COMPUTER ORGANIZATION & ASSEMBLY LANGUAGE

Session 2022 – 2026

Submitted by:

Muhammad Faizan Asim 2022-CS-111

Supervised by:

Sir Tehseen

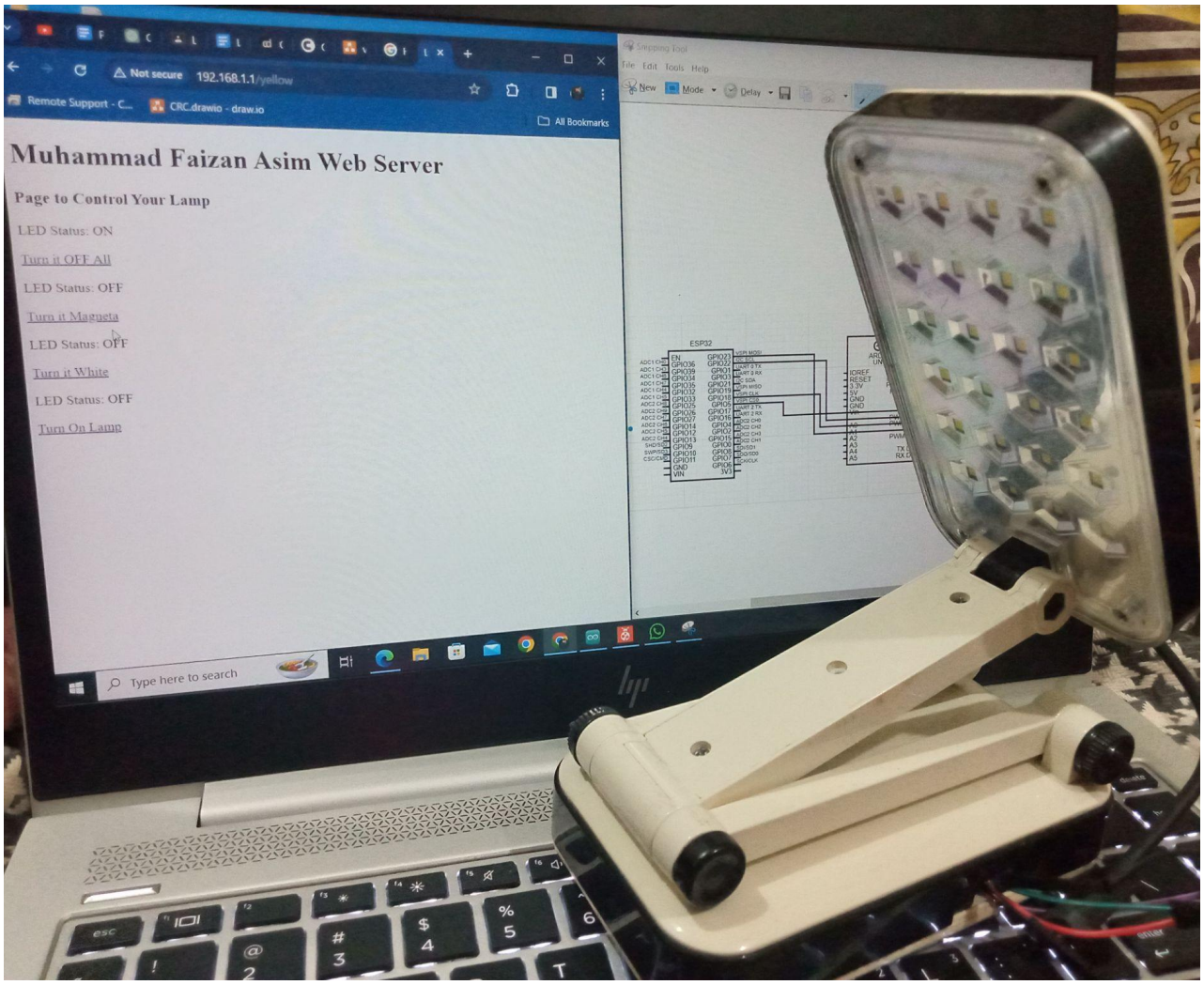
Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

ESP32 & Arduino-Powered Smart Lighting Control System

Project Photo



Description

WebSpectra brings together the power of ESP32 programmed in C++ and the intricate control of Arduino with assembly language, creating a harmonious symphony of brilliance and precision in lighting.

Crafted with meticulous attention to detail, the C++ code on the ESP32 ensures seamless communication with the web server, offering you unparalleled control through any mobile

device. Effortlessly switch between a spectrum of colors, from soothing yellows to dynamic magentas, adapting to your every mood and occasion.

Behind the scenes, the Arduino assembly code orchestrates the intricate dance of LEDs, transforming your space into a mesmerizing spectacle. Transition from an intimate ambiance to a luminous flood of white light, all at the touch of your fingertips.

WebSpectra isn't just a smart lamp; it's a technological marvel that elegantly marries the expressive power of C++ and the precision of assembly language. Experience the future of lighting control with WebSpectra – where brilliance meets precision, all under the command of your mobile device.

Methodology Used

AVR Module (Arduino UNO)

Hardware Setup:

Connect RGB Light and buzzer module to Arduino UNO using pin-to-hole jumper wires.

Integration:

Program Arduino UNO to detect signals from Esp32 and then take the decision of High which light.

Lights Control:

Configure Arduino UNO to activate the Lights according to the demand.

IoT Module (ESP32)

Hardware Setup:

Connect ESP32 to Arduino UNO using pin-to-hole jumper wires for UART Serial communication.

Communication Setup:

Establish Serial communication between Arduino UNO and ESP32 for data exchange.

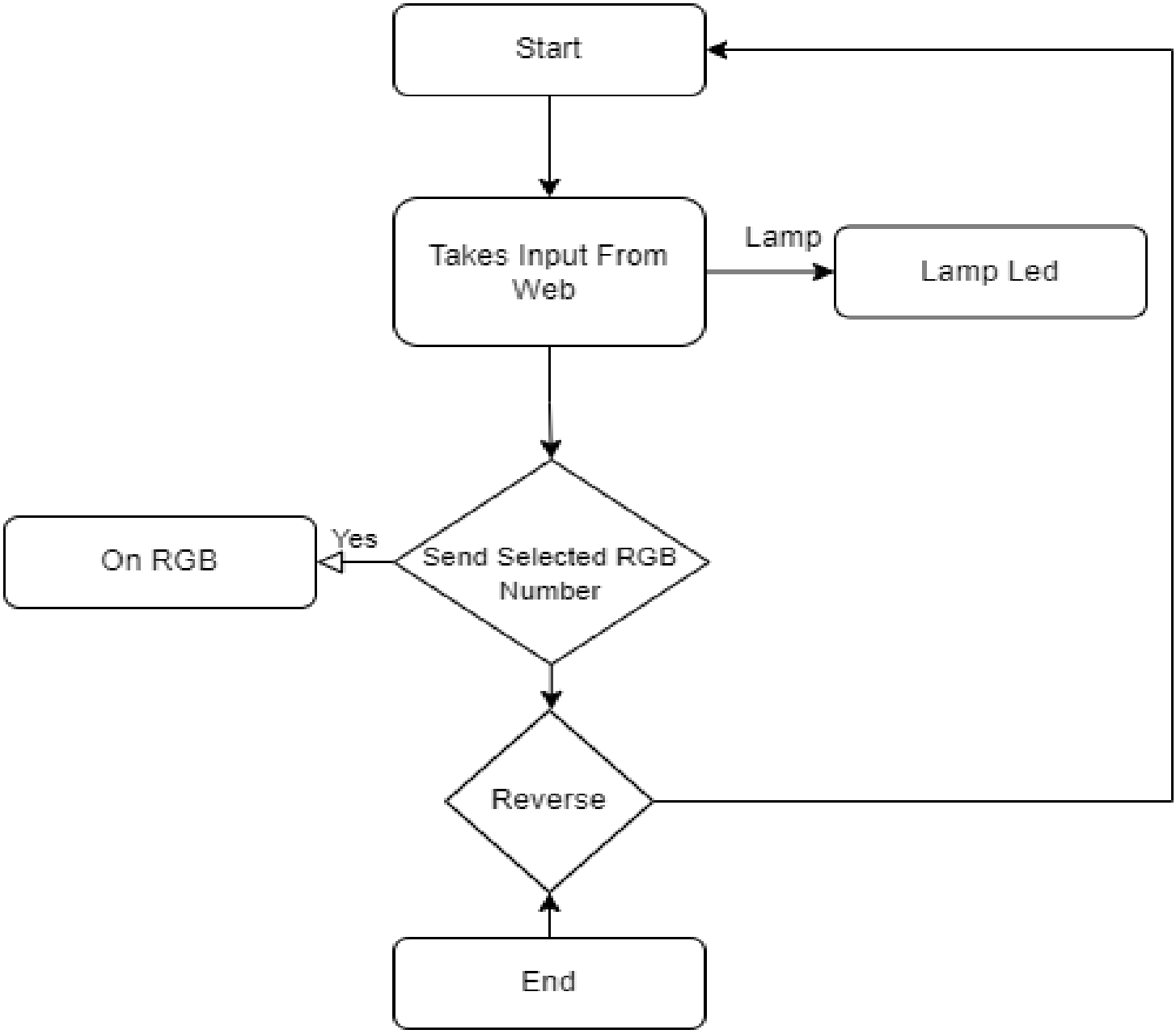
Web Server Protocol Implementation:

Program ESP32 to use web server protocol for communication between Arduino UNO and any device.

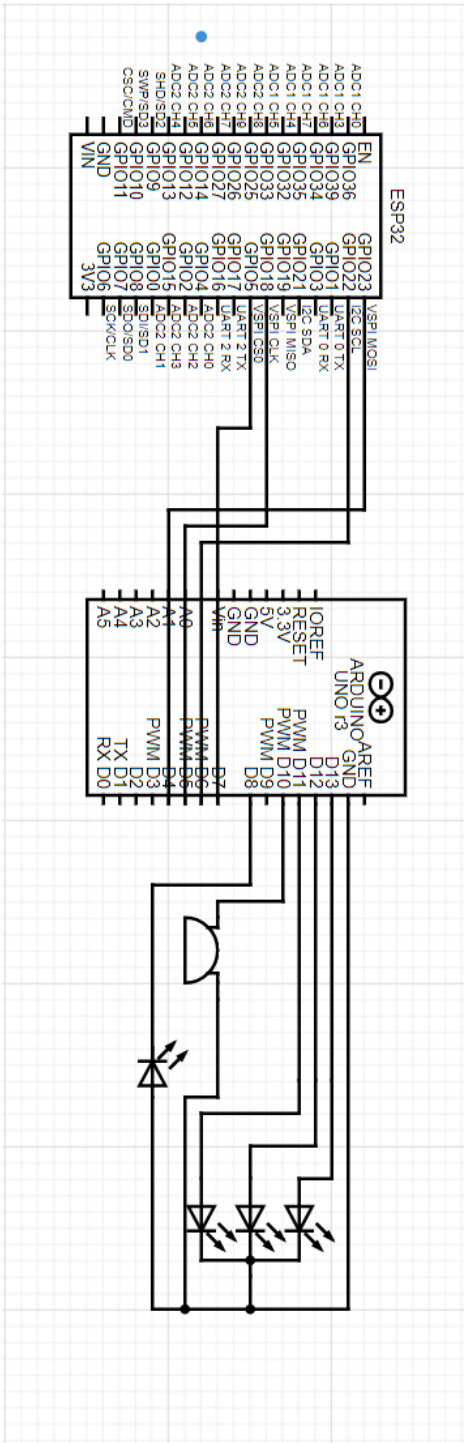
Manual Control via Website:

Develop five buttons on a website using Html and CSS to control the lights.

Flow Chart


Start
Takes Input From Web
Lamp
Lamp Led
Send Selected RGB Number
Yes
On RGB
Reverse
End

Circuit Diagram



Explanation of Components

Arduino UNO with USB Cable

The Arduino Uno is an open-source microcontroller board centered around the ATmega328P, equipped with a USB to Serial interface (such as the ATmega16U2 or CH340) for computer communication and programming via the USB connector. Power can be supplied through USB, an external 7-12V DC source, or a battery, regulated to 5V by the LM7805 voltage regulator. A 16 MHz crystal oscillator provides precise timing for the microcontroller's operations. The board features digital and analog input/output pins for connecting sensors and actuators, and a reset button restarts program execution. LEDs indicate power status, serial communication activity (TX and RX), and general-purpose usage (L). This combination of components makes the Arduino Uno a versatile and accessible platform for a wide range of electronic projects, suitable for hobbyists, students, and professionals.

ESP32 Development Board with Micro USB Cable

The ESP32 development board, featuring the ESP32 System-on-Chip (SoC), serves as a versatile platform for IoT projects. Connected to a computer via the Micro USB cable, this board facilitates programming and power supply. The SoC, based on the Xtensa LX6 architecture, integrates a dual-core processor, Wi-Fi, and Bluetooth modules, along with flash memory for program storage. The Micro USB connector supports both code uploading and power delivery. The voltage regulator ensures a stable 3.3V supply for the ESP32. Additionally, the board includes a reset button for restarting program execution and LEDs indicating power status and programmable status feedback. GPIO pins enable interfacing with external devices, and the integrated Wi-Fi and Bluetooth modules enable wireless communication. Overall, the ESP32 development board, with its rich feature set, is widely adopted for diverse IoT applications, making it a popular choice among developers for creating connected and embedded systems.

Breadboard

A breadboard is a crucial tool in electronics prototyping, providing a platform for temporarily connecting electronic components without soldering. It consists of a grid of metal clips enclosed in a plastic casing. The clips allow users to insert and connect wires, resistors, capacitors, and other components easily. The board is divided into rows and columns, with each row connected internally but isolated from others. Components are placed on the board, and their leads or wires are inserted into the clips, forming electrical connections. Breadboards facilitate rapid experimentation and testing, allowing designers to quickly modify and iterate on circuit designs. The interconnected clips make it easy to prototype circuits without the need for a permanent connection, making breadboards an essential tool for students, hobbyists, and professionals alike in the electronics field.

Pin-to-Hole Jumper Wires

Pin-to-Hole Jumper Wires are short, flexible wires with a pin on one end and a stripped or tinned wire on the other, used in electronics for making temporary connections on breadboards or connecting components. The pin end easily plugs into the pin sockets on microcontrollers, sensors, or other electronic devices, while the exposed wire end can be inserted into the holes on a breadboard for quick and temporary circuit connections. These jumper wires eliminate the need for soldering, providing a convenient and versatile solution for prototyping and experimenting with circuits. They enable the rapid construction and modification of electronic prototypes, making them an essential tool for both beginners and experienced electronics enthusiasts in building and testing various circuit configurations.

Pin-to-Pin Jumper Wires

Pin-to-Pin Jumper Wires are short, flexible wires with pins on both ends, designed for creating direct point-to-point connections between various electronic components. With one pin on each end, these wires easily plug into the pin sockets of microcontrollers, sensors, or other devices, allowing for seamless and temporary interconnections. Pin-to-Pin Jumper Wires are commonly used on breadboards or prototype boards, enabling users to establish direct links between specific pins without the need for soldering. This facilitates efficient and organized circuit prototyping, making it easy to test different configurations or integrate various components in electronics projects. The dual-pin design ensures a straightforward and versatile approach to building and modifying circuits during the development and experimentation phases.

Buzzer Module

A Buzzer Module is an electronic component designed to generate audible tones or sounds in response to an electrical signal. It typically consists of a piezoelectric buzzer element and a control circuit. When a voltage is applied to the module, the control circuit activates the piezoelectric element, causing it to vibrate and produce sound waves. The frequency and intensity of the sound can be influenced by the input signal characteristics. Buzzer Modules are commonly used in various applications such as alarms, notifications, or sound feedback in electronic projects. They are easy to interface with microcontrollers and other electronic circuits, making them valuable for creating audible alerts or signals in a wide range of projects, from simple hobbyist setups to more complex embedded systems.

RGB Module

RGB lights, which consist of red, green, and blue light-emitting diodes (LEDs), are a versatile lighting system widely used for their ability to generate a broad spectrum of colors. These lights operate on the principle of additive color mixing, where the intensity of each primary color—red, green, and blue—can be adjusted independently to create various hues. The RGB system enables precise color control, allowing users to tailor the lighting to their preferences. Whether installed in homes, commercial spaces, or entertainment setups, RGB lights provide dynamic and customizable lighting solutions. With the option to smoothly transition between colors, create

gradients, or even synchronize with music or other external stimuli, RGB lights have become a popular choice for ambient lighting, mood enhancement, and creative expression in diverse settings. Their adaptability and programmability make them not only a practical lighting solution but also a source of visual artistry.

P10 LED Matrix:

The P10 LED Matrix stands as a testament to the marriage of cutting-edge technology and visual brilliance, offering a dynamic canvas for creative expression. This matrix comprises a grid of P10 LED modules, each a pixel in a larger tapestry of vibrant displays.

At its core, the P10 LED Matrix is a captivating array of light-emitting diodes, meticulously organized to form a seamless display. This display is not merely a static entity; it is a dynamic platform capable of showcasing a spectrum of colors and patterns, orchestrated with precision.

AVR Module Code

```
.include "m328pdef.inc"
.include "delay_Macro.inc"
.include "UART_Macros.inc"

.cseg
.org 0x0000

    SBI DDRB, PB5       ; PB5 set as an OUTPUT pin (Blue)
    SBI DDRB, PB4       ; PB4 set as an OUTPUT pin (Green)
    SBI DDRB, PB3       ; PB3 set as an OUTPUT pin (Red)
    SBI DDRB, PB2       ; PB2 set as an OUTPUT pin (Buzzer)
    SBI DDRB, PB1       ; PB1 set as an OUTPUT pin (BIG LIGHT)
    CBI DDRB, PD7   ; 7 number
    CBI DDRB, PD6
    CBI DDRB, PD5

    ; Initialize UART for serial communication at 9600 bps
    Serial_begin

loop:
    ; Check if there is data available in the UART receive buffer


    ; Read the received byte from the UART
    ; Now r16 contains the received byte

    ; Check the received byte and select the appropriate sequence
    SBIS PIND,PD7
    rjmp P1

yellow_sequence:
    SBI PORTB, PB4      ; PB4 pin --> HIGH (5V) --> Green LED ON
    SBI PORTB, PB3      ; PB3 pin --> HIGH (5V) --> Red LED ON
    SBI PORTB, PB2      ; PB2 pin --> HIGH (5V) --> Buzzer ON
    delay 1000
    CBI PORTB, PB4      ; PB4 pin --> LOW  (0V) --> Green LED OFF
    CBI PORTB, PB3      ; PB3 pin --> LOW  (0V) --> Red LED OFF
    CBI PORTB, PB2      ; PB2 pin --> LOW  (0V) --> Buzzer OFF
```

```
    rjmp delay_and_continue
P1:
    SBIS PIND,PD6
    rjmp P2

magenta_sequence:
    SBI PORTB, PB3      ; PB3 pin --> HIGH (5V) --> Red LED ON
    SBI PORTB, PB5      ; PB5 pin --> HIGH (5V) --> Blue LED ON
    SBI PORTB, PB2      ; PB2 pin --> HIGH (5V) --> Buzzer ON
    delay 1000
    CBI PORTB, PB3      ; PB3 pin --> LOW  (0V) --> Red LED OFF
    CBI PORTB, PB5      ; PB5 pin --> LOW  (0V) --> Blue LED OFF
    CBI PORTB, PB2      ; PB2 pin --> LOW  (0V) --> Buzzer OFF
    rjmp delay_and_continue
P2:
    SBIS PIND,PD5
    rjmp P3

white_sequence:
    SBI PORTB, PB3      ; PB3 pin --> HIGH (5V) --> Red LED ON
    SBI PORTB, PB4      ; PB4 pin --> HIGH (5V) --> Green LED ON
    SBI PORTB, PB5      ; PB5 pin --> HIGH (5V) --> Blue LED ON
    SBI PORTB, PB2      ; PB2 pin --> HIGH (5V) --> Buzzer ON
    delay 1000
    CBI PORTB, PB3      ; PB3 pin --> LOW  (0V) --> Red LED OFF
    CBI PORTB, PB4      ; PB4 pin --> LOW  (0V) --> Green LED OFF
    CBI PORTB, PB5      ; PB5 pin --> LOW  (0V) --> Blue LED OFF
    CBI PORTB, PB2      ; PB2 pin --> LOW  (0V) --> Buzzer OFF
    rjmp delay_and_continue
P3:
    SBIS PIND,PD4
    rjmp P4

lamp_sequence:
        delay 1000
    SBI PORTB, PB1      ;
```

```
lamp_sequence:
        delay 1000
    SBI PORTB, PB1      ;
    rjmp delay_and_continue
P4:
    CBI PORTB, PB3      ; PB3 pin --> LOW  (0V) --> Red LED OFF
    CBI PORTB, PB4      ; PB4 pin --> LOW  (0V) --> Green LED OFF
    CBI PORTB, PB5      ; PB5 pin --> LOW  (0V) --> Blue LED OFF
    CBI PORTB, PB2      ; PB2 pin --> LOW  (0V) --> Buzzer OFF
    CBI PORTB, PB1      ; PB2 pin --> LOW  (0V) --> Buzzer OFF

delay_and_continue:
    ; Common delay and continue logic

    rjmp loop
```

IoT Module Code

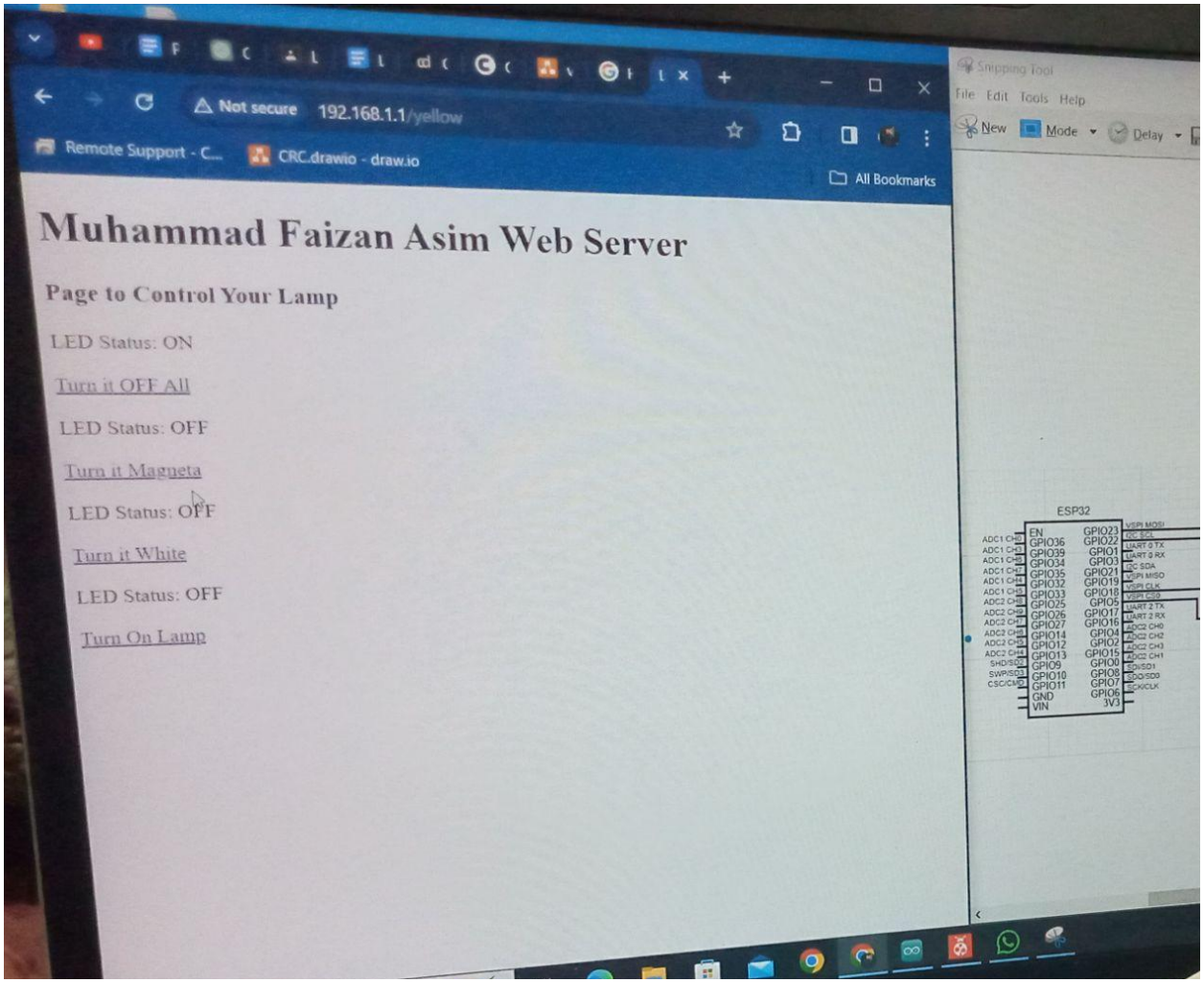
```
#include <WiFi.h>
#include <WebServer.h>

#define ssid "LampWifi"
#define password "12345678"

#define blueLedPin 5
#define LedPin1 22
#define LedPin2 18
#define LampPin 23

IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
WebServer server(80);

bool LEDStatus = LOW;
bool LED1Status = LOW;
bool LED2Status = LOW;
bool LampStatus = LOW;

void setup()
{
    Serial.begin(115200);
    pinMode(blueLedPin,OUTPUT);
    pinMode(LedPin1,OUTPUT);
    pinMode(LedPin2,OUTPUT);
    pinMode(LampPin,OUTPUT);

    WiFi.softAP(ssid,password);
    WiFi.softAPConfig(local_ip,gateway,subnet);
    delay(1000);

    server.on("/",handle_OnConnect);
    server.on("/yellow",handle_YellowON);
    server.on("/magneta",handle_LED1ON);
    server.on("/white",handle_LED2ON);
    server.on("/lamp",handle_LAMPON);
    server.on("/off",handle_ledoff);
    server.onNotFound(handle_NotFound);
    server.begin();
    Serial.println("HTTP SERVER STARTED");
}
String getHTML()
{
String htmlcode = "<!DOCTYPE html> <html>\n";
htmlcode +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
htmlcode += "<title>LED Control</title>\n";
htmlcode += "</head>\n";
htmlcode +="<body>\n";
htmlcode +="<h1>Muhammad Faizan Asim Web Server</h1>\n";
htmlcode +="<h3>Page to Control Your Lamp</h3>\n";
if (LEDStatus) {
htmlcode += "<p>LED Status: ON</p><a href=\"/off\">Turn it OFF All</a>\n";
}
else{
htmlcode +="<p>LED Status: OFF</p><a href=\"/yellow\">Turn it yellow</a>\n";
}
if (LED1Status) {
htmlcode += "<p>LED Status: ON</p><a href=\"/off\">Turn OFF All</a>\n";
}
else{
htmlcode +="<p>LED Status: OFF</p><a href=\"/magneta\">Turn it Magneta</a>\n";
}
if (LED2Status) {
htmlcode += "<p>LED Status: ON</p><a href=\"/off\">Turn OFF All</a>\n";
```

```
}
else{
htmlcode +="<p>LED Status: OFF</p><a href=\"/white\">Turn it White</a>\n";
}
if (LampStatus) {
htmlcode += "<p>LED Status: ON</p><a href=\"/off\">Turn OFF All</a>\n";
}
else{
htmlcode +="<p>LED Status: OFF</p><a href=\"/lamp\">Turn On Lamp</a>\n";
}
htmlcode += "</body>\n";
htmlcode += "</html>\n";
return htmlcode;
}
void handle_OnConnect()
{
  LEDStatus = LOW;
LED1Status = LOW;
LED2Status = LOW;
LampStatus = LOW;

  Serial.println("LED STATUS: OFF");
  server.send(200,"text/html",getHTML());
}
void handle_YellowON()
{
LEDStatus = HIGH;
LED1Status = LOW;
LED2Status = LOW;
LampStatus = LOW;
  Serial.println("LED STATUS: ON");
  server.send(200,"text/html",getHTML());
}
void handle_LED1ON()
```

```
void handle_YellowON()
{
LEDStatus = HIGH;
LED1Status = LOW;
LED2Status = LOW;
LampStatus = LOW;
  Serial.println("LED STATUS: ON");
  server.send(200,"text/html",getHTML());
}
void handle_LED1ON()
{
  LEDStatus = LOW;
LED1Status = HIGH;
LED2Status = LOW;
LampStatus = LOW;
  Serial.println("LED STATUS: ON");
  server.send(200,"text/html",getHTML());
}
void handle_LED2ON()
{
  LEDStatus = LOW;
LED2Status = HIGH;
LED1Status = LOW;
LampStatus = LOW;
  Serial.println("LED STATUS: ON");
  server.send(200,"text/html",getHTML());
}
void handle_LAMPON()
{
  LEDStatus = LOW;
LED1Status = LOW;
LED2Status = LOW;
LampStatus = HIGH;
  Serial.println("LED STATUS: ON");
  server.send(200,"text/html",getHTML());
}
void handle_ledoff()
{
    LEDStatus = LOW;
LED1Status = LOW;
LED2Status = LOW;
LampStatus = LOW;
  Serial.println("LED STATUS: OFF");
```

```
void loop()
{
    server.handleClient();
    if(LEDStatus)
    {
      digitalWrite(blueLedPin,HIGH);

    }
    else
    {
      digitalWrite(blueLedPin,LOW);

    }
    if(LED1Status)
    {
      digitalWrite(LedPin1,HIGH);

    }
    else
    {
      digitalWrite(LedPin1,LOW);

    }
    if(LED2Status)
    {
      digitalWrite(LedPin2,HIGH);

    }
    else
    {
      digitalWrite(LedPin2,LOW);

    }
    if(LampStatus)
    {
      digitalWrite(LampPin,HIGH);

    }
    else
    {
      digitalWrite(LampPin,LOW);

    }
}

void handle_NotFound()
{
  server.send(404,"text/plain","Not Found");
}
```

Website Interface



Project video links of YouTube and LinkedIn

https://www.linkedin.com/feed/update/urn:li:activity:7148467320787566592/

GitHub link of your project codes:

https://github.com/Mr-Faizan-Asim/Ardunio-Projects/tree/master/FinalLampProject

References

https://github.com/TehseenHasan

https://docs.arduino.cc/learn/