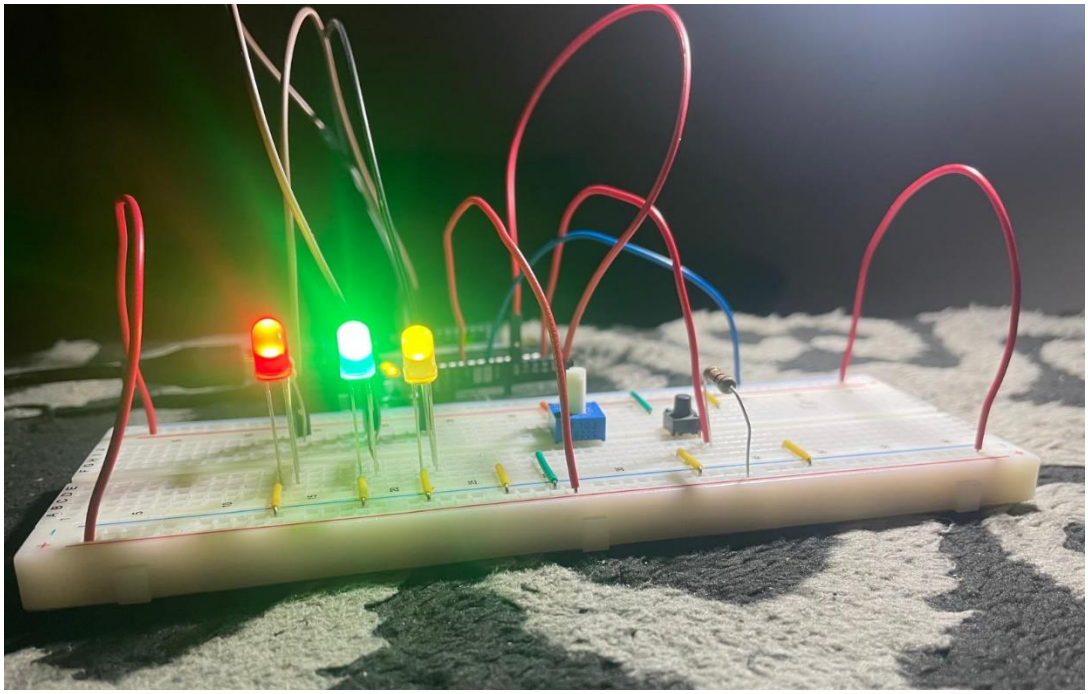
Making A Traffic Light With Arduino (FSM)



Author: B Hydara

Department: Dept. of Computer Eng. Tech.

Date: 3/31/2026

Table of Contents:

1. Objectives:

- Learn basic concepts of Finite State Machine (FSM).
- Learn basic application of Finite State Machine model in Computer Controlled System design.
- Learn to use the Finite State Machine Library for Arduino.
- Learn to design a simple computer-controlled system by using a Finite State Machine implementation in software, to control output devices by using input sensors.

2. Component / Equipment List:

- Microcontroller development board (UNO, MEGA etc.) with USB cable
- Breadboard and Jumper Wires
  (or Rapid Prototyping Shield (Sensor Shield) with GVS pin connections)
- Push Button and 10k-ohm (or similar value) pull-down resistor
  (or Push Button GVS module)
- One Potentiometer (10k-ohm or similar value)
  (or Potentiometer GVS module)
- Three LEDs (any color) with 330-ohm (or similar value) series resistors
  (or RGB LED GVS module)
- Multimeter for circuit testing and troubleshooting

3.. Procedure:

Preparation:

1. Download **Lab-#.zip** file on your PC.
2. Study the following reference material:
   - Finite State Machine Review.pdf
   - FiniteStateMachine Library for Arduino.pdf

Part 1 (Install third party libraries) *(skip Part 1 if these libraries were installed)*

1. Download the Lab...zip file from Labs section.
2. Open the folder on the PC where zip file was saved.
   - (It is usually "C:\Users\<UserName>\Downloads").
3. **Extract the Lab..zip file** to a folder on your PC.
4. Launch Arduino IDE and click **Sketch > Import Library > Add .ZIP Library**
5. Navigate to the folder where Lab...zip file was extracted and add the following three library files located in **FSM and Support Libraries** folder, **one by one**…
   - Button.zip
   - FSM.zip
   - LED.zip
6. *If you get any error messages while adding libraries, troubleshoot the library installation procedure, before proceeding to Part 2.*

Part 2 (Analysis): Hardware and Software Test

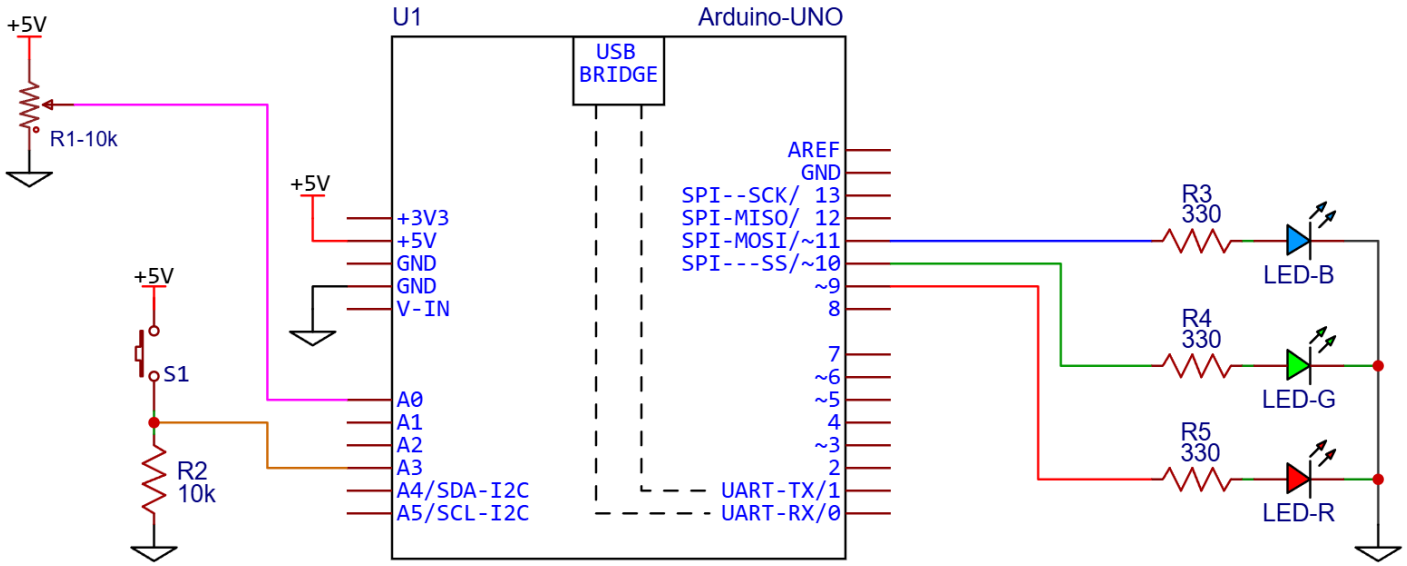
7. Wire the circuit according to the schematic diagram.
8. Upload **File > Examples > Analog > AnalogInOutSerial** example. This program will let you control pin-9 LED brightness by using the 10k potentiometer. This serves as a quick test of the analog input (ADC) and analog output (PWM) circuits. Test results are displayed in Serial Monitor window.
9. Open the **FSM_RGB_LED.ino** example program file included in the Lab extracted folder. Compile and upload the program and verify that the circuit works as expected, according to the **FSM_RGB_LED state diagram**. With each button press the circuit should step through each state as shown in **FSM_RGB_LED state diagram.**
10. *Study the example source code and make sure you understand how these programs work so that you can combine them correctly in Part 3.*

Part 3 (Design): Program Modification

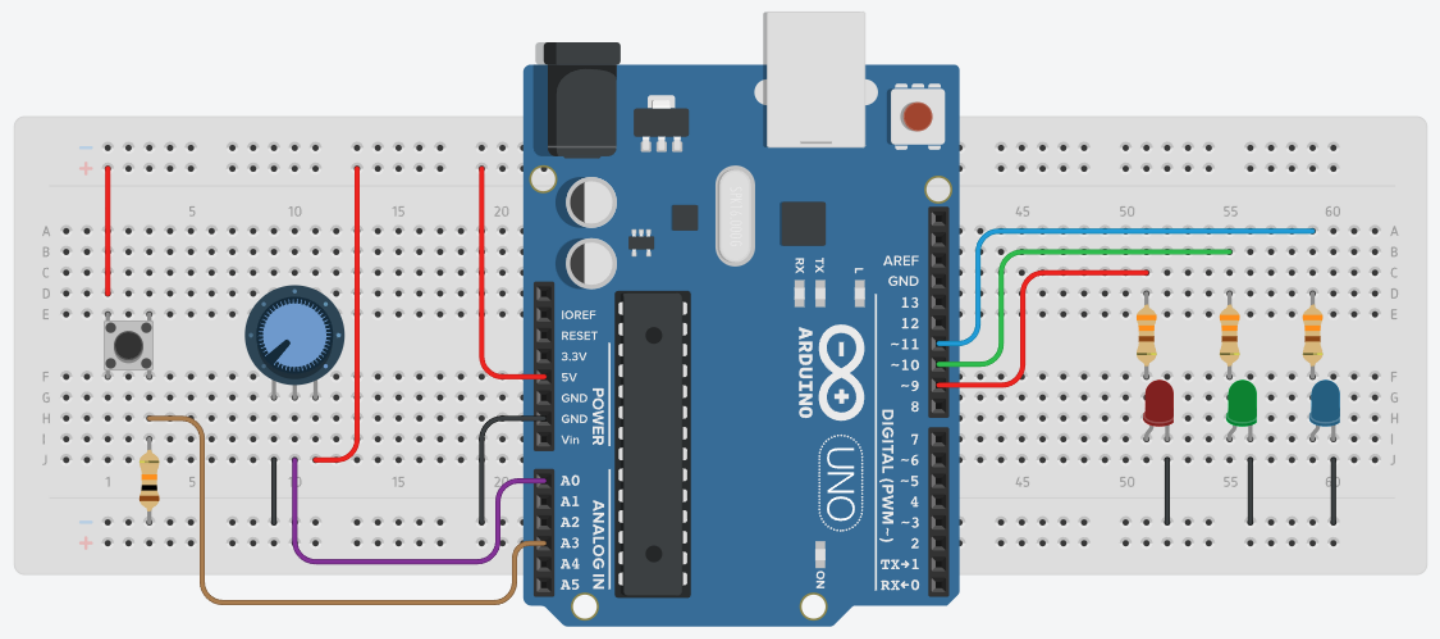
11. Save the FSM_RGB_LED program under a different name and modify it as follows.
12. Use appropriate lines of code from **AnalogInOutSerial** example program to modify the **FSM_RGB_LED** program, so that it works according to the **modified program state diagram**.
13. Hints:
    - Most of the modifications will be in the four functions that implement the **behavior of each of the four states**, at the end of the example program shown in the Appendix.
    - **Each state function** will require the use of **analogRead( )**, **map( )** and **analogWrite( )** functions as shown in **AnalogInOutSerial** example program.
    - Assign a pin name for Potentiometer pin A0 before setup().
    - Declare 4 appropriately named global variables to store the Potentiometer value and brightness values of Red, Green, Blue LEDs.
14. Compile and upload the modified program. Verify that the program works as expected, according to the **second state diagram**.

4.. Diagrams and Pictures:

Hardware: (schematic diagram)

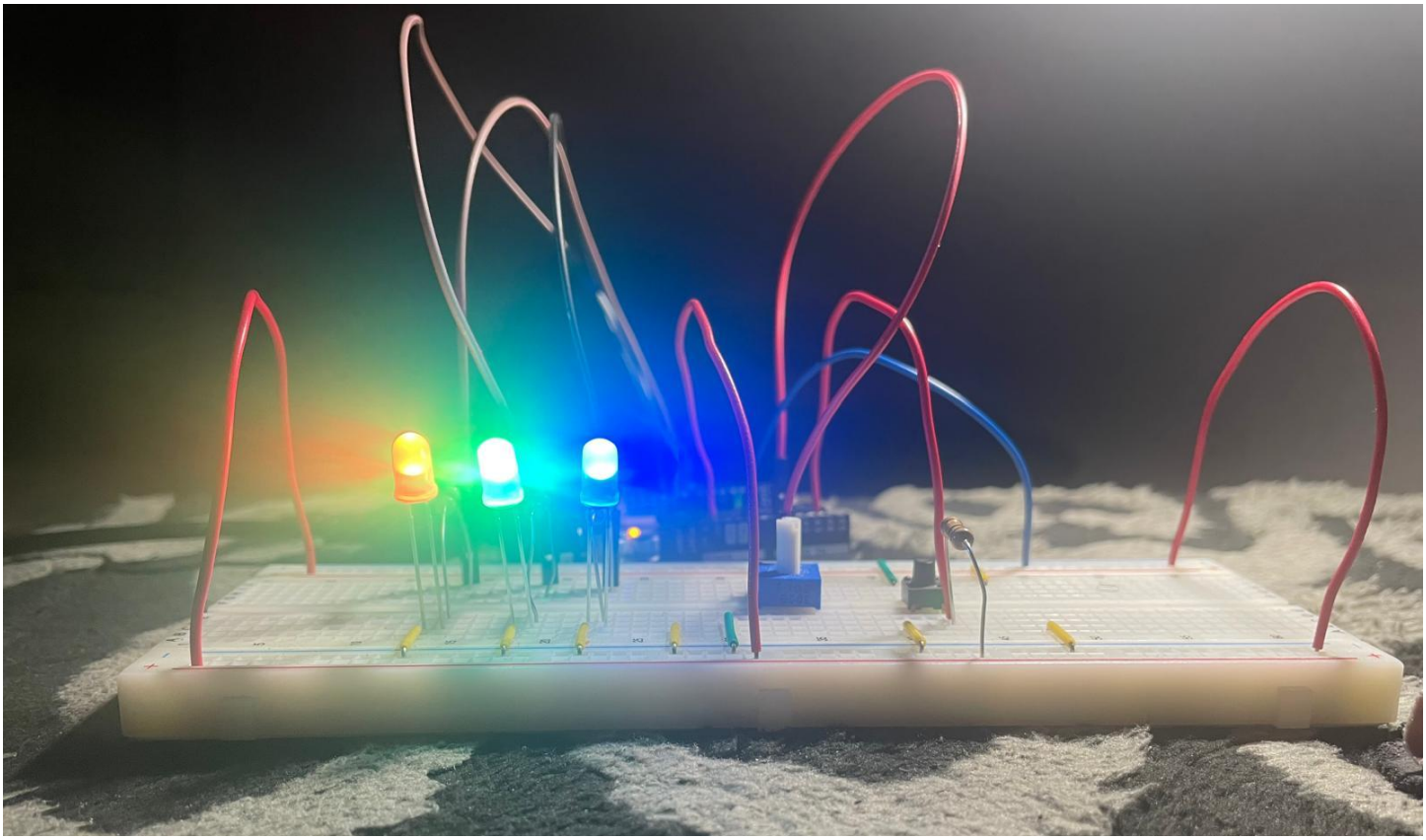


Hardware: (physical layout diagram)



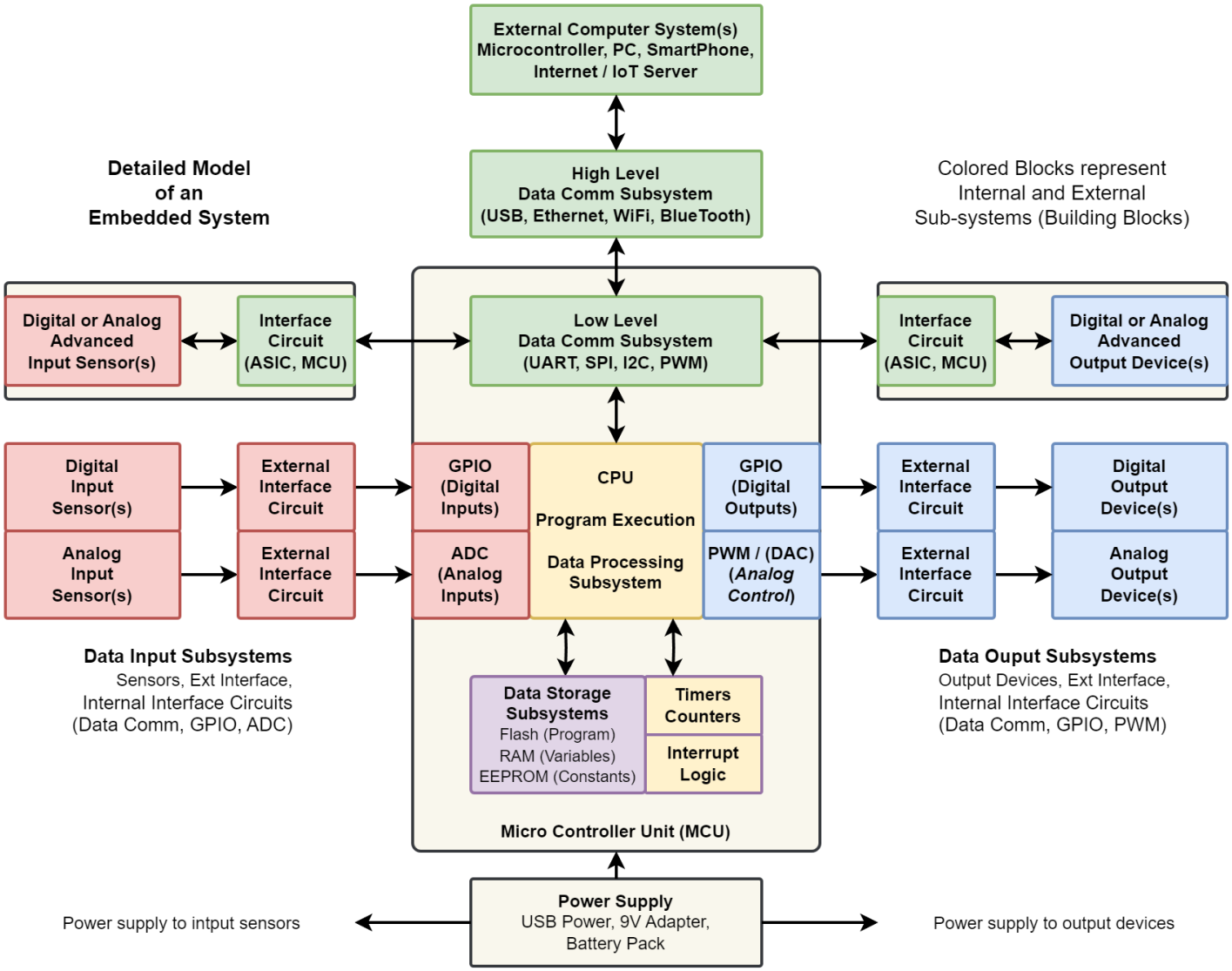
4.. Diagrams and Pictures:

Hardware: (picture(s) of the final version of lab circuit)



4.. Diagrams and Pictures:

Hardware: (reference block diagram)



Following sub-systems of a computer-controlled system are involved in this lab exercise.

Input Sub-systems:

- Digital Input Sensor: Push Button (with 10k pull down resistor as external interface circuit) connected to GPIO (General Purpose Input Output) internal interface circuit.
- Analog Input Sensor: Potentiometer connected to ADC (Analog to Digital Converter) internal interface circuit.

Data Processing and Data Storage Sub-systems:

- **Microcontroller:** ATmega328P (UNO) or ATmega2560 (MEGA) with Flash memory and RAM.

Output Sub-systems:

- Analog Output Devices: LEDs (with 330 ohm series resistor as external interface circuit) connected to PWM (Pulse Width Modulator, internal interface circuit) output pins.

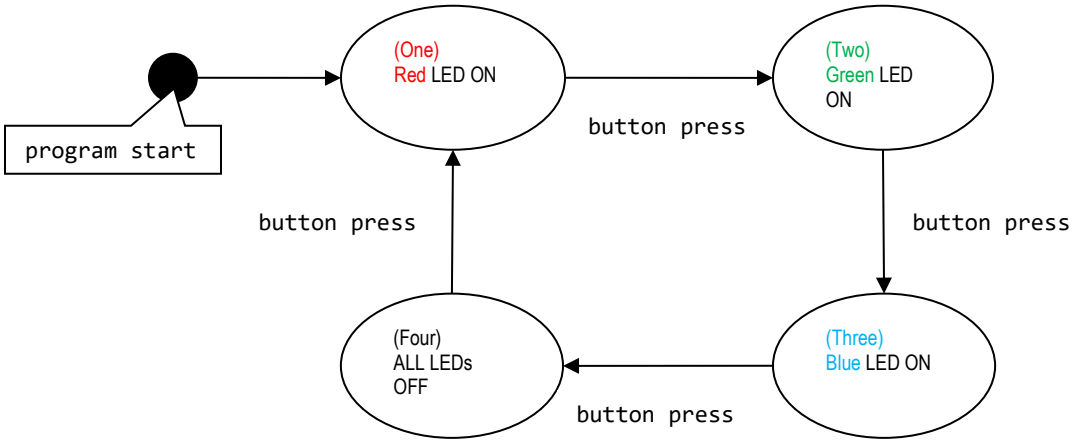
Data Communication Sub-systems:

- **Low Level:** Universal Asynchronous Receiver Transmitter (UART)
- **High Level:** Universal Serial Bus (USB)

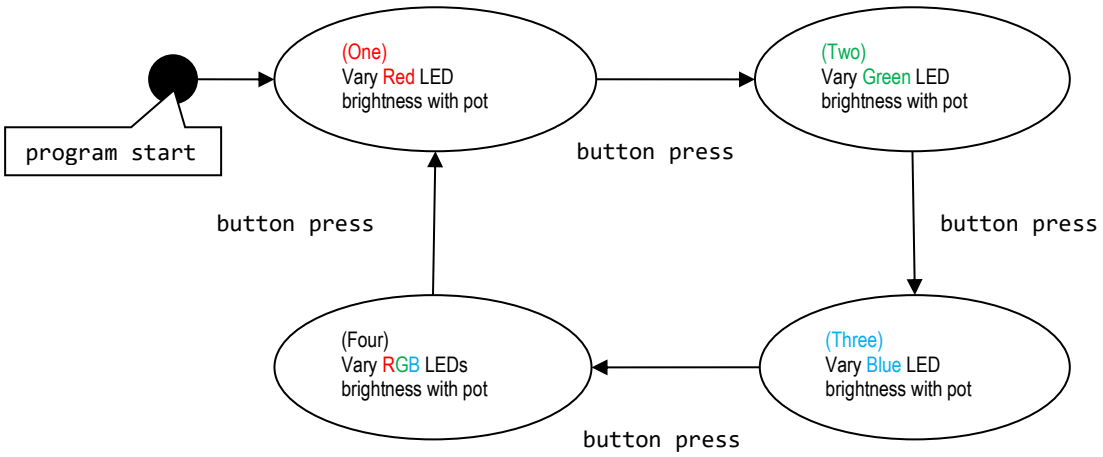
4.. Diagrams and Pictures:

Software:

State diagram for example program FSM_RGB_LED.ino



Modify FSM_RGB_LED program so that it works according to following state diagram



5.. Measurements:

- **Serial Monitor screenshots**
  (add caption to each screenshot to describe what data values are shown in the screen shot)

FSM_RGB_LED

FSM_RGB_LED.ino

```
#define ledRed    9
#define ledGreen 10
#define ledBlue  11
```

Output   Serial Monitor ×

Message (Enter to send message to 'Arduino UNO' on 'COM4')

```
Blu LED on
Blu LED on
Blu LED on
Blu LED on
Blu LED on
Blu LED on
Blu LED on
Blu LED on
```

Part 3 (Design): Program Modification

FSM_RGB_LED_P3.ino

```

    switch (buttonPresses) {
      case 0: FSM_RGB_LED.transitionTo(One);   break;
      case 1: FSM_RGB_LED.transitionTo(Two);   break;
      case 2: FSM_RGB_LED.transitionTo(Three); break;
      case 3: FSM_RGB_LED.transitionTo(Four);  break;
    }
  }
```

Output   Serial Monitor ×

Message (Enter to send message to 'Arduino UNO' on 'COM4')

```
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
State 4 - Pot = 21  Red = 5  Green = 5  Blue = 5
```

6.. Source Code:

```
/***
FSM_RGB_LED_Modified.ino
This program implements a Finite State Machine.
Button press moves to next state.
Potentiometer on A0 controls LED brightness.
Programe name:    Balagie Hydara
Date modified:     3/31/2026
You can watch the video on YouTube here: https://youtu.be/BxjJdxoj_4k
***/

#include <FiniteStateMachine.h>
#include <Button.h>

const byte NUMBER_OF_STATES = 4;

// create states
State One   = State(One_fn);
State Two   = State(Two_fn);
State Three = State(Three_fn);
State Four  = State(Four_fn);

FSM FSM_RGB_LED = FSM(One);

// button
Button button = Button(A3, INPUT);
byte buttonPresses = 0;

// LED pins
#define ledRed    9
#define ledGreen 10
#define ledBlue  11

// potentiometer pin
#define potPin A0

// global variables
int potValue = 0;
int redBrightness = 0;
int greenBrightness = 0;
int blueBrightness = 0;

void setup() {
  pinMode(ledRed, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(ledBlue, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop() {
  if (button.uniquePress()) {
    buttonPresses = ++buttonPresses % NUMBER_OF_STATES;

    switch (buttonPresses) {
      case 0: FSM_RGB_LED.transitionTo(One);   break;
      case 1: FSM_RGB_LED.transitionTo(Two);   break;
      case 2: FSM_RGB_LED.transitionTo(Three); break;
      case 3: FSM_RGB_LED.transitionTo(Four);  break;
    }
  }

  FSM_RGB_LED.update();
}

// State 1: vary Red LED brightness with pot
void One_fn() {
  potValue = analogRead(potPin);
  redBrightness = map(potValue, 0, 1023, 0, 255);

  analogWrite(ledRed, redBrightness);
  analogWrite(ledGreen, 0);
  analogWrite(ledBlue, 0);

  Serial.print("State 1 - Pot = ");
  Serial.print(potValue);
  Serial.print("  Red = ");
  Serial.println(redBrightness);
}

// State 2: vary Green LED brightness with pot
void Two_fn() {
  potValue = analogRead(potPin);
  greenBrightness = map(potValue, 0, 1023, 0, 255);

  analogWrite(ledRed, 0);
  analogWrite(ledGreen, greenBrightness);
  analogWrite(ledBlue, 0);

  Serial.print("State 2 - Pot = ");
  Serial.print(potValue);
  Serial.print("  Green = ");
  Serial.println(greenBrightness);
}

// State 3: vary Blue LED brightness with pot
void Three_fn() {
  potValue = analogRead(potPin);
  blueBrightness = map(potValue, 0, 1023, 0, 255);
```

```
  analogWrite(ledRed, 0);
  analogWrite(ledGreen, 0);
  analogWrite(ledBlue, blueBrightness);

  Serial.print("State 3 - Pot = ");
  Serial.print(potValue);
  Serial.print("  Blue = ");
  Serial.println(blueBrightness);
}

// State 4: vary all RGB LEDs brightness with pot
void Four_fn() {
  potValue = analogRead(potPin);

  redBrightness = map(potValue, 0, 1023, 0, 255);
  greenBrightness = map(potValue, 0, 1023, 0, 255);
  blueBrightness = map(potValue, 0, 1023, 0, 255);

  analogWrite(ledRed, redBrightness);
  analogWrite(ledGreen, greenBrightness);
  analogWrite(ledBlue, blueBrightness);

  Serial.print("State 4 - Pot = ");
  Serial.print(potValue);
  Serial.print("  Red = ");
  Serial.print(redBrightness);
  Serial.print("  Green = ");
  Serial.print(greenBrightness);
  Serial.print("  Blue = ");
  Serial.println(blueBrightness);
}
```

7. Critical Thinking and Discussion:

Software:

Data Processing operation(s):

<Describing code statement and/or library function(s) used for processing data>

*<**Syntax, Parameters, Return value, Description**>*

1. analogRead()
Syntax: analogRead(pin)
Parameters: pin (e.g., A0)
Return: 0–1023
Description: Reads potentiometer value

2. map()
Syntax: map(value, fromLow, fromHigh, toLow, toHigh)
Parameters: input value and ranges
Return: mapped value
Description: Converts 0–1023 → 0–255

3. analogWrite()
Syntax: analogWrite(pin, value)
Parameters: pin, brightness (0–255)
Return: none
Description: Controls LED brightness (PWM)

4. transitionTo()
Syntax: FSM.transitionTo(state)
Parameters: next state
Return: none
Description: Changes FSM state

5. update()
Syntax: FSM.update()
Parameters: none
Return: none
Description: Runs current state

Data Storage operation(s):

Size of **FLASH memory** (used / total) for storing program code (% or kB): **3794 bytes (11%)**
Size of **RAM** (used / total) for storing variables (% or kB): **331 bytes (16%)**

Data Output Sub-systems:

Hardware:

Output device(s) used:

Output device(s) used:	**Type:**
RGB LED	Analog output (PWM-based)

PWM output pins of the microcontroller are used to control the LEDs. The PWM signal varies the duty cycle to adjust brightness of each LED.

8. Conclusion:

- A real-life application, device or project example that represents a computer-controlled system and has similar types of input and output sub-systems as implemented in this project.

  A real-life example is a smart lighting system (like LED room lights with a dimmer or color control app). It uses sensors or user input to control light brightness and color, similar to this lab.

Similarities:

- Both use input
- Both use a microcontroller to process data
- Both control LED brightness using PWM

Differences:

- Real systems use apps or wireless control instead of a simple button
- Real systems are more complex and support automation
- Lab circuit is simple and used for learning only