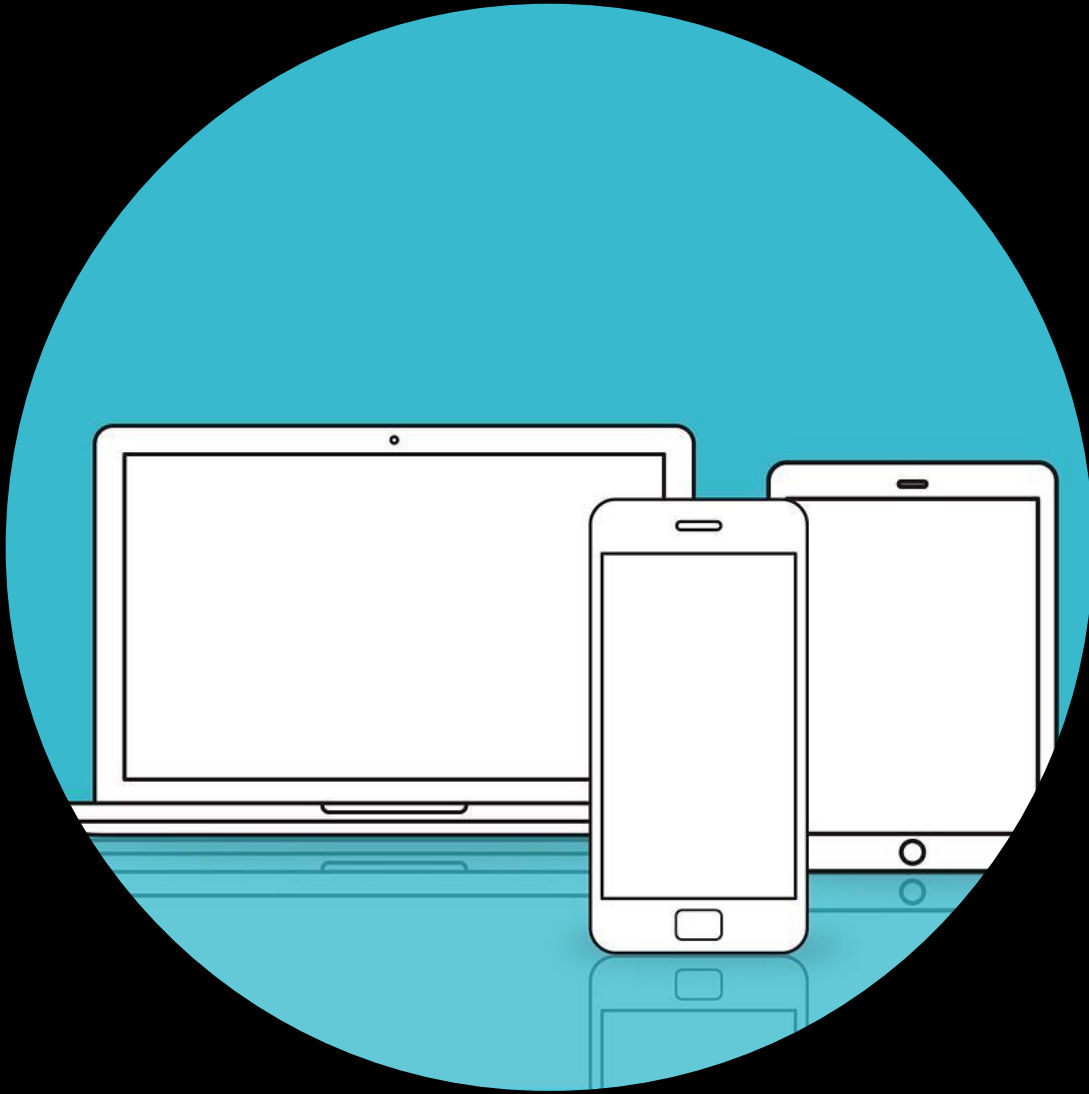


# Introduction to Digital Devices (CEIS114)



Final Project

Jonathan Waugh

Professor Sean Caruthers

Oct. 2024

# Introduction



This presentation covers my six-part project completed while taking Introduction to Digital Devices (CEIS114).



It includes examples of project planning for IoT traffic controller, creation of multi-light traffic controller, adding a crosswalk and emergency buzzer, secured IoT control via web

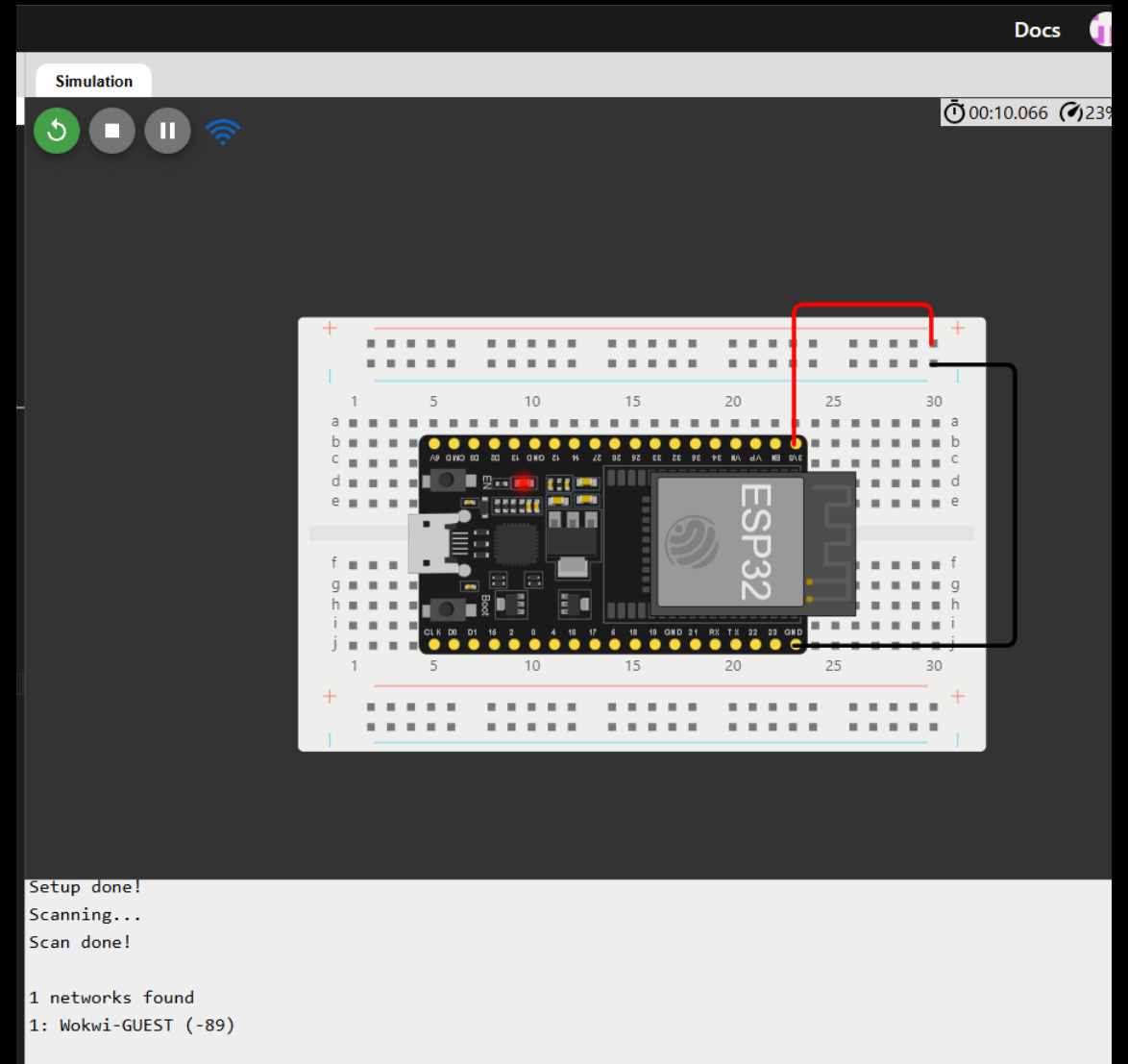
# Project Plan for IoT Traffic Controller

- ESP32 installed and powered on
- WIFI scan



# ESP32 (Screenshot)

- Microcontroller mounted and powered ON



```
Initializing WiFi...
```

```
Setup done!
```

```
Scanning...
```

```
Scan done!
```

```
1 networks found
```

```
1: Wokwi-GUEST (-89)
```



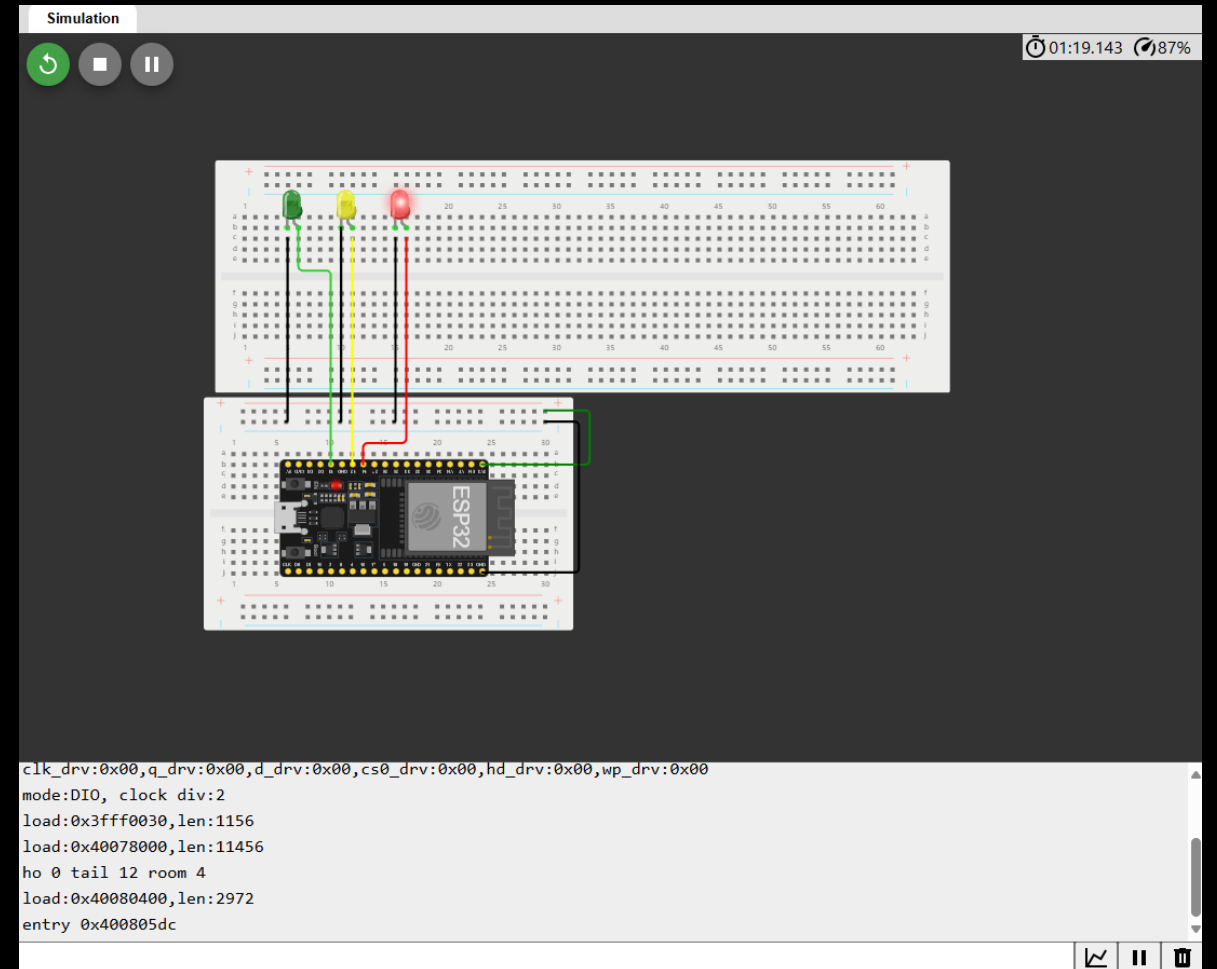
## ESP32 WiFi Scan

# Creating the Traffic Controller



# Picture of circuit with working LEDs

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





# Screenshot of code in the Code Editor

```
1 // === Jonathan Waugh ===  
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  
35   digitalWrite(yellow_LED1, HIGH); // This should turn on the YELLOW LED1  
36   digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1  
37  
38   delay(2000); // wait for 2 seconds  
39 }  
40 |
```

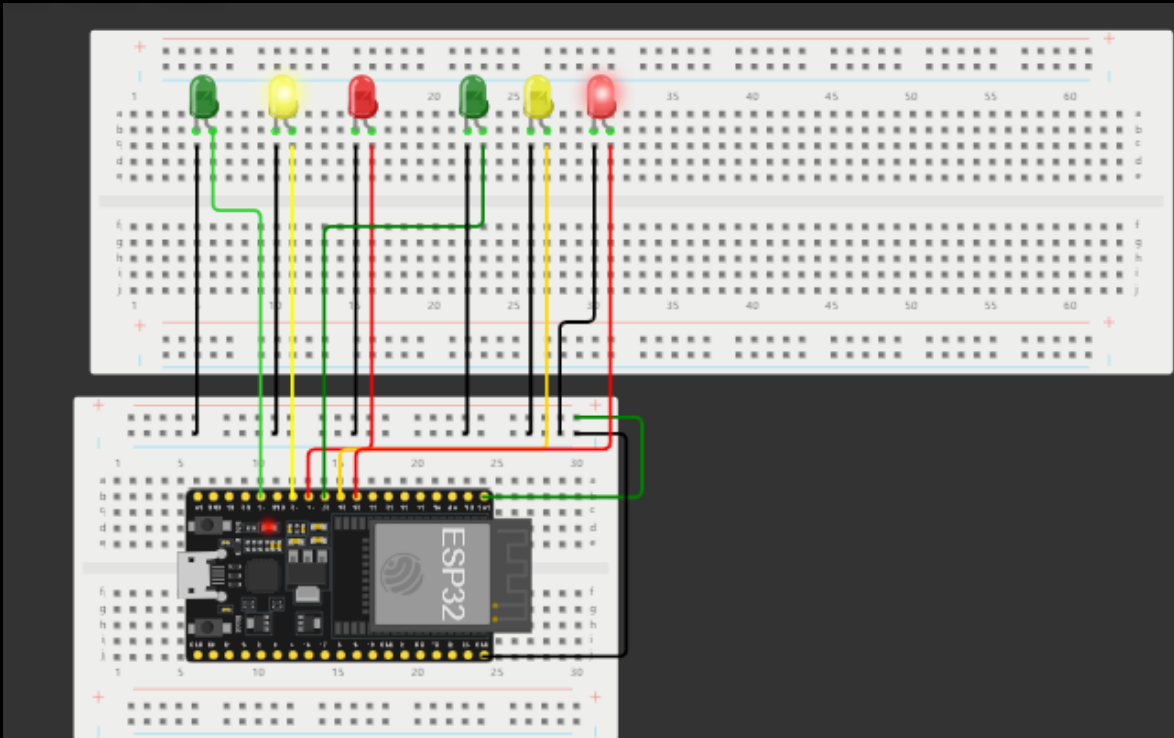


The background of the slide is a dark, out-of-focus photograph of city lights at night. It features numerous circular bokeh lights in warm yellow and white tones on the left side, and a dense cluster of red and orange bokeh lights on the right side. The overall effect is a blurred, artistic representation of a city skyline or traffic lights.

# Creating a Multiple Traffic Light Controller

# Picture of circuit with working LEDs

---



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

```
1 // === Jonathan Waugh ===
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 GPIO26
10 const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO27
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 s
31
32   // The next three lines of code turn on the green LED2 for 2 seconds
```

# Screenshot of code in Wokwi



A photograph of a city street scene. In the foreground, a crosswalk with thick yellow stripes is painted on a dark asphalt road. In the background, a dark-colored sedan is parked or stopped. To the right of the car, a person is standing next to a bicycle. On the left side of the road, there are black trash bins and some trees. The overall scene is slightly blurred, suggesting a candid shot.

# Adding a Cross Walk



WOKWI

SAVE

SHARE

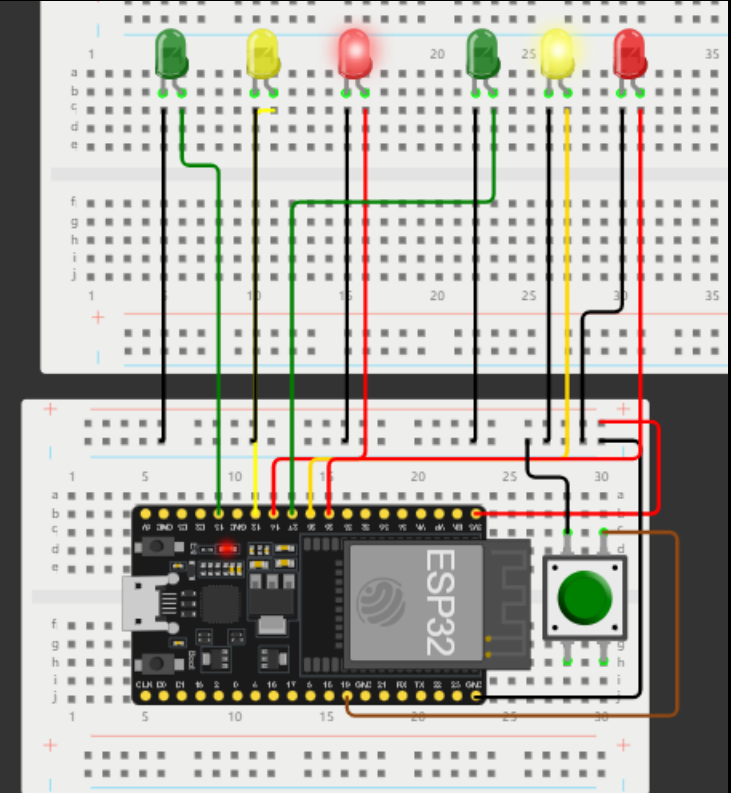
Module5\_Waugh

sketch.ino • diagram.json • Library Manager

```
1 // === Jonathan Waugh ===
2
3 // Module #5 project
4
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
8 const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13
9 const int red_LED2 = 25; // The red LED2 is wired to Mega board pin GPIO25
10 const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPIO 26
11 const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO 27
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) as an output.
23   pinMode(yellow_LED1, OUTPUT); // initialize digital pin 12 (yellow LED1) as an output.
24   pinMode(green_LED1, OUTPUT); // initialize digital pin 13 (green LED1) as an output.
25
26   pinMode(red_LED2, OUTPUT); // initialize digital pin 25(Red LED2) as an output.
27   pinMode(yellow_LED2, OUTPUT); // initialize digital pin 26 (yellow LED2) as an output.
28   pinMode(green_LED2, OUTPUT); // initialize digital pin 27 (green LED2) as an output.
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);
36
37   if (Xw_value == LOW ){ // if crosswalk button (X-button) pressed
38
39     digitalWrite(yellow_LED1 , LOW); // This should turn off the YELLOW LED1
40     digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1
41     digitalWrite(yellow_LED2 , LOW); // This should turn off the YELLOW LED2
42     digitalWrite(green_LED2, LOW); // This should turn off the GREEN LED2
43
44     for (int i=10; i>0; i--)
45
```

Screenshot  
of code in  
Wokwi

# Screenshot of Serial Monitor in Wokwi



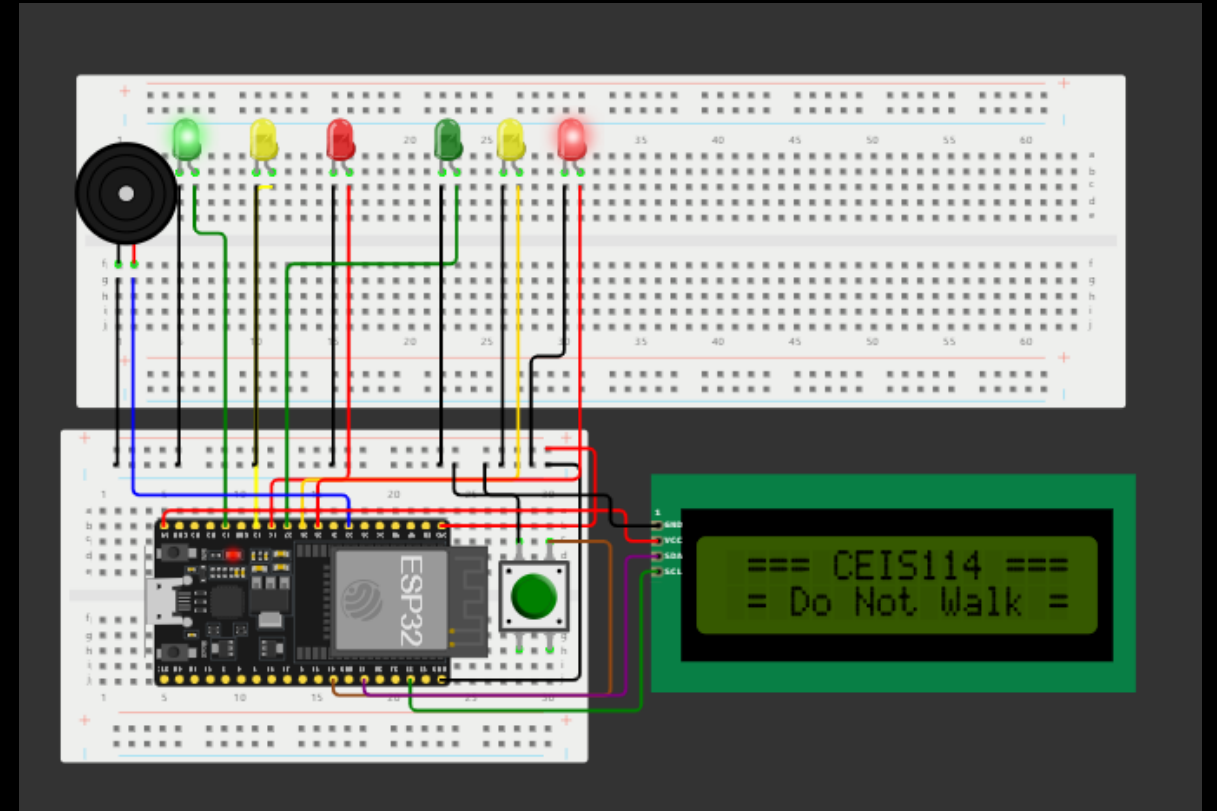
```
Count = 6 == Walk ==  
Count = 5 == Walk ==  
Count = 4 == Walk ==  
Count = 3 == Walk ==  
Count = 2 == Walk ==  
Count = 1 == Walk ==  
== Do Not Walk ==
```





## 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



WOKWI

SAVE

SHARE

Module6\_Waugh

sketch.ino

diagram.json

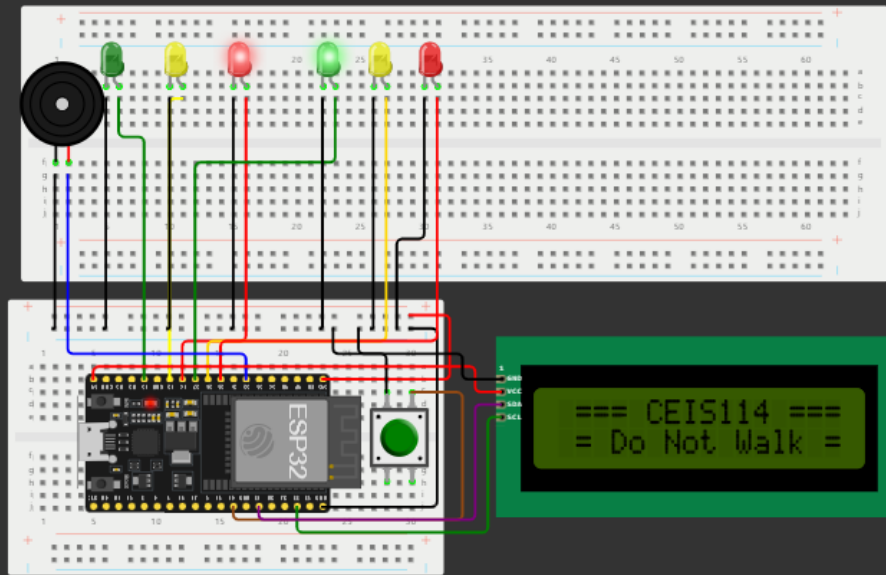
libraries.txt

Library Manager

```
1 // === Jonathan Waugh ===
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=presed, 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.
34
35   pinMode(red_LED2, OUTPUT); // initialize digital pin 25(Red LED2) as an output.
36   pinMode(yellow_LED2, OUTPUT); // initialize digital pin 26 (yellow LED2) as an output.
37   pinMode(green_LED2, OUTPUT); // initialize digital pin 27 (green LED2) as an output.
38
39 }
40
41 // the loop function runs over and over again forever
42 void loop()
43 {
44
45   // read the cross walk button value:
```

Screenshot  
of code in  
Code Editor

# Screenshot of Serial Monitor



```
Count = 4 == Walk ==  
Count = 3 == Walk ==  
Count = 2 == Walk ==  
Count = 1 == Walk ==  
Count = 0 == Walk ==  
== Do Not Walk ==
```

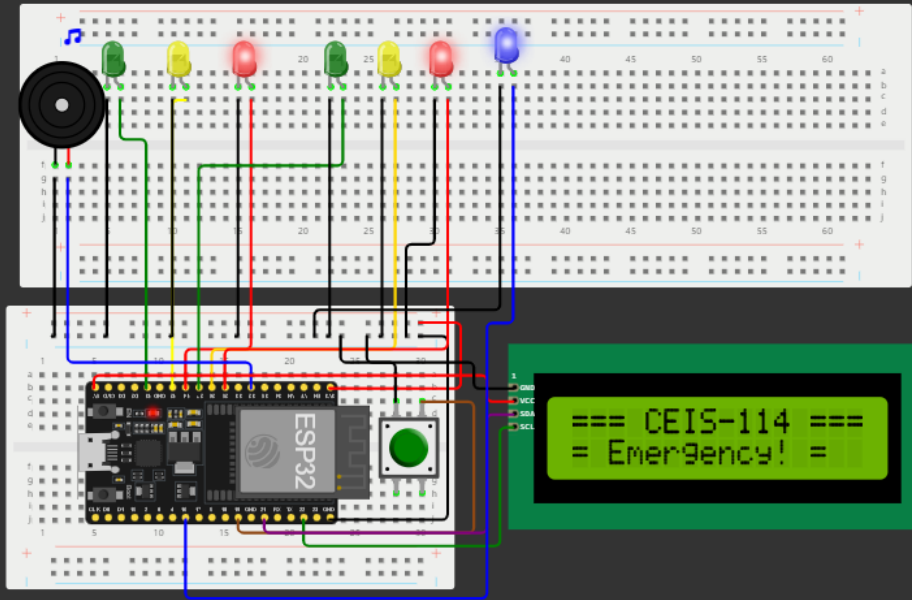




# Secured IoT Control via WEB

## 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

# Screenshot of code in Code Editor (Testing)

```
WOKWI SAVE SHARE Final-Project_Waugh
sketch.ino diagram.json libraries.txt Library Manager
1 // === Jonathan Waugh ===
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 router
11 const char* mqttServer = "test.mosquitto.org"; // This is the free MQTT broker we will use.
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 microcontroller
16 char clientId[50]; // This client ID is used to identify the user accessing the MQTT broker.
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-line display
22 // if it does not work then try 0x3F, if both addresses do not work then run the scan code
23
24 const int redLightNorthSouth = 14; // The red LED NS is wired to ESP32 board pin GPIO 14
25 const int yellowLightNorthSouth = 12; // The yellow LED NS is wired to ESP32 board pin GPIO 12
26 const int greenLightNorthSouth = 13; // The green LED NS is wired to ESP32 board pin GPIO 13
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 GPIO 26
29 const int greenLightEastWest = 27; // The green LED EW is wired to ESP32 board pin GPIO 27
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 buzzerPin = 32; // Active Buzzer pin is GPIO 32
35
36 int loopCount; // Variable will keep count of the number of times the light pattern repeats
37 int secondsLeft; // counter to keep track of number of seconds left for crossing intersection
38 int iotControl = 0; // Variable will be used to switch between emergency and normal operations of
39
40 // traffic controller
41 void setup() {
42
43     Serial.begin(115200); // set baud rate of serial monitor to 115200 bits per second
44     randomSeed(analogRead(0)); // seed the random() function
45     delay(10); // wait 10 milliseconds
46 }
```



```
== Do Not Walk ==  
= Emergency! =  
= Emergency! =  
= Emergency! =  
= Emergency! =  
= Emergency! =  
== Do Not Walk ==
```

# Screenshot of Serial Monitor (Testing)





# Challenges

---

- Challenges I faced while doing these projects were learning the code when not writing it myself
- I overcame this challenge by reading all the editor notes and making sure I understood each line and what its function is

The background features a light blue field with several large, semi-transparent blue gears of different sizes. A hand is visible in the bottom right corner, holding a black marker. A large, irregular black shape, resembling a hand-drawn brushstroke, frames the central text and list.

# Career Skills

- Diagram design
- Embedded systems knowledge
  - Project Management
- Electronics and circuit design
- Networking and communications
  - Programming
  - Attention to Detail



# Conclusion

---

This class provided a great teaching on the introduction to digital devices. With these teachings I now can design, code, troubleshoot, and manage different designs that include linking a secured IoT controller via web. Learning these skills will allow me to better understand device control, design, and troubleshooting and hone my skills to advance me in my career path. Learning about digital devices will allow me to design and bring to life products that could help people in their day to day lives.



# References

- Live Lessons. Professor Muqri, Mohammad and Prof. Jellouli, Prof. Majumder, Prof. McKenzie, Prof. Mortezaie, Prof. Duclos, Prof. Ng, Prof. Qian, Prof. Sommer, Prof. St. John, Dr. Tilghman, Prof. Wang. CEIS114, 2024
- Project Guides



# Wix Portfolio

Portfolio | Jonathan Waugh | IT services and logo/website design