CEIS 114 Final Project Deliverables PowerPoint

- Name: Anthony Norman
- Professor: Shadeeb Hossain
- Session: CEIS 114
- Date: 2/25/2025

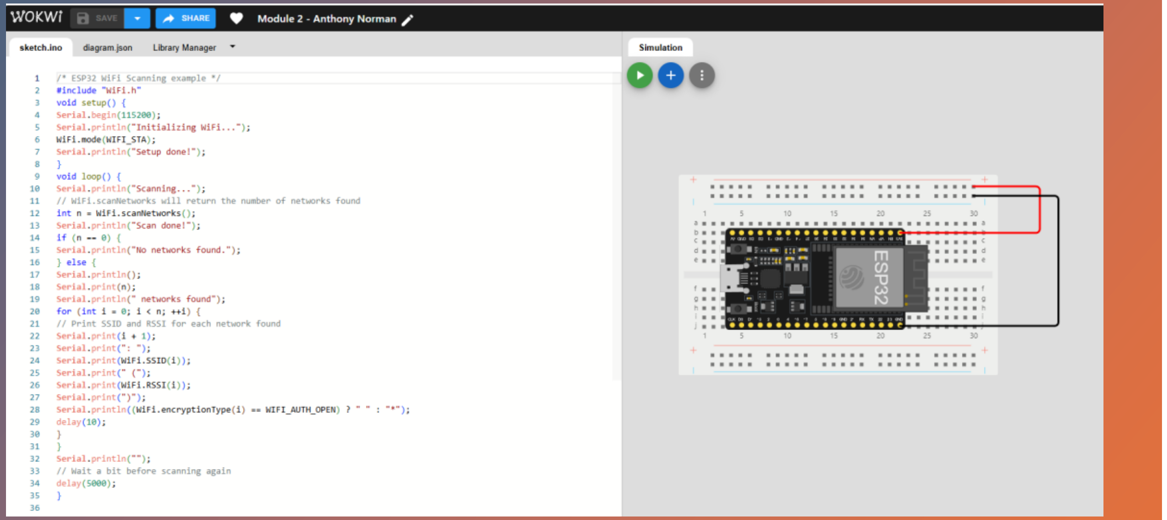
INTRODUCTION

- In this project we created a traffic alarm system.
- You will see through the slides the series of different wires, LEDs, and much more that we had to put together to make this system work.

CEIS 114 Module 2

Project Plan for IoT Traffic Controller

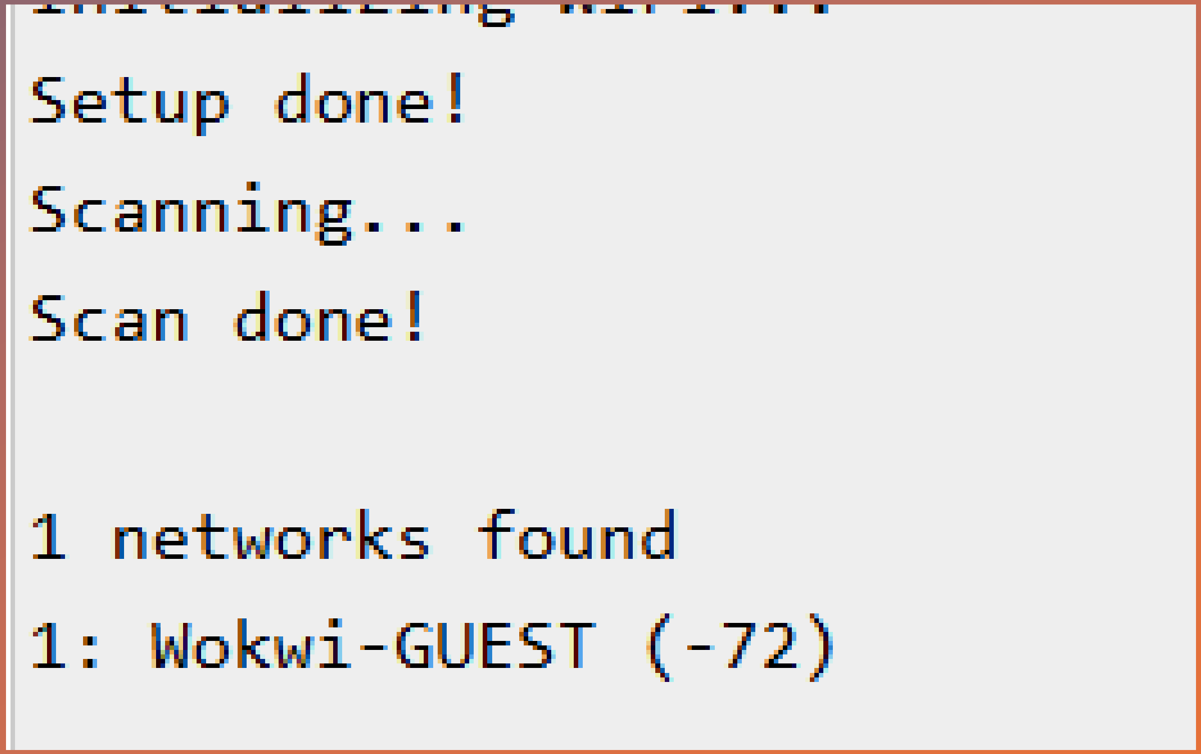
ESP32 (Screenshot)



```
/* ESP32 WiFi Scanning example */
#include "WiFi.h"
void setup() {
Serial.begin(115200);
Serial.println("Initializing WiFi...");
WiFi.mode(WIFI_STA);
Serial.println("Setup done!");
}
void loop() {
Serial.println("Scanning...");
// WiFi.scanNetworks will return the number of networks found
int n = WiFi.scanNetworks();
Serial.println("Scan done!");
if (n == 0) {
Serial.println("No networks found.");
} else {
Serial.println();
Serial.print(n);
Serial.println(" networks found");
for (int i = 0; i < n; ++i) {
// Print SSID and RSSI for each network found
Serial.print(i + 1);
Serial.print(": ");
Serial.print(WiFi.SSID(i));
Serial.print(" (");
Serial.print(WiFi.RSSI(i));
Serial.print(")");
Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN) ? " " : "*");
delay(10);
}
}
Serial.println("");
// Wait a bit before scanning again
delay(5000);
}

```

- Microcontroller mounted and powered ON

ESP32 WiFi Scan

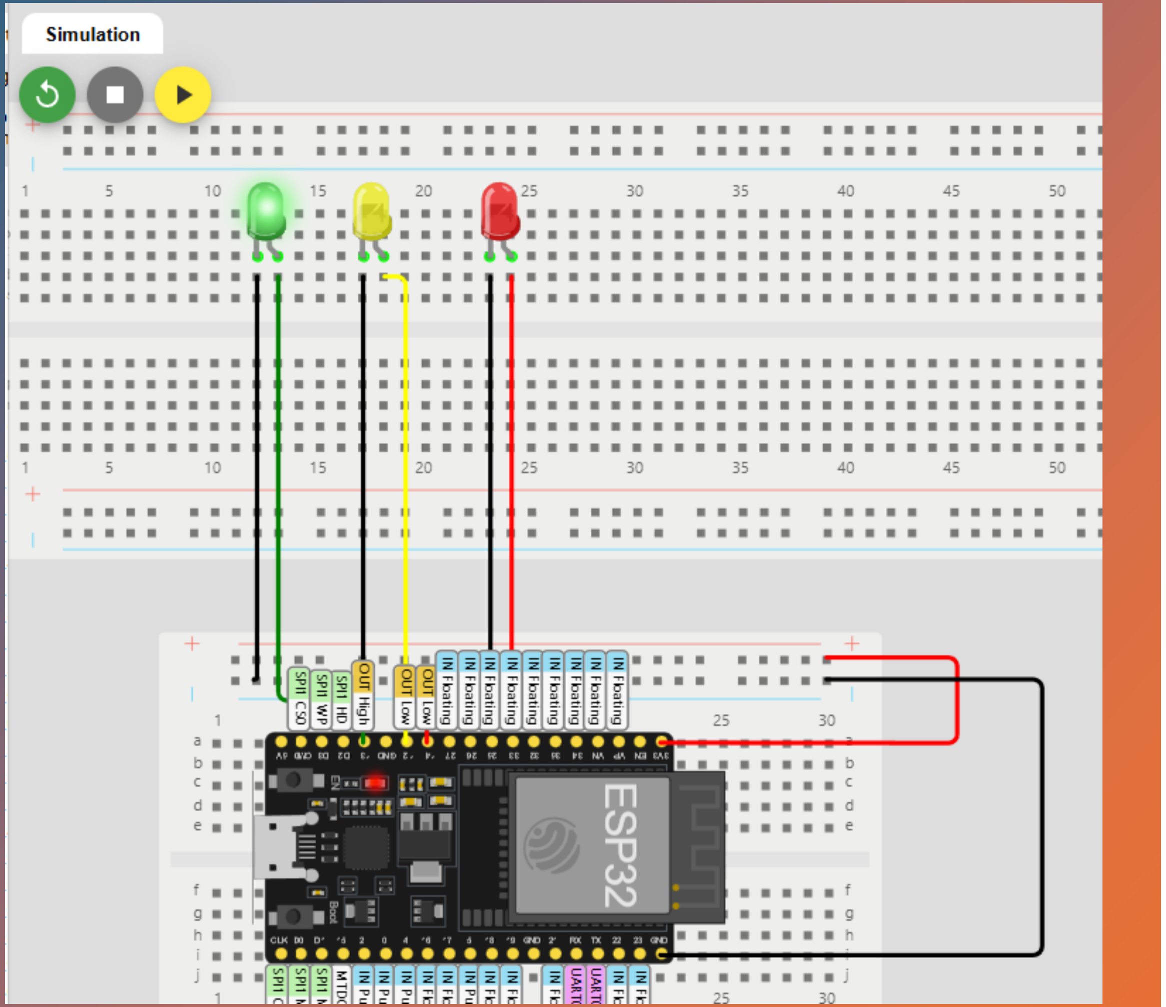


```
Initializing WiFi...
Setup done!
Scanning...
Scan done!

1 networks found
1: Wokwi-GUEST (-72)
```

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

CEIS 114 Module 3

Creating the Traffic Controller

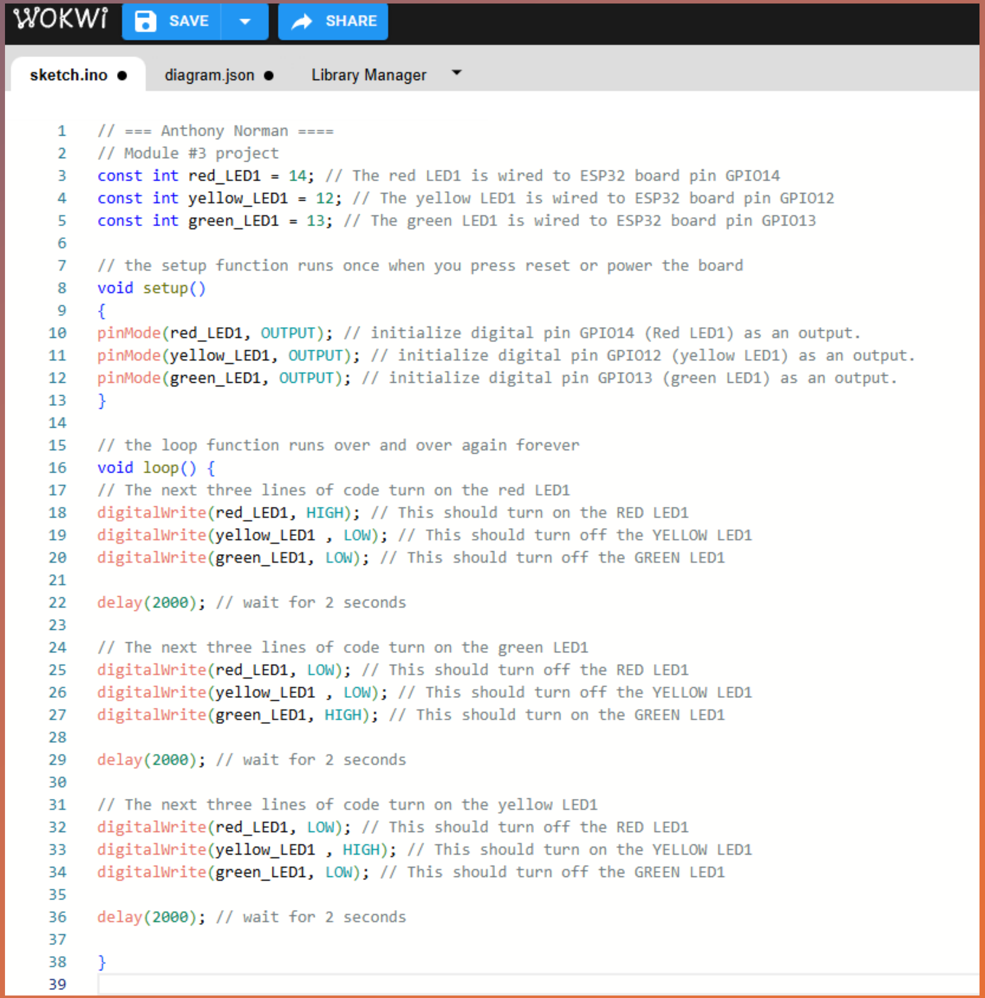


Picture of circuit with working LEDs

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

Screenshot of code in the Code Editor

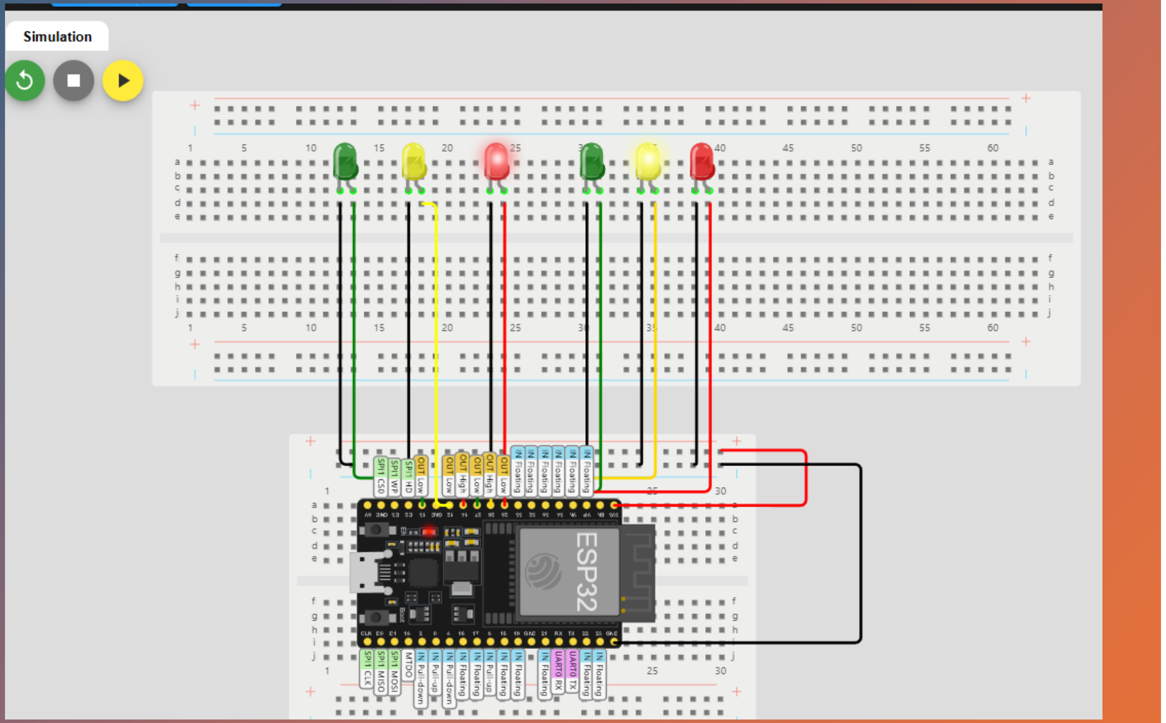
Screenshot of code in the Wokwi Code Editor showing **your name in the comment**



```
// === Anthony Norman ====
// Module #3 project
const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14
const int yellow_LED1 = 12; // The yellow LED1 is wired to ESP32 board pin GPIO12
const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13

// the setup function runs once when you press reset or power the board
void setup()
{
pinMode(red_LED1, OUTPUT); // initialize digital pin GPIO14 (Red LED1) as an output.
pinMode(yellow_LED1, OUTPUT); // initialize digital pin GPIO12 (yellow LED1) as an output.
pinMode(green_LED1, OUTPUT); // initialize digital pin GPIO13 (green LED1) as an output.
}

// the loop function runs over and over again forever
void loop() {
// The next three lines of code turn on the red LED1
digitalWrite(red_LED1, HIGH); // This should turn on the RED LED1
digitalWrite(yellow_LED1 , LOW); // This should turn off the YELLOW LED1
digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1

delay(2000); // wait for 2 seconds

// The next three lines of code turn on the green LED1
digitalWrite(red_LED1, LOW); // This should turn off the RED LED1
digitalWrite(yellow_LED1 , LOW); // This should turn off the YELLOW LED1
digitalWrite(green_LED1, HIGH); // This should turn on the GREEN LED1

delay(2000); // wait for 2 seconds

// The next three lines of code turn on the yellow LED1
digitalWrite(red_LED1, LOW); // This should turn off the RED LED1
digitalWrite(yellow_LED1 , HIGH); // This should turn on the YELLOW LED1
digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1

delay(2000); // wait for 2 seconds

}

```

CEIS 114 Module 4

Creating a Multiple Traffic Light Controller

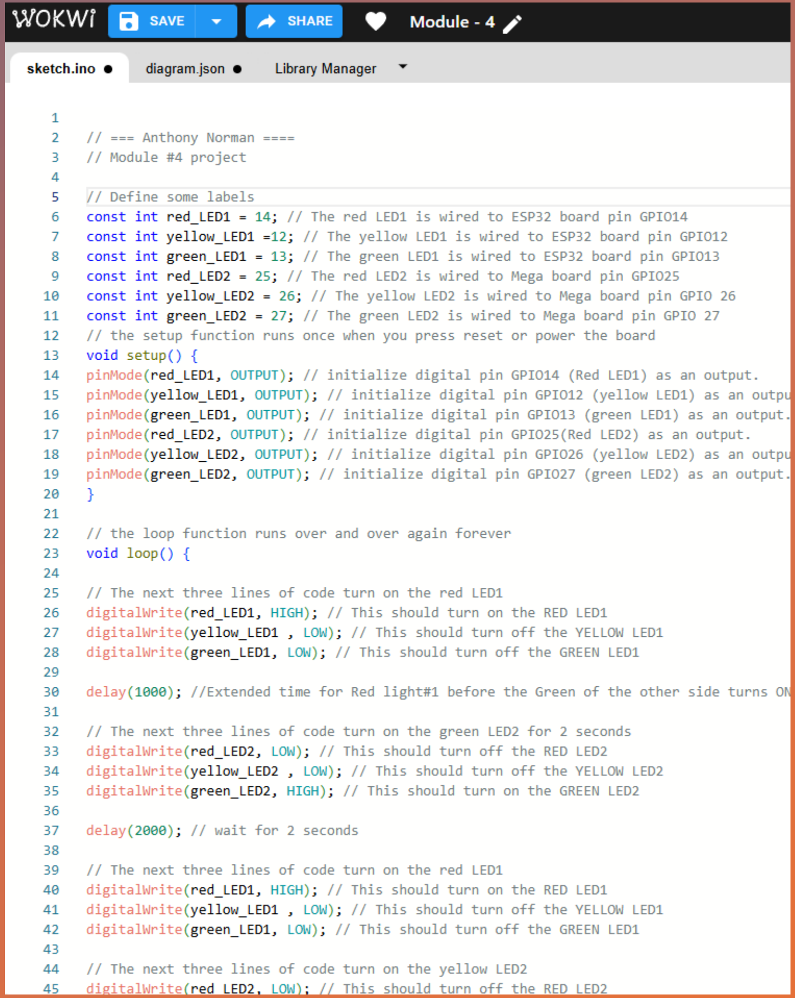


Picture of circuit with working LEDs

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

Screenshot of code in Wokwi

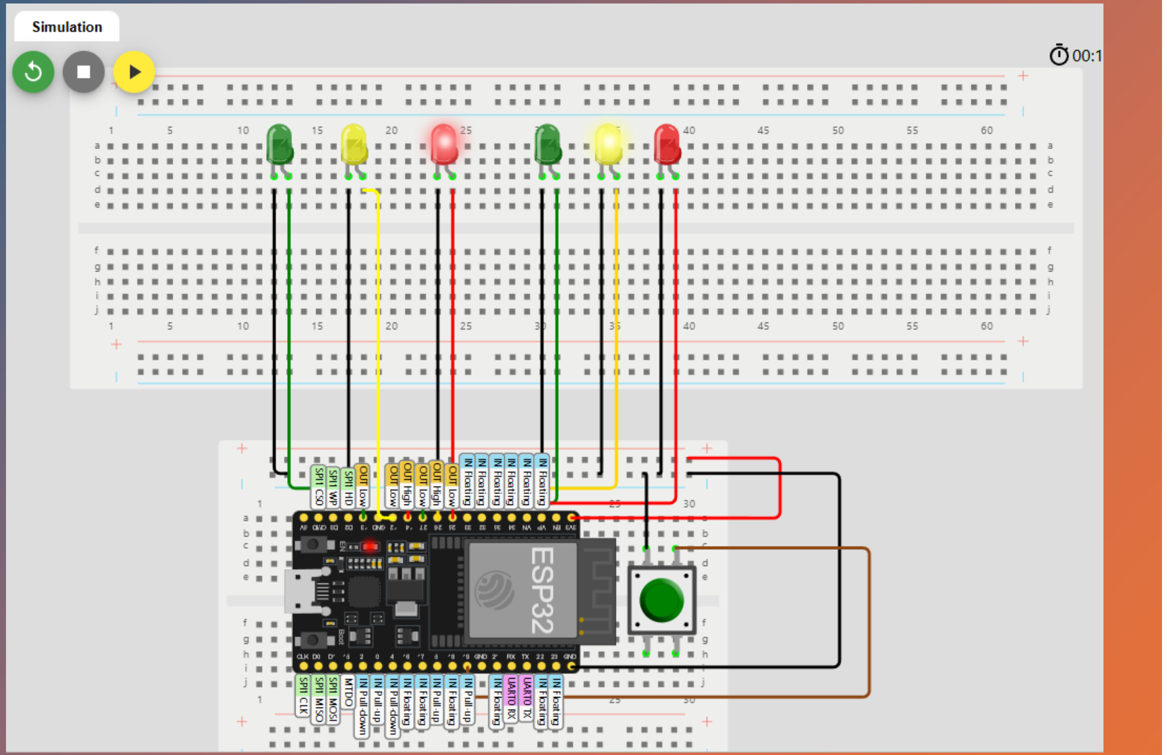
Screenshot of code Wokwi Code Editor showing **your name in the comment**



```

// === Anthony Norman ====
// Module #4 project

// Define some labels
const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14
const int yellow_LED1 =12; // The yellow LED1 is wired to ESP32 board pin GPIO12
const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13
const int red_LED2 = 25; // The red LED2 is wired to Mega board pin GPIO25
const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPIO 26
const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO 27
// the setup function runs once when you press reset or power the board
void setup() {
pinMode(red_LED1, OUTPUT); // initialize digital pin GPIO14 (Red LED1) as an output.
pinMode(yellow_LED1, OUTPUT); // initialize digital pin GPIO12 (yellow LED1) as an outpu
pinMode(green_LED1, OUTPUT); // initialize digital pin GPIO13 (green LED1) as an output.
pinMode(red_LED2, OUTPUT); // initialize digital pin GPIO25(Red LED2) as an output.
pinMode(yellow_LED2, OUTPUT); // initialize digital pin GPIO26 (yellow LED2) as an outpu
pinMode(green_LED2, OUTPUT); // initialize digital pin GPIO27 (green LED2) as an output.
}

// the loop function runs over and over again forever
void loop() {

// The next three lines of code turn on the red LED1
digitalWrite(red_LED1, HIGH); // This should turn on the RED LED1
digitalWrite(yellow_LED1 , LOW); // This should turn off the YELLOW LED1
digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1

delay(1000); //Extended time for Red light#1 before the Green of the other side turns ON

// The next three lines of code turn on the green LED2 for 2 seconds
digitalWrite(red_LED2, LOW); // This should turn off the RED LED2
digitalWrite(yellow_LED2 , LOW); // This should turn off the YELLOW LED2
digitalWrite(green_LED2, HIGH); // This should turn on the GREEN LED2

delay(2000); // wait for 2 seconds

// The next three lines of code turn on the red LED1
digitalWrite(red_LED1, HIGH); // This should turn on the RED LED1
digitalWrite(yellow_LED1 , LOW); // This should turn off the YELLOW LED1
digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1

// The next three lines of code turn on the yellow LED2
digitalWrite(red_LED2, LOW); // This should turn off the RED LED2
```

CEIS 114
Module 5

Creating a Multiple Traffic Light Controller
with a Cross Walk

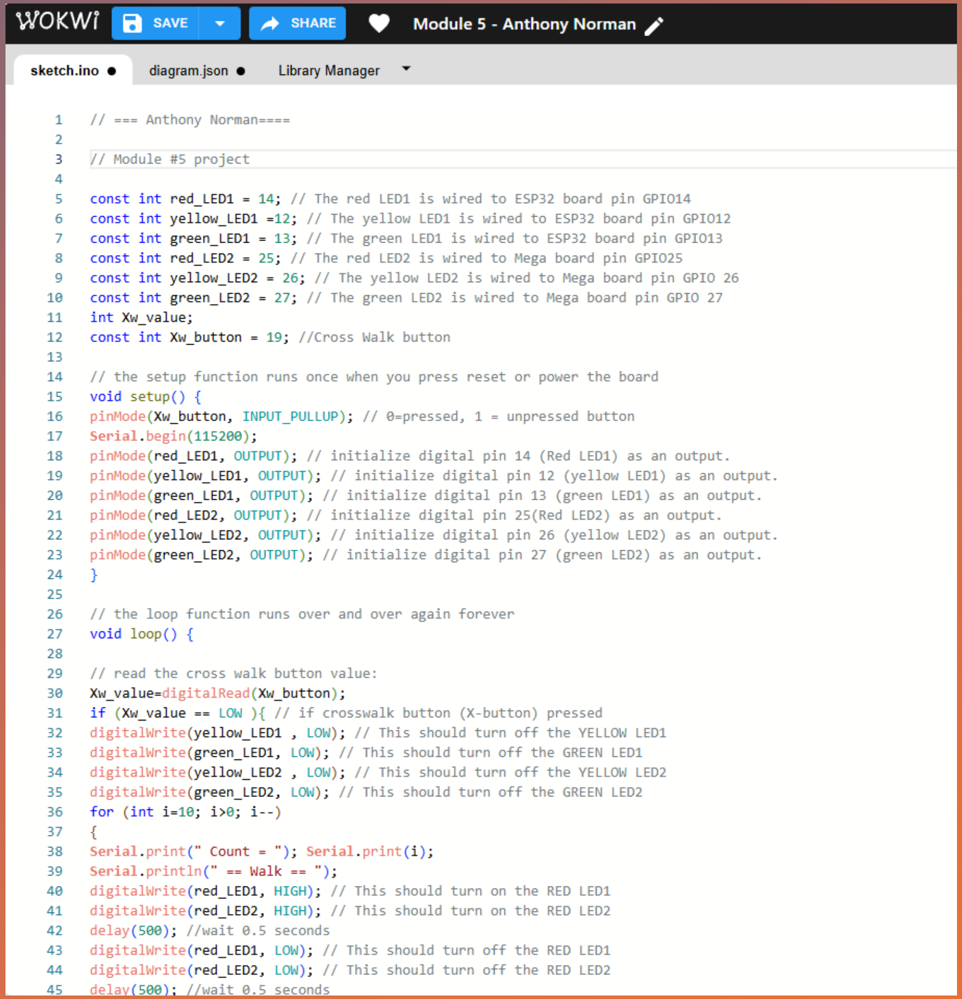


Screenshot of circuit with working LEDs

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

Screenshot of code in Wokwi

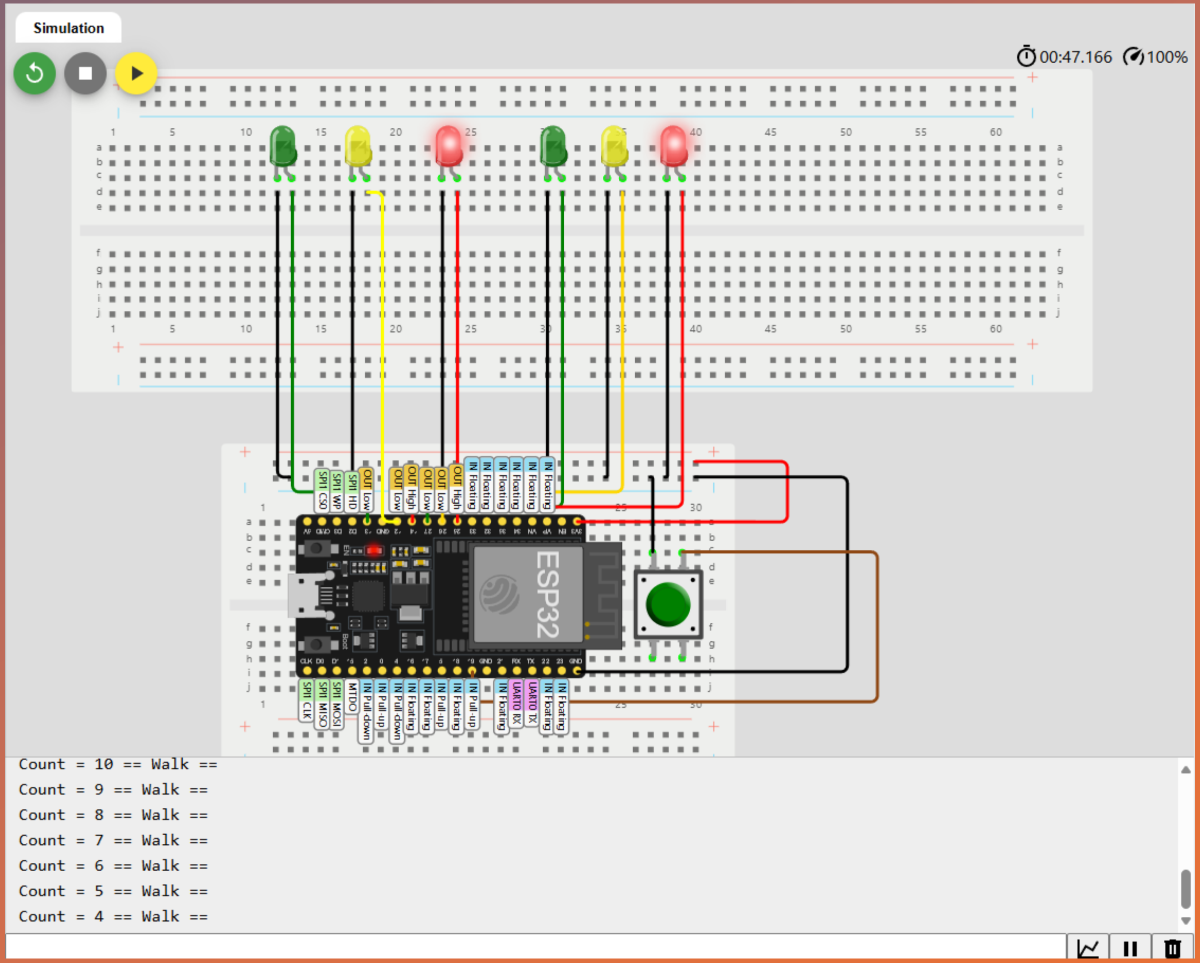
Screenshot of code in Wokwi Code Editor showing **your name in the comment**



```
// === Anthony Norman====

// Module #5 project

const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14
const int yellow_LED1 =12; // The yellow LED1 is wired to ESP32 board pin GPIO12
const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13
const int red_LED2 = 25; // The red LED2 is wired to Mega board pin GPIO25
const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPIO 26
const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO 27
int Xw_value;
const int Xw_button = 19; //Cross Walk button

// the setup function runs once when you press reset or power the board
void setup() {
pinMode(Xw_button, INPUT_PULLUP); // 0=pressed, 1 = unpressed button
Serial.begin(115200);
pinMode(red_LED1, OUTPUT); // initialize digital pin 14 (Red LED1) as an output.
pinMode(yellow_LED1, OUTPUT); // initialize digital pin 12 (yellow LED1) as an output.
pinMode(green_LED1, OUTPUT); // initialize digital pin 13 (green LED1) as an output.
pinMode(red_LED2, OUTPUT); // initialize digital pin 25(Red LED2) as an output.
pinMode(yellow_LED2, OUTPUT); // initialize digital pin 26 (yellow LED2) as an output.
pinMode(green_LED2, OUTPUT); // initialize digital pin 27 (green LED2) as an output.
}

// the loop function runs over and over again forever
void loop() {

// read the cross walk button value:
Xw_value=digitalRead(Xw_button);
if (Xw_value == LOW ){ // if crosswalk button (X-button) pressed
digitalWrite(yellow_LED1 , LOW); // This should turn off the YELLOW LED1
digitalWrite(green_LED1, LOW); // This should turn off the GREEN LED1
digitalWrite(yellow_LED2 , LOW); // This should turn off the YELLOW LED2
digitalWrite(green_LED2, LOW); // This should turn off the GREEN LED2
for (int i=10; i>0; i--)
{
Serial.print(" Count = "); Serial.print(i);
Serial.println(" == Walk == ");
digitalWrite(red_LED1, HIGH); // This should turn on the RED LED1
digitalWrite(red_LED2, HIGH); // This should turn on the RED LED2
delay(500); //wait 0.5 seconds
digitalWrite(red_LED1, LOW); // This should turn off the RED LED1
digitalWrite(red_LED2, LOW); // This should turn off the RED LED2
delay(500); //wait 0.5 seconds
```

Screenshot of Serial Monitor in Wokwi

Screenshot of output in Serial Monitor

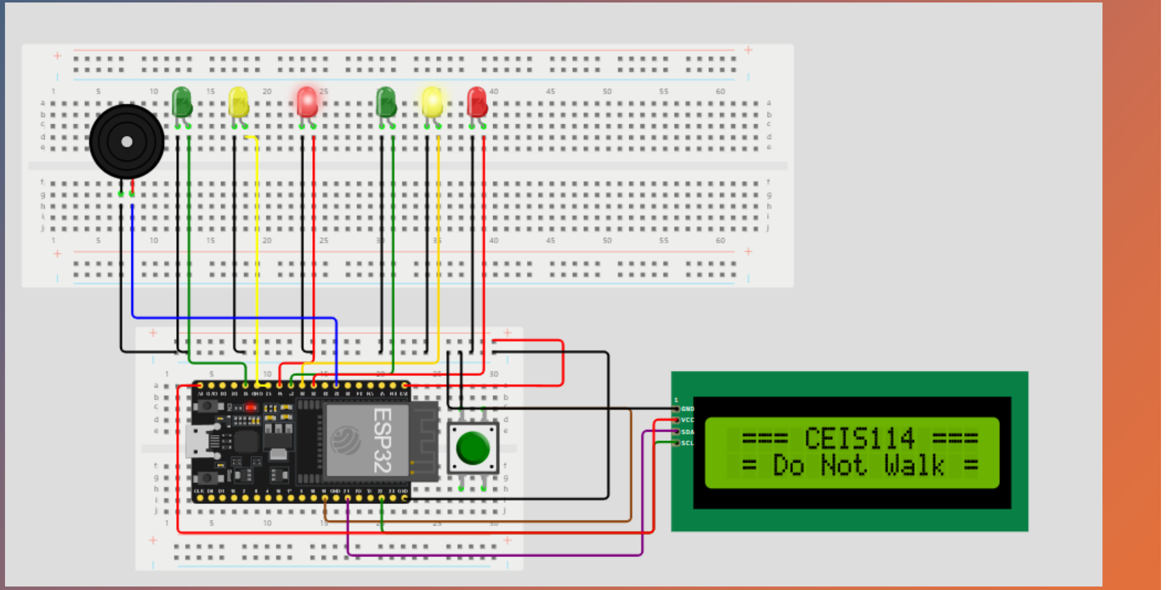


```
Count = 10 == Walk ==
Count = 9 == Walk ==
Count = 8 == Walk ==
Count = 7 == Walk ==
Count = 6 == Walk ==
Count = 5 == Walk ==
Count = 4 == Walk ==
```

CEIS 114
Module 6

Creating a Multiple Traffic Light Controller
with a Cross Walk and an Emergency Buzzer

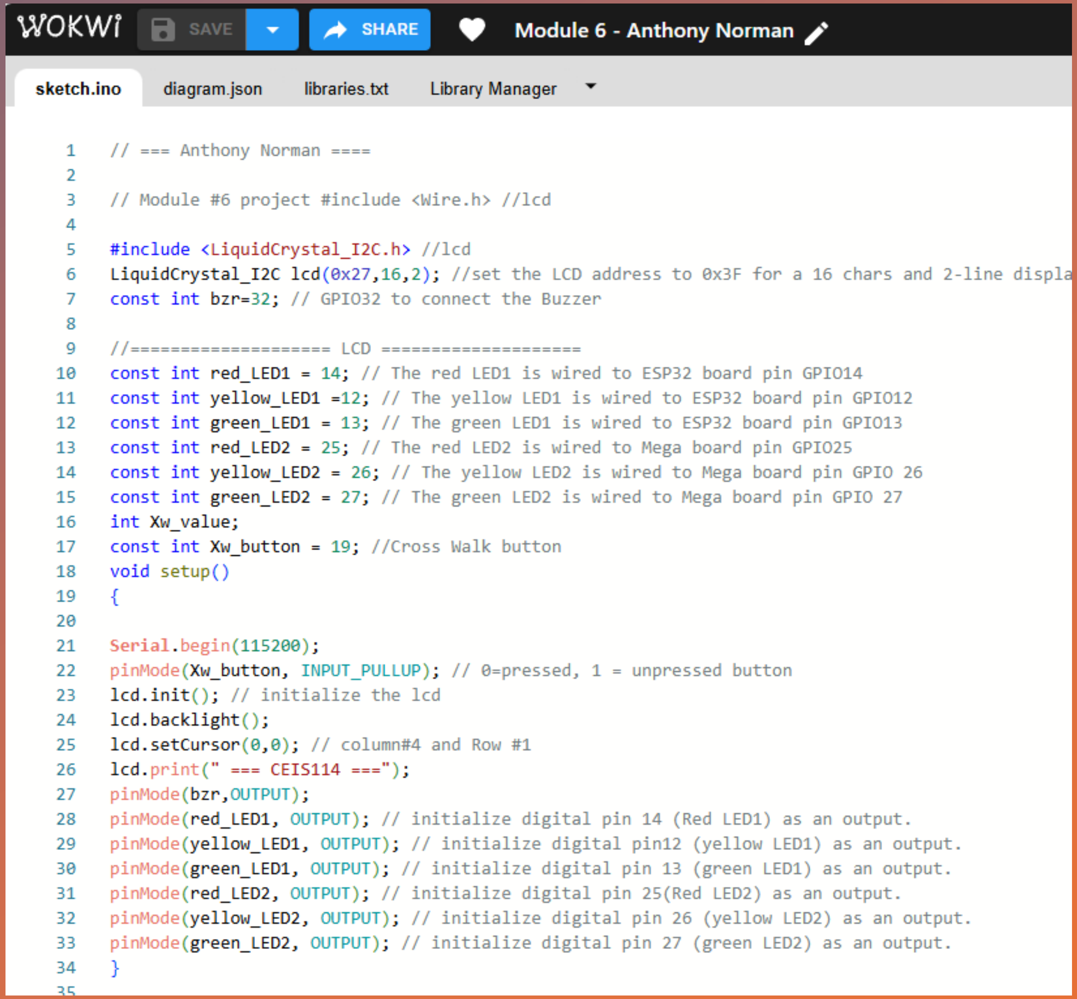
Picture of circuit with working LEDs and LCD display



- 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

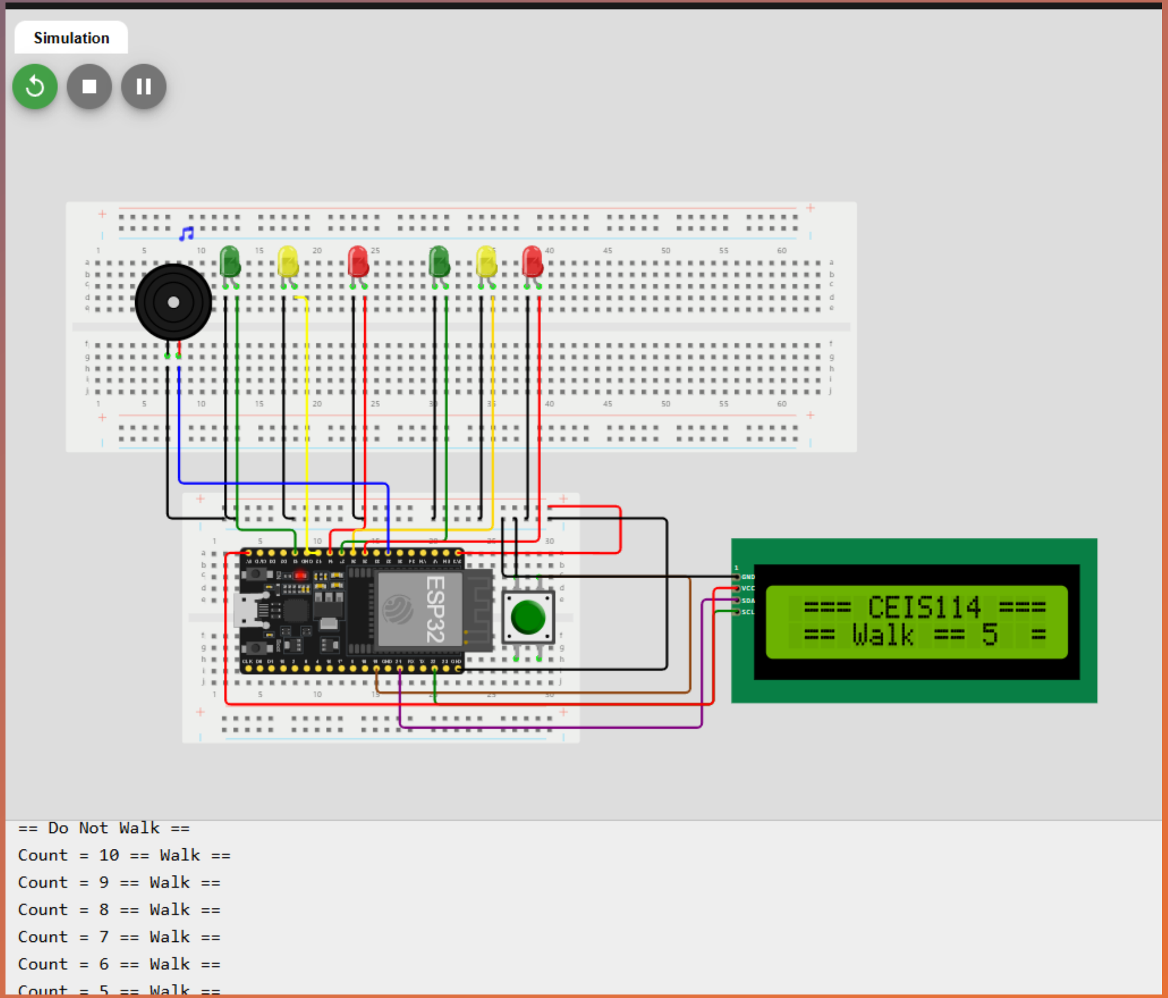
Screenshot of code in Code Editor showing **your name in the comment**



```
// === Anthony Norman ====

// Module #6 project #include <Wire.h> //lcd

#include <LiquidCrystal_I2C.h> //lcd
LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD address to 0x3F for a 16 chars and 2-line displa
const int bzr=32; // GPIO32 to connect the Buzzer

//==================== LCD ====================
const int red_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPIO14
const int yellow_LED1 =12; // The yellow LED1 is wired to ESP32 board pin GPIO12
const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPIO13
const int red_LED2 = 25; // The red LED2 is wired to Mega board pin GPIO25
const int yellow_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPIO 26
const int green_LED2 = 27; // The green LED2 is wired to Mega board pin GPIO 27
int Xw_value;
const int Xw_button = 19; //Cross Walk button
void setup()
{

Serial.begin(115200);
pinMode(Xw_button, INPUT_PULLUP); // 0=pressed, 1 = unpressed button
lcd.init(); // initialize the lcd
lcd.backlight();
lcd.setCursor(0,0); // column#4 and Row #1
lcd.print(" === CEIS114 ===");
pinMode(bzr,OUTPUT);
pinMode(red_LED1, OUTPUT); // initialize digital pin 14 (Red LED1) as an output.
pinMode(yellow_LED1, OUTPUT); // initialize digital pin12 (yellow LED1) as an output.
pinMode(green_LED1, OUTPUT); // initialize digital pin 13 (green LED1) as an output.
pinMode(red_LED2, OUTPUT); // initialize digital pin 25(Red LED2) as an output.
pinMode(yellow_LED2, OUTPUT); // initialize digital pin 26 (yellow LED2) as an output.
pinMode(green_LED2, OUTPUT); // initialize digital pin 27 (green LED2) as an output.
}
```

Screenshot of Serial Monitor

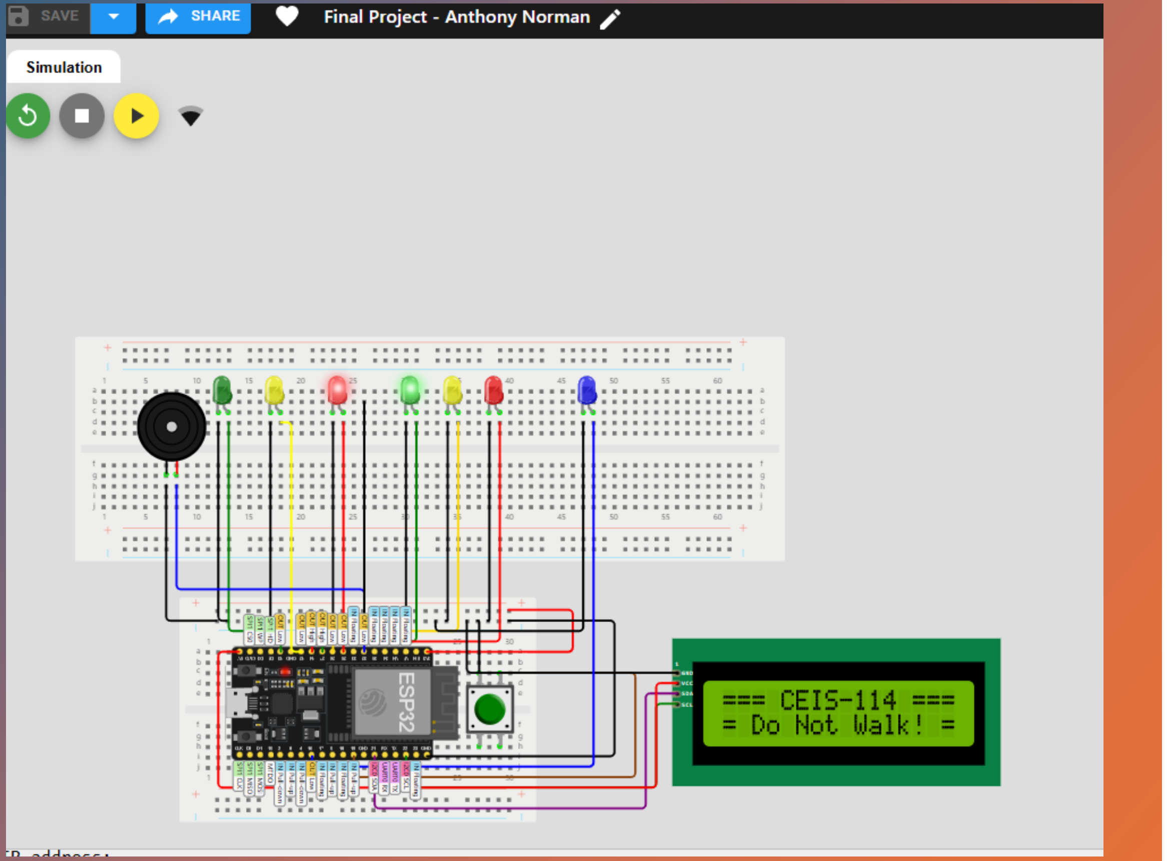
Screenshot of output in Serial Monitor



```
== Do Not Walk ==
Count = 10 == Walk ==
Count = 9 == Walk ==
Count = 8 == Walk ==
Count = 7 == Walk ==
Count = 6 == Walk ==
Count = 5 == Walk ==
```

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



Final Project - Anthony Norman

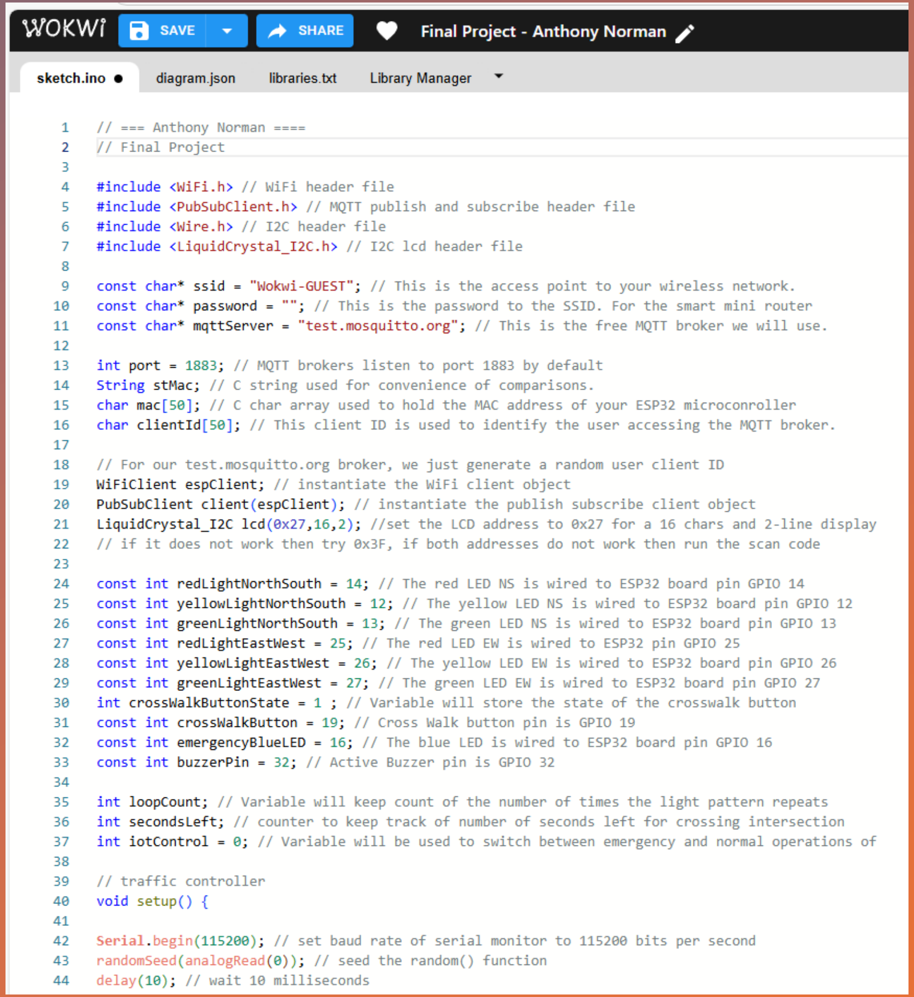
```
=== CEIS-114 ===
= Do Not Walk! =
```

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)

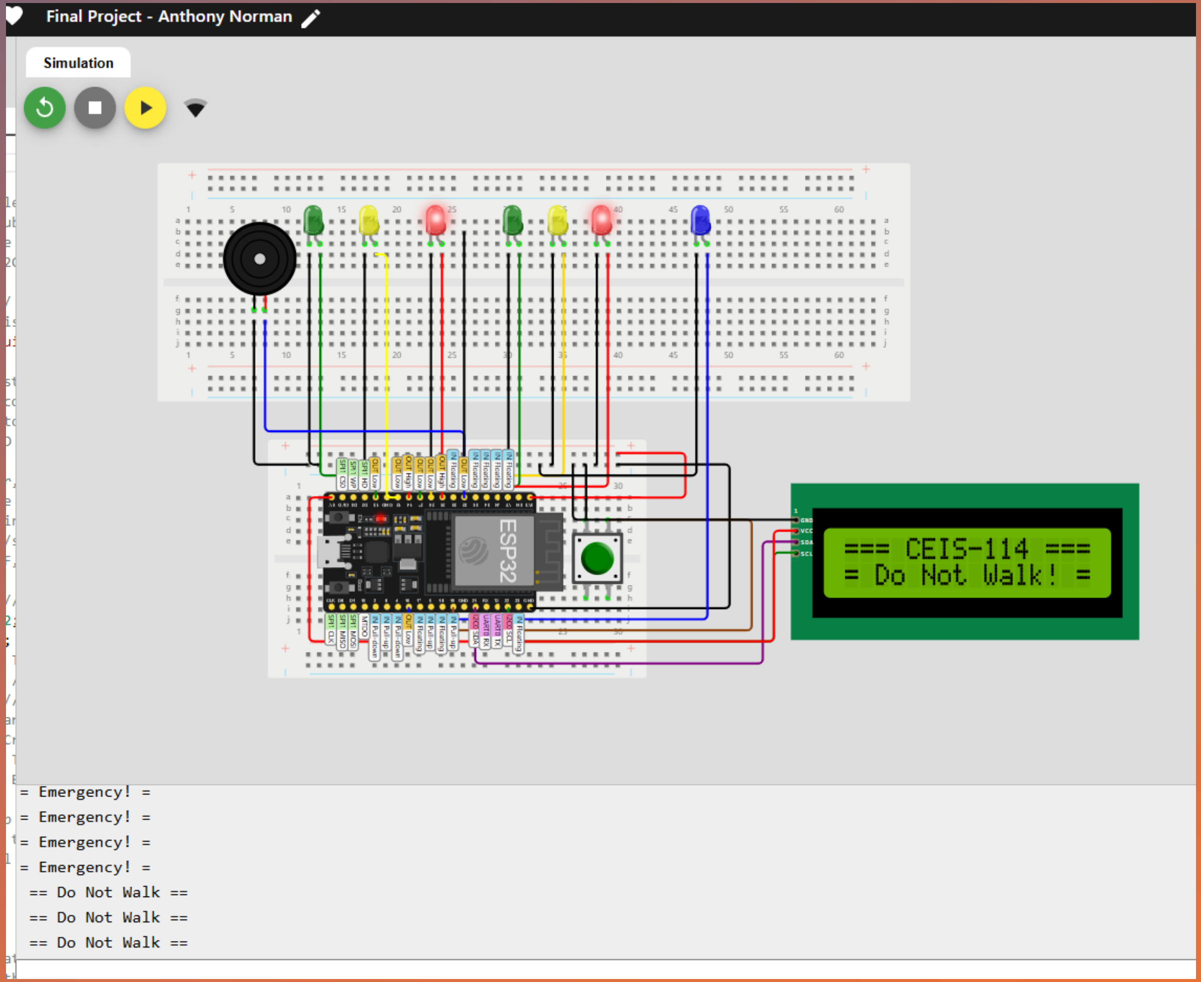
Screenshot of code in Code Editor showing **your name in the comment**



```
// === Anthony Norman ====
// Final Project

#include <WiFi.h> // WiFi header file
#include <PubSubClient.h> // MQTT publish and subscribe header file
#include <Wire.h> // I2C header file
#include <LiquidCrystal_I2C.h> // I2C lcd header file

const char* ssid = "Wokwi-GUEST"; // This is the access point to your wireless network.
const char* password = ""; // This is the password to the SSID. For the smart mini router
const char* mqttServer = "test.mosquitto.org"; // This is the free MQTT broker we will use.

int port = 1883; // MQTT brokers listen to port 1883 by default
String stMac; // C string used for convenience of comparisons.
char mac[50]; // C char array used to hold the MAC address of your ESP32 microconroller
char clientId[50]; // This client ID is used to identify the user accessing the MQTT broker.

// For our test.mosquitto.org broker, we just generate a random user client ID
WiFiClient espClient; // instantiate the WiFi client object
PubSubClient client(espClient); // instantiate the publish subscribe client object
LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD address to 0x27 for a 16 chars and 2-line display
// if it does not work then try 0x3F, if both addresses do not work then run the scan code

const int redLightNorthSouth = 14; // The red LED NS is wired to ESP32 board pin GPIO 14
const int yellowLightNorthSouth = 12; // The yellow LED NS is wired to ESP32 board pin GPIO 12
const int greenLightNorthSouth = 13; // The green LED NS is wired to ESP32 board pin GPIO 13
const int redLightEastWest = 25; // The red LED EW is wired to ESP32 pin GPIO 25
const int yellowLightEastWest = 26; // The yellow LED EW is wired to ESP32 board pin GPIO 26
const int greenLightEastWest = 27; // The green LED EW is wired to ESP32 board pin GPIO 27
int crossWalkButtonState = 1 ; // Variable will store the state of the crosswalk button
const int crossWalkButton = 19; // Cross Walk button pin is GPIO 19
const int emergencyBlueLED = 16; // The blue LED is wired to ESP32 board pin GPIO 16
const int buzzerPin = 32; // Active Buzzer pin is GPIO 32

int loopCount; // Variable will keep count of the number of times the light pattern repeats
int secondsLeft; // counter to keep track of number of seconds left for crossing intersection
int iotControl = 0; // Variable will be used to switch between emergency and normal operations of

// traffic controller
void setup() {

Serial.begin(115200); // set baud rate of serial monitor to 115200 bits per second
randomSeed(analogRead(0)); // seed the random() function
delay(10); // wait 10 milliseconds
```

Screenshot **of Serial Monitor** (Testing)

Screenshot of output in Serial Monitor



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

Challenges

- Some of the challenges I had to face during this was attention to detail.
- What I mean by this is whenever you are adding the ground wires and power wires from the LEDs, you really had to make sure that the wires were connected properly, or they wouldn't work.
- I found myself checking this a few different times.

Career Skills

- Some of the career skills I gained was coding for one.
- Implementing the code and then going back and figuring out why the code was not working was fun and interesting.
- Also, some skills in electrical engineering
- Connecting power and ground wires and understanding how they work was fun to me.

Conclusion

- In conclusion I really enjoyed this project
- Building alarm systems requires some work but it’s not as hard as I thought it would be.
- Not to say it’s not challenging because it is, but once you get to doing it more it becomes a little less stressful.