Assignment No. 5

Aim: Create a program so that when the user enters "B" the green light blinks, "g" the green light is illuminated "y" the yellow light is illuminated and "r" the red light is illuminated

Outcome: Connectivity, configuration and control of LED using Arduino circuit under different conditions.

- Hardware Requirement: Arduino, LED, 220 ohm resistor etc.
- Software Requirement: Arduino IDE
- Theory:

The problem statement is like Arduino traffic light, a fun little project that you can build in under an hour. Here's how to build your own using an Arduino, and how to change the circuit for an advanced variation.

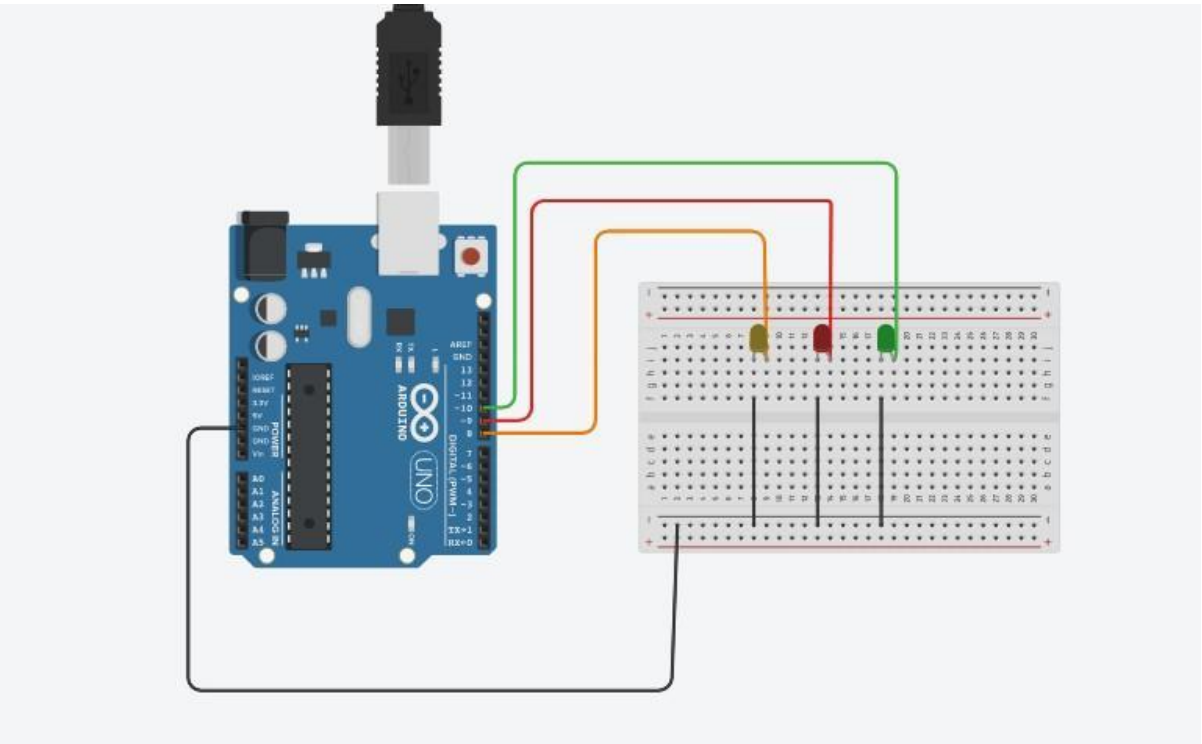
What You Need to Build an Arduino Traffic Light Controller

Apart from the basic Arduino, you'll need:

- 1 x 10k-ohm resistor
- 1 x pushbutton switch
- 6 x 220-ohm resistors
- A breadboard
- Connecting wires
- Red, yellow and green LEDs

Here's the circuit:

Connect the anode (long leg) of each LED to digital pins eight, nine, and ten (via a 220- ohm resistor). Connect the cathodes (short leg) to the Arduino's ground.



Start by defining variables so that you can address the lights by name rather than a number. Start a new Arduino project, and begin with these lines:
Int green=8
Int yellow=9
Int red=10

Next, let's add the setup function, where you'll configure the red, yellow and green LEDs to be outputs. Since you have created variables to represent the pin numbers, you can now refer to the pins by name instead:

```
void setup()
{
    Serial.begin(9600);
    pinMode(8,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
}
```

The pinMode function configures the Arduino to use a given pin as an output. You have to do this for your LEDs to work at all.Here's the code you need. Add this below your variable definitions and setup function:

Upload this code to your Arduino, and run (make sure to select the correct board and port from the Tools > Board and Tools > Port menus).

For each step, the code is very similar. The appropriate LED gets turned on or off using digitalWrite. This is an Arduino function used to set output pins to HIGH (for on), or LOW (for off).

After enabling or disabling the required LEDs, the delay makes the Arduino wait for a given amount of time.

```
void loop()
{
    Serial.println("Enter which colour you want to Blink\n1.'b' for green colour \n2.'g'
for yellow        colour\n3.'r' for red colour");

    while(Serial.available() >= 0)
    {
        mychar = Serial.read();
        if (mychar == 'b')
        {
            digitalWrite(8, HIGH);
            delay(1000);
            digitalWrite(8, LOW);
            delay(100);
            digitalWrite(9, LOW);
            digitalWrite(10, LOW);
         }
         if (mychar == 'g')
         {
             digitalWrite(8, LOW);
             digitalWrite(9, HIGH);
             delay(1000);
             digitalWrite(9, LOW);
             delay(100);
             digitalWrite(10, LOW);
             break;
          }
          if (mychar == 'r')
          {
              digitalWrite(8, LOW);
              digitalWrite(9, LOW);
              digitalWrite(10, HIGH);
              delay(1000);
              digitalWrite(9, LOW);
              delay(100);
          }
    }
}
```

Conclusion: -Here we Create a program so that when the user enters "B" the green light blinks, "g" the green light is illuminated "y" the yellow light is illuminated and "r" the red light is illuminated

Assignment No. 6

Aim: Write a program that asks the user for a number and outputs the number Squared that is entered

● Outcome: Connectivity, configuration and serial communication with Arduino.

● Hardware Requirement: Arduino, USB cable etc.
● Software Requirement: Arduino IDE

Theory:
● Arduino Serial Monitor for Beginners
Arduino serial monitor for beginners in electronics Send and receive data between the serial Monitor window on a computer and an Arduino. The serial monitor is a utility that is part of the Arduino IDE. Send text from an Arduino board to the serial monitor window on a computer. In addition, send text from the serial monitor window to an Arduino board. Communications between the serial monitor and Arduino board takes place over the USB connection between the computer and Arduino.

Demonstration of the Arduino Serial Monitor for Beginners:
Part 2 of this Arduino tutorial for beginners shows how to install the Arduino IDE. In addition, it shows how to load an example sketch to an Arduino. It is necessary to know how to load a sketch to an Arduino board in this part of the tutorial. Therefore, first finish the previous parts of this tutorial before continuing with this part. A sketch loaded to an Arduino board demonstrates how the serial monitor works in the sub-sections that follow.

● Load an Example Sketch that uses the Serial Monitor to an Arduino Board
Start the Arduino IDE application. Select File → Examples → 04.Communication → ASCIITable from the top Arduino IDE menu bar. As a result, the ASCIITable example sketch opens in a new Arduino IDE window. Upload the ASCIITable example sketch to the Arduino Uno or MEGA 2560 board.

After the ASCIITable sketch is uploaded, nothing is seen to happen. This is because this example sketch sends text out of the USB port of the Arduino board. Because there is nothing running on the computer to receive this text, nothing is seen.

● How to Open the Arduino Serial Monitor Window for Beginners
Click the Serial Monitor icon near the top right of the Arduino IDE to open the serial monitor window.
Because the ASCIITable example is loaded on the Arduino board, when the serial monitor window opens, the Arduino sends text to the serial monitor window. This is also because opening the serial monitor window resets the Arduino board, causing the ASCIITable sketch to run from the beginning again.

The ASCIITable sketch sends text out of the USB port of the Arduino. Because the serial monitor is connected to the USB port, it receives the text and displays it in the big receive area of the window. As a result, text scrolls on the serial monitor window for a while. The text then stops because the Arduino has finished sending text. Use the right scrollbar in the serial monitor window to scroll up. Scrolling up reveals all of the text that the Arduino sent.

● What to do When Junk Characters are Displayed

When junk, or garbage characters, or even nothing is displayed in the serial monitor, it is usually because of an incorrect baud rate setting. Look at the bottom of the serial monitor in the above image. Notice the value 9600 baud in a box. This is the baud setting of communications between the Arduino and serial monitor. The ASCIITable, and most other built-in example sketches, set the Arduino to communicate at 9600 baud. If your serial monitor window shows a different baud rate, change it to 9600 baud. Do this by clicking the baud drop-down list. Select 9600 baud on the list that drops down.

● Reset the Arduino Board with the RESET Button

Press and release the RESET button on the Arduino board and the ASCIITable sketch runs from the beginning again. As a result of the reset, the same text scrolls down the serial monitor window and then stops again. The RESET button is the only push button on the Arduino Uno or MEGA 2560.

Pushing the RESET button in holds the board in reset. This means that the sketch currently loaded on the board stops running. Releasing the RESET button takes the board out of reset. As a result, the sketch currently loaded on the Arduino starts running from the beginning again.

Conclusion: -Here we are write a program that asks the user for a number and outputs the number squared that is entered

```
int out;
int cube;
void setup()
{
 Serial.begin(9600); // opens serial port, sets data rate to 9600

}
void loop()
{
 // send data only when you receive data:
 if (Serial.available() > 0)
 {
 // read the incoming byte:
```

```
 int num=Serial.readString().toInt();
 // say what you got:
Serial.print("I received: ");
Serial.println(num);

out = num*num;
Serial.print("Sq of no.: ");
Serial.println(out);
Serial.println();
cube=(num*num*num);
Serial.print("cube of no.: ");
Serial.println(cube);
Serial.println("---------------------------------------");

 }
}
```

Assignment No. 7

Aim: Write a program read the temperature sensor and send the values to the serial monitor on the computer

● Outcome: Understanding working principle of DHT11, LM35 temperature sensor.

● Hardware Requirement: Arduino, LED, LM35

● Software Requirement: Arduino IDE

● Theory:

LM35 Temperature Sensor:

Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3	Ground	Ground Connected to ground of circuit

● LM35 Sensor Features
Minimum and Maximum Input Voltage is 35V and -2V respectively typically 5V.
Can measure temperature ranging from -55°C to 150°C
Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
±0.5°C Accuracy
Drain current is less than 60uA
Low cost temperature sensor
Small and hence suitable for remote applications
Available in TO-92, TO-220, TO-CAN and SOIC package

- How to use LM35 Temperature Sensor:

LM35 is a precession Integrated circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between -55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function or any development platform like Arduino.
Power the IC by applying a regulated voltage like +5V (VS) to the input pin and connected the ground pin to the ground of the circuit. Now, you can measure the temperature in form of voltage .

If the temperature is 0°C, then the output voltage will also be 0V. There will be rise of 0.01V (10mV) for every degree Celsius rise in temperature. The voltage can converted into temperature using the below formulae.

Vout=10mv/°C *T

- LM35 Temperature Sensor Applications:
- Measuring temperature of a particular environment
- Providing thermal shut down for a circuit/component
- Monitoring Battery Temperature
- Measuring Temperatures for HVAC applications.

LM35 can also be directly connected to Arduino. The output of LM35 temperature can also be given to comparator circuit and can be used for over temperature indication or by using a simple relay can be used as a temperature controller.

Conclusion: Here we are Write a program read the temperature sensor and send the values to the serial monitor on the computer

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int temp = analogRead(A1);    //Read the analog pin
  temp = temp * 0.48828125;   // convert output (mv) to readable celcius
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println("C");  //print the temperature status
  delay(5000);
}
```

.

Assignment No. 8

Aim: Write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue

● Outcome: Connectivity, configuration and control of LED using Arduino circuit under different conditions.

● Software Requirement: Arduino IDE

- Hardware Required

- Arduino board
- Potentiometer
- 1x Red, 1x Green, 1x Blue LED
- 3x 220 Ohm Resistors

- Circuit

- Potentiometer + Pin to 5V
- Potentiometer - Pin to GND
- Potentiometer Data Pin to A3
- A Red LED connected to pin 9
- A Green LED connected to pin 10
- A Blue LED connected to pin 11

potentiometer is a simple mechanical device that comes in many different forms. It provides a variable amount of resistance that changes as you manipulate it. By passing voltage through a potentiometer into an analog input on your Arduino, it is possible to measure the amount of resistance of the potentiometer as an analog value. This practical will showcase use cases of potentiometers, as well as teach you how to connect and read data from them. One shows how you can use a potentiometer as an input for a color mixer, and the other shows how to accurately choose colors and how to smoothly fade between them.

The typical potentiometer will have 3 pins, two power supply pins (+5V and GND), and one pin that connects to an analog input pin on your Arduino to read the value output.

- analogWrite()

Description

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

Syntax

```
analogWrite(pin, value)
```

Parameters

`pin`: the Arduino pin to write to. Allowed data types: `int`.
`value`: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: `int`.

Returns

Nothing

- analogRead()

Description

Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

Syntax

```
analogRead(pin)
```

Parameters

`pin`: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns

The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits). Data type: `int`.

Conclusion: Here we Write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue

```
int input0 = A0;
int input1 = A1;
int input2 = A2;
int LEDR = 9;
int LEDG = 10;
int LEDB = 11;
int value0=0;

int value1=0;
int value2=0;
void setup ()
{
Serial.begin(9600);

pinMode(LEDR,OUTPUT);
pinMode(LEDG,OUTPUT);
pinMode(LEDB,OUTPUT);
}
void loop()
{
int value0 = analogRead(input0);

int value1 = analogRead(input1);
int value2 = analogRead(input2);

analogWrite(LEDR, value0/4);
analogWrite(LEDG, value1/4);
analogWrite(LEDB, value2/4);
}
```