SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT213- IoT SYSTEM ARCHITECTURE

II ECE / IV SEMESTER

UNIT 2 – MICROCONTROLLER AND INTERFACING TECHNIQUES FOR IoT DEVICES

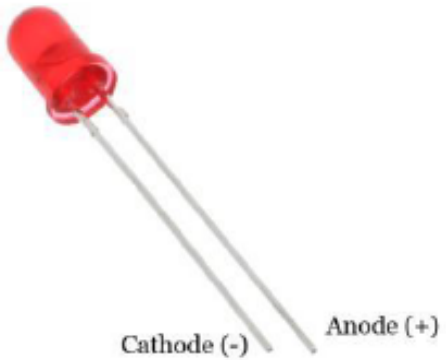
TOPIC 7 –Digital Sensor Interfacing

Interfacing programs on Arduino using LED

1. Blinking an LED
2. Toggle the state of LED using Switch
3. Traffic light simulation for pedestrians
4. Create Dimmable LED using Potentiometer

Blinking an LED

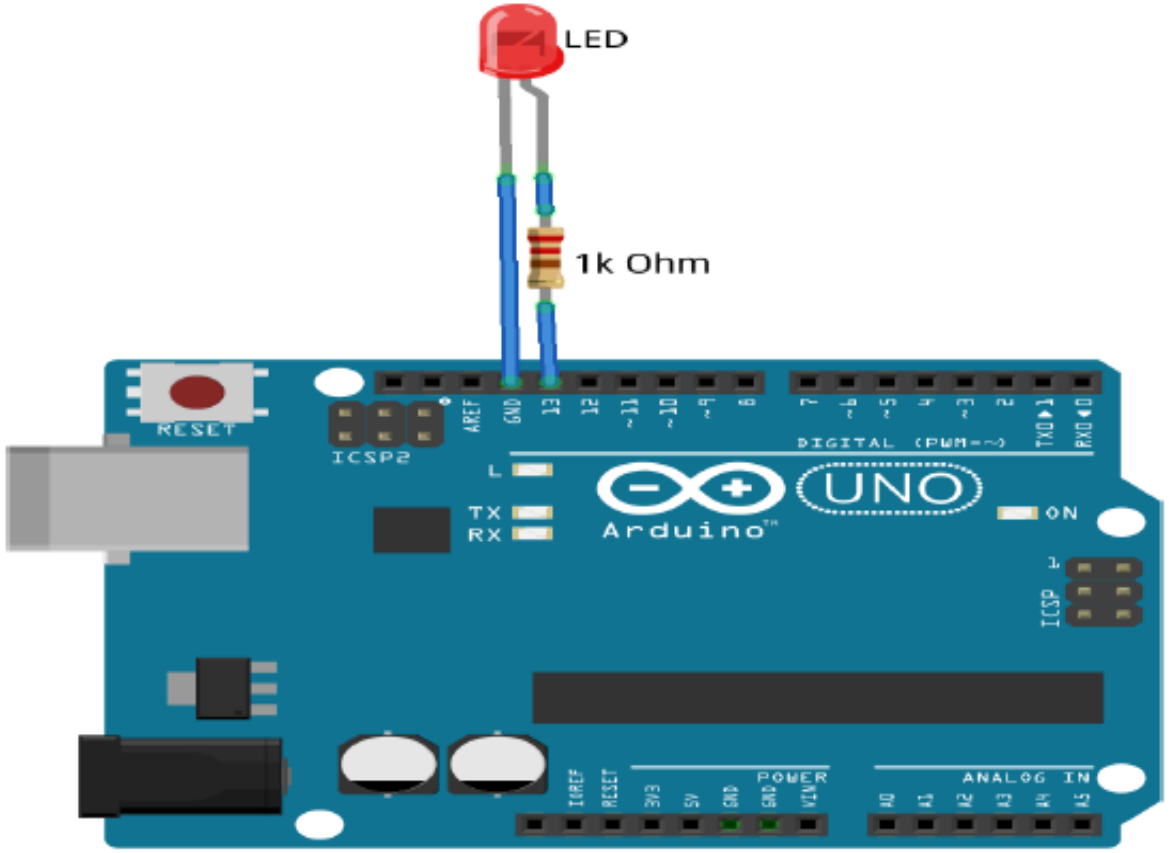
Components required	1-LED, 1-KΩ resistor, Jumper wires, Breadboard



The longest lead is the anode and the shortest is the cathode.

Blinking an LED

Circuit Diagram



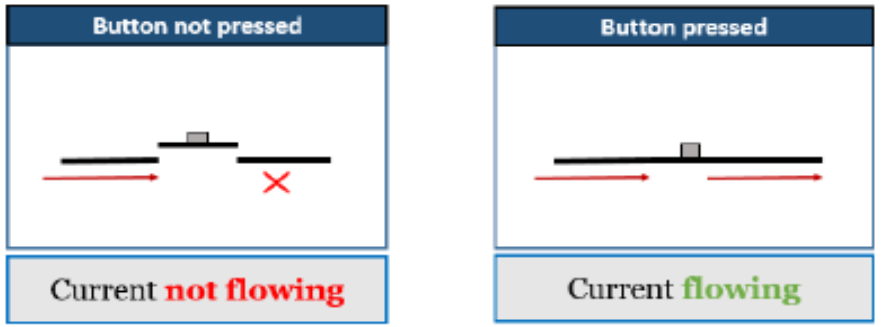
Blinking an LED

Code

```
/*The Function setup runs only once when Arduino board is first powered up
or a rest button the board is pressed */
void setup()
{
pinMode(13, OUTPUT); //pin 13 is set as an OUTPUT pin
}
//loop function iterates forever
void loop() {
digitalWrite(13, HIGH); //Sets LED to HIGH voltage
delay(1000); //delay by a second
digitalWrite(13, LOW); //Sets LED to LOW voltage
delay(1000); //delay by a second
}
```

Toggle the state of LED using Switch

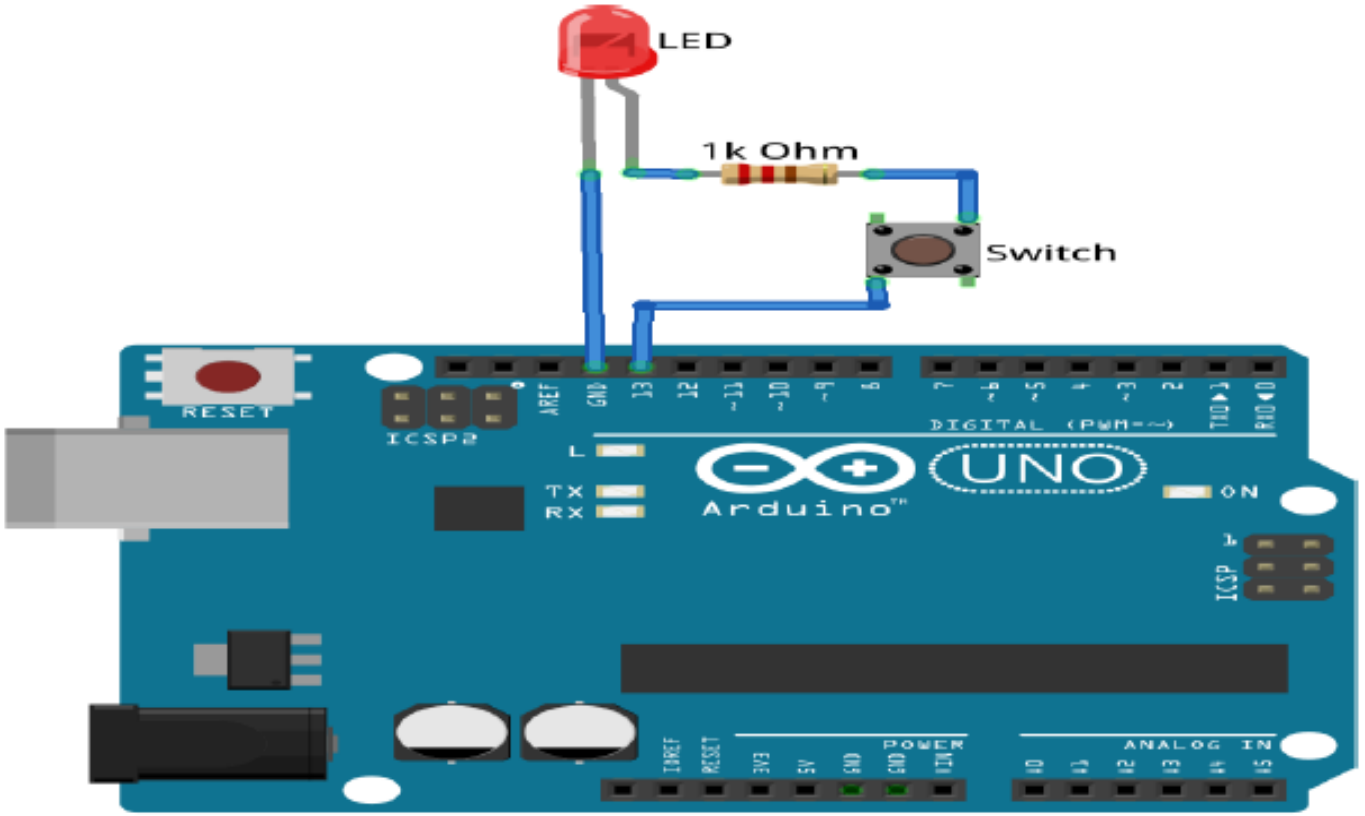
Components required	1-LED, 1-KΩ resistor, 1-push button, Jumper wires, Breadboard



Here an **open pushbutton** mechanism is used. In Normal state(not pushed) of the button current doesn't flow, only when button is pushed flow of current is allowed

Toggle the state of LED using Switch

Circuit diagram



Toggle the state of LED using Switch

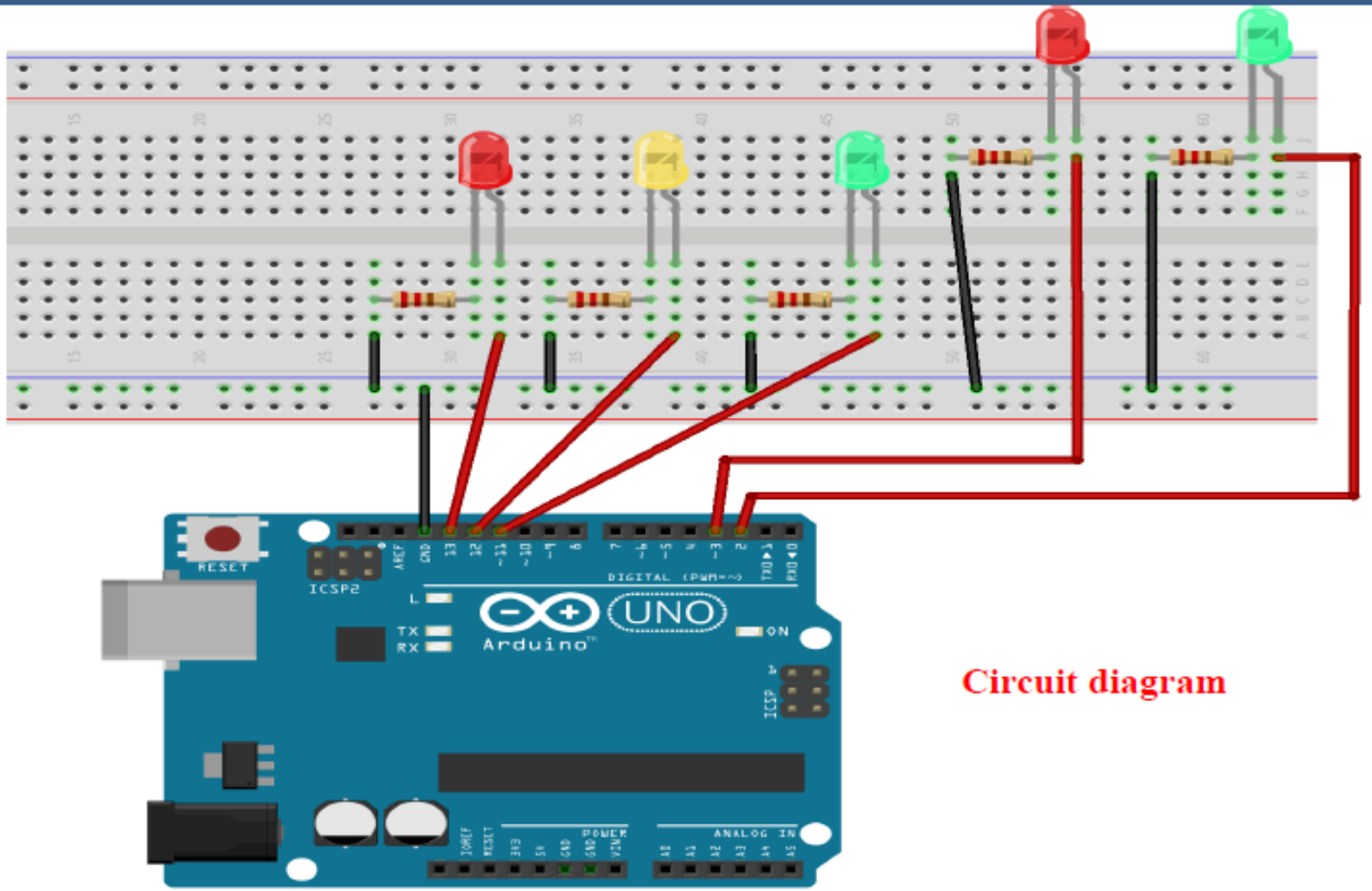
Code

```
/*The Function setup runs only once when Arduino board is first
powered up or a rest button the board is pressed */
void setup()
{
pinMode(13, OUTPUT); //pin 13 is set as an OUTPUT pin
}
//loop function iterates forever
void loop()
{
digitalWrite(13, HIGH); //Sets LED to HIGH voltage when a button is
//pressed else it remains LOW
//delay by a second
delay(1000);
}
```

Traffic light Simulation for Pedestrians

Components required	2-Red LED, 2-Green LED, 1-Yellow LED, 5-220Ω resistor, Jumper wires, Breadboard

Traffic light Simulation for Pedestrians



Circuit diagram

Traffic light Simulation for Pedestrians

Code

```
// Declare the variables for different colors of LEDs.
int red_vehicle = 13;
int yellow_vehicle = 12;
int green_vehicle = 11;
int green_Pedestrian =2;
int red_Pedestrian= 3;
void setup( )
{
// Initialize the pins for output
pinMode(red_vehicle, OUTPUT);
pinMode(yellow_vehicle, OUTPUT);
pinMode(green_vehicle, OUTPUT);
pinMode(red_Pedestrian, OUTPUT);
pinMode(green_Pedestrian, OUTPUT);
}
```

Traffic light Simulation for Pedestrians

```
void loop( )
{
digitalWrite(green_Vehicle, HIGH); // green LED turns ON
digitalWrite(red_Pedestrian, HIGH);
delay(5000);
digitalWrite(green_Vehicle, LOW); // green LED turns OFF
digitalWrite(yellow_Vehicle, HIGH); // Yellow LED turns ON for 2second.
delay(2000);
digitalWrite(yellow_Vehicle, LOW); // yellow LED will turn OFF
digitalWrite(red_Pedestrain, LOW);
digitalWrite(red_Vehicle, HIGH); // Red LED turns ON for 5 seconds
digitalWrite (green_Pedestrian, HIGH);
delay(5000);
digitalWrite(red_Vehicle, LOW); // Red LED turns OFF
digitalWrite(green_Pedestrian, LOW);
}
```

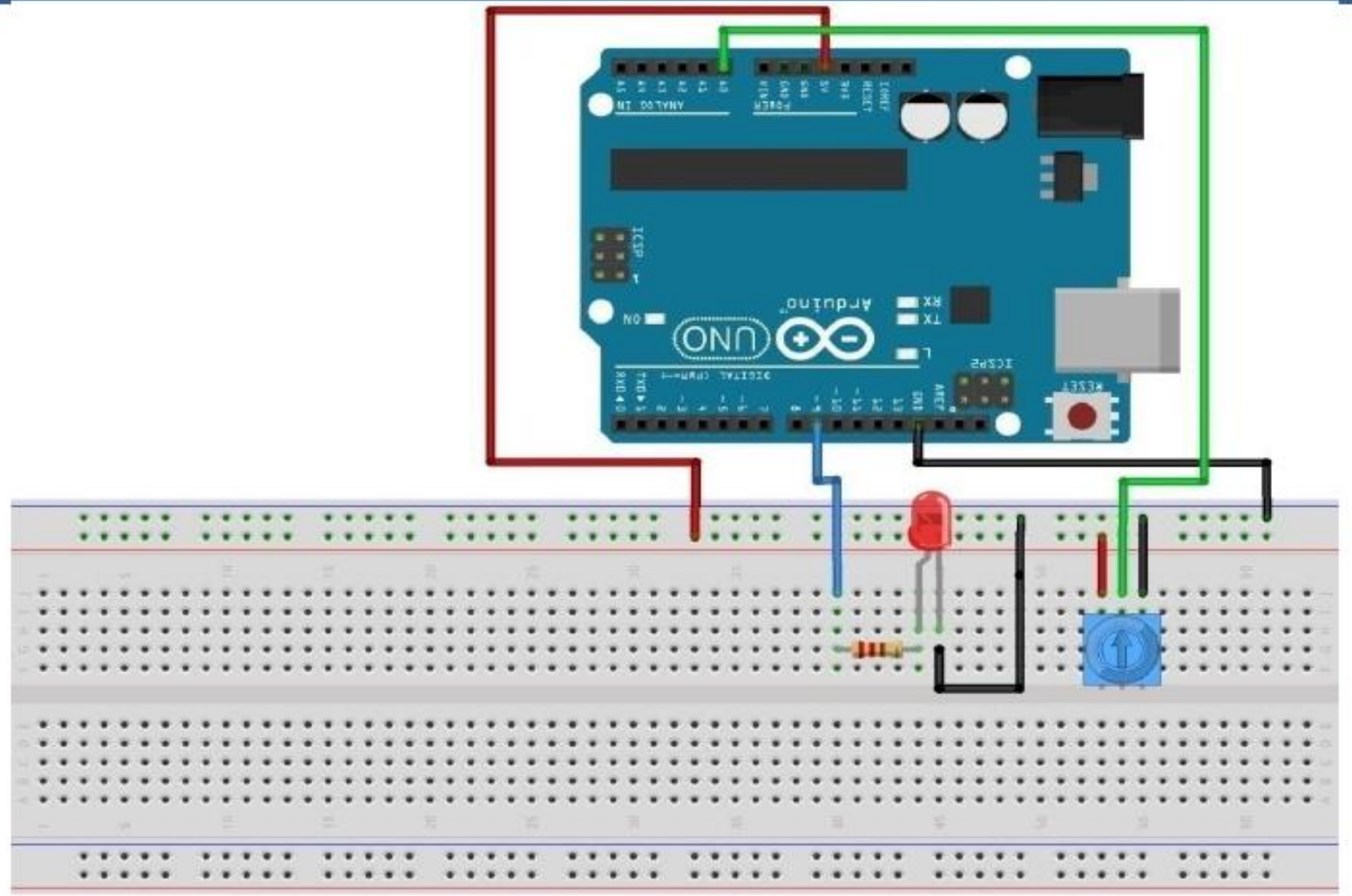
Creating a Dimmable LED using Potentiometer

Components Required	1-LED, 220Ω resistor, 1-Potentiometer, Jumper wires, Breadboard

In this program we dim the LED based on the value read from the potentiometer. A "**0**" value from potentiometer is a "**0V**" and a value "**1023**" from potentiometer is a "**5V**", which means we need to write a value of **255**. Hence we need to scale our read values from the potentiometer which falls between 0 to 1023 to suitable write values to be between 0 to 255 using the below given formulae.

write value=(255/1023)* read_value

Creating a Dimmable LED using Potentiometer



Creating a Dimmable LED using Potentiometer

```
//Declaring the pins corresponds to an LED-to pin 9 and a Potentiometer- to
//pinA0
int pot_Pin= A0;
int LED_Pin= 9;
int read_Value;  // To store the value read by potentiometer
int write_Value; // To write the value to LED
void setup( )
{ pinMode(pot_Pin, INPUT);
  pinMode(LED_Pin, OUTPUT);
  Serial.begin(9600);       }
void loop( )
{ read_Value = analogRead(pot_Pin);  //Potentiometer reading
 write_Value = (255./1023.) * readValue; //Write value for LED is calculated
 analogWrite(LEDPin, writeValue);      //Write to the LED
 Serial.print("The writing vlues to the LED is ");  //Debugging purpose
 Serial.println(write_Value); }
```

Programs to interact with Serial Monitor of our Computer Screen

To print the status of our computer Screen

Now, let's introduce the interaction with the **Serial monitor**. In this program we perform Arithmetic operations on the variables defined in the program, variables are initialized inside the program. Serial monitor communication will be processed when we call the method **Serial.begin( )** with appropriate **Baud rate**. Serial monitor displays the desired message of a program using the method **Serial.print( )** method.

Syntax:

Serial.begin(speed) /* to communicate between your computer and Serial monitor */

Serial.begin(speed, config)

Serial.print() #To print desired message on the Serial monitor

Programs to interact with Serial Monitor of our Computer Screen

```
//In this program we compute basic aritmetic operations to print the result on
//to the Serial monitor.
int a = 5, b = 10, c = 20;
void setup( )                    // run once, when the sketch starts
{ Serial.begin(9600);            // set up Serial library at 9600 bps
  Serial.println("Here is some math: ");
  Serial.print("a = ");
  Serial.println(a);
  Serial.print("b = ");
  Serial.println(b);
  Serial.print("c = ");
  Serial.println(c);
```

Programs to interact with Serial Monitor of our Computer Screen

```
Serial.print("a + b = ");       // add
  Serial.println(a + b);
  Serial.print("a * c = ");       // multiply
  Serial.println(a * c);
  Serial.print("c / b = ");       // divide
  Serial.println(c / b);
  Serial.print("b - c = ");       // subtract
  Serial.println(b - c);
}
void loop( )  {  }
```

Interfacing Sensors to the Arduino

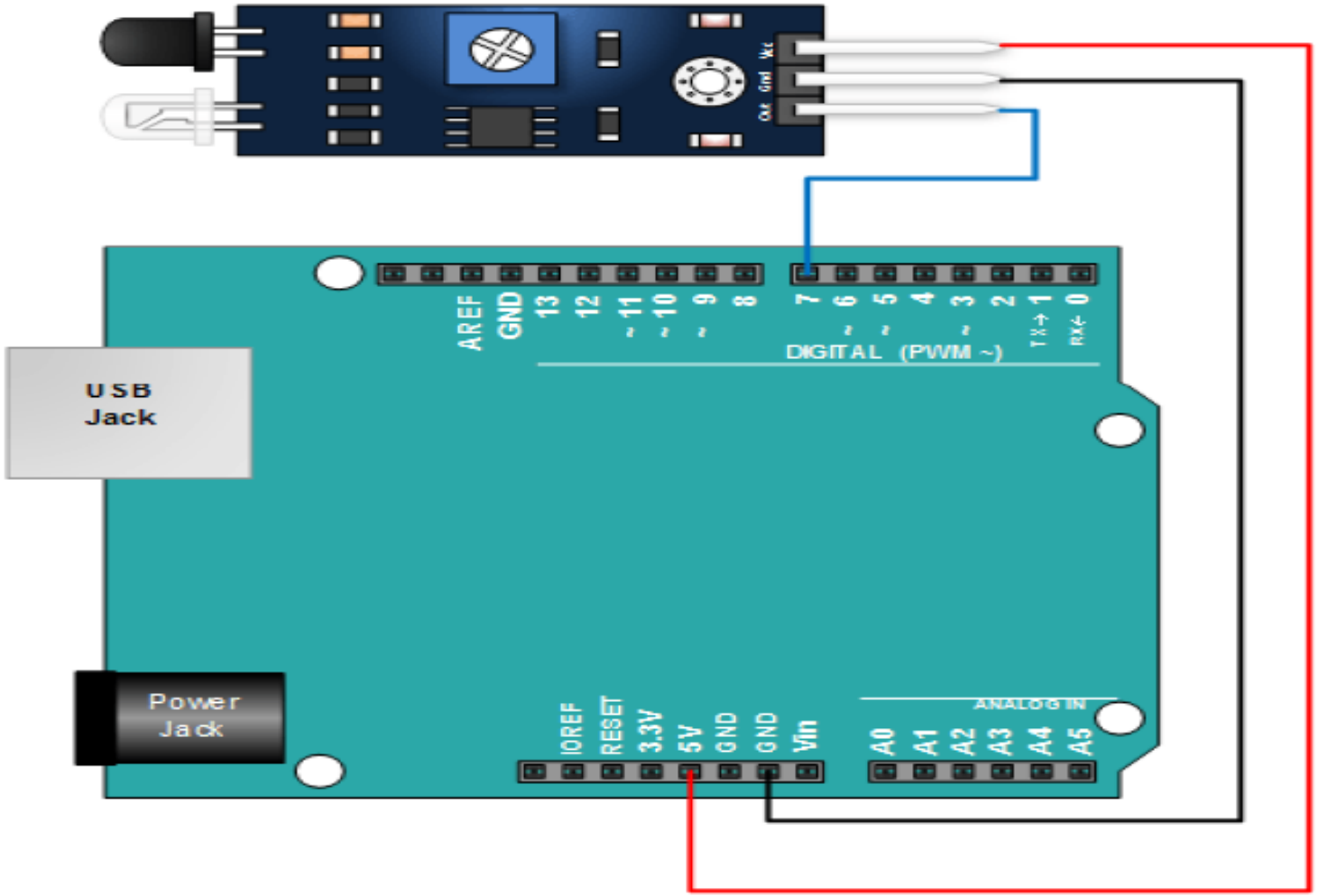
- **Temperature Sensor**
- **Light Sensor**
- **Ultrasonic distance sensor**
- **Line sensor** (infrared).

Obstacle collision module using IR(Infrared) sensor

Components Required	1-IR sensor, Jumper wire, Breadboard

An **Infrared sensor** is an electronic instrument which is used to sense certain characteristics of its surroundings by either emitting and/or detecting infrared radiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion.

Obstacle collision module using IR(Infrared) sensor



Obstacle collision module using IR(Infrared) sensor

```
// IR Obstacle Collision Detection Module

int LED = 13;
int is_Obstacle_Pin = 7;  // input pin for ostacle
int is_Obstacle = HIGH;   // value HIGH tells
there's no obstacle

void setup() {
 pinMode(LED, OUTPUT);
 pinMode(is_Obstacle_Pin, INPUT);
 Serial.begin(9600);

}
```

Obstacle collision module using IR(Infrared) sensor

```
void loop() {
  is_Obstacle = digitalRead(is_Obstacle_Pin);
  if (is_Obstacle == LOW)
  {
    Serial.println("OBSTACLE!!,
OBSTACLE!!");
    digitalWrite(LED, HIGH);
  }
  else
  {
    Serial.println("clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
}
```