4- WEEKS TRAINING REPORT (IoT)

SUBMITTED TO-

MR. PRAKASH PRATIK

(TRAINING CO-ORDINATOR)

______________________________ signature.

SUBMITTED BY-

-PRATEEK UMRAO (Electronics Engg. Intern)

-SURYA GIRI (Electronics Engg. Intern)

-KUNDAN PAWAR (Tech Support TinkerCad)

-HIMANSHU GANDHI (Mechanical Engg. Intern)

DAY 1

INTRODUCTION TO IOT

The Internet of Things (IoT) describes the network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools. With more than 7 billion connected IoT devices today, experts are expecting this number to grow to 10 billion by 2020 and 22 billion by 2025. Oracle has a network of device partners.

Why is Internet of Things (IoT) so important?
Over the past few years, IoT has become one of the most important technologies of the 21st century. Now that we can connect everyday objects—kitchen appliances, cars, thermostats, baby monitors—to the internet via embedded devices, seamless communication is possible between people, processes, and things.
By means of low-cost computing, the cloud, big data, analytics, and mobile technologies, physical things can share and collect data with minimal human intervention. In this hyperconnected world, digital systems can record, monitor, and adjust each interaction between connected things. The physical world meets the digital world—and they cooperate.
What technologies have made IoT possible?
While the idea of IoT has been in existence for a long time, a collection of recent advances in a number of different technologies has made it practical.

- **Access to low-cost, low-power sensor technology.** Affordable and reliable sensors are making IoT technology possible for more manufacturers.
- **Connectivity.** A host of network protocols for the internet has made it easy to connect sensors to the cloud and to other “things” for efficient data transfer.
- **Cloud computing platforms.** The increase in the availability of cloud platforms enables both businesses and consumers to access the infrastructure they need to scale up without actually having to manage it all.
- **Machine learning and analytics.** With advances in machine learning and analytics, along with access to varied and vast amounts of data stored in the cloud, businesses can gather insights faster and more easily. The emergence of these allied technologies continues to push the boundaries of IoT and the data produced by IoT also feeds these technologies.
- **Conversational artificial intelligence (AI).** Advances in neural networks have brought natural-language processing (NLP) to IoT devices (such as digital personal assistants Alexa, Cortana, and Siri) and made them appealing, affordable, and viable for home use.

What is industrial IoT?
Industrial IoT (IIoT) refers to the application of IoT technology in industrial settings, especially with respect to instrumentation and control of sensors and devices that engage cloud technologies. Refer to thisTitan use case PDF for a good example of IIoT. Recently, industries have used machine-to-machine communication (M2M) to achieve wireless automation and control. But with the emergence of cloud and allied technologies (such as analytics and machine learning), industries can achieve a new automation layer and with it create new revenue and

business models. IIoT is sometimes called the fourth wave of the industrial revolution, or Industry 4.0. The following are some common uses for IIoT:

- Smart manufacturing
- Connected assets and preventive and predictive maintenance
- Smart power grids
- Smart cities
- Connected logistics
- Smart digital supply chains

Unlock business value with IoT

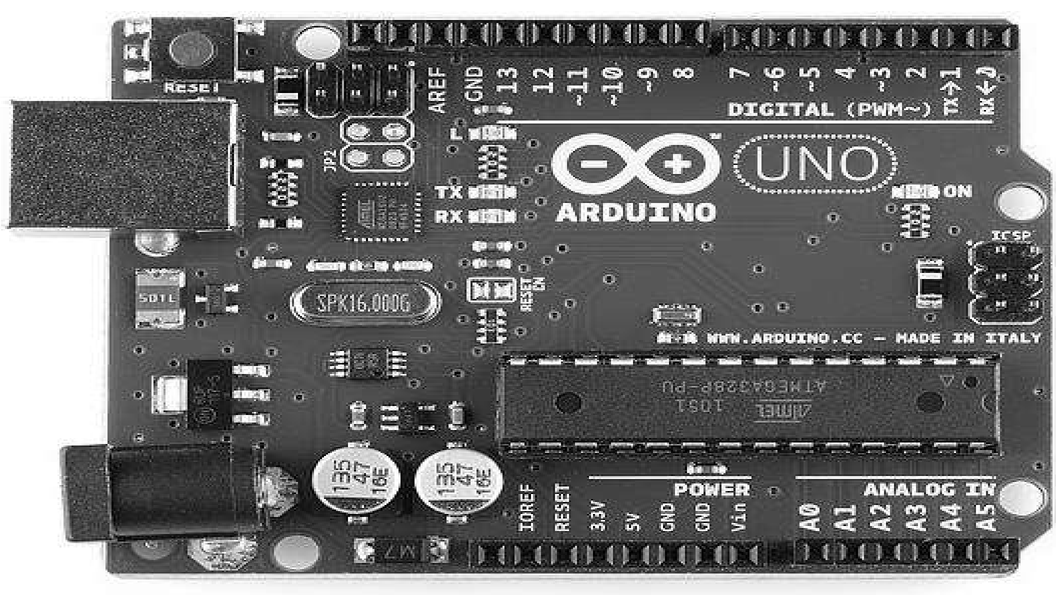
As IoT becomes more widespread in the marketplace, companies are capitalizing on the tremendous business value it can offer. These benefits include:

- Deriving data-driven insights from IoT data to help better manage the business
- Increasing productivity and efficiency of business operations
- Creating new business models and revenue streams
- Easily and seamlessly connecting the physical business world to the digital world to drive quick time to value

DAY 2

ARDUINO –

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.



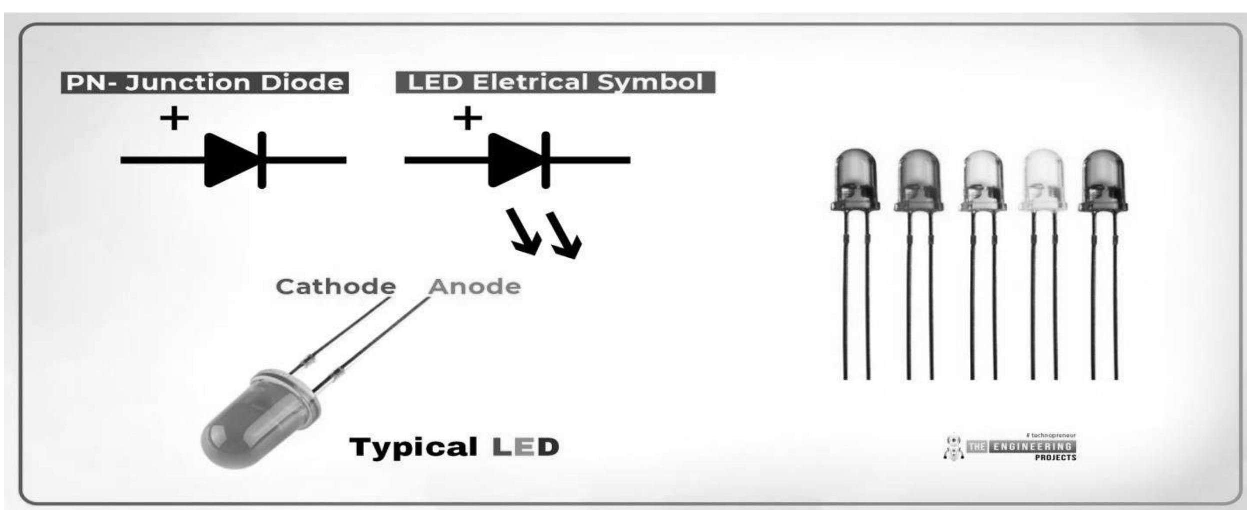
BREAD BOARD-

A **breadboard**, **solder less breadboard**, **protoboard**, or **terminal array board** is a construction base used to build semi-permanent prototypes of electronic circuits. Unlike strip board (Vero board), breadboards do not require soldering or destruction to tracks and are hence reusable. For this reason, breadboards are also popular with students and in technological education.

A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

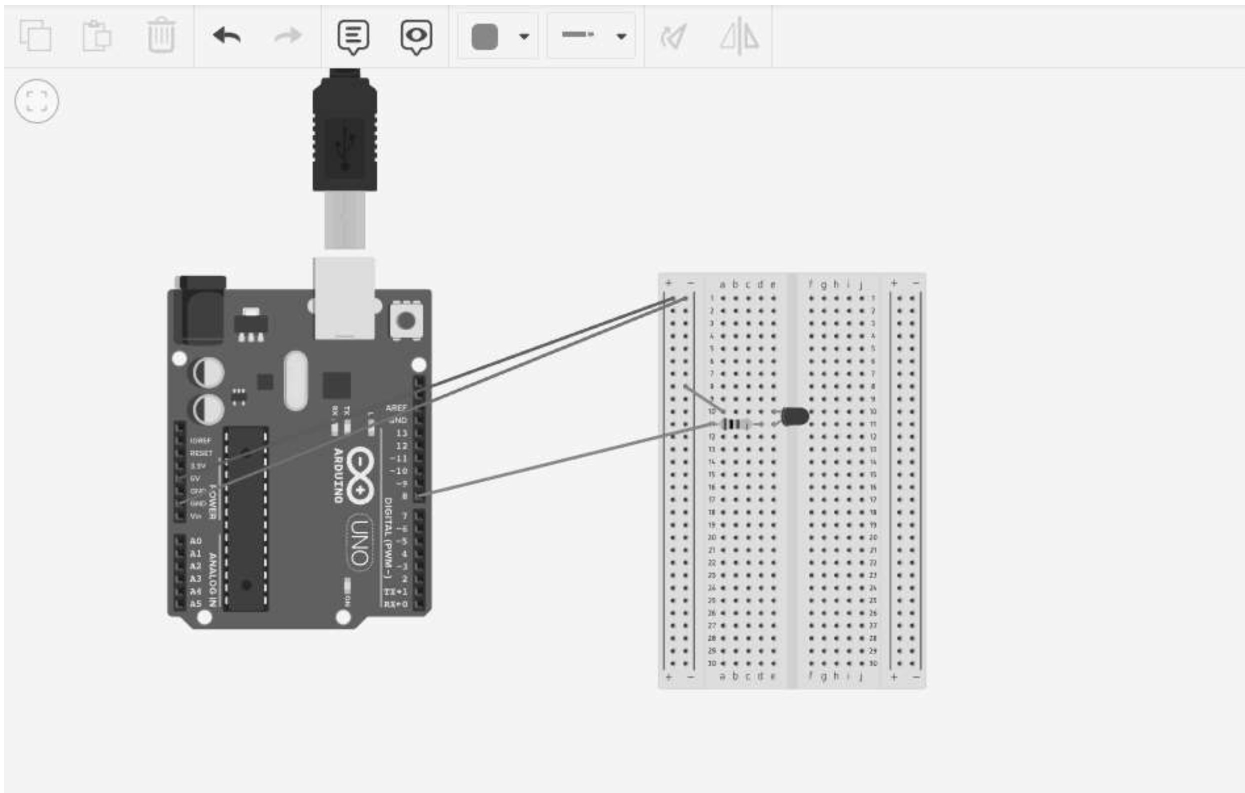
LED-

LED stands for Light Emitting Diode, and this light source should not be confused with a light fixture or luminaire. An LED is a component of the entire fixture. LED lighting can also be referred to as solid-state lighting (SSL) because an LED is solid-state technology similar to the memory in your computer.



DAY 3

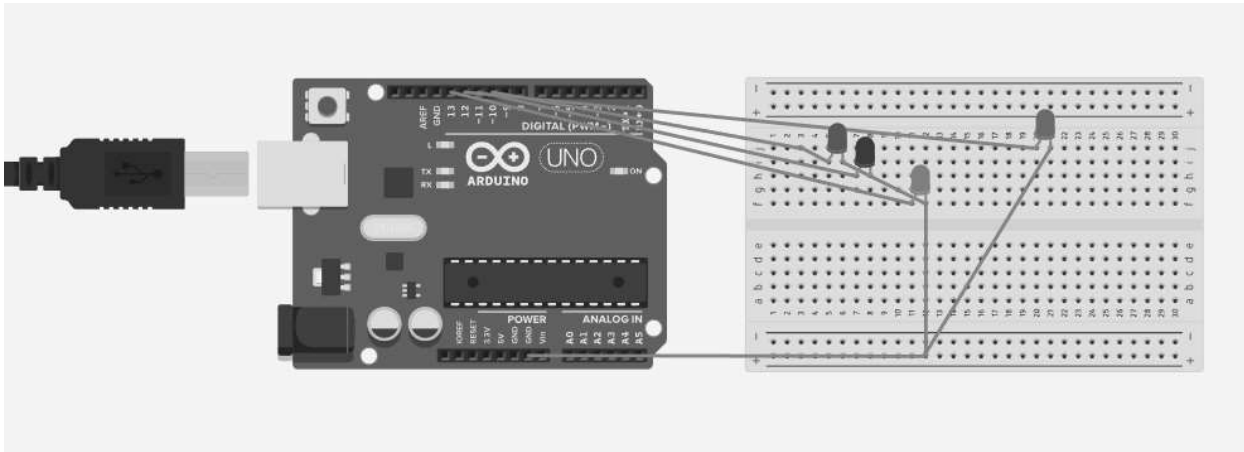
LED BLINKING



CODE

```
void setup() {

  pinMode(8, OUTPUT);
}

void loop() {
  digitalWrite(8, HIGH);
  delay(1000);
  digitalWrite(8, LOW);
  delay(1000);
}
```

MULTIPLE LED BLINKING

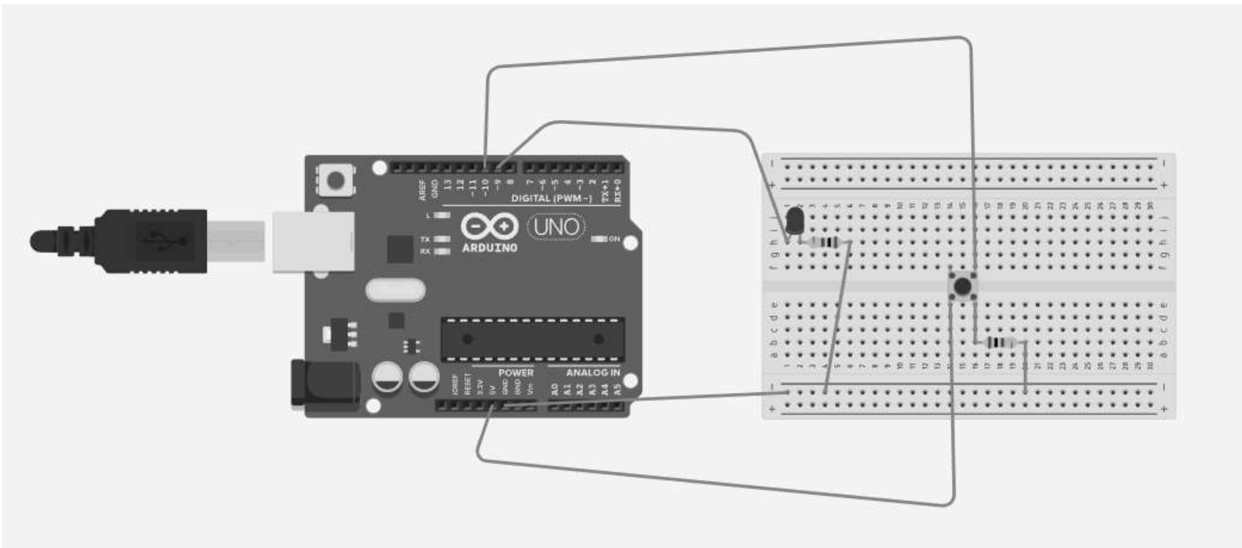


CODE

```
void setup()
{
pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
digitalWrite(10,HIGH);
  delay(1000);
    digitalWrite(10,LOW);
    delay(1000);
  digitalWrite(11,HIGH);
  delay(1000);
    digitalWrite(11,LOW);
    delay(1000);
  digitalWrite(12,HIGH);
  delay(1000);
    digitalWrite(12,LOW);
    delay(1000);
digitalWrite(13,HIGH);
  delay(1000);
    digitalWrite(13,LOW);
    delay(1000);
}
```

DAY 4

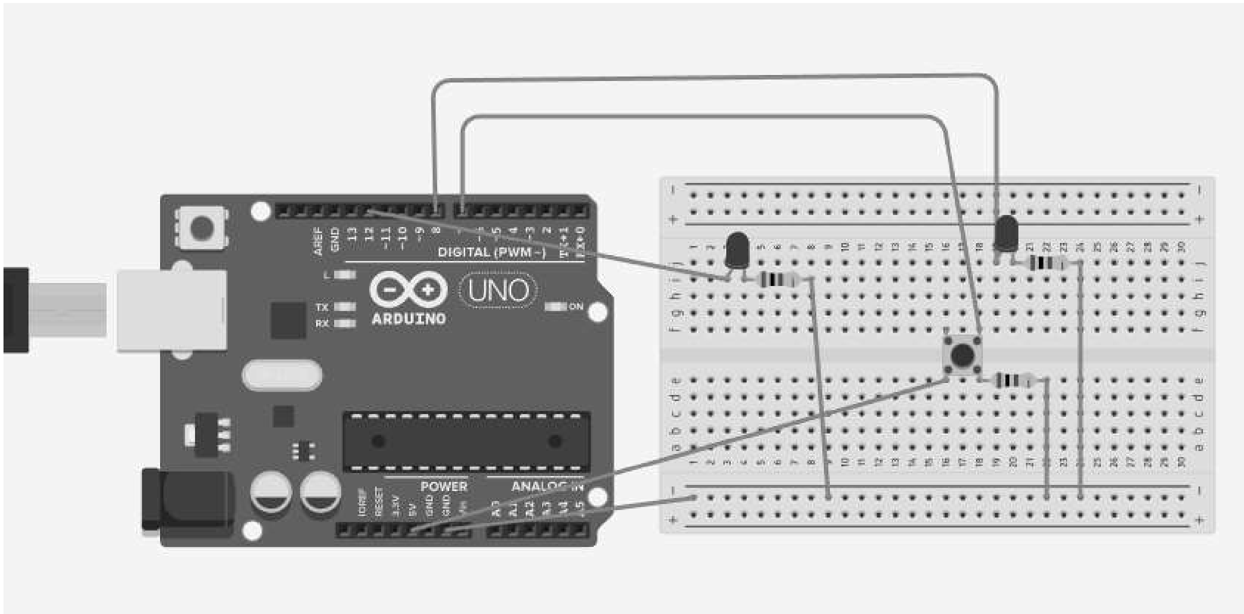
LED ON/OFF USING SWITCH



CODE

```
void setup()
{
  pinMode(9, OUTPUT);
  pinMode(10, INPUT);
}
void loop()
{
 if(digitalRead(10)== HIGH)
   {
    digitalWrite(9, HIGH);
 }
{
     (digitalWrite(9,LOW));
 }
}
```

MULTIPLE LED ON/OFF USING SWITCH



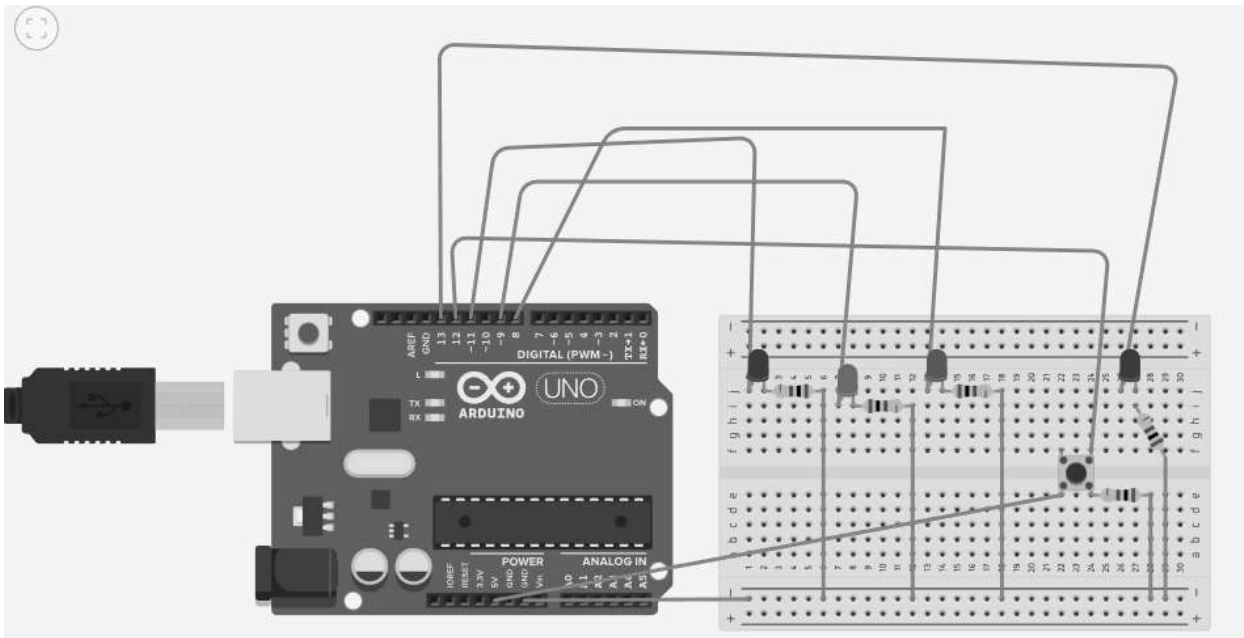
CODE

```
int i=0;
void setup()
{
  pinMode(12, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, INPUT);
}
void loop()
{
  delay(100);
  if(digitalRead(7)==HIGH)
  {
    i++;
  }
    if(i%2==0)
    {
  digitalWrite(12, HIGH);
   digitalWrite(8, LOW);
      }
  else if(i%2==1)
    {
  digitalWrite(8, HIGH);
  digitalWrite(12, LOW);
  }
  else
{
  digitalWrite(12, LOW);
```

```
    digitalWrite(8, LOW);
}
```

DAY 5

MULTIPLE LED ON/OFF USING SWITCH

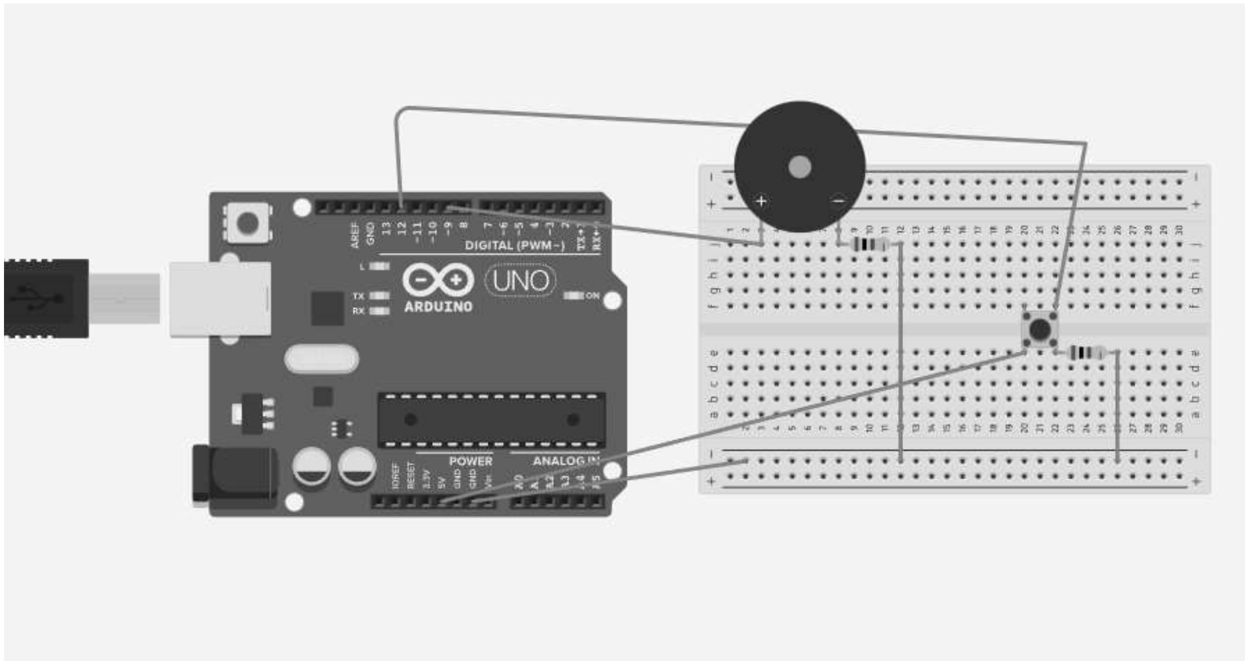


CODE

```
int i = 0;
void setup() {
  pinMode(12, INPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(11, OUTPUT);
   pinMode(13, OUTPUT);
}
void loop()  {
  int b;
  b = digitalRead(12);
  delay(200);
  if (b == HIGH)
        {
    ++i;
    }
```

```
 if (i == 0) {
   digitalWrite(11, LOW);
   digitalWrite(9, LOW);
   digitalWrite(13, LOW);
   digitalWrite(8, LOW);
 }
else if (i == 1)
{
   digitalWrite(11, HIGH);
   digitalWrite(9, LOW);
   digitalWrite(13, LOW);
   digitalWrite(8, LOW);
 }
   else if (i == 2) {
   digitalWrite(8, LOW);
   digitalWrite(9, HIGH);
   digitalWrite(11, LOW);
   digitalWrite(13, LOW);
 }
  else if (i == 3) {
   digitalWrite(9, LOW);
   digitalWrite(11, LOW);
   digitalWrite(8, HIGH);
   digitalWrite(13, LOW);
   }
  else if (i == 4) {
   digitalWrite(9, LOW);
   digitalWrite(8, LOW);
   digitalWrite(11, LOW);
   digitalWrite(13, HIGH);
 }

 else {
   i = 1;
 }
}
```

BUZZER ON OFF USING SWITCH



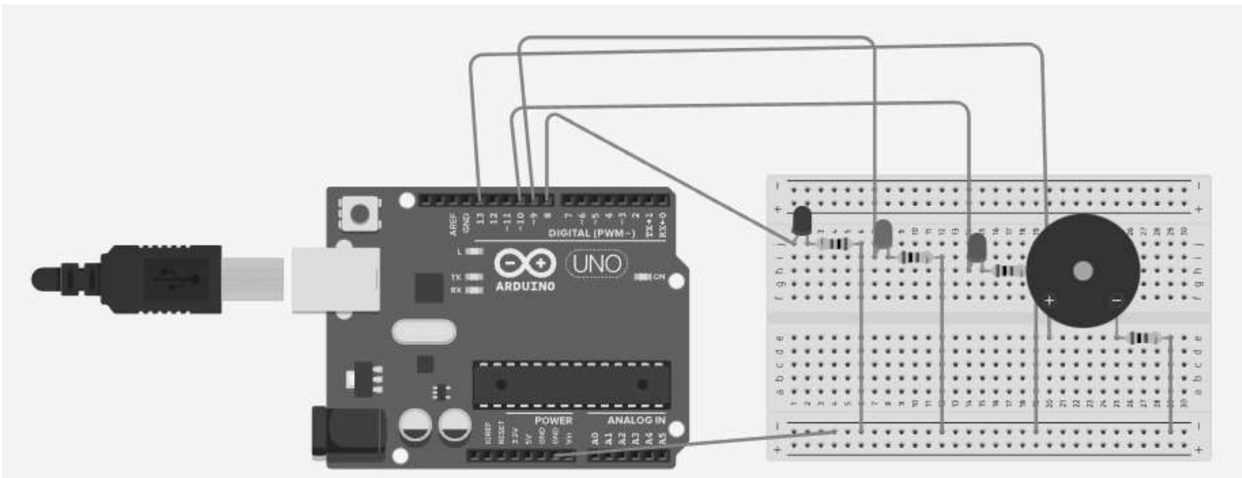
CODE

```
int i=0;
void setup()
{
 pinMode(9, OUTPUT);
  pinMode(12, INPUT);
}

void loop()
{
 if(digitalRead(12)==HIGH)
 {
  i++;
 }
 if(i%2==1)
 {
 digitalWrite(9, HIGH);
 delay(1000);
 }
 else
 {
 digitalWrite(9, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
}
```

}

PROJECT 1 -

TRAFIC LIGHT SYSTEM USING LED AND BUZZER



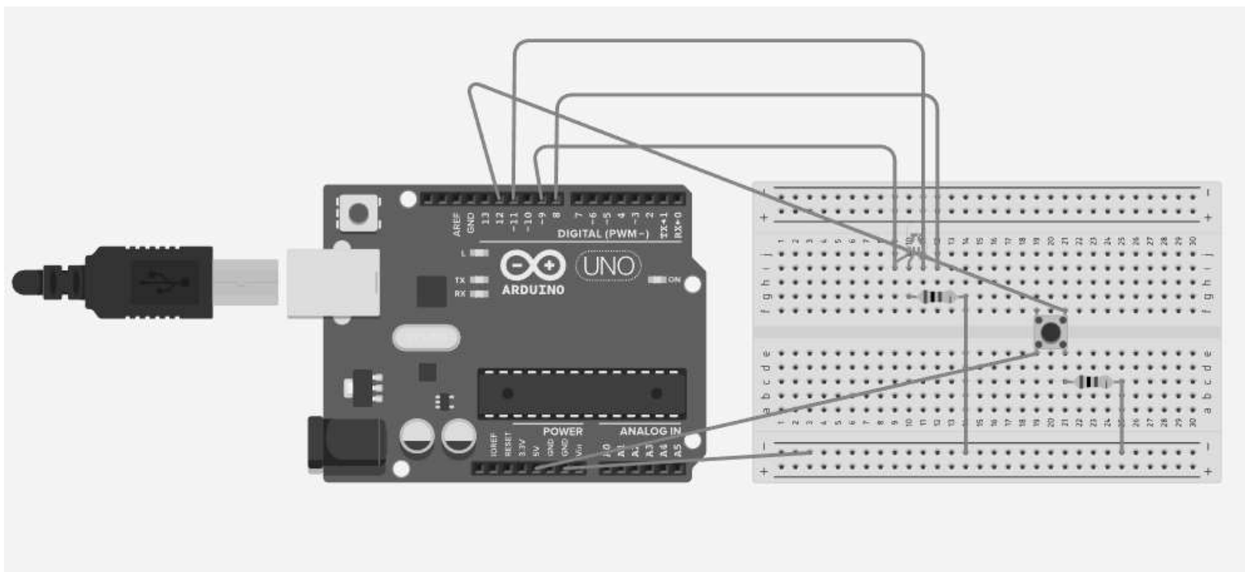
CODE

```
void setup()
{
  pinMode(8, OUTPUT);
   pinMode(9, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(8, HIGH);
  delay(3000);
  digitalWrite(8, LOW);
  delay(300);
  {
  digitalWrite(9, HIGH);

  digitalWrite(13, HIGH);
   delay(3000);
  digitalWrite(13, LOW);

   digitalWrite(9, LOW);
  delay(300);
  }
  digitalWrite(10, HIGH);
```

```
  delay(3000);
 digitalWrite(10, LOW);
 delay(300); }
```

DAY 8

RGB USING SWITCH



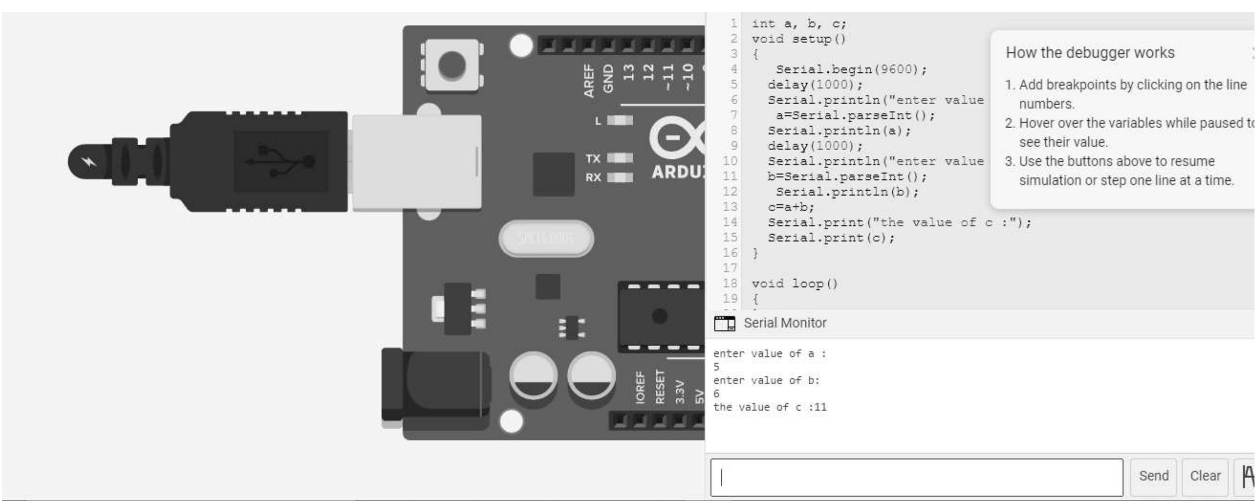
CODE

```
int i = 0;
void setup() {
 pinMode(12, INPUT);
 pinMode(9, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(11, OUTPUT);
 }
void loop() {
 int b;
 b = digitalRead(12);
 if (b == HIGH) {
   i++;
   delay(300);
 }
  else if (i == 0) {
   digitalWrite(9, LOW);
   digitalWrite(8, LOW);
   digitalWrite(11, LOW);
   digitalWrite(13, LOW);
```

```
}
 else if (i == 1) {
   digitalWrite(9, HIGH);
   delay(300);
   digitalWrite(8, LOW);
   digitalWrite(11, LOW);
     }
  else if (i == 2) {
   digitalWrite(9, LOW);
   digitalWrite(8, HIGH);
   delay(300);
   digitalWrite(11, LOW);
    }
  else if (i == 3) {
   digitalWrite(9, LOW);
   digitalWrite(8, LOW);
   digitalWrite(11, HIGH);
    delay(300);
}
 else {
   i = 1;
 }
}
```

PROGRAMS USING SERIAL MONITOR

PROGRM OF ADDITION OF TWO NUMBERS



CODE

```
int a, b, c;
void setup()
```

```
{
  Serial.begin(9600);
 delay(1000);
 Serial.println("enter  value of a :");
  a=Serial.parseInt();
 Serial.println(a);
 delay(1000);
 Serial.println("enter  value of b:");
 b=Serial.parseInt();
  Serial.println(b);
 c=a+b;
 Serial.print("the  value of c :");
 Serial.print(c);
}
void loop()
{
}
```

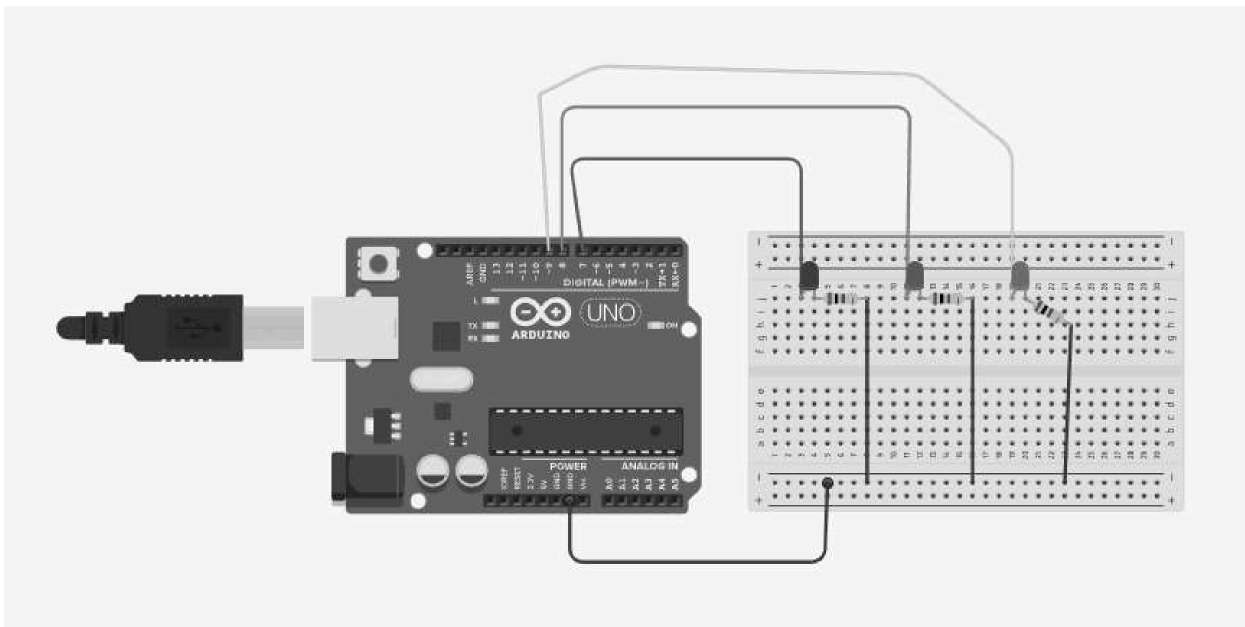
PROGRAM TO DO MULTIPLE MATHEMATICAL OPERATIONS USING SERAIL MONITER

```
int a, b, c,d,e,f;
void setup()
{
  Serial.begin(9600);
 delay(1000);
 Serial.println("enter  value of a :");
  a=Serial.parseInt();
 Serial.println(a);
 delay(1000);
 Serial.println("enter  value of b:");
 b=Serial.parseInt();
  Serial.println(b);
 c=a+b;
 Serial.print("the  value of c :");
 Serial.print(c);
d=a-b;
Serial.print("the  value of d :");
 Serial.print(d);
e=a*b;
Serial.print("the  value of e :");
 Serial.print(e);
f=a/b;
Serial.print("the  value of f :");
 Serial.print(f);
```

```
}
void loop()
{    }
```

DAY 9

ALTERNATE LED ON/OFF USING SERIAL MONITOR



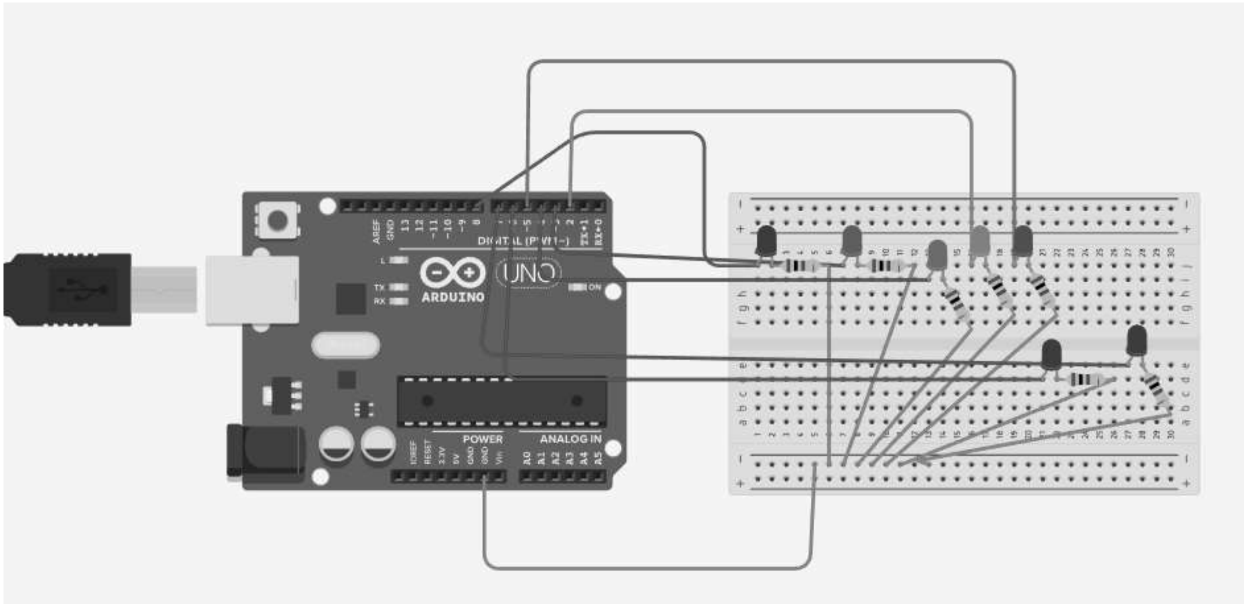
CODE

```
int Rled=7;
int Gled=8;
int Yled=9;
char input;
void setup()
{
  Serial.begin(9600);
  Serial.println("press  1 to on Red LED");
  Serial.println("2  to on green LED");
  Serial.println("3  for yellow LED");
  pinMode(Rled, OUTPUT);
  pinMode(Gled, OUTPUT);
  pinMode(Yled, OUTPUT);
}
void loop()
{
```

```
 if(Serial.available()>0)
  input=Serial.read();
 {
  if(input=='1')
  {
  digitalWrite(Rled, HIGH);
    digitalWrite(Yled, LOW);
    digitalWrite(Gled, LOW);
  }
  else if(input=='2')
  {
  digitalWrite(Gled, HIGH);
    digitalWrite(Rled, LOW);
    digitalWrite(Yled, LOW);
  }
  else if(input=='0')
  {
  digitalWrite(Gled, LOW);
    digitalWrite(Rled, LOW);
    digitalWrite(Yled, LOW);
  }
  else if(input=='3')
  {
  digitalWrite(Yled, HIGH);
   digitalWrite(Rled, LOW);
    digitalWrite(Gled, LOW);
  }
  else if(input=='0')
   {
  digitalWrite(Yled, HIGH);
    delay(300);
   digitalWrite(Rled, HIGH);
    delay(300);
    digitalWrite(Gled, HIGH);
  }
  else
  {
    digitalWrite(Yled, HIGH);
   digitalWrite(Rled, HIGH);
    digitalWrite(Gled, HIGH);}
    }
  }
```

PROJECT 2 -

RANDOM LED ON/OFF USING SERIAL MONITOR



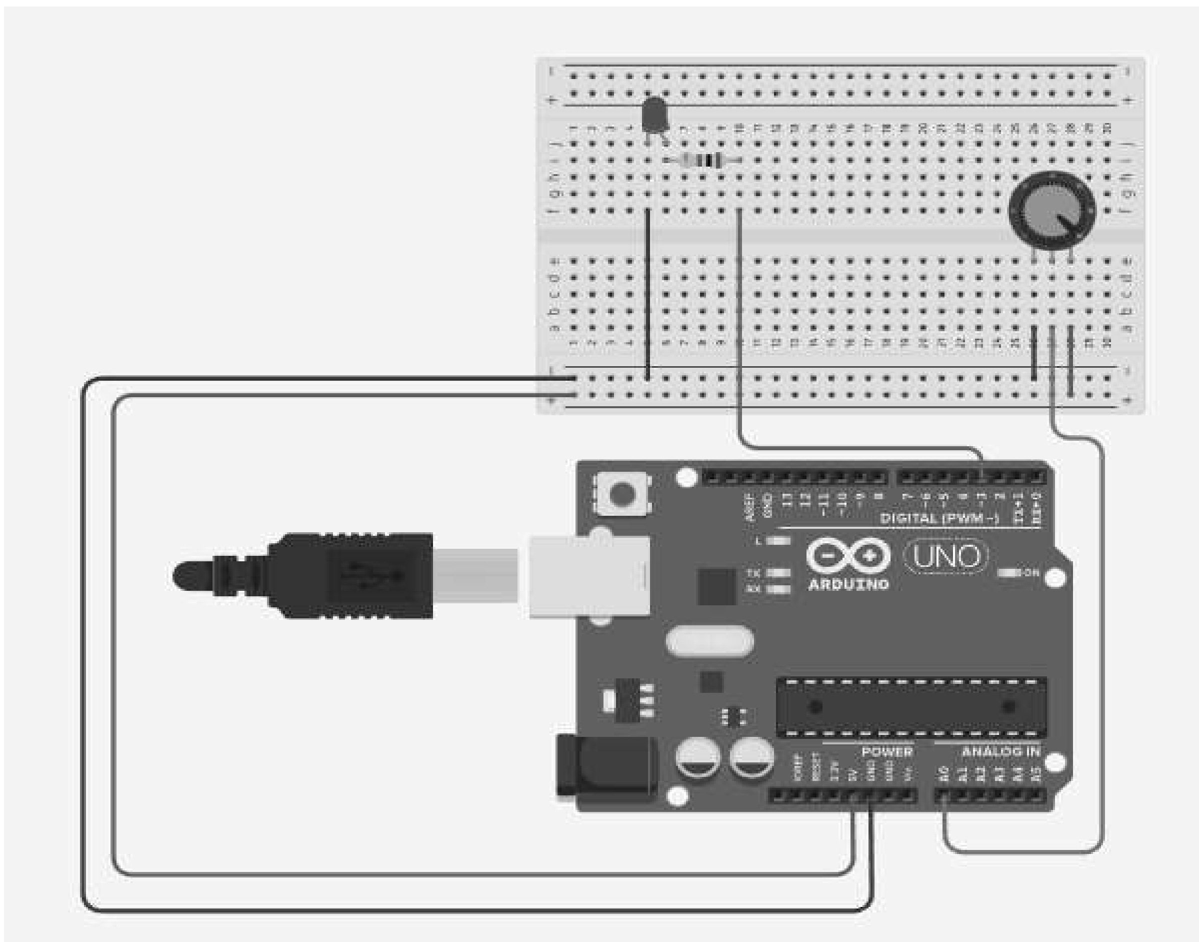
CODE

```
int led1 = 8;
int led2 = 2;
int led3 = 3;
int led4 = 4;
int led5 = 5;
int led6 = 6;
int led7 = 7;
long randomNumber;
void setup()
{
  pinMode(1 ,OUTPUT);
  pinMode(2 ,OUTPUT);
  pinMode(3 ,OUTPUT);
  pinMode(4 ,OUTPUT);
  pinMode(5 ,OUTPUT);
  pinMode(6 ,OUTPUT);
  pinMode(7 ,OUTPUT);
  Serial.begin(9600);
}
void loop(){
 randomNumber=random(2,8);
 Serial.println(randomNumber);
digitalWrite(randomNumber,HIGH);
   delay(2000);
  digitalWrite(randomNumber,LOW);
 delay(100);
```

```
}
```

DAY 10

CONTROLING BRIGHTNESS OF LED USING POTENTIOMETER

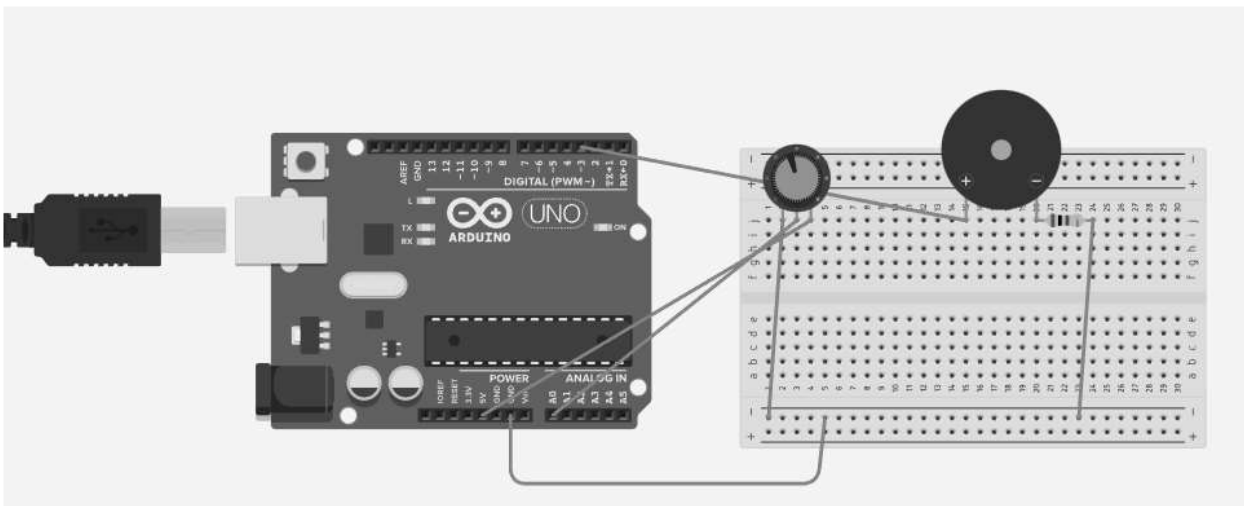


CODE

```
#include <SoftwareSerial.h>
#define LED 3
#define KNOB 0
void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop() {
```

```
  int val = analogRead(KNOB);
  int ledPower = map(val,  1, 1024, 1, 255);
  String stringOne = "Sensor value:  ";
  Serial.println(stringOne + ledPower);
  analogWrite(LED, ledPower);
}
```

CONTROLING BUZZER WITH POTENTIOMETER



CODE

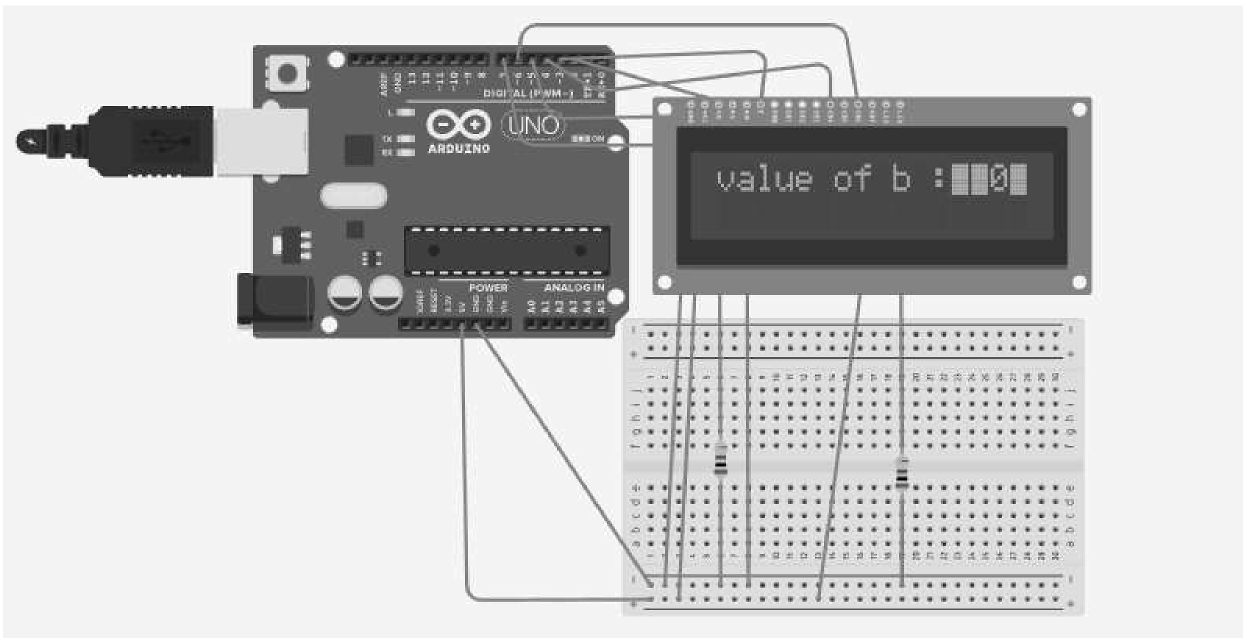
```
void setup()
{
  pinMode(3, OUTPUT);
  pinMode(A0, INPUT);
}

void loop()
{
  int potvalue = analogRead(A0);
  int BUZZERON = map(potvalue, 0, 1023, 0, 255);
  analogWrite(3,BUZZERON);
}
```

DAY 11

LCD-

This 2×16-character LCD Module with YELLOW Backlight uses an I2C interface to communicate with the host microcontroller. This budget-conscious LCD is used on projects requiring the display of text, data, or ASCII characters of all types. This is a output device which is used to display output.

LIVE DATA PRINTING USING LCD



CODE

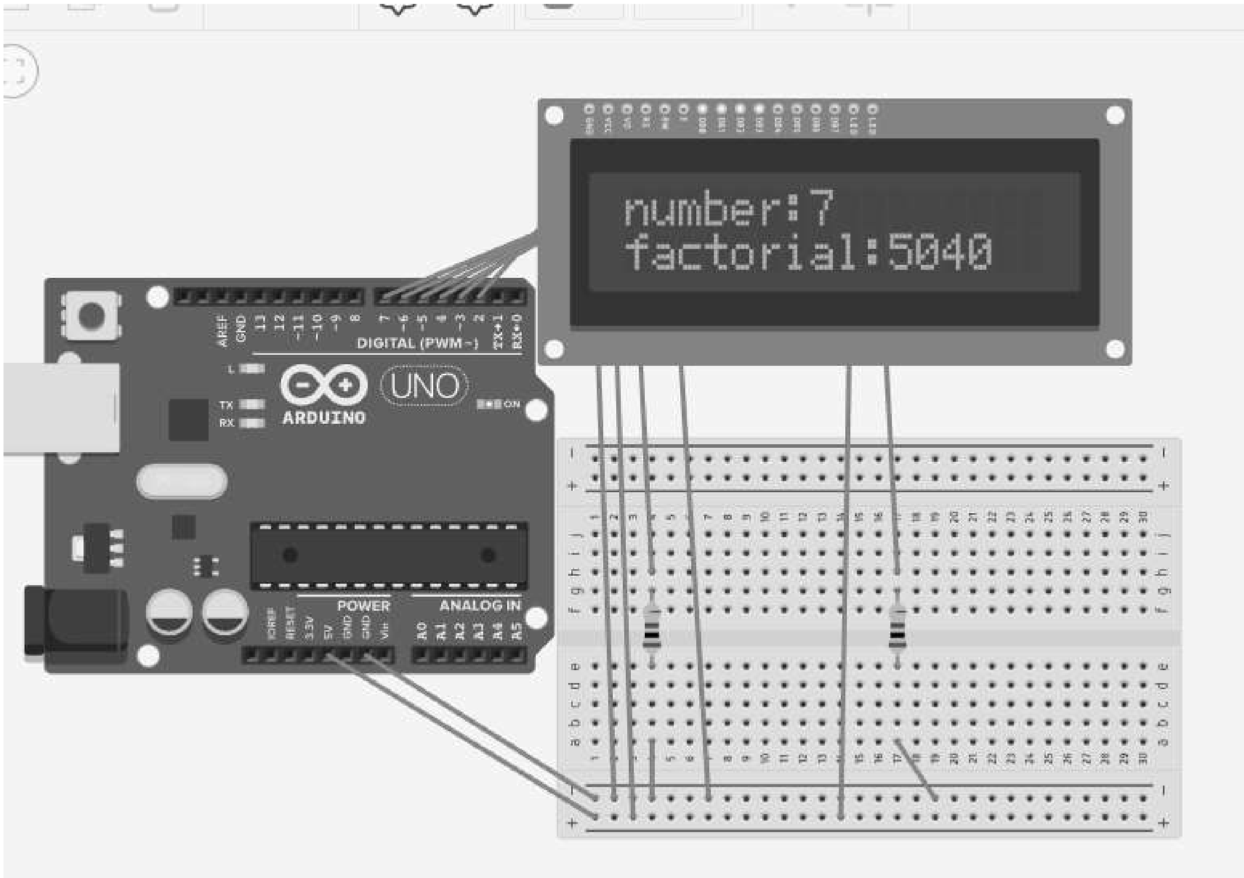
```
#include<LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);
int a, b, c, d, e, g, f;
void setup()
{
 lcd.begin(16,2);
  Serial.begin(9600);
 delay(1000);
 Serial.println("enter  two numbers for multiplication");
 Serial.println("enter  value of a :");
  a=Serial.parseInt();
 Serial.println(a);
 delay(2000);
 Serial.print("enter  value of b:");
 b=Serial.parseInt();
 Serial.println(b);
 c=a*b;
```

```
  Serial.print("the value after multiplication :");
  Serial.println(c);
  d=a+b;
  Serial.print("the value after addition:");
  Serial.println(d);
  e=a-b;
  Serial.print("the value after subtraction :");
  Serial.println(e);
  f=a/b;
  Serial.print("the value after division:");
  Serial.println(f);
  g=a%b;
  Serial.print("the value after modulus:");
  Serial.println(g);
  lcd.println("value of a :");
  lcd.println(a);
  delay(2000);
  lcd.setCursor(0,0);
  lcd.println("value of b :");
  lcd.println(b);
  delay(2000);
  lcd.setCursor(0,0);
   lcd.println("value of c :");
  lcd.println(c);
  delay(2000);
  lcd.setCursor(0,0);
   lcd.println("value of d :");
  lcd.println(d);
  delay(2000);
  lcd.setCursor(0,0);
  lcd.println("value of e :");
  lcd.println(e);
  delay(2000);
  lcd.setCursor(0,0);
   lcd.println("value of f :");
  lcd.println(f);
  delay(2000);
  lcd.setCursor(0,0);
   lcd.println("value of g :");
  lcd.println(g);
  delay(2000);
  lcd.setCursor(0,0);
  delay(2000);
  lcd.print("byeee :)");
}
void loop()
```

```
{
}
```

PROJECT 3

PRINTING FACTORIAL OF RANDOM NUMBER ON LCD USING SRIAL MONITOR



CODE

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);
int randomNumber,n;
int long fact =1;
void setup()
{
  lcd.begin(16,2);
  Serial.begin(9600);
  delay(1000);
  randomNumber=random(-9,20);
  Serial.println("randomNumber");
  Serial.println(randomNumber);
```

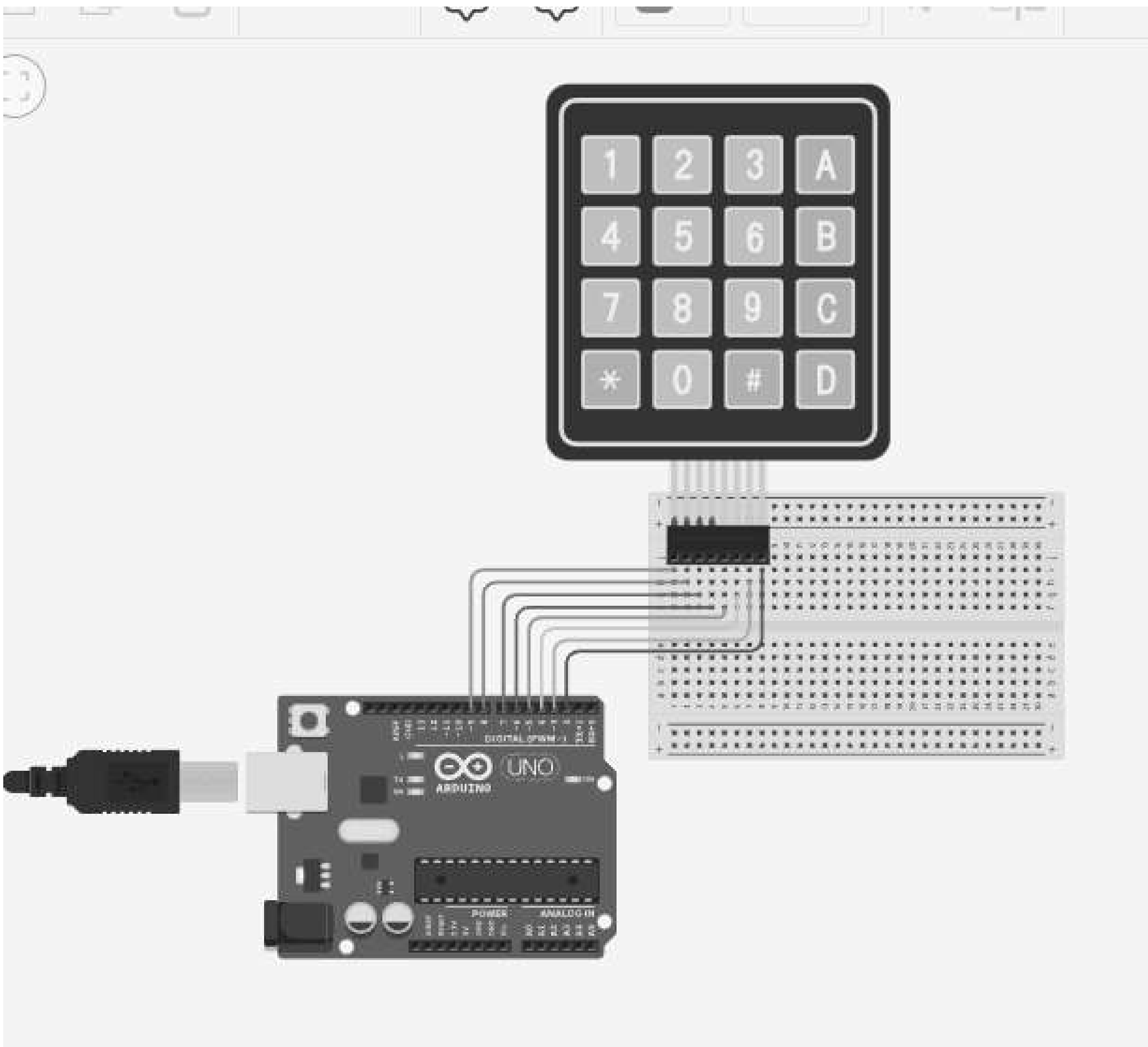
```
  lcd.setCursor(0,0);
    delay(1000);
   lcd.print("number:");
    lcd.print(randomNumber);
  if (randomNumber<0)
  {
Serial.println("fact  doesn't exists");
  lcd.setCursor(0,0);
    delay(1000);
     lcd.print("fact not exists");
}
  else {
    for(n=1;n<=randomNumber;++n)
     {
    fact=fact*n;
     }
  Serial.println("factorial");
  Serial.println(fact);
  lcd.setCursor(0,1);
    delay(1000);
   lcd.print("factorial:");
    lcd.print(fact);
}
}
void loop()
{
}
```

DAY 12

KEYPAD-

A keypad is a block or pad of buttons set with an arrangement of digits, symbols, or alphabetical letters. Pads mostly containing numbers and used with computers are numeric keypads. Keypads are found on devices which require mainly numeric input such as calculators, television remotes, push-button telephones, vending machines, ATMs, point of sale terminals, combination locks, safes, and digital door locks

KEYPAD INTERFACING USING ARDUINO

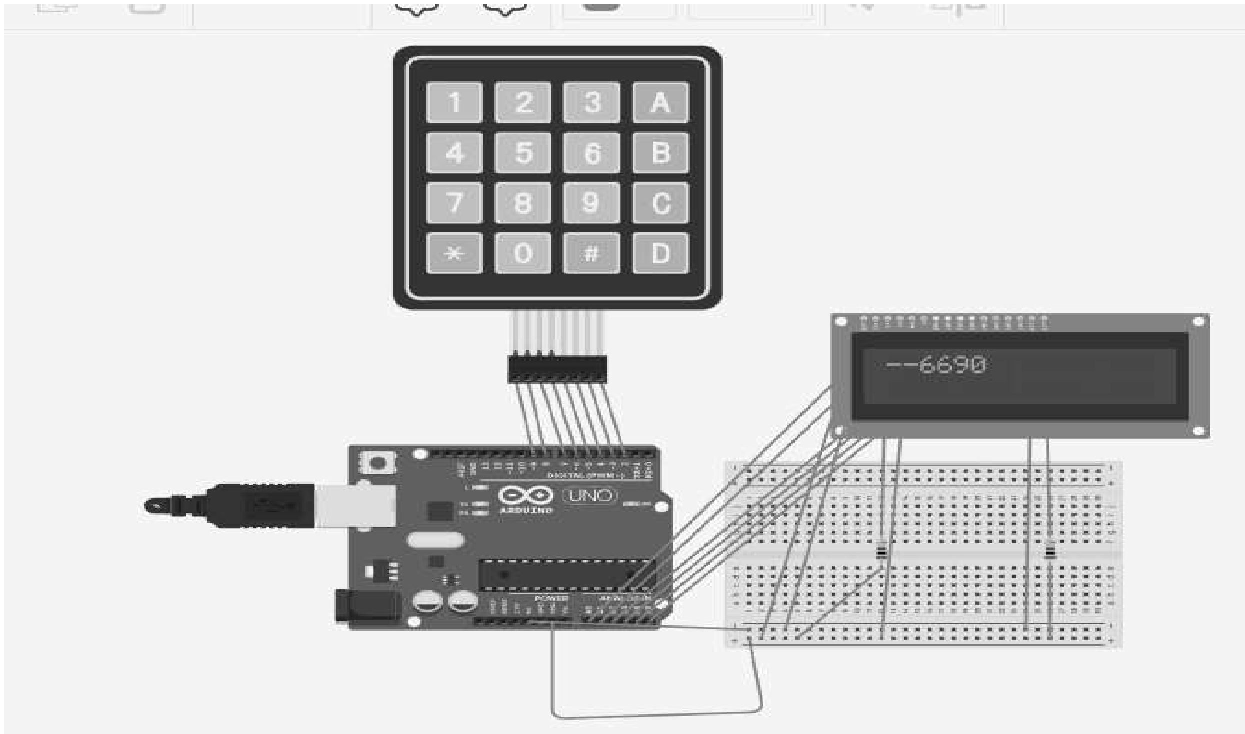


CODE

```
#include <Keypad.h>
const byte numRows= 4;
const byte numCols= 4;
char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
byte rowPins[numRows] = {9,8,7,6};
byte colPins[numCols]=  {5,4,3,2};
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);
void setup()
{
Serial.begin(9600);
}
void loop()
{
   char customKey = customKeypad.getKey();
  if(customKey)
  {
   Serial.print(customKey);
  }
  }
```

PROJECT 4-

PRINTING LIVE DATA ON LCD USING KEYPAD



CODE

```
#include <Keypad.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(14,15,16,17,18,19);
int op, customKey;
double a,b,c;
const byte ROWS=4;
const byte COLS=4;
char keys[ROWS][COLS]=
{
  {'1','2','3','+'},
  {'4','5','6','-'},
  {'7','8','9','*'},
  {'c','0','=','/'},
};
byte rowPins[ROWS] = {9,8,7,6};
byte colPins[COLS] = {5,4,3,2};
Keypad customKeypad = Keypad(makeKeymap(keys),rowPins,colPins,ROWS,COLS);
void setup()
{
  lcd.setCursor(0,0);
```

```
  lcd.begin(16,2);
  Serial.begin(9600);
  lcd.print("hello");
  delay(1000);
  lcd.clear();
  }
void loop()
 {
 char  a = customKeypad.getKey();
  if(a)
  {
    Serial.println(a);
    lcd.print(a); }}
```

DAY 15

SEVEN SEGEMENT DISPLAY-

A 7 segment display is one of the oldest forms of the electronic display device for displaying decimal numerals in embedded applications. It is An alternative to the complex dot matrix displays.
These displays have 8 Light-emitting diodes (LEDs) inside for displaying numbers and alphabets.
You can trace the use of these gadgets as far back as 1903 by an individual known as Carl Kinsley. He was the inventor of a method to transmit letters and numbers telegraphically.

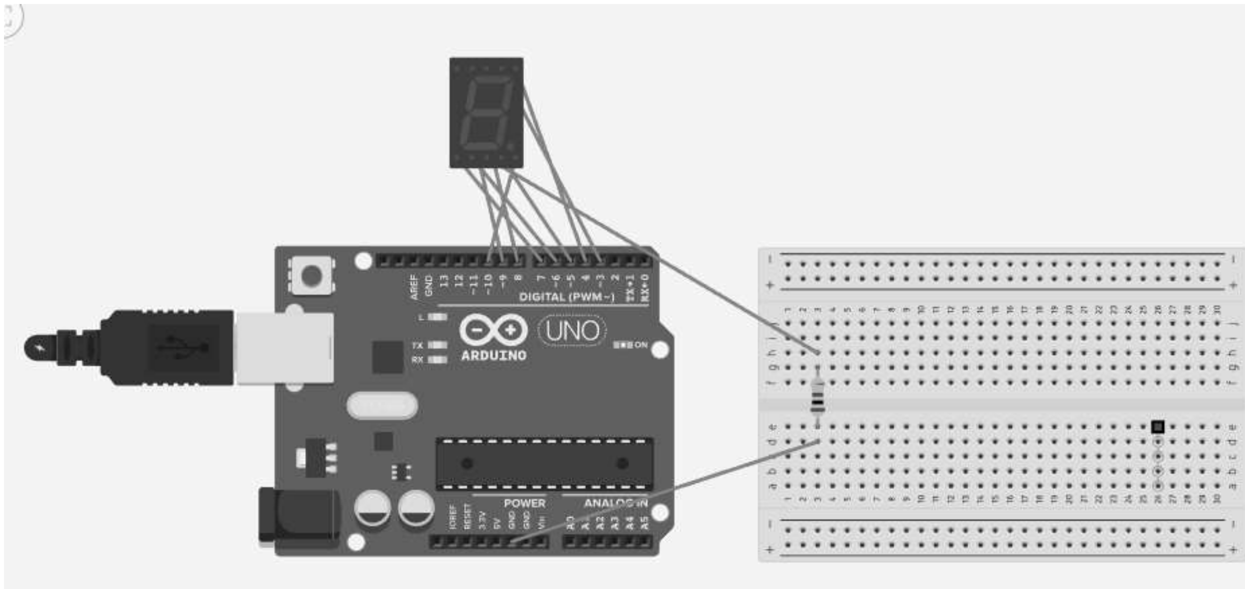
FUNCTIONING

A seven-segment display uses Light Emitting Diodes to release light energy in photons. The production emits light to show digits in all seven segments, with the eighth segment being a decimal point.
While the phenomenon happens, bear in mind that an LED is a solid-state optical p-n junction diode. Hence it allows current flow through its junction.
The diode favors current from external voltage to flow forward, and this process is known as electroluminescence. The colors range from red to blue to green depending on the various impurities in the semiconductor materials that help produce it.
The decimal point comes in handy when connecting two or more seven-segment displays. The reason is to show decimal digits or give a multi-digit presentation.
Preference for LEDs is due to reasons such as:

- their cheapness,
- long life,
- small size,
- availability in various colors,
- easy availability and
- It is easy to integrate them into other electronic components and digital circuits.

However, their main advantage is that you can fit multiple of them in a small compact package forming a seven-segment display. A concept possible thanks to the display's small size.

SEVEN SEGEMENT DISPLAY INTERFACING



CODE

```
int A=3;
  int B=4;
  int C=5;
  int D=6;
  int E=7;
  int F=8;
  int G=9;
  int H=10;
  int i;
void setup()
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(H, OUTPUT);
  Serial.begin(9600);

}
void loop()
{
 Serial.println(i);
 delay(1000);
  i++;
```

```
  {
  switch(i)
  {
    case 0:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
    case 1:
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
case 2:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 3:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
```

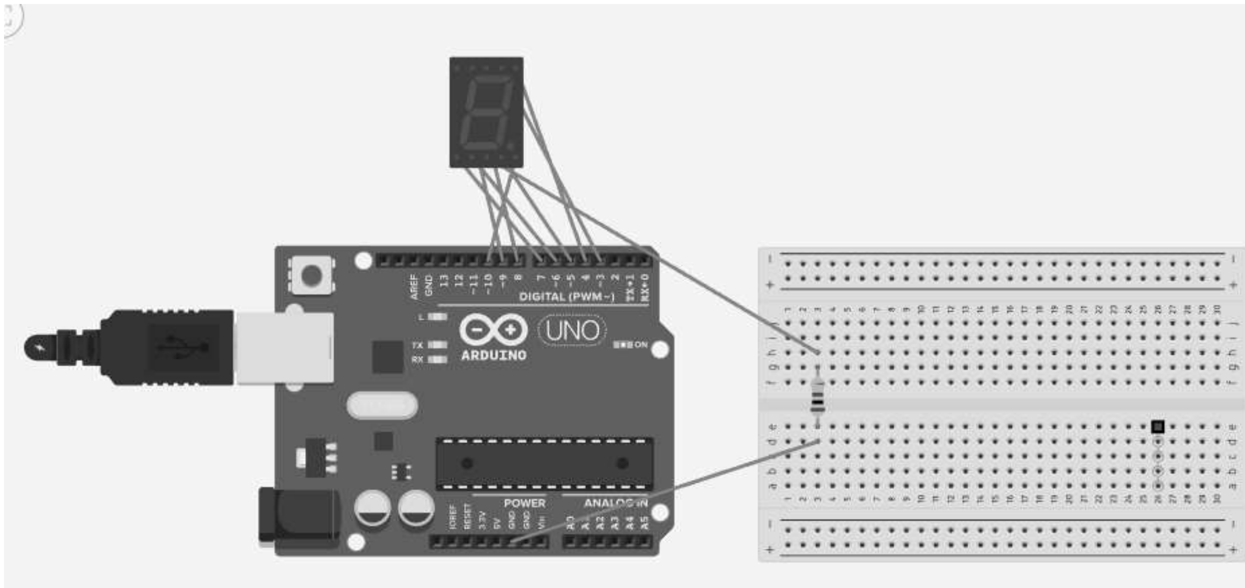
```
break ;
case 4:
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 5:
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW );
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
case 6:
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 7:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 8:
```

```
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
case 9:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
   break ;
   case 10:
    digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
   break ;
    case 11:
    digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
   break ;
    case 12:
    digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
```

```
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
     delay(2000);
    break ;
    case 13:
     digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
     delay(2000);
    break ;
    case 14:
     digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
    break ;
     case 15:
     digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
  }  } }
```

PROJECT 5-

PRINTING LIVE DATA ON 7-SEGENEMT DISPLAY USING SERIAL MONITOR



CODE

```
int A=3;
  int B=4;
  int C=5;
  int D=6;
  int E=7;
  int F=8;
  int G=9;
  int H=10;
  int i;
void setup()
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(H, OUTPUT);
  Serial.begin(9600);

}
void loop()
{
  Serial.println("enter  value of i(0 to 15) :");
```

```
 i=Serial.parseInt();
 Serial.println(i);
 delay(1000);
  {
  switch(i)
  {
    case 0:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
    case 1:
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
case 2:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 3:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
```

```
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 4:
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 5:
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW );
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
case 6:
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 7:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
```

```
     delay(2000);
break ;
case 8:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
break ;
case 9:
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
    break ;
    case 10:
     digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
    break ;
     case 11:
     digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
     delay(2000);
    break ;
```

```
    case 12:
    digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
    delay(2000);
   break ;
   case 13:
    digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
    delay(2000);
   break ;
   case 14:
    digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
   break ;
    case 15:
    digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
    delay(2000);
break ;
  }
  }
```

}

DAY 16

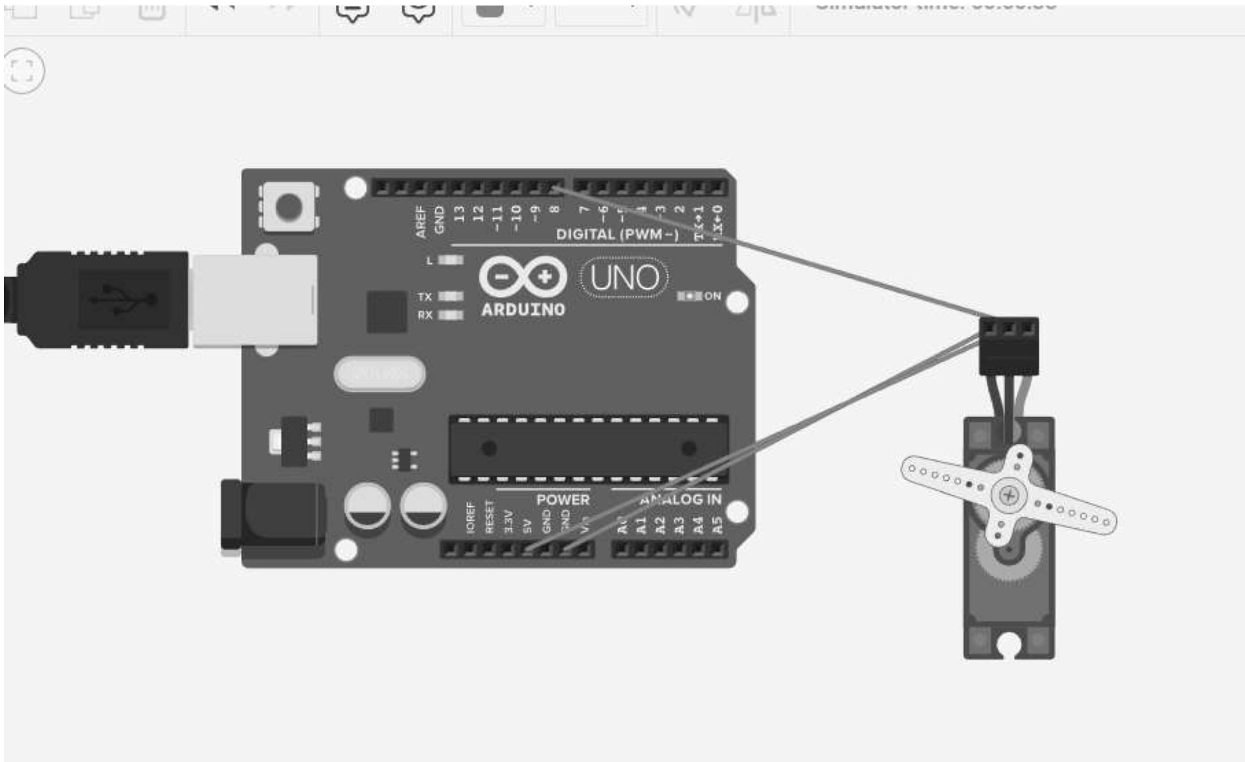
SERVO MOTOR-

A servomotor (or servo motor) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.
Servomotors are not a specific class of motor, although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system.
Servomotors are used in applications such as robotics, CNC machinery, and automated manufacturing.

MECHANISM

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.
The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.
The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.
More sophisticated servomotors use optical rotary encoders to measure the speed of the output shaft] and a variable-speed drive to control the motor speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

SERVO MOTOR INTERFACING

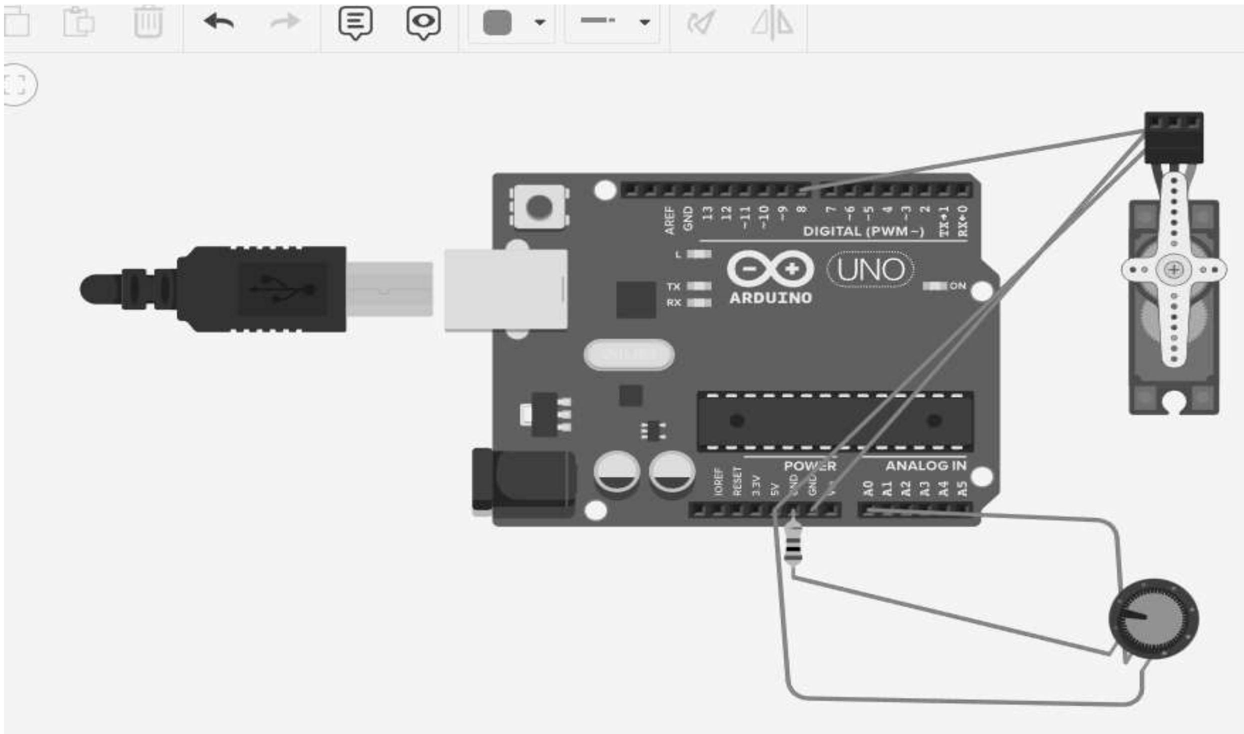


CODE

```
#include<Servo.h>
Servo motion;
int i;
void setup()
{
  Serial.begin(9600);
  motion.attach(8);
}

void loop()
{

 for(i=0;i<=180;i++)
 {
  motion.write(i);
  delay(75);
 }

 for(i=180;i>=0;i--)
 {
  motion.write(i);
```

```
   delay(75);
  }
}
```

CONTROLING SERVOMOTOR USING POTENTIOMETER



CODE

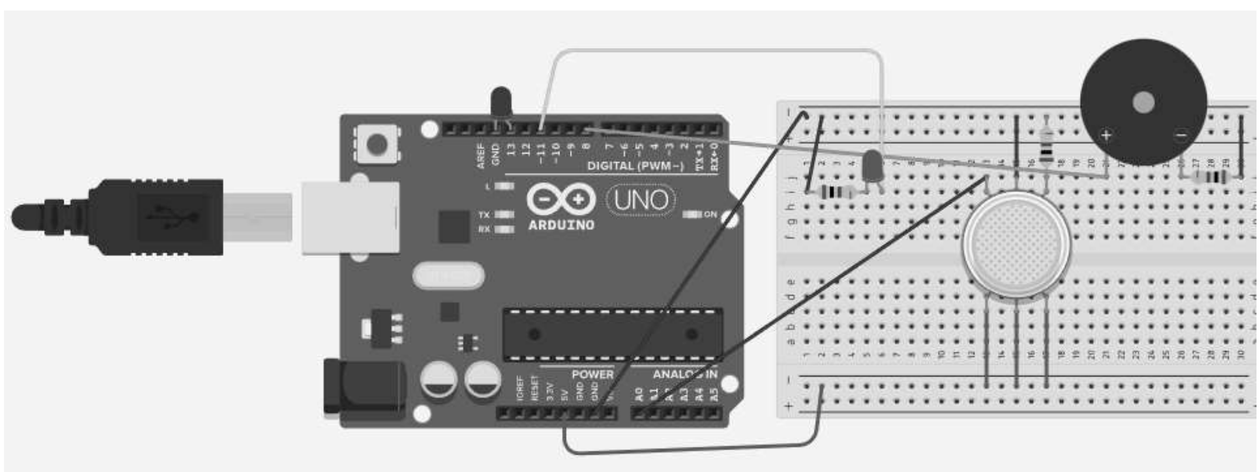
```
#include<Servo.h>
Servo servo;
int val;
int potpin = A0;
void setup()
{
  Serial.begin(9600);
  pinMode(potpin, INPUT);
  servo.attach(8);
}
void loop()  {
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 180);
 servo.write(val);
  delay(15);
  Serial.println(val);
  delay(2000);  }
```

DAY 17

GAS SENSOR-

Gas sensors are devices that help us understand the amount of gas in the environment and the natural state of its movement. Gas sensors reveal the amount of gas in the environment and the nature of the gas composition with electrical signals and can provide its change
Gas sensors are generally understood as providing a measurement of the concentration of some analyte of interest, such as CO, CO_2, NO_x, SO_2, without at this point dwelling on the plethora of underlying approaches such as optical absorption, electrical conductivity, electrochemical (EC), and catalytic bead (see Section 3). However, and as discussed in Section 2, many other gas sensors measure a physical property of the environment around them, such as simple temperature, pressure, flow, thermal conductivity, and specific heat, or more complex properties such as heating value, super compressibility, and octane number for gaseous fuels. The latter may require capital-intensive (engines) or destructive testing, for example, via combustion, or involve the measurement of a number of parameters to serve as inputs to a correlation with the complex property of interest.

GAS SENSOR INTERFACING



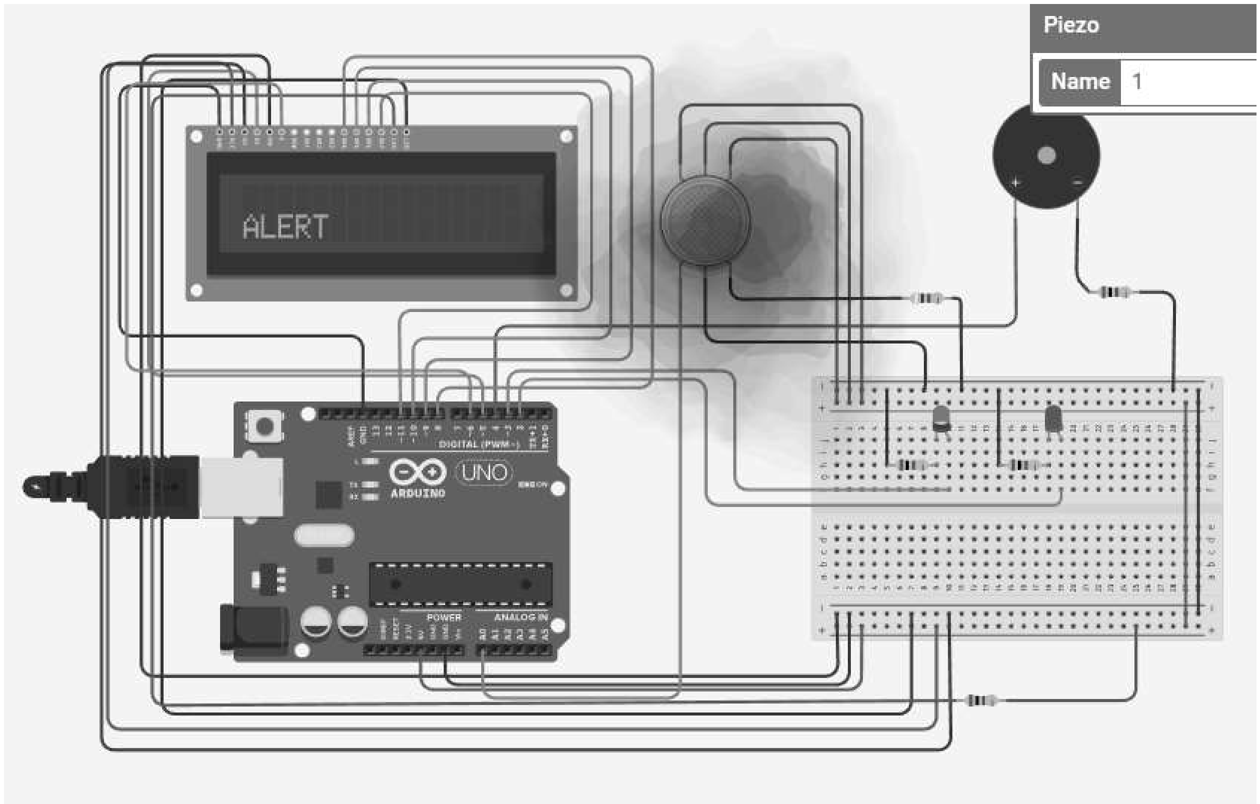
CODE

```
int redLed = 13;
int greenLed = 9;
int buzzer = 12;
int sensor = A1;
int warning=400;
void setup()
{
```

```
pinMode(redLed, OUTPUT);
   pinMode(greenLed, OUTPUT);
   pinMode(buzzer, OUTPUT);
   pinMode(sensor, INPUT);
  Serial.begin(9600);
}
void loop()
{
int analogValue= analogRead(sensor);
Serial.println(analogValue);
if(analogValue>warning)
{
digitalWrite(greenLed, HIGH);
  digitalWrite(redLed, LOW);
  noTone(buzzer);
  delay(500);
}
  else
  {
  digitalWrite(greenLed, LOW);
   tone(buzzer,100);
  digitalWrite(redLed, HIGH);
  delay(500);
  }
}
```

PROJECT - 6

SMOKE DETECTION SYSTEM



CODE

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(5,6,8,9,10,11);

int redled=3;
int greenled=2;
int buzzer=4;
int sensor= A0;
int sensorThresh=400;
void setup()
{
pinMode(redled,OUTPUT);
pinMode(greenled,OUTPUT);
pinMode(buzzer,OUTPUT);
pinMode(sensor,INPUT);
Serial.begin(9600);
lcd.begin(16,2);
}
void loop()
{
int analogValue= analogRead(sensor);
Serial.print(analogValue);
```

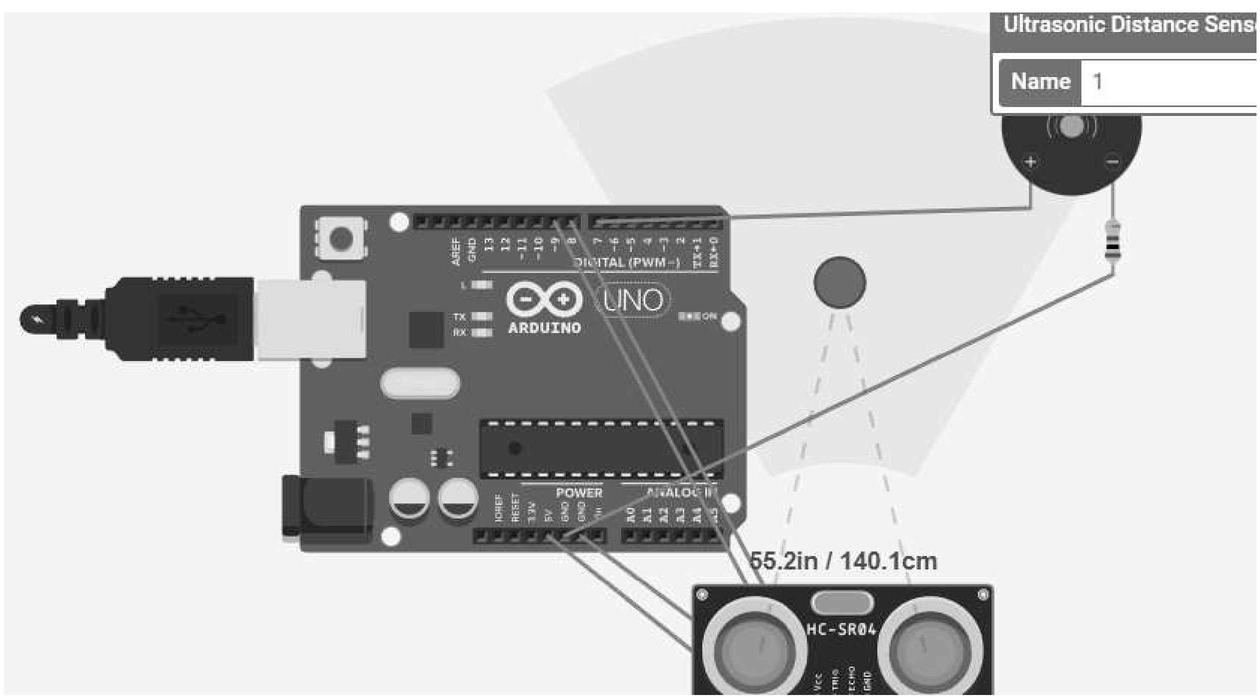
```
if(analogValue>sensorThresh)
{

digitalWrite(redled,HIGH);
digitalWrite(greenled,LOW);
tone(buzzer,1000,10000);
lcd.clear();
lcd.setCursor(0,1);
lcd.print("ALERT");
delay(1000);
lcd.clear();
lcd.setCursor(0,1);
lcd.print("EVACUATE");
delay(1000);
}
else
{
digitalWrite(greenled,HIGH);
digitalWrite(redled,LOW);
noTone(buzzer);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("SAFE");
delay(1000);
lcd.clear();
lcd.setCursor(0,1);
lcd.print("ALL CLEAR");
delay(1000);
}
}
```

DAY 18

ULTRASONIC SENSOR –

Ultrasonic Sensors also known as transceivers when they both send and receive work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object. This technology can be used for measuring: wind speed and direction (anemometer), fullness of a tank and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure the amount of liquid in a tank, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms and non-destructive testing. Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed. The technology is limited by the shapes of surfaces and the density or consistency of the material. For example foam on the surface of a fluid in a tank could distort a reading.

ULTRASONIC SENSOR INTERFACING

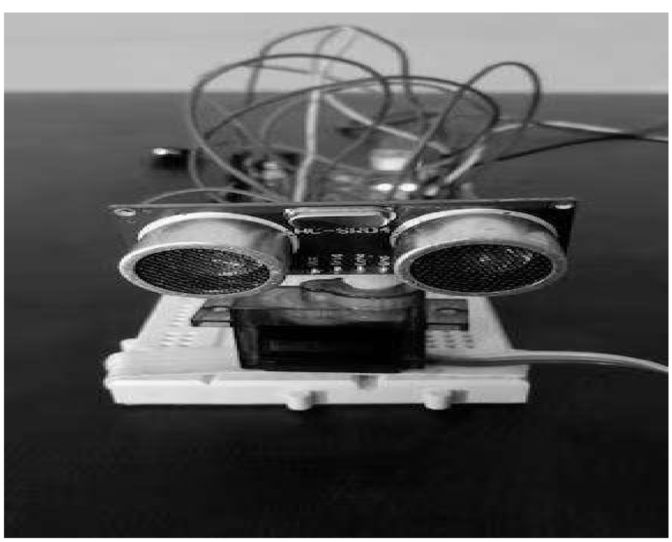


CODE

```
long time;
double distance;
```

```
#define speed 0.034
int triger = 9;
int echo = 8;
int buzzer = 7;
void setup()
{
  Serial.begin(9600);
  pinMode(triger, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(echo, INPUT);
}
void loop()
{
  digitalWrite(triger, LOW);
  delayMicroseconds(4);
  digitalWrite(triger, HIGH);
  delayMicroseconds(4);
   digitalWrite(triger, LOW);
  delayMicroseconds(4);
  time=pulseIn(echo,HIGH,12000);
    distance=0.5*speed*time;
  Serial.println(distance);
  delay(300);
 if(distance>0)
 {
digitalWrite(buzzer, HIGH);
} else
 {
digitalWrite(buzzer, LOW);
 }}
```

PROJECT -7



CODE

```
#include<Servo.h>
Servo srv;
long time;
#define speed 0.034
double distance;
int i;
void setup()
{
  Serial.begin(9600);
  pinMode(11, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(12, INPUT);
  srv.attach(7);
}
void loop()
{
  for(i=0;i<=180;i++)
  {
    srv.write(i);
    delay(75);
      srv.write(i);
    delay(75);
  digitalWrite(11, LOW);
  delayMicroseconds(4);
  digitalWrite(11, HIGH);
  delayMicroseconds(4);
  digitalWrite(11, LOW);
 time=pulseIn(12,HIGH,12000);
    distance=0.5*speed*time;
  Serial.println(distance);
  if(distance>0)
  {
   digitalWrite(13, HIGH);
    delay(1000);
  }
  else
  {
    digitalWrite(13, LOW);
    delay(1000);
    }
   for(i=180;i>=0;i--)
```

```
{
  srv.write(i);
  delay(75);
 digitalWrite(11, LOW);
 delayMicroseconds(4);
 digitalWrite(11, HIGH);
 delayMicroseconds(4);
 digitalWrite(11, LOW);
time=pulseIn(12,HIGH,12000);
  distance=0.5*speed*time;
 Serial.println(distance);
 if(distance>0)
 {
  digitalWrite(13, HIGH);
  delay(1000);
 }
 else
 {
  digitalWrite(13, LOW);
  delay(1000);
 }
 }
 }
}
```

DAY 19

PIR SENSOR

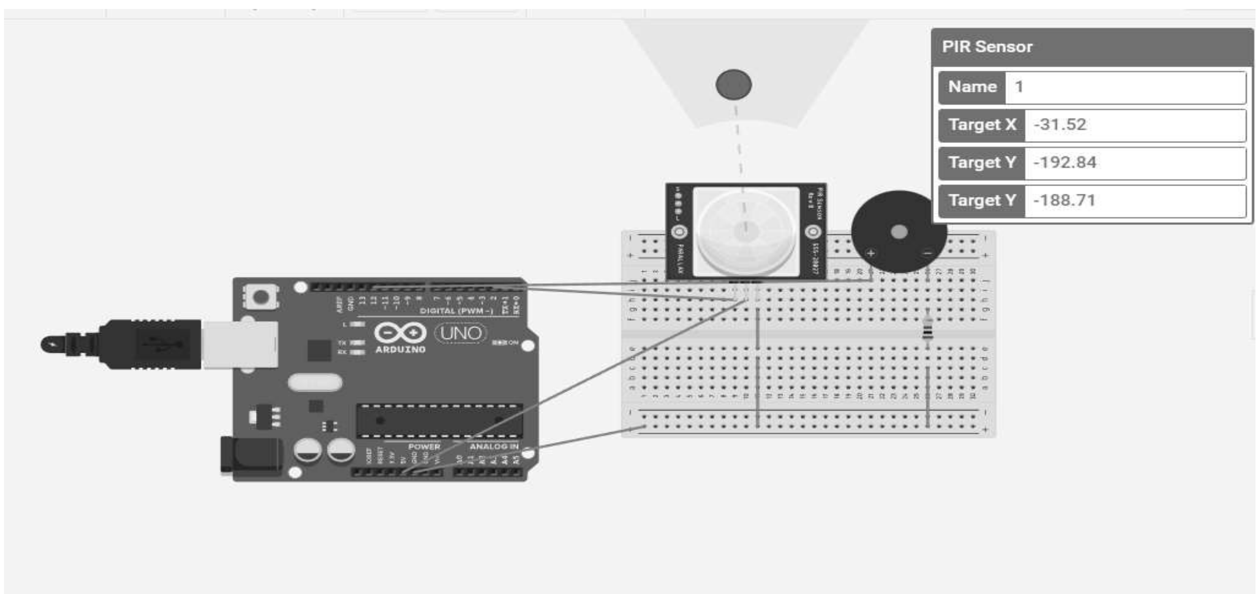
A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors are commonly used in security alarms and automatic lighting applications. PIR sensors detect general movement, but do not give information on who or what moved. For that purpose, an imaging IR sensor is required.
PIR sensors are commonly called simply "PIR", or sometimes "PID", for "passive infrared detector". The term passive refers to the fact that PIR devices do not radiate energy for detection purposes. They work entirely by detecting infrared radiation (radiant heat) emitted by or reflected from objects

OPERATION

A PIR sensor can detect changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. When an object, such as a person, passes in front of the background, such as a wall, the

temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. Objects of similar temperature but different surface characteristics may also have a different infrared emission pattern, and thus moving them with respect to the background may trigger the detector as well.
PIRs come in many configurations for a wide variety of applications. The most common models have numerous Fresnel lenses or mirror segments, an effective range of about 10 meters (30 feet), and a field of view less than 180°. Models with wider fields of view, including 360°, are available, typically designed to mount on a ceiling. Some larger PIRs are made with single segment mirrors and can sense changes in infrared energy over 30 meters (100 feet) from the PIR. There are also PIRs designed with reversible orientation mirrors which allow either broad coverage (110° wide) or very narrow "curtain" coverage, or with individually selectable segments to "shape" the coverage.

PIR SENSOR INTERFACING



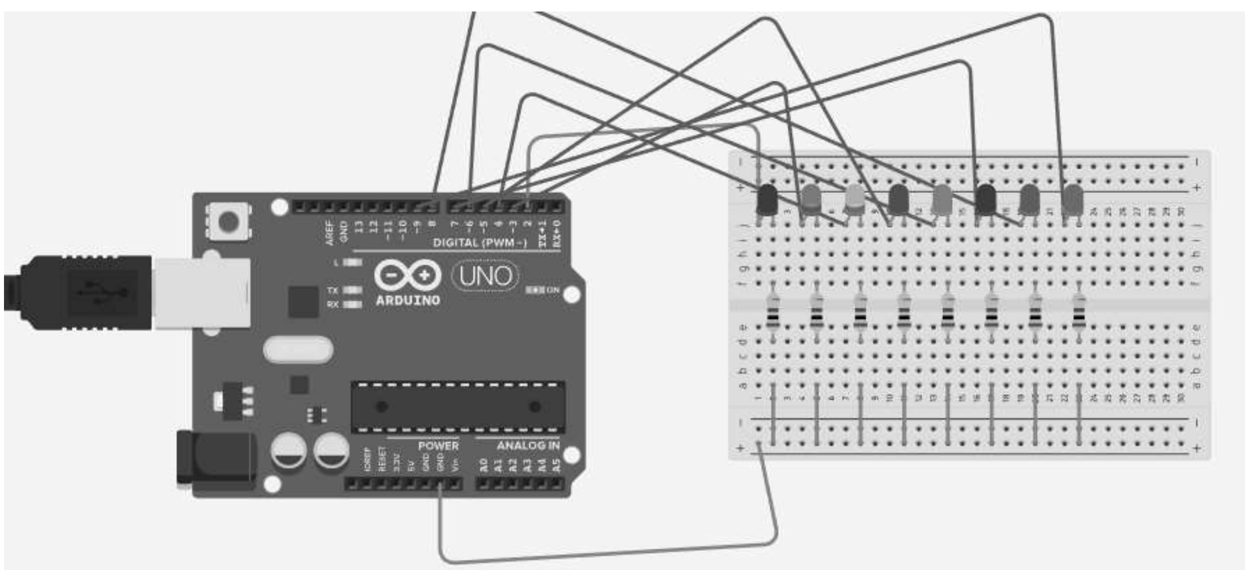
CODE

```
int buzzer = 12;
int i,pirpin = 2;
void setup() {
Serial.begin(9600);
pinMode(buzzer, OUTPUT);
 pinMode(pirpin, INPUT);
}
void loop(){
int pir = digitalRead(pirpin);
```

```
  Serial.println(pir);
 if (pir == HIGH) {
   digitalWrite(buzzer, HIGH);
   delay(500);
   Serial.println("alert!!!");
 }
   else
   {
     Serial.println("safe");
     digitalWrite(buzzer, LOW);
     delay(500);
   }
}
```

DAY 22

LED PATERN USING FOR LOOP



CODE

```
int i;
void setup()
```

```
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}
void loop()
{
  for(i=2;i<=9;i++)
   {
  digitalWrite(i, HIGH);
  delay(1000);
  digitalWrite(i, LOW);
  delay(10);
}
  for(i=9;i>=1;i--)
   {
    digitalWrite(i,LOW);
  delay(1000);
  digitalWrite(i,HIGH);
  delay(10);
}
}
```

DAY 23

PROJECT-7

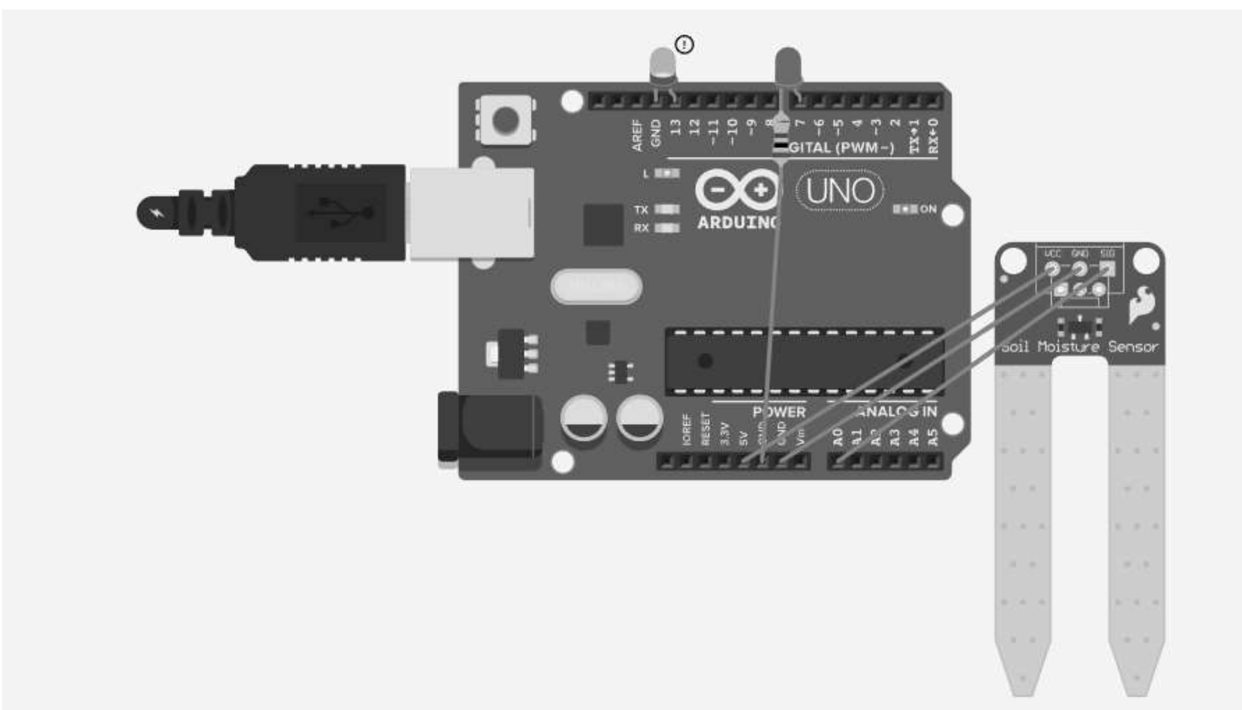
SOIL MOISTURE SENSOR-

oil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.

The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners.

Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.

SOIL MOISTURE SENSOR INTERFACING



CODE

```
int led_pin =13;
```

```
int yled_pin=7;
int sensor_pin =A0;
void setup()
{
 pinMode(led_pin, OUTPUT);
 pinMode(yled_pin, OUTPUT);
 pinMode(sensor_pin, INPUT);
 Serial.begin(9600);
}
void loop() {
 float moisture_percentage;
 int sensor_analog;
 sensor_analog = analogRead(sensor_pin);
 Serial.println(sensor_analog);
 delay(1000);
 moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
 Serial.print("Moisture Percentage = ");
 Serial.print(moisture_percentage);
 Serial.print("%\n\n");
 delay(1000);
 if(moisture_percentage<50){
 digitalWrite(led_pin, HIGH);
 digitalWrite(yled_pin, LOW);
 }
 else {
 digitalWrite(led_pin, LOW);
 digitalWrite(yled_pin, HIGH);
 delay(1000);
}
}
```

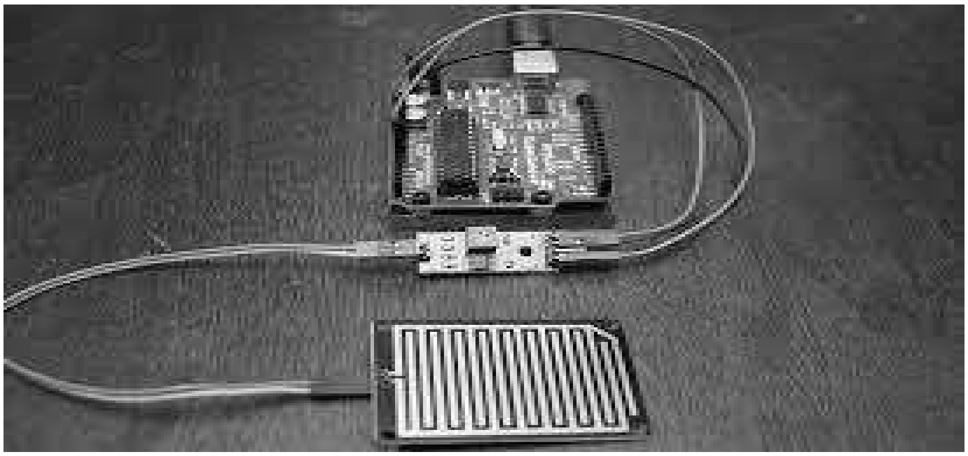
PROJECT-8

RAIN DROP SENSOR MODULE

Raindrop sensor is basically a board on which nickel is coated in the form of lines. It works on the principal of resistance.
Rain Sensor module allows to measure moisture via analog output pins and it provides a digital output when a threshold of moisture exceeds.
The module is based on the LM393 op amp. It includes the electronics module and a printed circuit board that “collects” the rain drops. As rain drops are collected on the circuit board, they create paths of parallel resistance that are measured via the op amp.
The sensor is a resistive dipole that shows less resistance when wet and more resistance when dry. When there is no rain drop on board it increases the Resistance so we gets high voltage according to V=IR.

When rain drop present it reduces the resistance because water is a conductor of electricity and presence of water connects nickel lines in parallel so reduces resistance and reduces voltage drop across it.

RAIN SENSOR INTERFACING



CODE

```
const int capteur_D = 4;
const int capteur_A = A0;
int val_analogique;
void setup()
{
  pinMode(capteur_D, INPUT);
  pinMode(capteur_A, INPUT);
  Serial.begin(9600);
}
void loop()
{
if(digitalRead(capteur_D) == LOW)
  {
    Serial.println("Digital value : wet");
    delay(10);
  }
else
  {
    Serial.println("Digital value : dry");
    delay(10);
  }
val_analogique=analogRead(capteur_A);
 Serial.print("Analog value : ");
 Serial.println(val_analogique);
 Serial.println("");
  delay(1000);
```

```
}
```

DAY 24

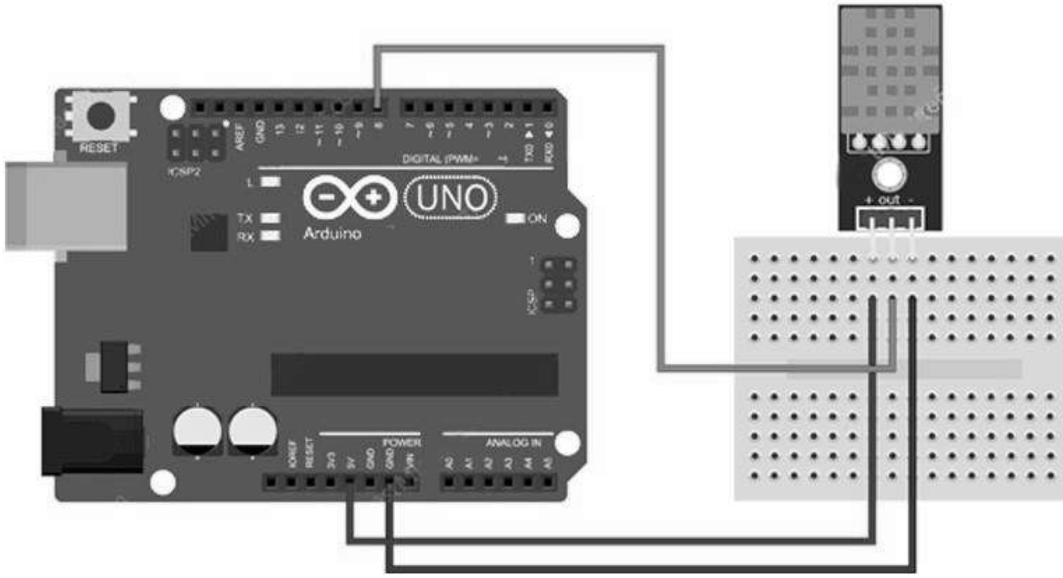
DHT SENSOR

The DHT11 is a commonly used Temperature and humidity sensor for prototypes monitoring the ambient temperature and humidity of a given area.
The sensor can measure temperature from 0°C to 50°C with an accuracy of ±2°C and humidity from 20% to 90% with an accuracy of ±5% RH.

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±2°C and ±5%

DHT SENSOR INTERFACING

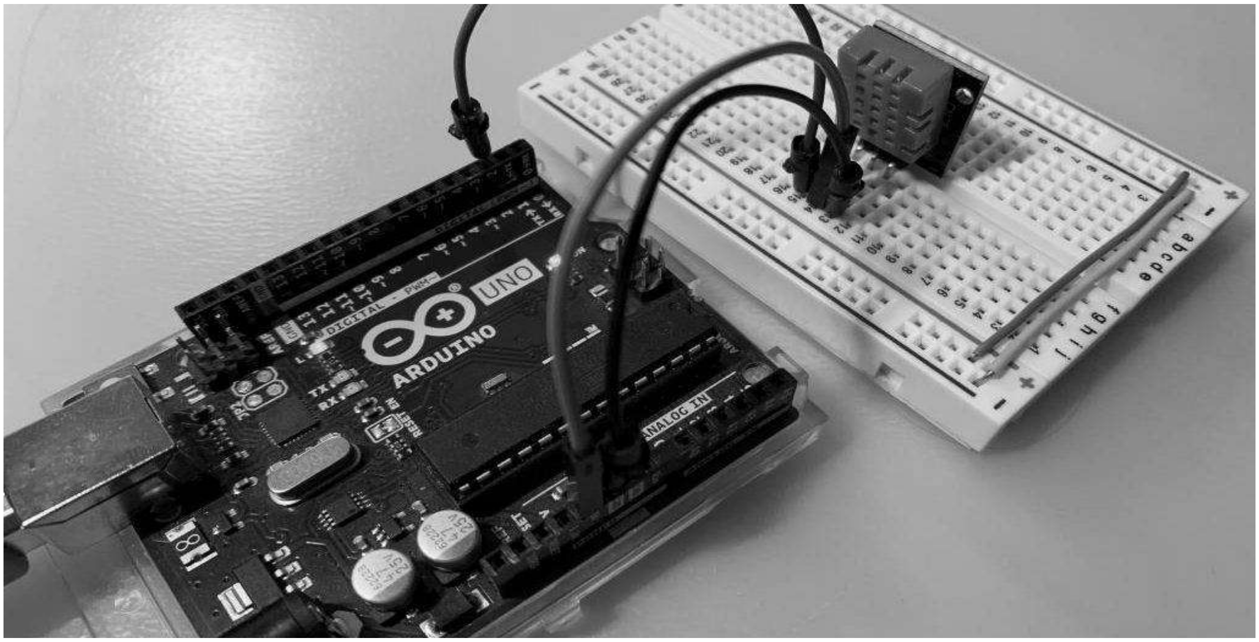


CODE

```
#include <DHT.h>
#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f))
 {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  Serial.print(F(" Humidity: "));
  Serial.println(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("C "));
  delay(2000);
}
```

PROJECT-8

REAL TIME IMPLEMENTATION



CODE

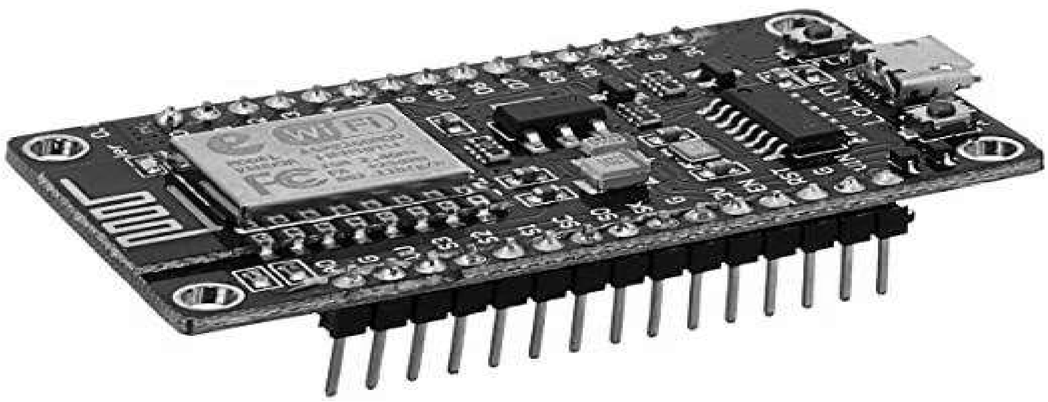
```
#include <DHT.h>
#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f))
 {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  Serial.print(F(" Humidity: "));
  Serial.println(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("C "));
  delay(2000);
}
```

DAY 25

WIFI MODULE

Today, IOT applications are on the rise, and connecting objects are getting more and more important. There are several ways to connect objects such as Wi-Fi protocol.
NodeMCU is an open-source platform based on ESP8266 which can connect objects and let data transfer using the Wi-Fi protocol. In addition, by providing some of the most important features of microcontrollers such as GPIO, PWM, ADC, and etc, it can solve many of the project's needs alone.
The general features of this board are as follows:

- Easy to use
- Programmability with Arduino IDE or IUA languages
- Available as an access point or station
- practicable in Event-driven API applications
- Having an internal antenna
- Containing 13 GPIO pins, 10 PWM channels, I2C, SPI, ADC, UART, and 1-Wire



CODE

```
#include<ESP8266WiFi.h>
void setup()
{
 Serial.begin(115200);
WiFi.begin("Redmi","123456789");
 while(WiFi.status()!=WL_CONNECTED)
 {
  Serial.println("not connected try again");
  delay(500);
  }
  Serial.println("connected");
  Serial.println(WiFi.localIP());
}
void loop() {}
```

DAY 26

OSI MODLE-

The Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network. It was the first standard model for network communications, adopted by all major computer and telecommunication companies in the early 1980s
The modern Internet is not based on OSI, but on the simpler TCP/IP model. However, the OSI 7-layer model is still widely used, as it helps visualize and communicate how networks operate, and helps isolate and troubleshoot networking problems.
OSI was introduced in 1983 by representatives of the major computer and telecom companies, and was adopted by ISO as an international standard in 1984.

LAYERS OF OSI MODLE-

7. Application Layer-

The application layer is used by end-user software such as web browsers and email clients. It provides protocols that allow software to send and receive information and present meaningful data to users. A few examples of application layer protocols are the Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP), and Domain Name System (DNS).

6. Presentation Layer-

The presentation layer prepares data for the application layer. It defines how two devices should encode, encrypt, and compress data so it is received correctly on the other end. The presentation layer takes any data transmitted by the application layer and prepares it for transmission over the session layer.

5. Session Layer-

The session layer creates communication channels, called sessions, between devices. It is responsible for opening sessions, ensuring they remain open and functional while data is being transferred, and closing them when communication ends. The session layer can also set checkpoints during a data transfer—if the session is interrupted, devices can resume data transfer from the last checkpoint.

4. Transport Layer-

The transport layer takes data transferred in the session layer and breaks it into “segments” on the transmitting end. It is responsible for reassembling the segments on the receiving end, turning it back into data that can be used by the session layer. The transport layer carries out flow control, sending data at a rate that matches the connection speed of the receiving device, and error control, checking if data was received incorrectly and if not, requesting it again.

3. Network Layer-

The network layer has two main functions. One is breaking up segments into network packets, and reassembling the packets on the receiving end. The other is routing packets by discovering the best path across a physical network. The network layer uses network addresses (typically Internet Protocol addresses) to route packets to a destination node.

2. Data Link Layer-

The data link layer establishes and terminates a connection between two physically-connected nodes on a network. It breaks up packets into frames and sends them from source to destination. This layer is composed of two parts—Logical Link Control (LLC), which identifies network protocols, performs error checking and synchronizes frames, and Media Access Control (MAC) which uses MAC addresses to connect devices and define permissions to transmit and receive data.

1.Physical Layer-

The physical layer is responsible for the physical cable or wireless connection between network nodes. It defines the connector, the electrical cable or wireless technology connecting the devices, and is responsible for transmission of the raw data, which is simply a series of 0s and 1s, while taking care of bit rate control.

ADVANTAGE OF OSI MODLE-

The OSI model helps users and operators of computer networks:

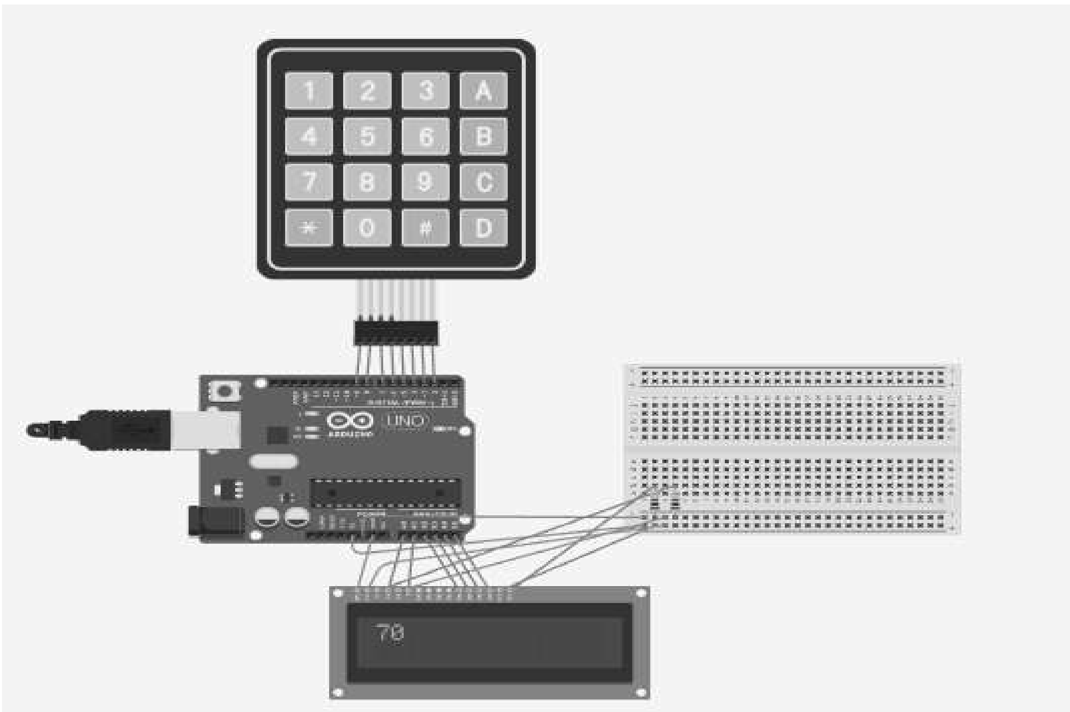
- Determine the required hardware and software to build their network.
- Understand and communicate the process followed by components communicating across a network.
- Perform troubleshooting, by identifying which network layer is causing an issue and focusing efforts on that layer.

The OSI model helps network device manufacturers and networking software vendors:

- Create devices and software that can communicate with products from any other vendor, allowing open interoperability
- Define which parts of the network their products should work with.
- Communicate to users at which network layers their product operates – for example, only at the application layer, or across the stack.

DAY 27 & 28

PROJECT CALCULATOR



CODE

```
#include<Keypad.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(14,15,16,17,18,19);
long first=0;
long second=0;
long c;
char customKey;
char op;
const byte rows =4;
const byte coln = 4;
char keys[rows][coln] =
{

  {'1','2','3','+'},
  {'4','5','6','-'},
  {'7','8','9','*'},
  {'%','0','=','/'},
};
```

```
byte rowPins[rows] = {9,8,7,6};
byte colPins[coln] = {5,4,3,2};
Keypad customKeypad = Keypad(makeKeymap(keys),rowPins,colPins,rows,coln);

void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("calculator.");
  delay(3000);
  lcd.clear();
  }
char opt=' ' ;
void loop()
{
 customKey = customKeypad.getKey();
  switch(customKey)
  {
    case '0'...'9':
        lcd.setCursor(0,0);
        first = first*10 + (customKey - '0');
        lcd.print(first);
        Serial.println(first);
        break;
    case '+' : lcd.print("+");
                        opt='+';
                        second= first; first=0;
                        Serial.print(first);
                        Serial.print(second);
                        break;
    case '-' : lcd.print("-");
                        opt='-';
                        second= first; first=0;
                        Serial.print(first);
                        Serial.print(second);
                        break;
     case '*' : lcd.print("*");
                        opt='*';
                        second= first; first=0;
                        Serial.print(first);
                        Serial.print(second);
                        break;
    case '%' : lcd.print("%");
                        opt='%';
                        second= first; first=0;
```

```
                    Serial.print(first);
                    Serial.print(second);
                    break;
case '/' : lcd.print("/");
                    opt='/';
                    second= first; first=0;
                    Serial.print(first);
                    Serial.print(second);
                    break;
case '=' :
        lcd.print("=");
    if (opt=='+'){
          first= second+ first;
      lcd.clear();
          Serial.println(first);
          lcd.print(first);
                    second= first; first=0;
    }
      else if (opt=='-'){
          first= second- first;
      lcd.clear();
          Serial.println(first);
          lcd.print(first);
                    second= first; first=0;
      }
       else if (opt=='*'){
          first= second* first;
      lcd.clear();
          Serial.println(first);
          lcd.print(first);
                    second= first; first=0;
         break;
       }
          else if (opt=='%'){
          first= second% first;
      lcd.clear();
          Serial.println(first);
          lcd.print(first);
                    second= first; first=0;
         break;
          }
         else if (opt=='/'){
          first= second/ first;
      lcd.clear();
          Serial.println(first);
          lcd.print(first);
```

```
                second= first; first=0;
        break;
    }
}

    }
```