



CEIS114

Final Project Deliverables

Akira Suain

Introduction to Digital Devices- 11882-May 2025

06/25/25

Rubric

| EIS.CEIS114.W8COURSE_PROJECT.MAR24 | | | | | | |
|--|--|--|--|---|--|--------|
| Criteria | Ratings | | | | | Pts |
| <p>🕒 Apply basic programming to support IoT device (Smart Traffic Light Controller)</p> <p>threshold: 15.0 pts</p> | <p>20 pts</p> <p>The project has no programming concepts errors.</p> | <p>15 pts</p> <p>The project has 1 or more programming concepts errors.</p> | <p>10 pts</p> <p>The project has 2 or more programming concepts errors.</p> | <p>5 pts</p> <p>The project has 3 or more programming concepts errors.</p> | <p>0 pts</p> <p>No Programming Completed</p> | 20 pts |
| <p>🕒 Classify the things or devices that make up the IoT (Smart Traffic Light Controller)</p> <p>threshold: 24.0 pts</p> | <p>30 pts</p> <p>The project was built using relevant technologies and IoT principles effectively and appropriately with no errors. All five circuit setup pictures were included: 1. Two sets of Traffic Lights 2. Push Button 3. LCD Panel 4. Buzzer 5. IOT Button or Motion Detector.</p> | <p>24 pts</p> <p>The project was built using relevant technologies and IoT principles effectively and appropriately with one of the following missing: 1. Two sets of Traffic Lights 2. Push Button 3. LCD Panel 4. Buzzer 5. IOT Button or Motion Detector.</p> | <p>15 pts</p> <p>The project was built using relevant technologies and IoT principles effectively and appropriately with two of the following missing: 1. Two sets of Traffic Lights 2. Push Button 3. LCD Panel 4. Buzzer 5. IOT Button or Motion Detector.</p> | <p>7 pts</p> <p>The project was built using relevant technologies and IoT principles effectively and appropriately with three of the following missing: 1. Two sets of Traffic Lights 2. Push Button 3. LCD Panel 4. Buzzer 5. IOT Button or Motion Detector.</p> | <p>0 pts</p> <p>No project was built.</p> | 30 pts |

| | | | | | | |
|--|--|---|---|--|---|--------|
| <p>🕒 Design and produce an IoT system (Smart Traffic Light Controller)</p> <p>threshold: 15.0 pts</p> | <p>20 pts</p> <p>System was built, coded, and tested in a Virtual environment and/or IoT Kit with no errors.</p> | <p>15 pts</p> <p>System was built, coded, and tested in Virtual Environment and/or IoT Kit with one or more errors.</p> | <p>10 pts</p> <p>System was built, coded, and tested in Virtual Environment and/or IoT Kit with two or more errors.</p> | <p>5 pts</p> <p>System was built, coded, and tested in Virtual Environment and/or IoT Kit with three or more errors.</p> | <p>0 pts</p> <p>No Project was built.</p> | 20 pts |
| <p>🕒 Explain the interconnectivity of devices in the IoT (Smart Traffic Light Controller)</p> <p>threshold: 10.0 pts</p> | <p>15 pts</p> <p>Final project submitted in the form of a professional presentation including transitions between modules projects. Other slides include title slide, introduction slide, challenges in the project slide, career skills slide, slides describing each module, and conclusion slide.</p> | <p>10 pts</p> <p>Final project submitted in the form of a professional presentation including transitions between modules. Missing one or two of the following: title slide, introduction slide, challenges in the project slide, career skills slide, slides describing each module, and conclusion slide.</p> | <p>5 pts</p> <p>Final project submitted in the form of a professional presentation including transitions between modules. Missing three to four of the following: title slide, introduction slide, challenges in the project slide, career skills slide, slides describing each module, and conclusion slide.</p> | <p>3 pts</p> <p>Final project submitted in the form of a professional presentation including transitions between modules. Missing all additional slides.</p> | <p>0 pts</p> <p>Final project presentation was not completed.</p> | 15 pts |

Total Points: 85



Project Plan for IoT Traffic Controller

ESP32 (Screenshot)

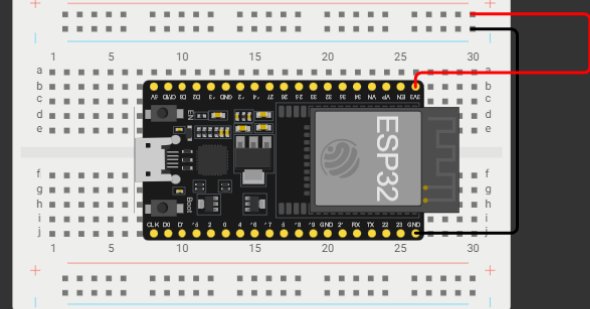
Microcontroller mounted and powered ON

WOKWI SAVE SHARE Module Two_ Akira Suain_ May 2025 Docs

sketch.ino diagram.json Library Manager

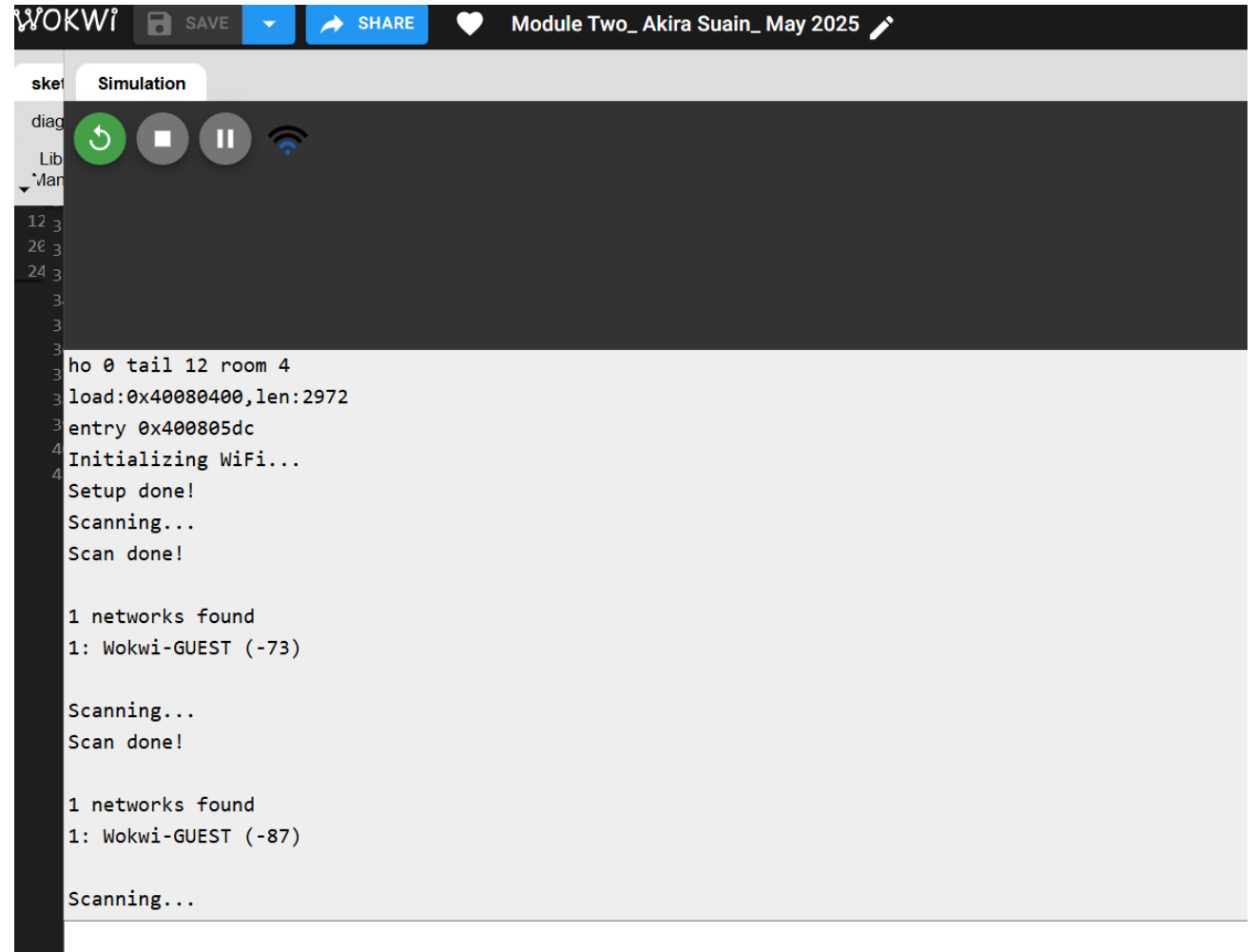
```
1  /* ESP32 WiFi Scanning example */
2
3  #include "WiFi.h"
4
5  void setup() {
6      Serial.begin(115200);
7      Serial.println("Initializing WiFi...");
8      WiFi.mode(WIFI_STA);
9      Serial.println("Setup done!");
10 }
11
12 void loop() {
13     Serial.println("Scanning...");
14
15     // WiFi.scanNetworks will return the number of networks found
16     int n = WiFi.scanNetworks();
17     Serial.println("Scan done!");
18     if (n == 0) {
19         Serial.println("No networks found.");
20     } else {
21         Serial.println();
22         Serial.print(n);
23         Serial.println(" networks found");
24         for (int i = 0; i < n; ++i) {
25             // Print SSID and RSSI for each network found
26             Serial.print(i + 1);
27             Serial.print(" ");
28             Serial.print(WiFi.SSID(i));
29             Serial.print(" (");
30             Serial.print(WiFi.RSSI(i));
31             Serial.print(")");
32             Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN) ? " " : "***");
33             delay(10);
34         }
35     }
36 }
```

Simulation



ESP32 WiFi Scan

Screenshot of **Serial Monitor** showing the available networks



```
WOKWI SAVE SHARE Module Two_ Akira Suain_ May 2025
Simulation
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Initializing WiFi...
Setup done!
Scanning...
Scan done!

1 networks found
1: Wokwi-GUEST (-73)

Scanning...
Scan done!

1 networks found
1: Wokwi-GUEST (-87)

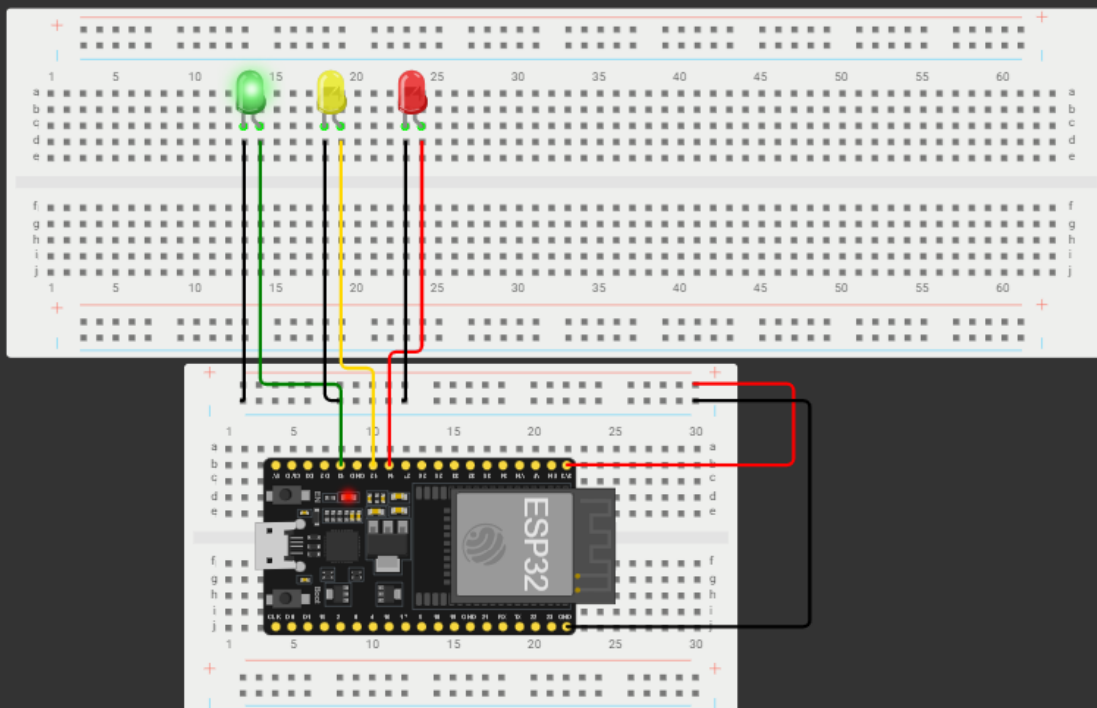
Scanning...
```



Creating the Traffic Controller

Rubric – Due at the end of Week 3

| Activity | Requirement(s) | Points |
|---|---|--------|
| Include a screenshot of your circuit | Picture of the breadboard with LEDs ON. | 15 |
| Screenshot of code showing your name in comment | Screenshot of code in the Code Editor showing your name in one of the comments. | 15 |



Call 12 room 4

```
d:0x40080400,len:2972
```

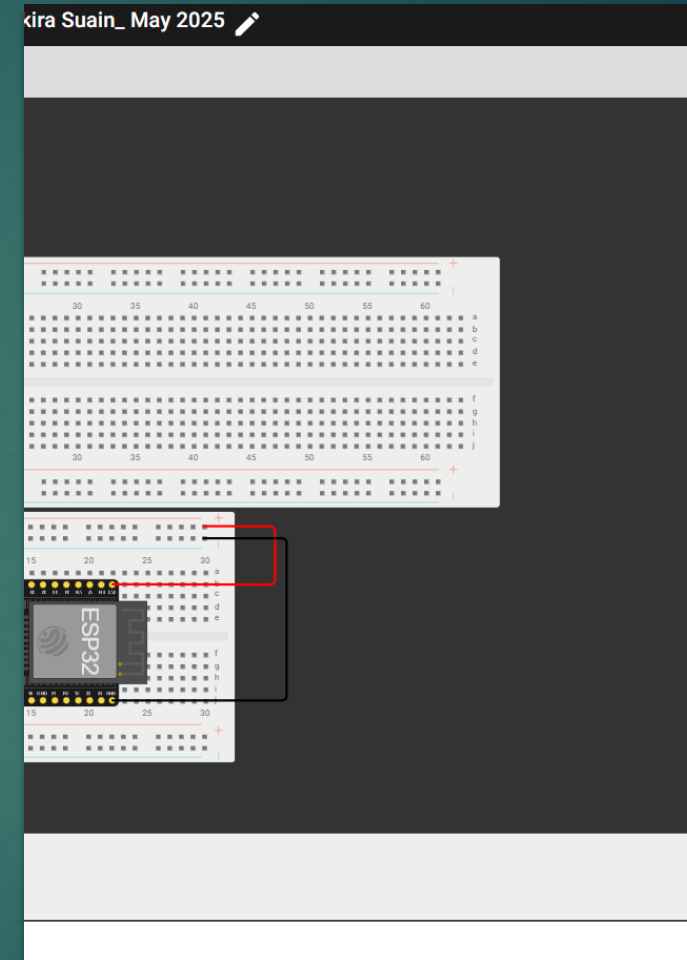
ry 0x400805dc

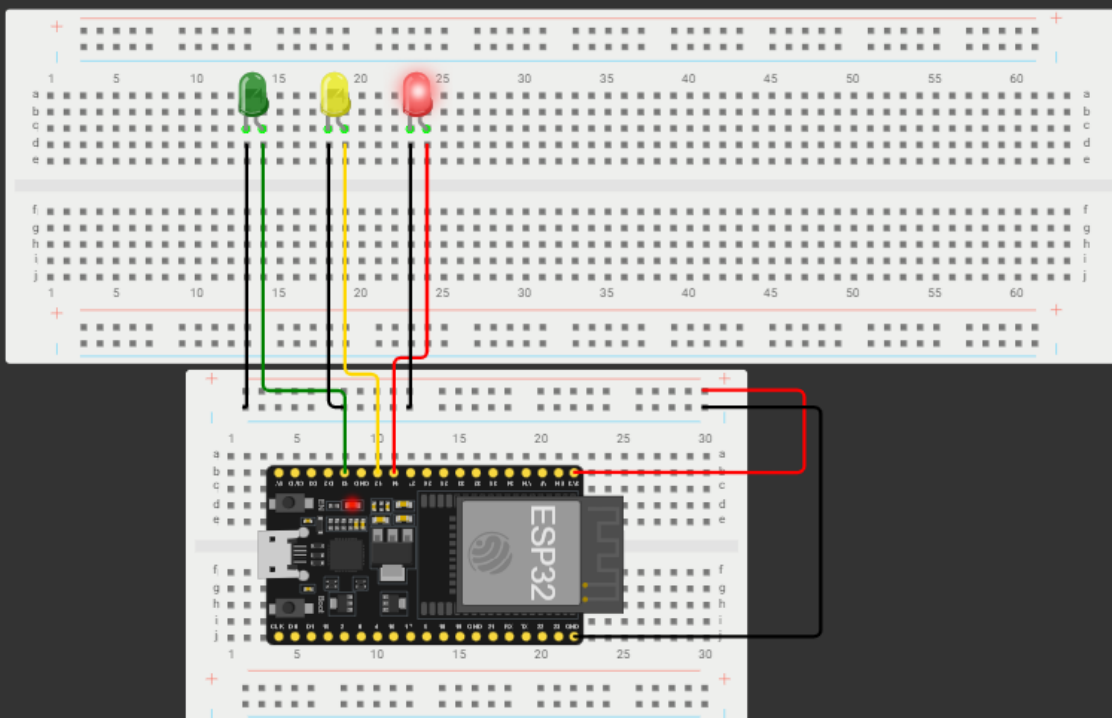
Picture of
circuit with
working LEDs

- ESP 32 Board
- Colored LEDs: Red, Yellow and Green
- Wires
- Breadboard

Picture of circuit with working LEDs

- ESP 32 Board
- Colored LEDs: Red, Yellow and Green
- Wires
- Breadboard





Picture of circuit with working LEDs

- ESP 32 Board
- Colored LEDs: Red, Yellow and Green
- Wires
- Breadboard

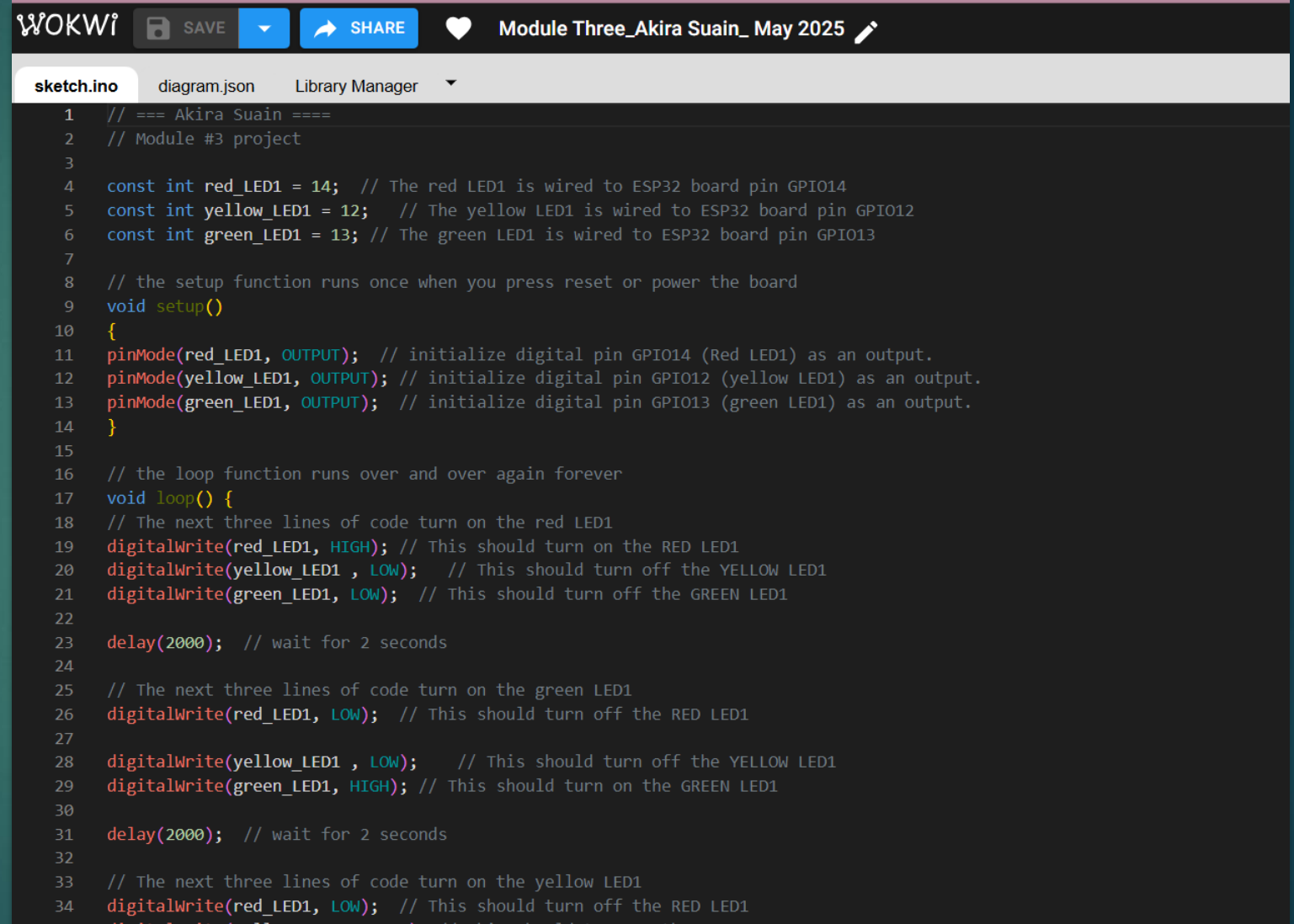
call 12 room 4

0x40080400,len:2972


0x400805dc

Screenshot of code in
the Wokwi Code
Editor showing **your**
name in the comment

Screenshot
of code in
the Code
Editor



```
WOKWI SAVE SHARE Module Three_Akira Suain_ May 2025  
sketch.ino diagram.json Library Manager  
1 // === Akira Suain ===  
2 // Module #3 project  
3  
4 const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14  
5 const int yellow_LED1 = 12; // The yellow LED1 is wired to ESP32 board pin GPIO12  
6 const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13  
7  
8 // the setup function runs once when you press reset or power the board  
9 void setup()  
10 {  
11 pinMode(red_LED1, OUTPUT); // initialize digital pin GPIO14 (Red LED1) as an output.  
12 pinMode(yellow_LED1, OUTPUT); // initialize digital pin GPIO12 (yellow LED1) as an output.  
13 pinMode(green_LED1, OUTPUT); // initialize digital pin GPIO13 (green LED1) as an output.  
14 }  
15  
16 // the loop function runs over and over again forever  
17 void loop() {  
18 // The next three lines of code turn on the red LED1  
19 digitalWrite(red_LED1, HIGH); // This should turn on the RED LED1  
20 digitalWrite(yellow_LED1, LOW); // This should turn off the YELLOW LED1  
21 digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1  
22  
23 delay(2000); // wait for 2 seconds  
24  
25 // The next three lines of code turn on the green LED1  
26 digitalWrite(red_LED1, LOW); // This should turn off the RED LED1  
27  
28 digitalWrite(yellow_LED1, LOW); // This should turn off the YELLOW LED1  
29 digitalWrite(green_LED1, HIGH); // This should turn on the GREEN LED1  
30  
31 delay(2000); // wait for 2 seconds  
32  
33 // The next three lines of code turn on the yellow LED1  
34 digitalWrite(red_LED1, LOW); // This should turn off the RED LED1
```



CEIS 114

Module 4

CREATING A MULTIPLE TRAFFIC LIGHT CONTROLLER

AKIRA SUAIN

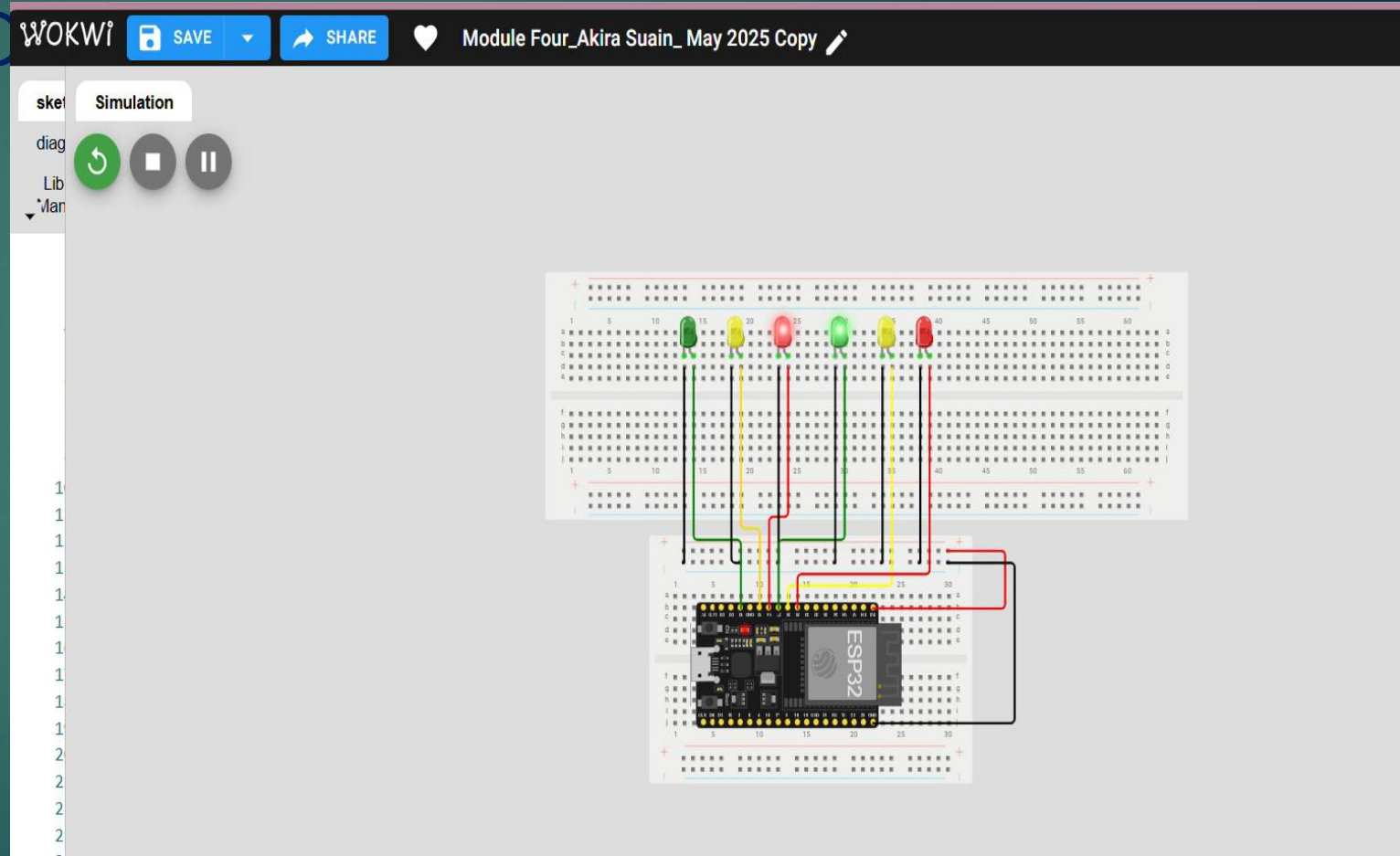
DUE DATE: JUNE 1ST 2025

Rubric – Due at the end of Week 4

| Activity | Requirement(s) | Points |
|---|--|--------|
| Include a screenshot of your circuit | Screenshot of the breadboard with LEDs ON. | 15 |
| Screenshot of code showing your name in comment | Screenshot of code in the Code Editor showing your name in one of the comments | 15 |

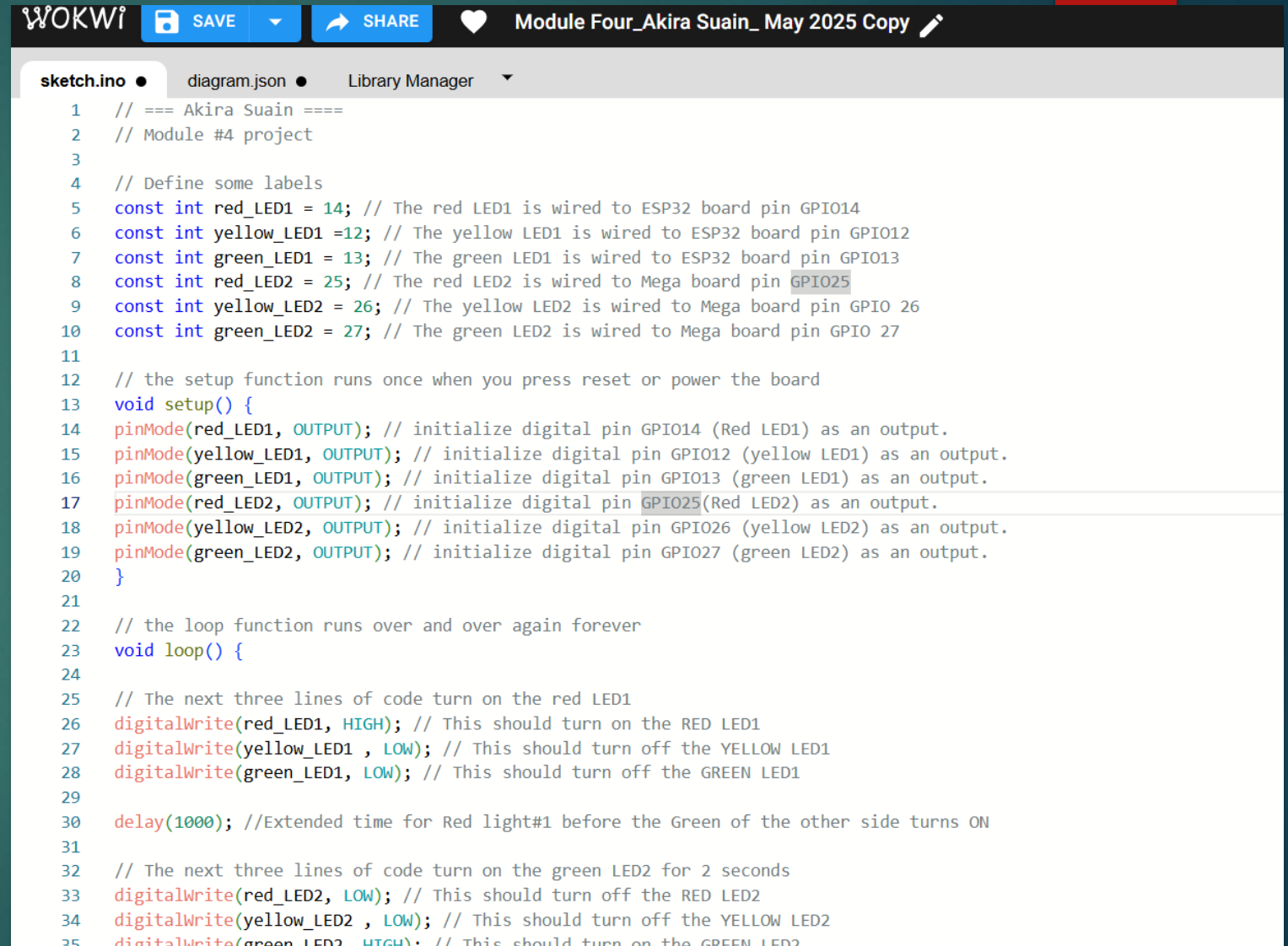
Picture of circuit with working LED

- ESP 32 Board
- Colored LEDs: Red, Yellow and Green (two sets)
- Wires
- Breadboard



Screenshot of code
Wokwi Code Editor
showing **your name in
the comment**

Screensho
t of code
in Wokwi



The screenshot shows the Wokwi Code Editor interface. At the top, there's a header bar with the Wokwi logo, a 'SAVE' button, a 'SHARE' button, a heart icon, and the text 'Module Four_Akira Suain_ May 2025 Copy'. Below the header, there's a tab bar with 'sketch.ino', 'diagram.json', and 'Library Manager'. The main area displays a C++ sketch for controlling LEDs. The code includes comments for pin definitions and setup, and a loop function that turns on the red LED1, then the yellow LED1, then the green LED1, and finally the green LED2 for 2 seconds.

```
1 // === Akira Suain ===
2 // Module #4 project
3
4 // Define some labels
5 const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14
6 const int yellow_LED1 = 12; // The yellow LED1 is wired to ESP32 board pin GPIO12
7 const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13
8 const int red_LED2 = 25; // The red LED2 is wired to Mega board pin GPIO25
9 const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPIO 26
10 const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO 27
11
12 // the setup function runs once when you press reset or power the board
13 void setup() {
14   pinMode(red_LED1, OUTPUT); // initialize digital pin GPIO14 (Red LED1) as an output.
15   pinMode(yellow_LED1, OUTPUT); // initialize digital pin GPIO12 (yellow LED1) as an output.
16   pinMode(green_LED1, OUTPUT); // initialize digital pin GPIO13 (green LED1) as an output.
17   pinMode(red_LED2, OUTPUT); // initialize digital pin GPIO25 (Red LED2) as an output.
18   pinMode(yellow_LED2, OUTPUT); // initialize digital pin GPIO26 (yellow LED2) as an output.
19   pinMode(green_LED2, OUTPUT); // initialize digital pin GPIO27 (green LED2) as an output.
20 }
21
22 // the loop function runs over and over again forever
23 void loop() {
24
25   // The next three lines of code turn on the red LED1
26   digitalWrite(red_LED1, HIGH); // This should turn on the RED LED1
27   digitalWrite(yellow_LED1, LOW); // This should turn off the YELLOW LED1
28   digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1
29
30   delay(1000); //Extended time for Red light#1 before the Green of the other side turns ON
31
32   // The next three lines of code turn on the green LED2 for 2 seconds
33   digitalWrite(red_LED2, LOW); // This should turn off the RED LED2
34   digitalWrite(yellow_LED2, LOW); // This should turn off the YELLOW LED2
35   digitalWrite(green_LED2, HIGH); // This should turn on the GREEN LED2
```




CEIS 114

Module 5

CREATING A MULTIPLE TRAFFIC LIGHT CONTROLLER WITH A CROSS WALK

AKIRA SUAIN

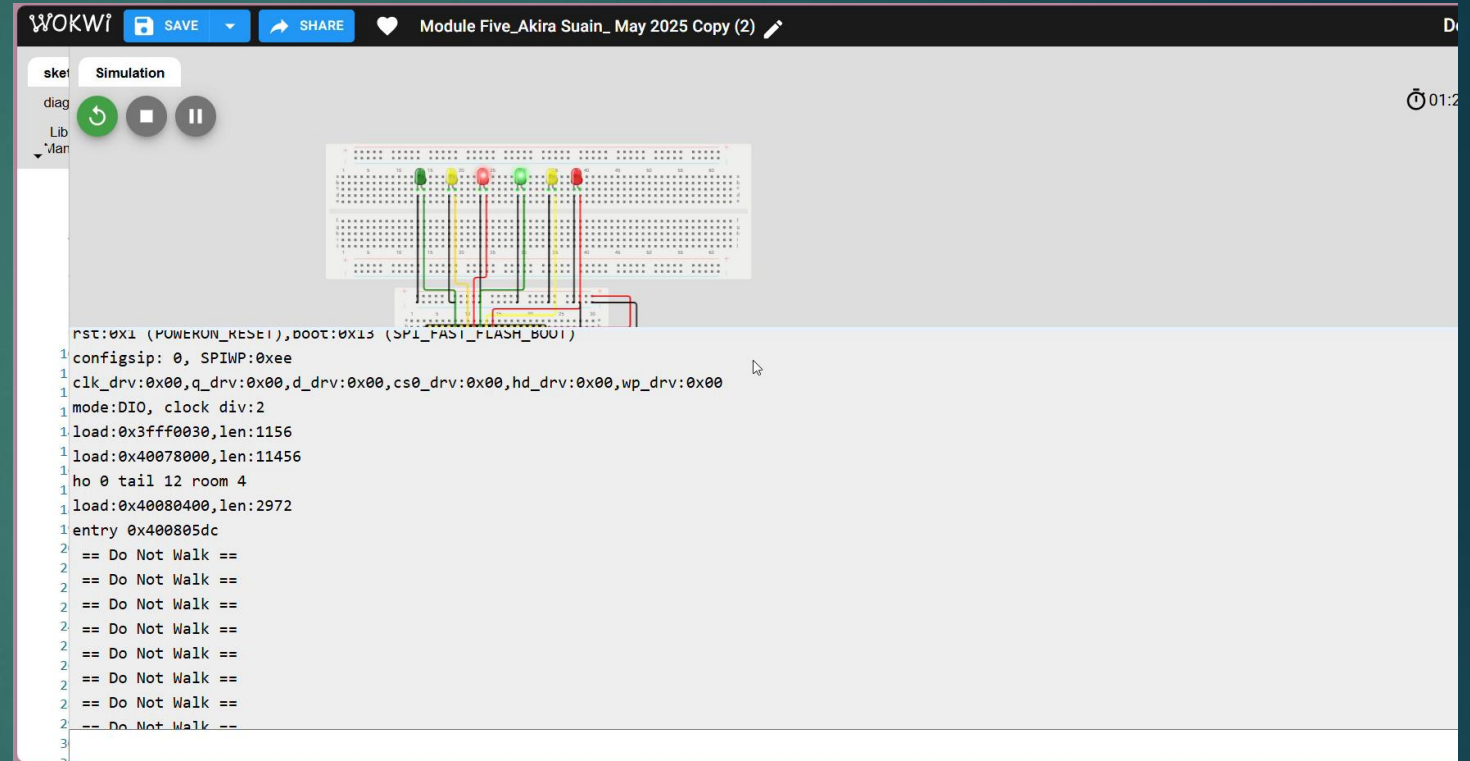
DUE: JUNE 8TH, 2025

Rubric – Due at the end of Week 5

| Activity | Requirement(s) | Points |
|---|--|--------|
| Include a picture of your circuit | Picture of the breadboard with LEDs ON | 20 |
| Screenshot of code showing your name in comment | Screenshot of code in Wokwi showing your name in one of the comments | 20 |
| Screenshot of output in Serial Monitor | Screenshot of Serial Monitor from Wokwi | 20 |

Screenshot of circuit with working LEDs

- ESP 32 Board
- Colored LEDs: Red, Yellow and Green (two sets)
- 220 Ohm Resistors (optional)
- Push Button
- Wires
- Breadboard





```
1 // === Akira Suain ===
2
3 // Module #5 project
4
5 const int red_LED1 = 14; // The red LED1 is wired to ESP32 board
6 const int yellow_LED1 = 12; // The yellow LED1 is wired to ESP32 board
7
8 const int green_LED1 = 13; // The green LED1 is wired to ESP32 board
9 const int red_LED2 = 25; // The red LED2 is wired to Mega board pin
10 const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin
11 const int green_LED2 = 27; // The green LED2 is wired to Mega board pin
12
13 int Xw_value;
14
15 const int Xw_button = 19; //Cross Walk button
16
17 // the setup function runs once when you press reset or power the board
18 void setup() {
19
20   pinMode(Xw_button, INPUT_PULLUP); // 0=pressed, 1 = unpressed button
21   Serial.begin(115200);
22   pinMode(red_LED1, OUTPUT); // initialize digital pin 14 (Red LED1)
23   pinMode(yellow_LED1, OUTPUT); // initialize digital pin 12 (yellow LED1)
24   pinMode(green_LED1, OUTPUT); // initialize digital pin 13 (green LED1)
25
26   pinMode(red_LED2, OUTPUT); // initialize digital pin 25 (Red LED2)
27   pinMode(yellow_LED2, OUTPUT); // initialize digital pin 26 (yellow LED2)
28   pinMode(green_LED2, OUTPUT); // initialize digital pin 27 (green LED2)
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33
34   // read the cross walk button value:
35   Xw_value = digitalRead(Xw_button);
```

SCREENSHOT OF CODE IN WOKWI

Screenshot of code in
Wokwi Code Editor showing
your name in the comment

sketch Simulation

diag

Lib

ets Jul 29 2019 12:21:46

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
1 load:0x40080400,len:2972
1 entry 0x400805dc
1 == Do Not Walk ==
1 == Do Not Walk ==
1 Count = 10 == Walk ==
1 Count = 9 == Walk ==
1 Count = 8 == Walk ==
1 Count = 7 == Walk ==
2 Count = 6 == Walk ==
2 Count = 5 == Walk ==
2 Count = 4 == Walk ==
2 Count = 3 == Walk ==
2 Count = 2 == Walk ==
2 Count = 1 == Walk ==
2 == Do Not Walk ==
```

SCREENSHOT OF SERIAL MONITOR IN WOKWI

Screenshot of output in
Serial Monitor



CEIS 114

Module 6

CREATING A MULTIPLE TRAFFIC LIGHT CONTROLLER WITH A CROSS WALK AND AN EMERGENCY BUZZER

AKIRA SUAIN

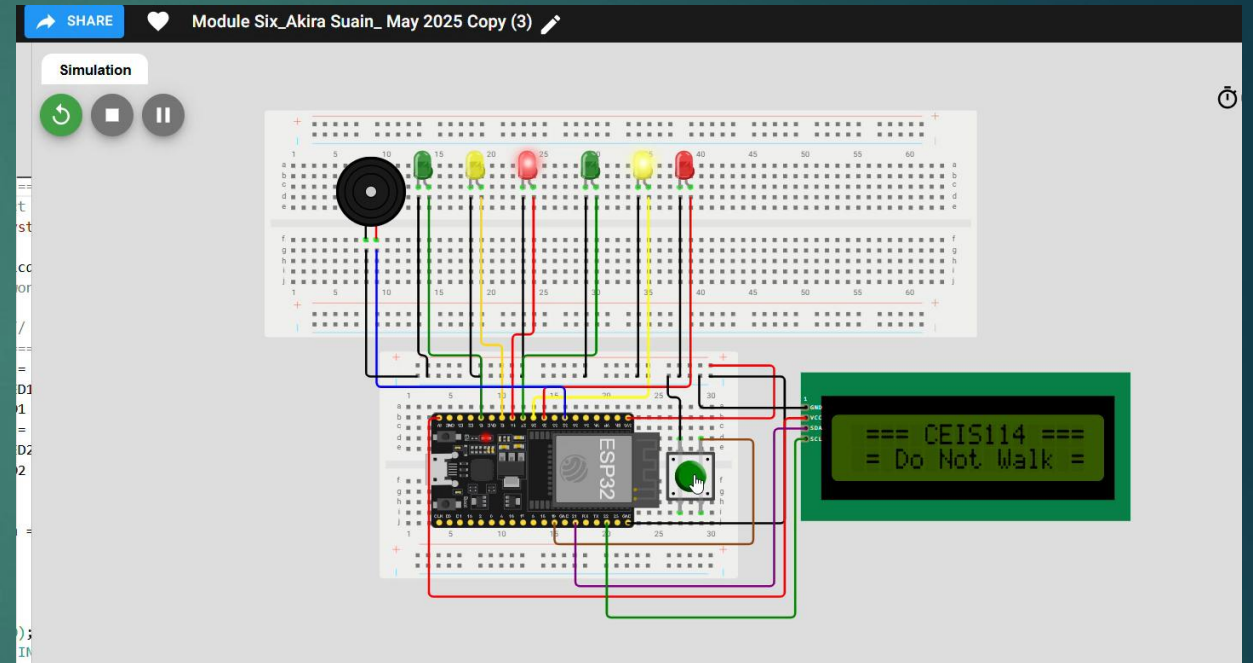
DUE JUNE 15TH 2025

Rubric – Due at the end of Week 6

| Activity | Requirement(s) | Points |
|---|--|--------|
| Include a picture of your circuit | Picture of the breadboard with LEDs ON and LCD displaying message. | 20 |
| Screenshot of code showing your name in comment | Screenshot of code showing your name in one of the comments | 20 |
| Screenshot of output in Serial Monitor | Screenshot of Serial Monitor | 20 |

Picture of circuit with working LEDs and LCD display

- ESP 32 Board
- Colored LEDs: Red, Yellow and Green (two sets)
- 220 Ohm Resistors (optional)
- Push Button
- LCD Unit with Message Display
- Wires
- Breadboard



Screenshot of code in Code Editor

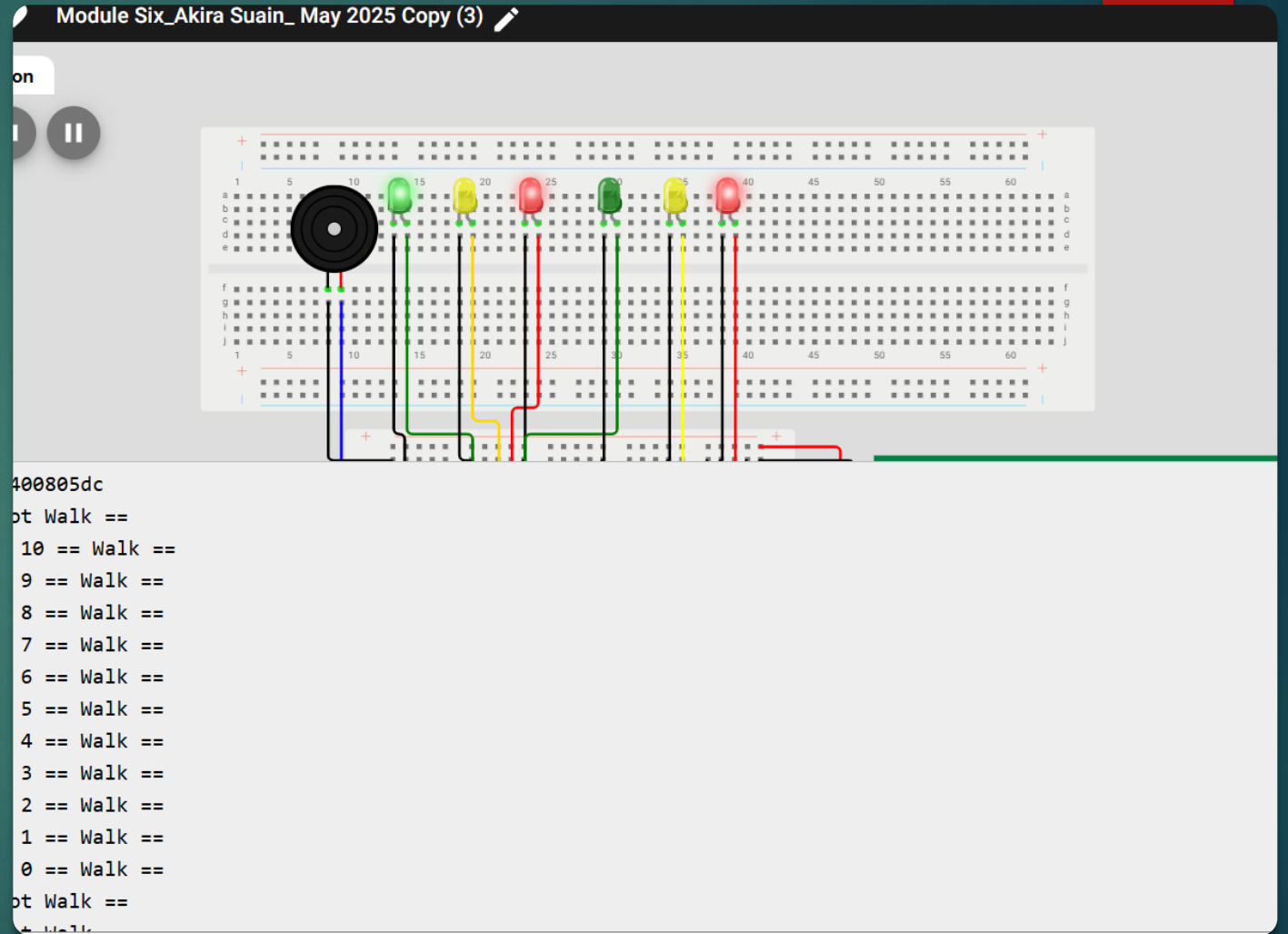
Screenshot of code in Code
Editor showing **your name in
the comment**



```
WOKWI SAVE SHARE Module Six_Akira Suain_ May 2025 Copy (3)
sketch.ino diagram.json libraries.txt Library Manager
1 // === Akira Suain ===
2 // Module #6 project #include <Wire.h> //lcd
3 #include <LiquidCrystal_I2C.h> //lcd
4
5 LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD address to 0x3F for a 16 chars and 2-line display
6 // if it does not work then try 0x3F, if both addresses do not work then run the scan code below
7
8 const int bzt=32; // GPIO32 to connect the Buzzer
9 //===== LCD =====
10 const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14
11 const int yellow_LED1 =12; // The yellow LED1 is wired to ESP32 board pin GPIO12
12 const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13
13 const int red_LED2 = 25; // The red LED2 is wired to Mega board pin GPIO25
14 const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPIO 26
15 const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO 27
16
17 int Xw_value;
18 const int Xw_button = 19; //Cross Walk button
19
20 void setup()
21 {
22
23   Serial.begin(115200);
24   pinMode(Xw_button, INPUT_PULLUP); // 0=pressed, 1 = unpressed button
25
26   lcd.init(); // initialize the lcd lcd.backlight();
27   lcd.setCursor(0,0); // column#4 and Row #1
28   lcd.print(" === CEIS114 ===");
29   pinMode(bzt,OUTPUT);
30
31   pinMode(red_LED1, OUTPUT); // initialize digital pin 14 (Red LED1) as an output.
32   pinMode(yellow_LED1, OUTPUT); // initialize digital pin12 (yellow LED1) as an output.
33   pinMode(green_LED1, OUTPUT); // initialize digital pin 13 (green LED1) as an output.
```


Screenshot of Serial Monitor

Screenshot of output in Serial Monitor



CEIS 114

Week 7 Project

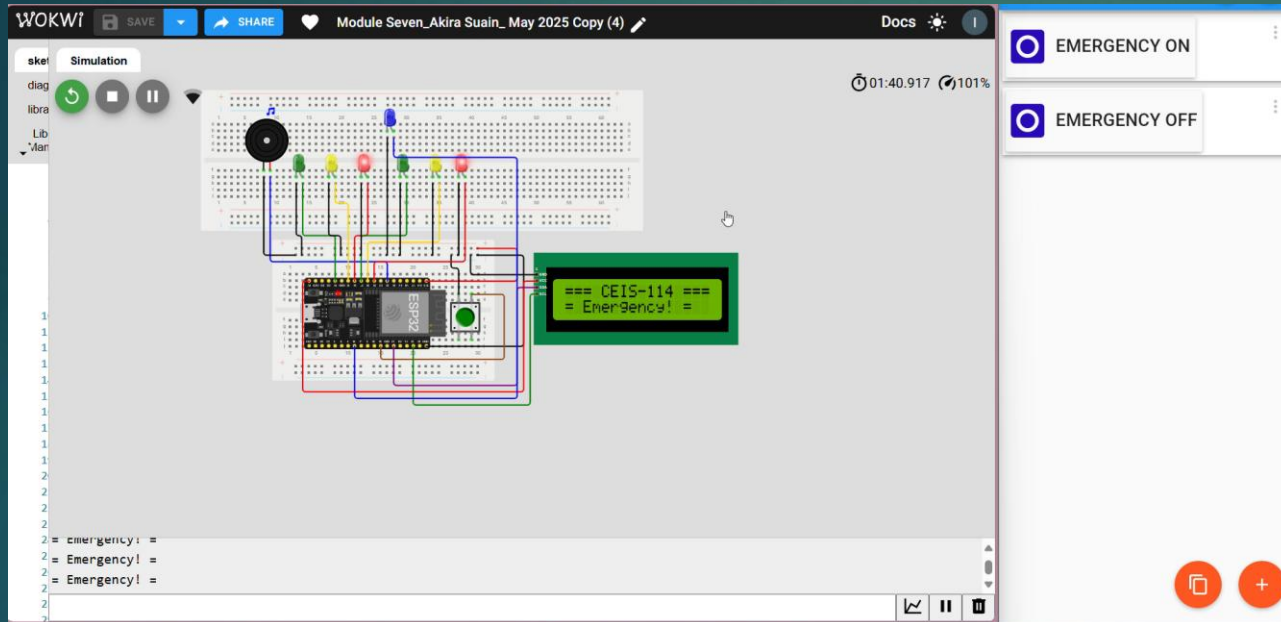
CREATING A MULTIPLE TRAFFIC LIGHT CONTROLLER
WITH A CROSS WALK AND AN EMERGENCY BUZZER
WITH SECURED IOT CONTROL VIA WEB
AKIRA SUAIN

DUE JUNE 22ND 2025

Rubric – Due at the end of Week 7

| Activity | Requirement(s) | Points |
|--------------------|--|--------|
| Building/Operation | Screenshot of circuit with working LEDs and LCD display | 20 |
| Testing | Screenshot of code in Code Editor | 20 |
| Testing | Screenshot of Serial Monitor | 20 |

Screenshot of circuit with working LEDs and LCD display (Building/Operation)



ESP 32 Board

Colored LEDs:
Red, Yellow and
Green (two sets)

One Blue LED –
Emergency Light

Push Button

LCD Unit

Buzzer

Wires

Breadboard



sketch.ino

diagram.json

libraries.txt

Library Manager

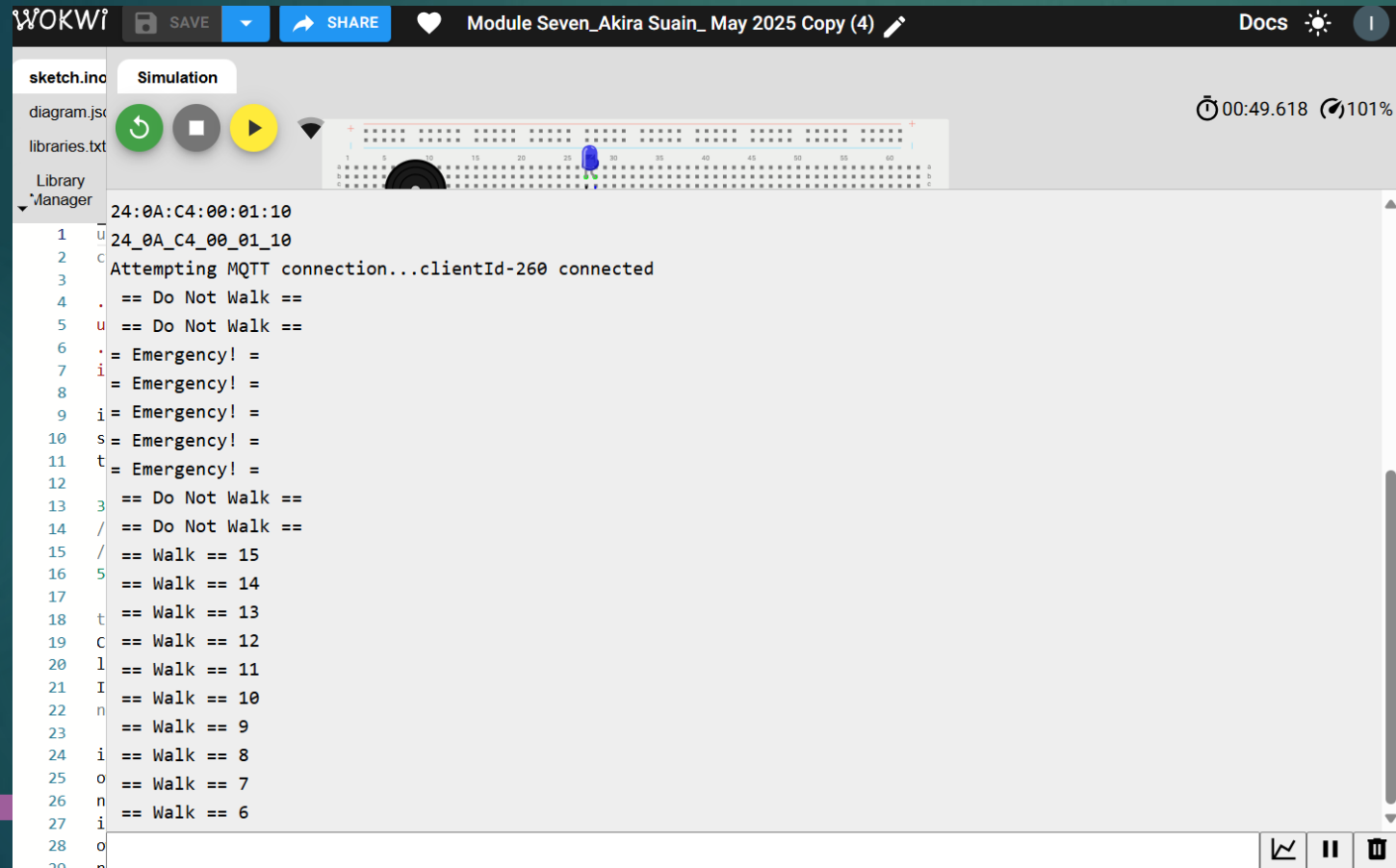


```
1 // === Akira Suain ===
2 // Final Project Component, Option 1
3
4 #include <WiFi.h> // WiFi header file
5 #include <PubSubClient.h> // MQTT publish and subscribe header file
6 #include <Wire.h> // I2C header file
7 #include <LiquidCrystal_I2C.h> // I2C lcd header file
8
9 const char* ssid = "Wokwi-GUEST"; // This is the access point to your wireless network.
10 const char* password = ""; // This is the password to the SSID. For the smart mini rout
11 const char* mqttServer = "test.mosquitto.org"; // This is the free MQTT broker we will
12
13 int port = 1883; // MQTT brokers listen to port 1883 by default
14 String stMac; // C string used for convenience of comparisons.
15 char mac[50]; // C char array used to hold the MAC address of your ESP32 microconroller
16 char clientId[50]; // This client ID is used to identify the user accessing the MQTT br
17
18 // For our test.mosquitto.org broker, we just generate a random user client ID
19 WiFiClient espClient; // instantiate the WiFi client object
20 PubSubClient client(espClient); // instantiate the publish subscribe client object
21 LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD address to 0x27 for a 16 chars and 2-l
22 // if it does not work then try 0x3F, if both addresses do not work then run the scan o
23
24 const int redLightNorthSouth = 14; // The red LED NS is wired to ESP32 board pin GPIO 1
25 const int yellowLightNorthSouth = 12; // The yellow LED NS is wired to ESP32 board pin
26 const int greenLightNorthSouth = 13; // The green LED NS is wired to ESP32 board pin GP
27 const int redLightEastWest = 25; // The red LED EW is wired to ESP32 pin GPIO 25
28 const int yellowLightEastWest = 26; // The yellow LED EW is wired to ESP32 board pin GP
29 const int greenLightEastWest = 27; // The green LED EW is wired to ESP32 board pin GPIO
30
31 int crossWalkButtonState = 1 ; // Variable will store the state of the crosswalk button
32 const int crossWalkButton = 19; // Cross walk button pin is GPIO 19
33 const int emergencyBlueLED = 16; // The blue LED is wired to ESP32 board pin GPIO 16
34 const int bumperPin = 22; // Active Bumper pin is GPIO 22
```

Screenshot of code in Code Editor (Testing)

Screenshot of code in Code Editor showing **your name in the comment**

Screenshot of Serial Monitor (Testing)



The screenshot shows the Wokwi IDE interface. At the top, the title bar reads "WOKWI" with "SAVE" and "SHARE" buttons, followed by the project name "Module Seven_Akira Suain_ May 2025 Copy (4)" and a "Docs" button. Below the title bar, the "Simulation" tab is active, showing a green play button, a red stop button, and a yellow refresh button. A progress bar at the top right indicates a runtime of 00:49.618 and 101% zoom. The main area displays the serial monitor output, which includes the MAC address "24:0A:C4:00:01:10", the MQTT connection attempt, and a list of messages. The messages are as follows:

```
1 u 24_0A_C4_00_01_10
2 c Attempting MQTT connection...clientId=260 connected
3 . == Do Not Walk ==
4 u == Do Not Walk ==
5 . = Emergency! =
6 i = Emergency! =
7 i = Emergency! =
8 s = Emergency! =
9 t = Emergency! =
10 3 == Do Not Walk ==
11 / == Do Not Walk ==
12 / == Walk == 15
13 5 == Walk == 14
14 t == Walk == 13
15 c == Walk == 12
16 l == Walk == 11
17 I == Walk == 10
18 n == Walk == 9
19 i == Walk == 8
20 o == Walk == 7
21 n == Walk == 6
```

Screenshot of output in Serial Monitor

Challenges

- ▶ Website Functionality: Working in Wokwi.com came with a few technical headaches. The interface was often glitchy, especially the mouse alignment. When dragging components like LEDs onto the screen, the delay made it hard to place them precisely, leading to a lot of frustrating trial and error.
- ▶ Wiring Overload: Although the individual wiring steps weren't too difficult, the sheer number of connections became overwhelming by the end. The circuit was so cluttered that I ended up having to rebuild large portions of it just to troubleshoot and clean up the layout.
- ▶ Coding from Scratch: Instead of copying and pasting code, I challenged myself to write everything from the ground up. It took more time, but it taught me how to properly debug—reading error codes and tracing them to the right line, instead of giving up and restarting the whole script.
- ▶ Module Five Button Bug: The module five button had a five-second delay that wasn't explained in the documentation or videos. I kept redoing it, thinking something was broken, until I realized it was a built-in delay. That experience taught me to be more observant about how modules behave beyond what's explicitly taught.

Career Skills

Problem Solving & Debugging: I learned to approach errors methodically, especially while coding from scratch. This helped me build confidence in debugging and taught me how to read error messages to fix issues efficiently, a valuable skill in any tech role.

Attention to Detail: From aligning glitchy components in Wokwi to untangling messy wiring, I had to stay focused and precise. That level of attention will be useful in both hardware design and cybersecurity work, where small mistakes can have big consequences.

Resilience & Adaptability: When things didn't work as expected—like the unexpected delay in the module five button—I kept experimenting until I figured it out. This built my patience and adaptability, which are key in fast-paced IT environments.

Independent Learning & Self-Motivation: Choosing to write my own code rather than copy it showed me how much I enjoy solving problems on my own. That kind of initiative is something I hope to bring to internships and future job roles

Conclusion

Building a smart traffic control system in Wokwi pushed me beyond just connecting wires and uploading code—it challenged my patience, sharpened my problem-solving skills, and reminded me that progress often comes through trial and error.

Despite the technical glitches, confusing wiring, and coding setbacks, I stuck with it. I walked away with more than just a functioning project—I gained real experience in how to think like a developer and troubleshoot like a professional.

This project gave me a glimpse into what it's like to build and improve complex systems, which ties perfectly into the career I'm pursuing in cybersecurity and IT. I'm proud of how far I've come and excited for the challenges I'll face next.

Wix: [Blog | Cybersercuity Project](#)