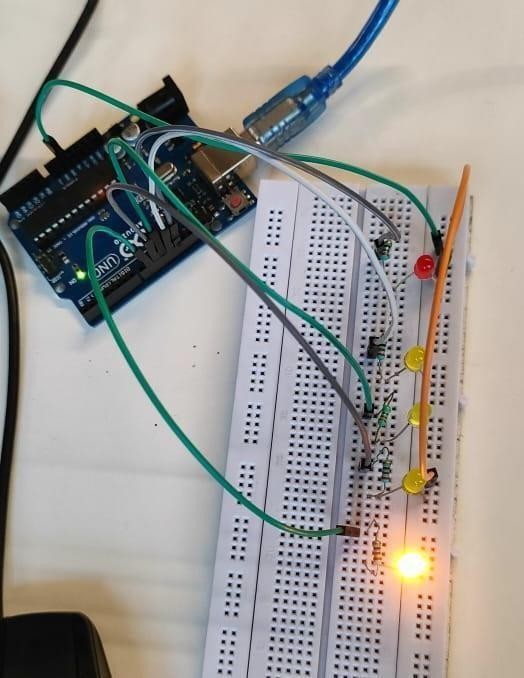


Fig 10.1 Dimmer effect of varying intensities

**Results and Analysis:** In this experiment, we have learnt how we can make different LED’s glow at different intensities. The circuit is as shown in the snapshot where different LED’s are glowing in different intensity ranges.

After making the circuit connections as shown, we connected the board of Arduino to battery . using USB, and LED’s started blinking at their respective range zones.

## Signature of the Faculty member

**Experiment: 11**

## Objective:

WAP to perform the following movements of Robo Car (Buggy) using function:

1. Forward
2. Reverse
3. Left
4. Right
5. Clockwise
6. Anti Clockwise

## Software Used:

Arduino Software

## Hardware Component Used:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of Components** | **Value** |
| 1. | Arduino Microcontroller | NA |
| 2. | USB Cable | NA |
| 3. | Nvis 3302ARD RoboCar | NA |

**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 machine is driven by DC motors which are powered by rechargeable batteries. This Nvis 3302ARD is Atmega328P Microcontroller 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.



Fig 11.1 NVIS Robocar

## Code:



**Results and Analysis:**

We see that the buggy can easily move in forward, then backward, then left, then right, then clockwise and then finally anticlockwise direction after a gap of 1 seconds.

## Signature of the Faculty member

## 

## Experiment: 12

**Objective:**

To read values of IR Sensor using analog and digital read and convert buggy into normal line follower robot car.

**Hardware Used:**

LED, Arduino microcontroller, Connecting wires, Breadboard, USB connector, Resistor, buggy, IR sensor.

**Software Used:**

Arduino IDE.

### Theory:

An IR sensor is a device that emits in order to sense some aspects of the surroundings which detects IR radiation falling on it. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength is emitted by the IR LED. When IR light falls on the photodiode, the resistances and output voltages, change in proportion to the magnitude of the IR light received. In our buggy, the IR sensor helps it to move only on the black lines of our path defined. We use the user-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.

**Code:**

void setup() {

// put your setup code here, to run once: pinMode(A0,INPUT); pinMode(A1,INPUT); pinMode(5,OUTPUT); pinMode(6,OUTPUT); pinMode(7,OUTPUT); pinMode(8,OUTPUT);

}

void forward()

{

digitalWrite(5,HIGH)

;

digitalWrite(6,LOW); digitalWrite(7,LOW); digitalWrite(8,HIGH)

;

}

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 loop() {

// put your main code here, to run repeatedly:

int l,r;

l = digitalRead(A0); r = digitalRead(A1);

if(l==0 && r==0) forward(); else if(l==0 && r==1) right(); else if(l==1 && r==0) left();

else if(l==1 && r==1) forward();

### RESULT ANALYSIS:

We can clearly see that our buggy moves on the black path. We learned how to use the IR sensor on the buggy for sensing the black-and-white color of the signal. We also learned how to control the movement of the buggy on the predefined path.

**Signature of the Faculty member**

## Experiment:13

**Objective:** To demonstrate the use of an ultrasonic sensor by integrating line follow robot car with obstacle avoidance capability.

**Hardware used:** LED, Arduino microcontroller, connecting wires, Breadboard, USB connector, Resistor, buggy, IR sensor, Ultrasonic sensor

**Software used:** Arduino IDE

**Theory:** An ultrasonic sensor is an instrument that measures the distance to an object 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. High-frequency sound waves reflect from boundaries to produce distinct echo patterns. 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.

### Logic:

int r1, r2, count=0; int buggypin5 = 5; int buggypin6 = 6; int buggypin7 = 7; int buggypin8 = 8;

#include <NewPing.h> #define TRIGGER\_PIN 13

#define ECHO\_PIN 12

#define MAX\_DISTANCE 200

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE);

void setup() { pinMode(5,OUTPUT); pinMode(6,OUTPUT); pinMode(7,OUTPUT); pinMode(8,OUTPUT); pinMode(A0,INPUT); pinMode(A1,INPUT); Serial.begin(9600);

}

void forward() { digitalWrite(5,HIGH);

digitalWrite(6,LOW); digitalWrite(7,LOW);

digitalWrite(8,HIGH);

}

void backward() { digitalWrite(5,LOW); digitalWrite(6,HIGH); digitalWrite(7,HIGH);

digitalWrite(8,LOW);

void clockwise() { digitalWrite(5,LOW); digitalWrite(6,HIGH); digitalWrite(7,LOW); digitalWrite(8,HIH);

}

void anticlockwise(int t) { digitalWrite(5,HIGH); digitalWrite(6,LOW); 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 buggystop() { digitalWrite(5,LOW); digitalWrite(6,LOW); digitalWrite(7,LOW); digitalWrite(8,LOW);

}

void loop()

{r1=digitalRead(A0); r2=digitalRead(A1)

;

int dist = sonar.ping\_cm(); //give readings in cm

//int dist = Sonar.ping\_in(); ---> gives distance in inches Serial.println(dist);

if(dist <15 && dist>0)

{

buggystop();

}

else if(r1==1 && r2==1)

{

forward();

}

else if(r1==0 && r2==0)

{

count++; if(count%2==0)

{

clockwise();

}

else

{

forward();

}

}

else if(r1==1 && r2==0)

{

left();

}

else if(r1==0 && r2==1)

{

right();

}



### Fig. Ultrasonic Sensor

**Result analysis**:

We learned how to use the Ultrasonic sensor on the buggy for distance measurement. We learned to design the code for controlling the motion of the buggy and detecting the obstacles in its path.

**Signature of the Faculty membe**

**EXPERIMENT-14(a)**

**Objective:**

Write a program to read the pulse width of gantry transmitter and trigger stop\_buggy function by detecting individual gantry.

**Hardware used:**

Arduino Board, Connecting Wires, Robot Car, Receiver Circuit, Ultrasonic Sensor

**Software used:**

Arduino IDE

**Theory:**

The gantries are equipped with three pulses of different bandwidths from the transmitter circuit. The range of pulses can be changed by programming the 5 reprogrammable pins of the ATtiny85 microcontroller used in the construction of the transmitter circuit. The receiver circuit is connected to buggy and programmed to do operations like stop the buggy when it passes through the suitable gantry and receives the corresponding pulse. In this way, buggy stops after it passes the gantry receiving the pulse from the transmitter connected to the gantry and if the pulse received is in the range according to the code then the required function is performed by Buggy.

### Logic:

int t1=A0; int t2=A2; int pin5=5; int pin6=6; int pin7=8; int pin8=7; int pin=4; int flag=0;

unsigned long d=0;

static ntgantryCounter=0; static long StartTime=0; static long CurrentTime =0;

unsigned long ElapsedTime = 0; static long StartTimeG=millis();; static long CurrentTimeG = 0; unsigned long ElapsedTimeG = 0;

long previousMillisU = millis(); long intervalU = 500;

#include <NewPing.h> #define TRIGGER\_PIN 13

#define ECHO\_PIN 12

#define MAX\_DISTANCE 200 NewPing    sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE);

void setup() { pinMode(pin5,OUTPUT); pinMode(pin6,OUTPUT); pinMode(pin7,OUTPUT); pinMode(pin8,OUTPUT); pinMode(t1,INPUT);

pinMode(t2,INPUT); Serial.begin(9600); Serial.print("+++"); delay(1500);

Serial.println("ATID 3333, CH C, CN");

} void loop()

{ if(flag==0)

{ if (Serial.available() > 0)

{ char s = Serial.read(); switch (s) { case 'G':

{ flag=1;

}

}

} }

unsigned    long    currentMillisU  = millis(); if(currentMillisU - previousMillisU>intervalU)

{

previousMillisU = currentMillisU; detectObstacle();

}

if (flag==1)

{

gantry();

}

if (flag==3)

{

CurrentTimeG=millis(); ElapsedTimeG = CurrentTimeG- StartTimeG; if(ElapsedTimeG<1500)

{

flag=3; leftBlind();

}

if(ElapsedTimeG>1500 &&ElapsedTimeG<3500)

{

flag=3; normalLineFollow();

}

if(ElapsedTimeG>3500)

stopBuggy();

Serial.print("Buggy:1 Parked"); Serial.println(ElapsedTimeG

); delay(2000)

; flag=-1;

}

}

}

void gantry()

{

int r1=digitalRead(t1); int r2=digitalRead(t2);

if(r1==LOW && r2==LOW)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r1==LOW && r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,LOW); digitalWrite(pin8,LOW);

} if(r1==HIGH && r2==LOW)

{

digitalWrite(pin5,LOW);

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r1==HIGH && r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

Void stopBuggy()

{

digitalWrite(pin5,LOW)

;

digitalWrite(pin6,LOW)

;

digitalWrite(pin7,LOW)

;

digitalWrite(pin8,LOW)

;

}

void detectObstacle()

{

delay(50);

unsigned int distanceCm;

distanceCm = sonar.ping\_cm(); pinMode(ECHO\_PIN,OUTPUT); digitalWrite(ECHO\_PIN,LOW);

pinMode(ECHO\_PIN,INPUT);

Serial.print("Ping: "); Serial.println(distanceCm); Serial.println("cm");

if((distanceCm<15) && (distanceCm>0))

{

stopBuggy(); delay(1000);

}

**Result analysis:**

From this experiment, we get to understand the working of the gantry circuit along receiver circuit. The pulse sends to the computer via receiver circuit which detects the pulse in the form of a Square Wave Function. The time of the high pulse is obtained using pulseIn function along with Serial.begin(9600).

### Signature of the Faculty member

**EXPERIMENT:14(b)**

**Objective:**

Write a configuration program to demonstrate Xbee module communication between two PCs using XCTU.

**Hardware used:**

Arduino Board, Connecting Wires, Robot Car, Zigbee module.

**Software used:**

Arduino Software (IDE), XCTU.

### Theory:

Zigbee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols used to create personal area networks with small, low-power digital radios, such as for home automation, medical device data collection, and other low-power low-bandwidth needs, designed for small scale projects which need wireless connection. Hence, Zigbee is a low-power, low data rate, and close proximity (i.e., personal area) wireless ad hoc network. The Xbee is the brand name of a wireless transceiver device introduced by Digi international which works on the ZigBee protocol and can form PAN networks. They have an approximate range of 10 to 100 meters and are used in industries, scientific fields, medical fields etc. The Xbee module even though uses complex packet data based Zigbee protocol for communicating with each other; But also they can communicate with other devices using simplest serial communication protocol and hence they are widely used in microcontroller baseboards

.

### LOGIC:

void setup()

{

pinMode(5,OUTPUT); pinMode(6,OUTPUT); pinMode(7,OUTPUT); pinMode(8,OUTPUT); Serial.begin(9600);

}

void forward()

{

digitalWrite(5,HIGH); digitalWrite(6,LOW); digitalWrite(7,LOW); digitalWrite(8,HIGH);

}

void loop()

{

if(Serial.available()>0)

{

Char s=Serail.read(); if(S== ‘G’)

{

forward();

}

else

{

digitalWrite(5,LOW);

digitalWrite(6,LOW)

;

digitalWrite(7,LOW)

;

digitalWrite(8,LOW)

;

### Fig. Xbee

**Result analysis:**

From this experiment, we get to understand the working of the Xbee.We have connected the Xbee to buggy and done its configuration on XCTU software. The coordinator will be attached to the co mputer which will control the buggy and the router will be attached to the buggy Then after doing the configurations, we have controlled the movement of buggy from the computer using Xbee communication.

### Signature of the Faculty member

**Experiment: 15**

**Objective:**

Write a Program to demonstrate the full bronze challenge.

**Hardware used:**

Arduino Board, Connecting Wires, Robot Car, Receiver Circuit, Ultrasonic Sensor, Zigbee Module**.**

**Software used:**

Arduino software (IDE), XCTU.

**Theory:**

Bronze Challenge: Single buggy capable of following main track twice in an clockwise direction under full supervisory control. Buggy must be capable of detecting an obstacle whilst following the track, coming to a halt if it does. The buggy must safely park in the parking bay. It prints the state of the track and buggy at each gantry stop. No external enduser manual control input is permitted once the initial start is signalled.

**Logic:**

int t1=A0; int t2=A2; int pin5=5; int pin6=6; int pin7=8; int pin8=7; int irPin=4; int flag=0;

int inside = 0; unsigned long d=0;

static int gantryCounter=0; static long StartTime=0; static long CurrentTime = 0;

unsigned long ElapsedTime = 0; static long StartTimeG=millis();; static long CurrentTimeG = 0; unsigned long ElapsedTimeG = 0; long previousMillisU = millis(); long intervalU = 500;

#include

<NewPing.h> #define TRIGGER\_PIN 13

#define ECHO\_PIN 12

#define MAX\_DISTANCE 200

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE);

void    setup() { pinMode(pin5,OUTPUT); pinMode(pin6,OUTPUT); pinMode(pin7,OUTPUT); pinMode(pin8,OUTPUT); pinMode(t1,INPUT); pinMode(t2,INPUT); Serial.begin(9600); Serial.print("+++");

// Enter Xbee AT command mode, NB no carriage return here delay(1500);  // Guard time Serial.println("ATID 3333, CH C, CN");}

void loop()

{

if(flag==0)

{

if (Serial.available() > 0)

{

char s = Serial.read(); switch (s) { case 'K':

{

flag=1;

}

}

}

}

unsigned    long    currentMillisU  =   millis(); if(currentMillisU - previousMillisU > intervalU)

{

previousMillisU = currentMillisU; detectObstacle();

}

if (flag==1)

{

gantry();

}

if (flag==3)

{

int r1=digitalRead(t1); int r2=digitalRead(t2);

if (r1 == LOW && r2 == LOW)

{

if (inside == 0)

{

digitalWrite(pin5,LOW); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW); inside = 1;

delay(200);

}

else

{

stopBuggy(); delay(100000);

}

}

else

{

normalLineFollow();

}}}

void gantry()

{

int r1=digitalRead(t1); int r2=digitalRead(t2);

if(r1==LOW&&r2==LOW)

{

digitalWrite(pin5,HIGH); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH); digitalWrite(pin8,LOW)

} if(r1==LOW&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,LOW); digitalWrite(pin8,LOW);

} if(r1==HIGH&&r2==LOW)

{

digitalWrite(pin5,LOW); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r1==HIGH&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

if (digitalRead(irPin)==HIGH)

{

StartTime = millis();

d = pulseIn(irPin,HIGH);

if (d>500 and d<1500) //(d > 500 and d < 1500)

{

Serial.println(d); Serial.println("Gantry: 1"); stopBuggy();

delay(1000); if(r1==HIGH&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

delay(300); gantryCounter=gantryCounter+1;

}

else if (d>1500 and d< 2700) //(d> 1500 and d < 2500)

{

Serial.println(d); Serial.println("Gantry: 2"); stopBuggy();

delay(1000); if(r1==HIGH&&r2==HIGH)

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH); digitalWrite(pin8,LOW);

}

delay(300); gantryCounter=gantryCounter+1;

}

else if (d>2500 and d<3500) //(d > 2500 and d < 3500)

{

Serial.println(d); stopBuggy(); Serial.println("Gantry:

3"); delay(1000); if(r1==HIGH&&r2==HIGH

)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

delay(300); gantryCounter=gantryCounter+1;

}

else

{

//Serial.println(d); Serial.println("Gantry: Unknown");

}

Serial.print("The gantry Counter is: "); Serial.println(gantryCounter);

if (gantryCounter>=2)

{

flag=3;

}

els{

gantry();

}

}

}

void stopBuggy()

{

digitalWrite(pin5,LOW)

;

digitalWrite(pin6,LOW)

;

digitalWrite(pin7,LOW)

;

digitalWrite(pin8,LOW)

;

}

void normalLineFollow()

{

int r1=digitalRead(t1); int r2=digitalRead(t2);

if(r1==LOW&&r2==LOW)

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH); digitalWrite(pin8,LOW);

} if(r1==HIGH&&r2==LOW)

{

digitalWrite(pin5,LOW); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r1==LOW&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,LOW); digitalWrite(pin8,LOW);

} if(r1==HIGH&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

}

void leftBlind()

{

int r2=digitalRead(t2); if(r2==LOW)

{

digitalWrite(pin5,LOW); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW);

digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

}

void detectObstacle()

{

delay(50);

unsigned int distanceCm; distanceCm = sonar.ping\_cm(); pinMode(ECHO\_PIN,OUTPUT);

digitalWrite(ECHO\_PIN,LOW); pinMode(ECHO\_PIN,INPUT);

Serial.print("Ping: ");

Serial.println(distanceCm); Serial.println("cm"); if((distanceCm<15) && (distanceCm>0))

{

stopBuggy(); delay(1000); if(r1==HIG H&&r2==HI GH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

delay(300); gantryCounter=gantryCounter+1;

} else if (d>1500 and d< 2700) //(d> 1500 and d < 2500)

{

Serial.println(d); Serial.println("Gantry: 2"); stopBuggy();

delay(1000); if(r1==HIGH&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

delay(300); gantryCounter=gantryCounter+1;

}

else if (d>2500 and d<3500) //(d > 2500 and d < 3500)

{

Serial.println(d); stopBuggy(); Serial.println("Gantry:

3");

delay(1000); if(r1==HIGH&&r2==HIGH

)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

delay(300); gantryCounter=gantryCounter+1;

}

else {

//Serial.println(d); Serial.println("Gantry: Unknown");

}

Serial.print("The gantry Counter is: "); Serial.println(gantryCounter);

if (gantryCounter>=2)

{

flag=3;

}

else

{

gantry();

}

}

}

void stopBuggy()

{

digitalWrite(pin5,LOW)

;

digitalWrite(pin6,LOW)

;

digitalWrite(pin7,LOW)

;

digitalWrite(pin8,LOW)

;

}

void normalLineFollow()

{

int r1=digitalRead(t1); int r2=digitalRead(t2);

if(r1==LOW&&r2==LOW)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW);

digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r1==HIGH&&r2==LOW)

{

digitalWrite(pin5,LOW); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

} if(r1==LOW&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,LOW); digitalWrite(pin8,LOW);

} if(r1==HIGH&&r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

}

void leftBlind()

{

int r2=digitalRead(t2); if(r2==LOW)

{

digitalWrite(pin5,LOW); digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

if(r2==HIGH)

{

digitalWrite(pin5,HIGH)

;

digitalWrite(pin6,LOW); digitalWrite(pin7,HIGH)

;

digitalWrite(pin8,LOW);

}

}

void detectObstacle()

{

delay(50);

unsigned    int distanceCm;

distanceCm  =   sonar.ping\_cm(); pinMode(ECHO\_PIN,OUTPUT);

digitalWrite(ECHO\_PIN,LOW); pinMode(ECHO\_PIN,INPUT);

Serial.print("Ping: "); Serial.println(distanceCm); Serial.println("cm");

if((distanceCm<15) && (distanceCm>0))

{

stopBuggy(); delay(1000);

}

**Result analysis:**

We have successfully completed the Bronze challenge by using various sensors like IR sensors, ultrasonic sensors, receiver circuit and ZigBee module.

**Signature of the Faculty member**