CEIS114 Course Project
Traffic Controller

Developed by James Garlie

DeVry University: April 2022

Introduction

This presentation shows the creation of a Multiple Traffic Light Controller with a Cross Walk and an Emergency Buzzer with Secured IoT Control via Web.

It begins with a project plan and taking inventory of the parts needed followed by mounting the ESP32 Microcontroller on a breadboard and powered ON. Then the Arduino IDE is installed and an ESP32 WiFi Scan is generated. This creates the Traffic Controller.

The presentation moves on to show Creating a Multiple Traffic Light Controller, adding a Cross Walk, an Emergency Buzzer, and a Secured Iot Control via Web.

Project Plan for IoT Traffic Controller

The next three slides show the inventory of parts, the ESP32 Microcontroller mounted on a breadboard and powered ON, the installation of Arduino IDE, and ESP32 WiFi Scan

Inventory

ESP 32 Board

Colored LEDs: Red, Yellow, Green, and Blue

220 Ohm Resistors (optional)

Wires

Breadboard(s)

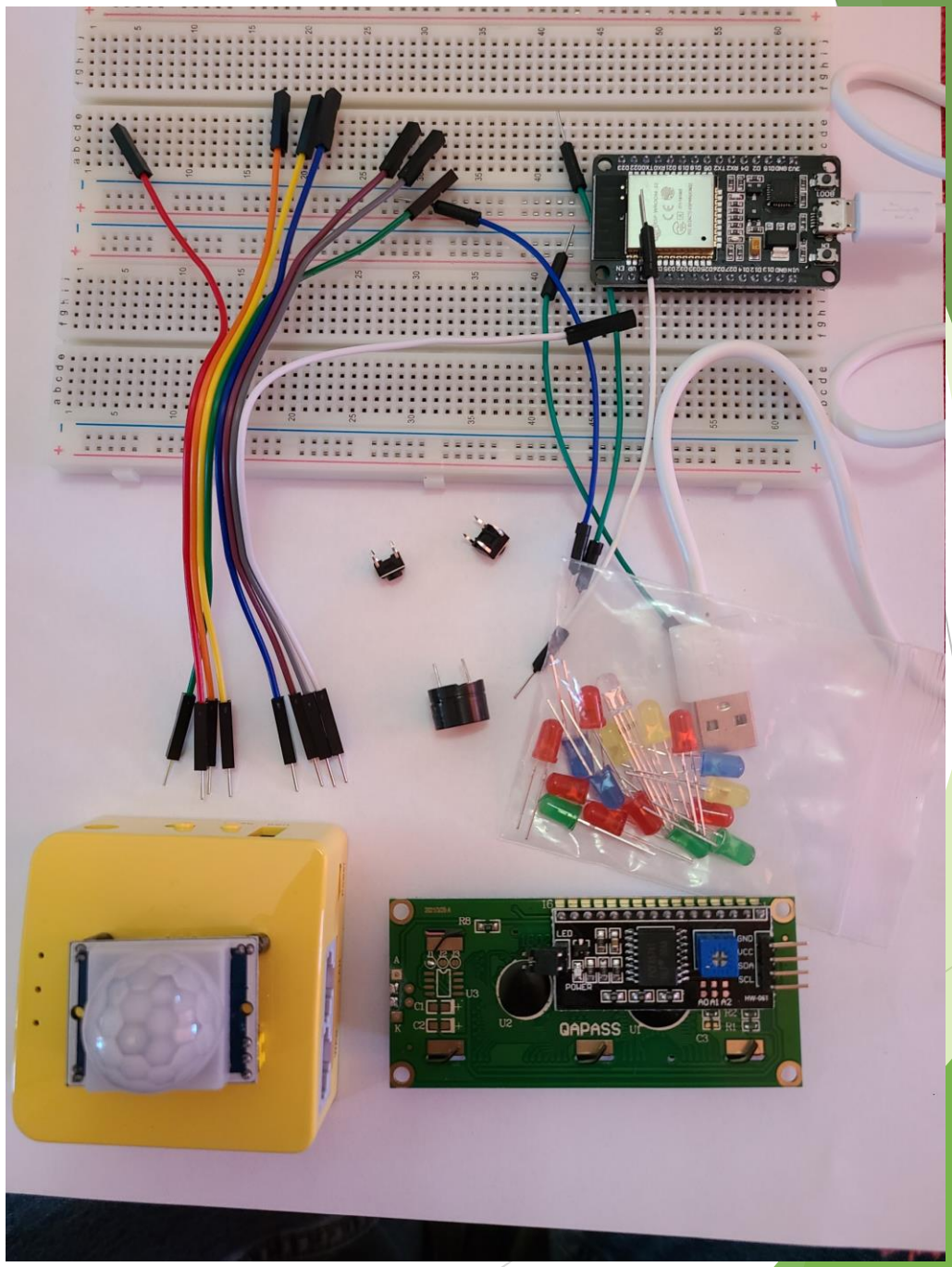
LCD Unit with I2C Adapter

Active Buzzer

Mini Router

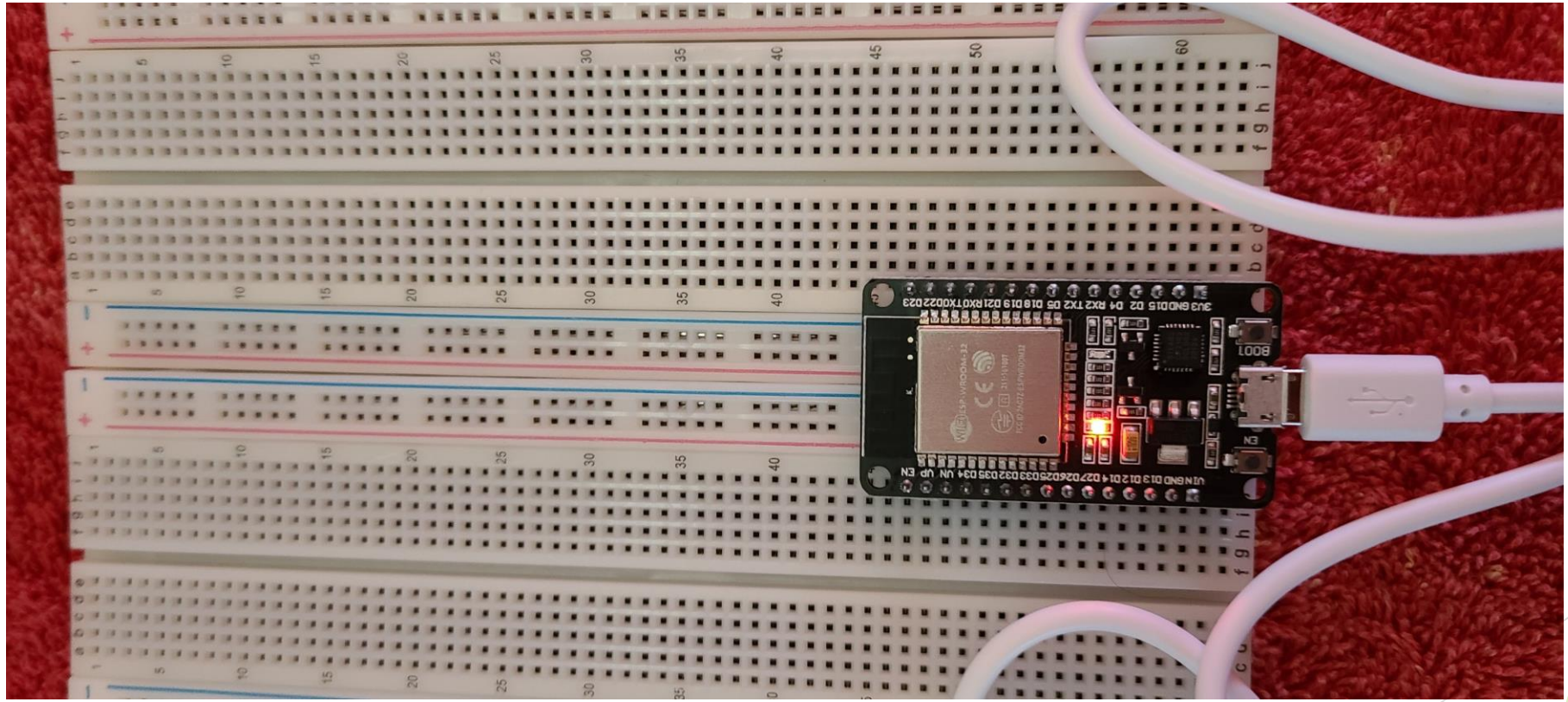
Push Button(s)

PIR Motion Sensor



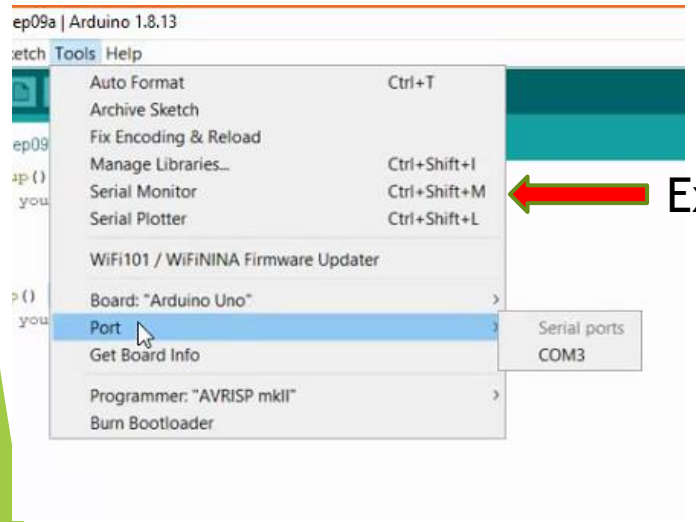
ESP32 Microcontroller

Microcontroller mounted and powered ON

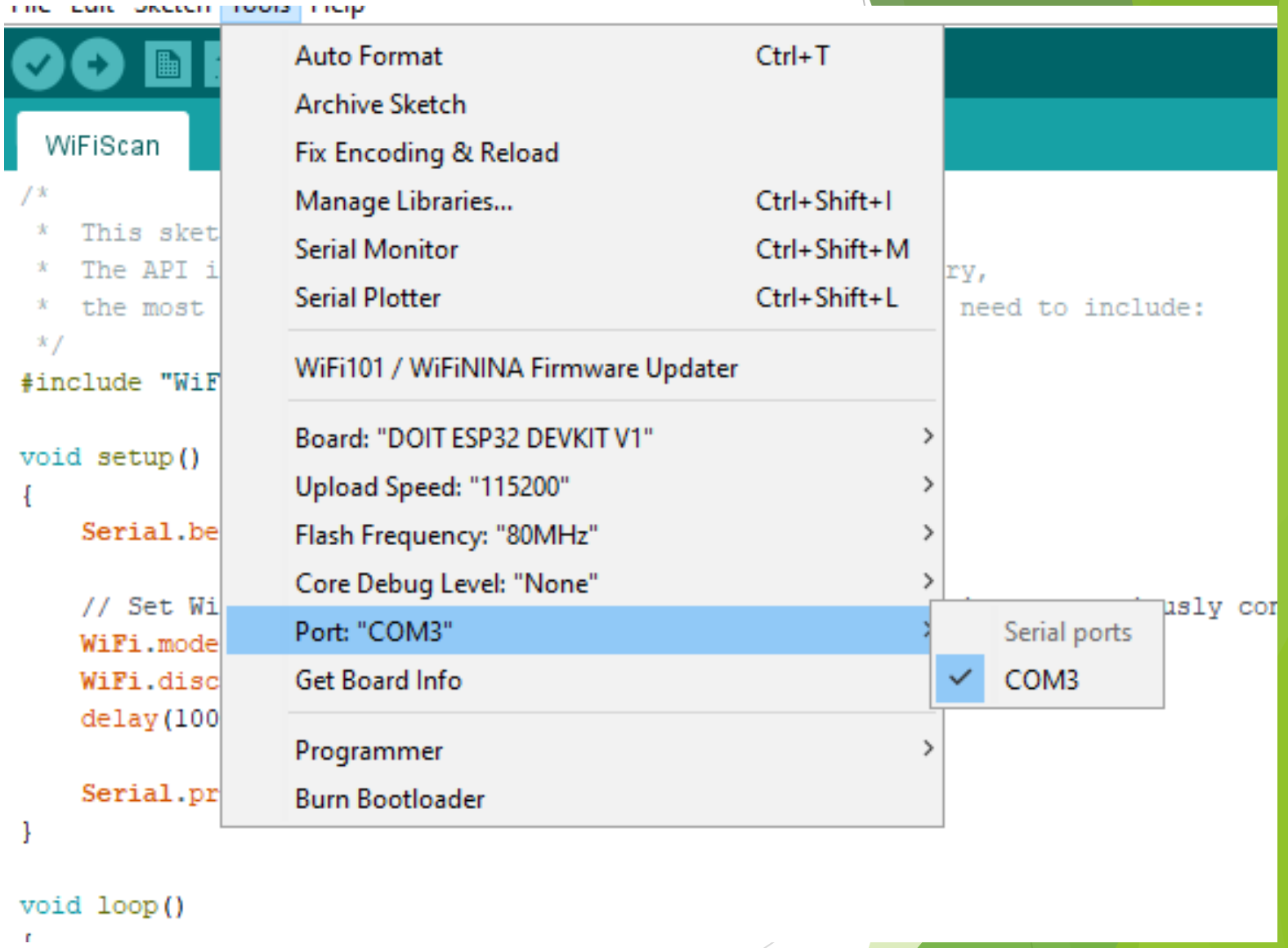


Installation of Arduino IDE

Screenshot of Arduino IDE with **Port** selected from Tools menu.

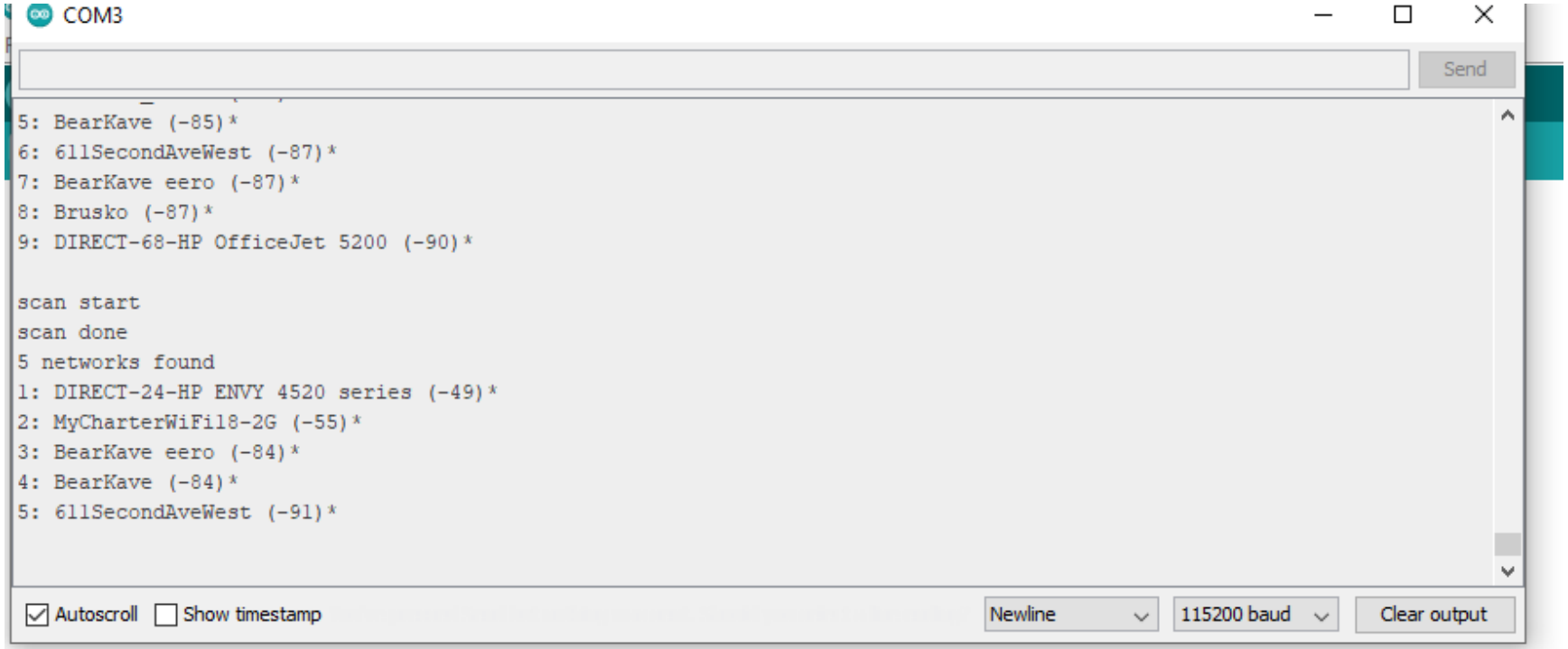


Example



ESP32 WiFi Scan

Screenshot of **Serial Monitor** in Arduino IDE showing the available networks



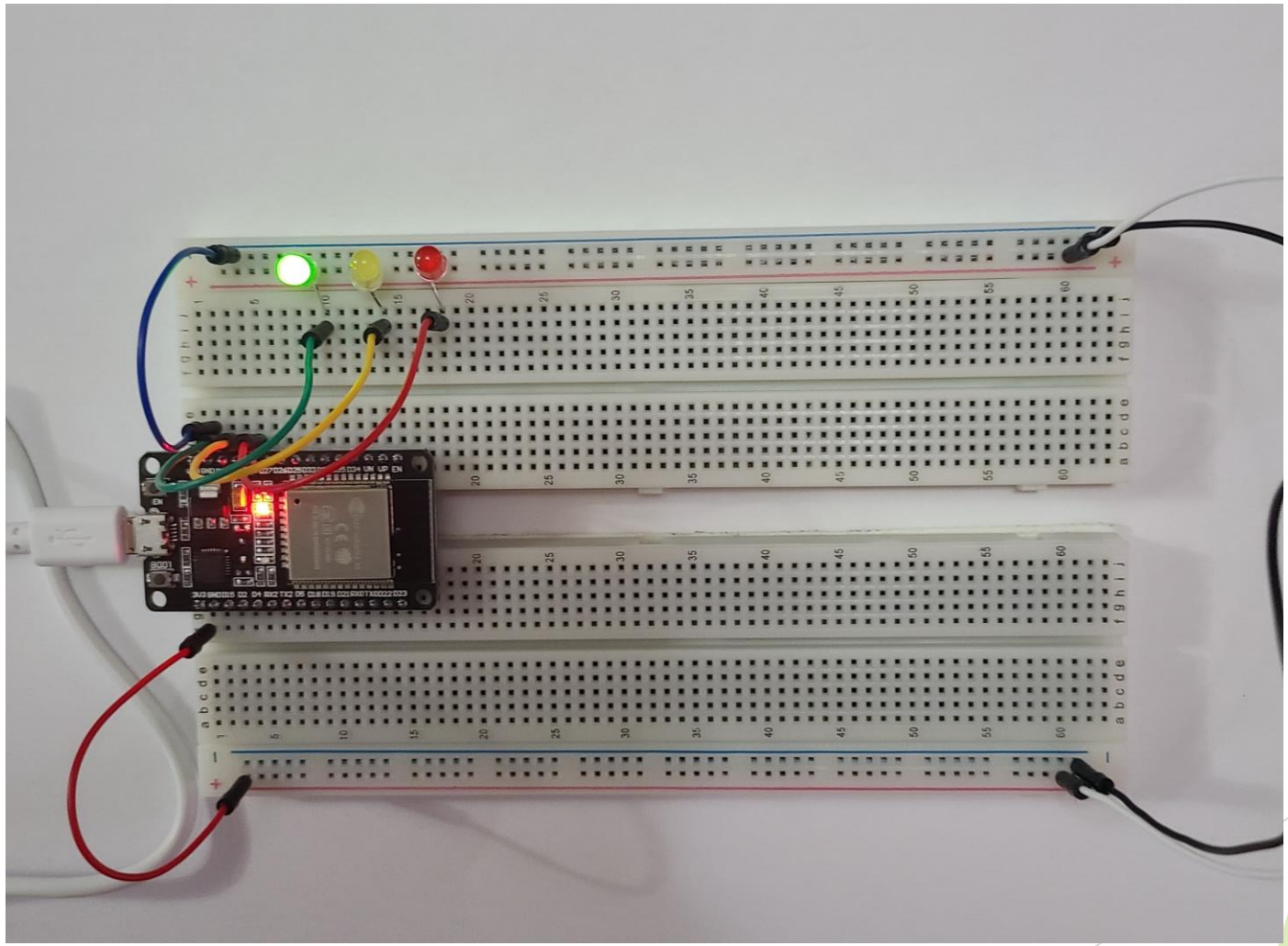

```
    Serial.println("Setup done");
}

void loop()
{
    Serial.println("scan start");

    // WiFi.scanNetworks will return the number of networks found
    int n = WiFi.scanNetworks();
    Serial.println("scan done");
```

Creating the Traffic Controller

The next two slide show creating a Traffic Controller

Picture of circuit with working LEDs

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



Screenshot of Code in Arduino IDE

sketch_mar13a §

```
// === James Garlie ====
// 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
```

Creating a Multiple Traffic Light Controller

The next two slide show creating a Multiple Traffic Light Controller

Picture of circuit with working LEDs

ESP 32 Board

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

220 Ohm Resistors (optional)

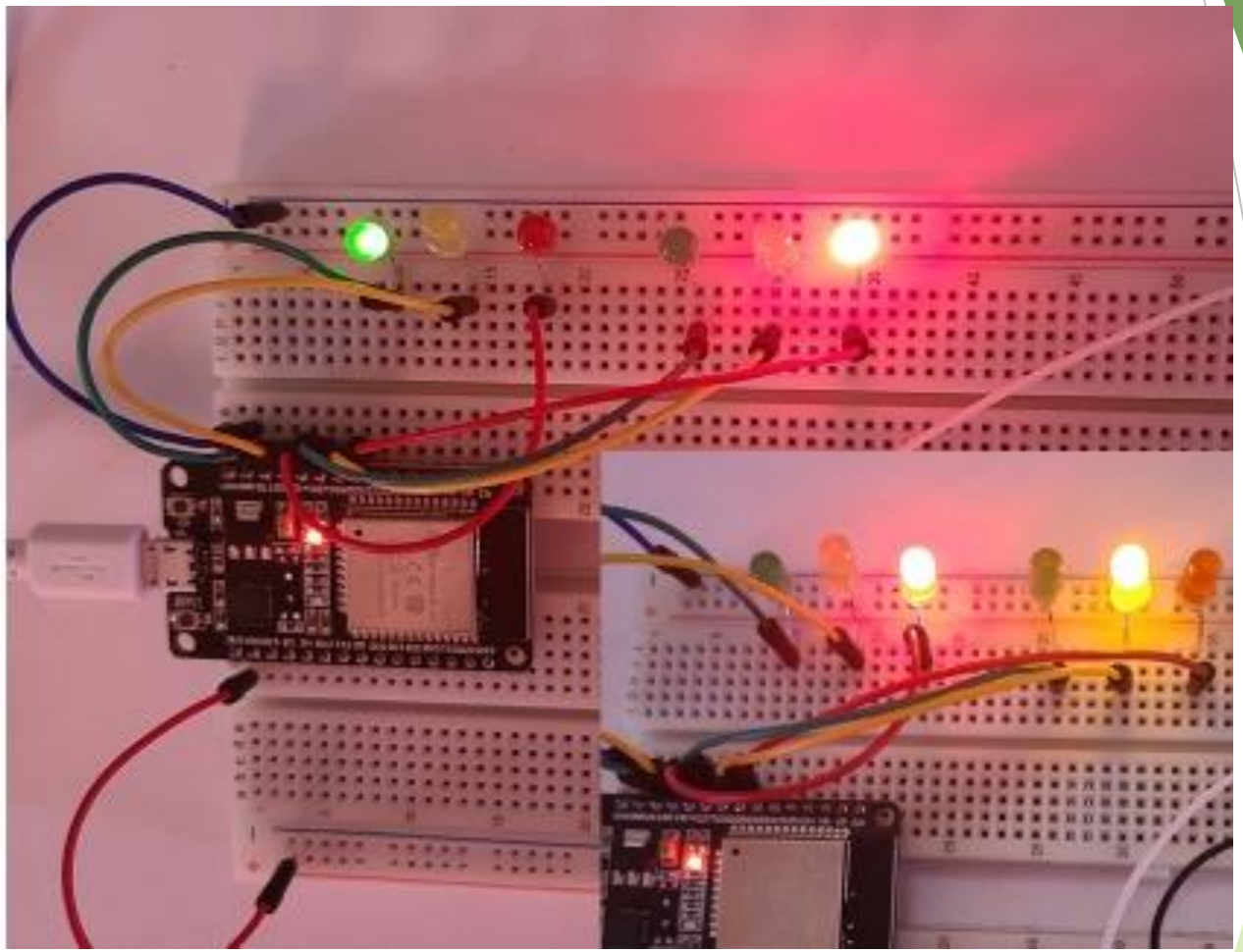
Wires

Breadboard

Note:

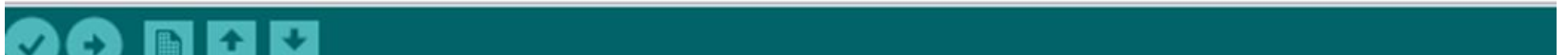
Top is Green and Red

Bottom is Red and Yellow



Screenshot of Code in Arduino IDE

File Edit Sketch Tools Help



sketch_mar13a

```
// === James Garlie ====
// 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 output.
 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 output.
 pinMode(green_LED2, OUTPUT); // initialize digital pin GPIO27 (green LED2) as an output.
}
```

Creating a Multiple Traffic Light Controller with a Cross Walk

The next three slides show adding a Cross Walk

Picture of circuit with working LEDs

ESP 32 Board

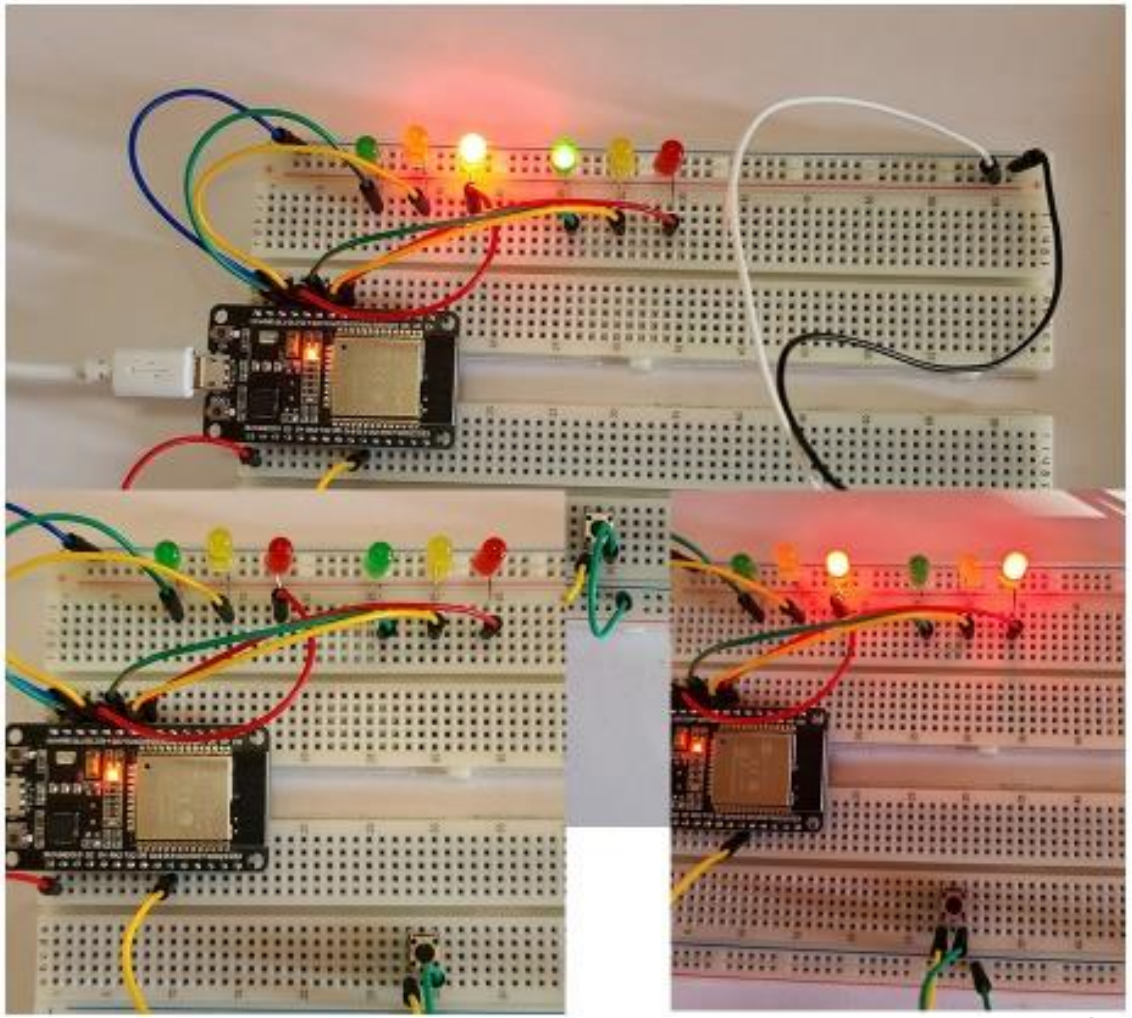
Colored LEDs: Red, Yellow and Green (two sets)

220 Ohm Resistors (optional)

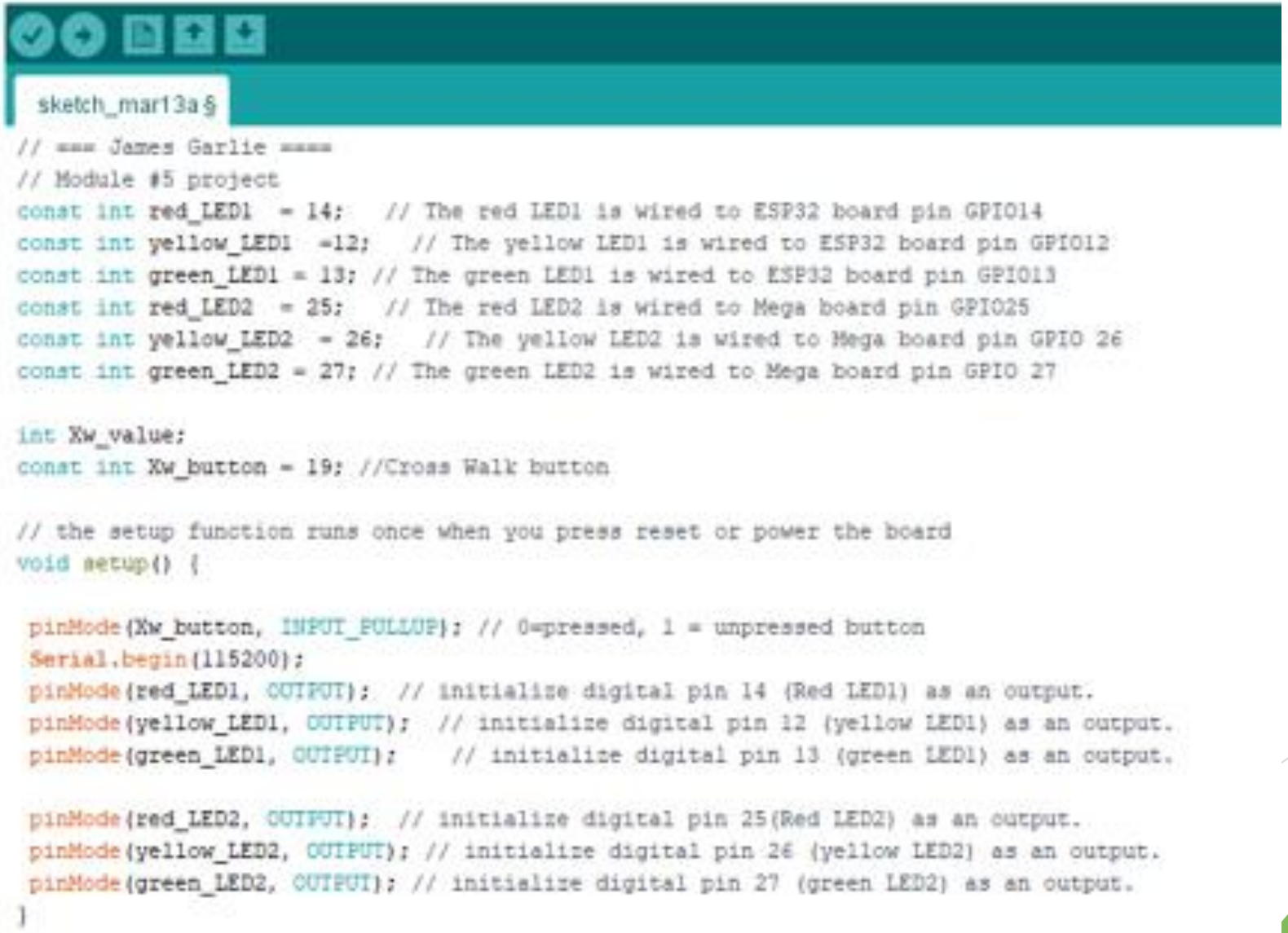
Push Button

Wires

Breadboard



Screenshot of Code in Arduino IDE

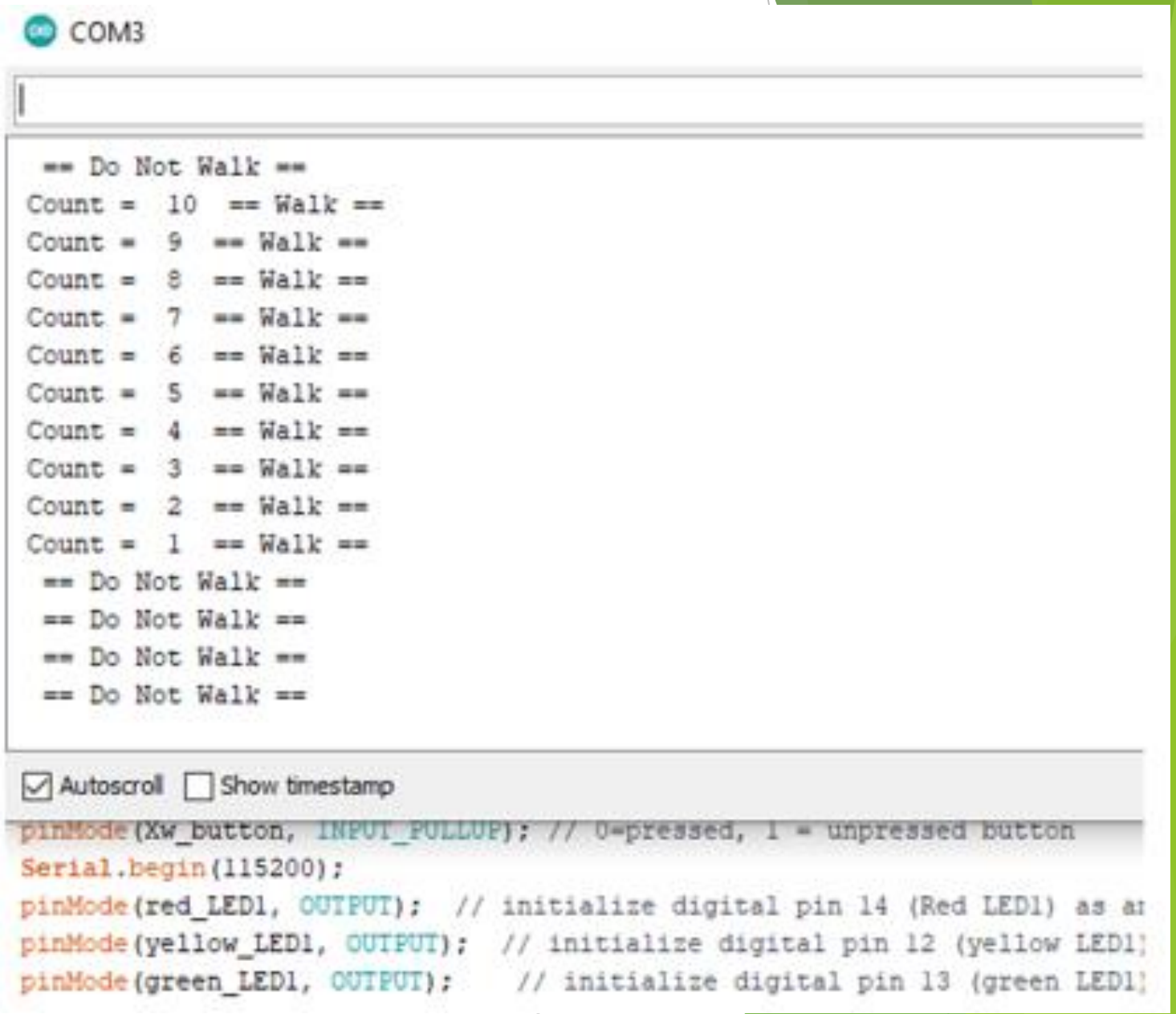


sketch_mar13a §

```
// === James Garlie ====
// 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.
}
```

Screenshot of Serial Monitor in Arduino IDE

Screenshot of output in Serial Monitor



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

The next three slides show adding an Emergency Buzzer

Pictures of Circuits with Working LEDs and LCD Display

ESP 32 Board

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

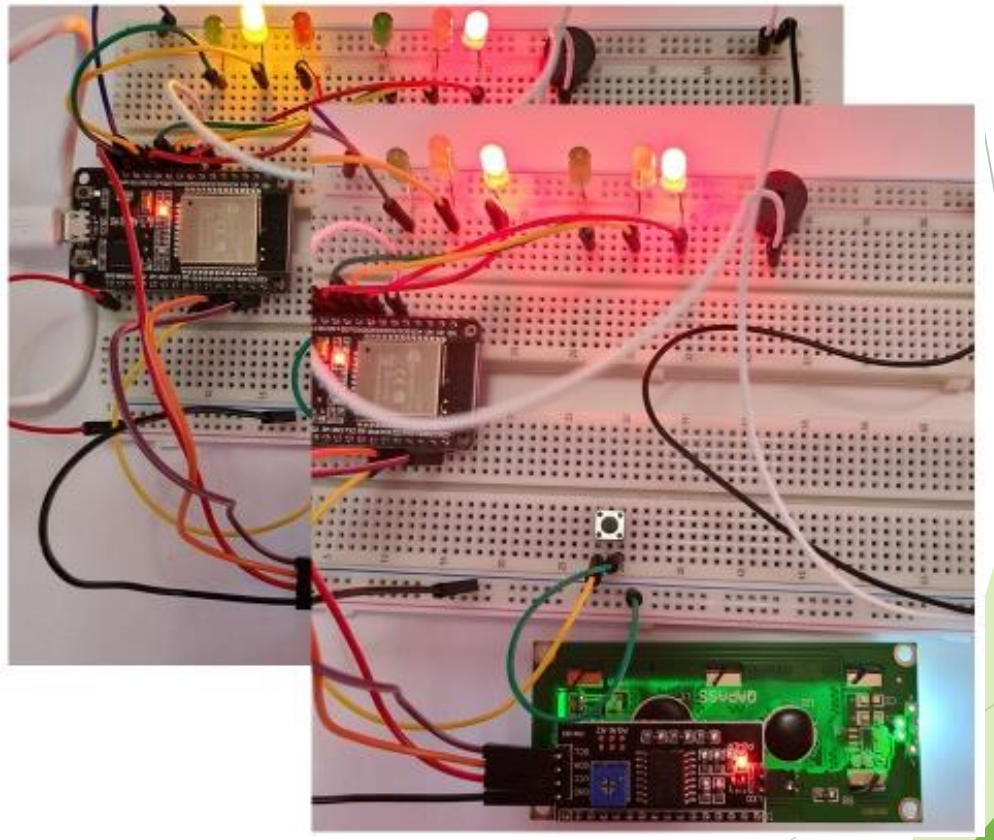
220 Ohm Resistors (optional)

Push Button

LCD Unit with Message Display

Wires

Breadboard

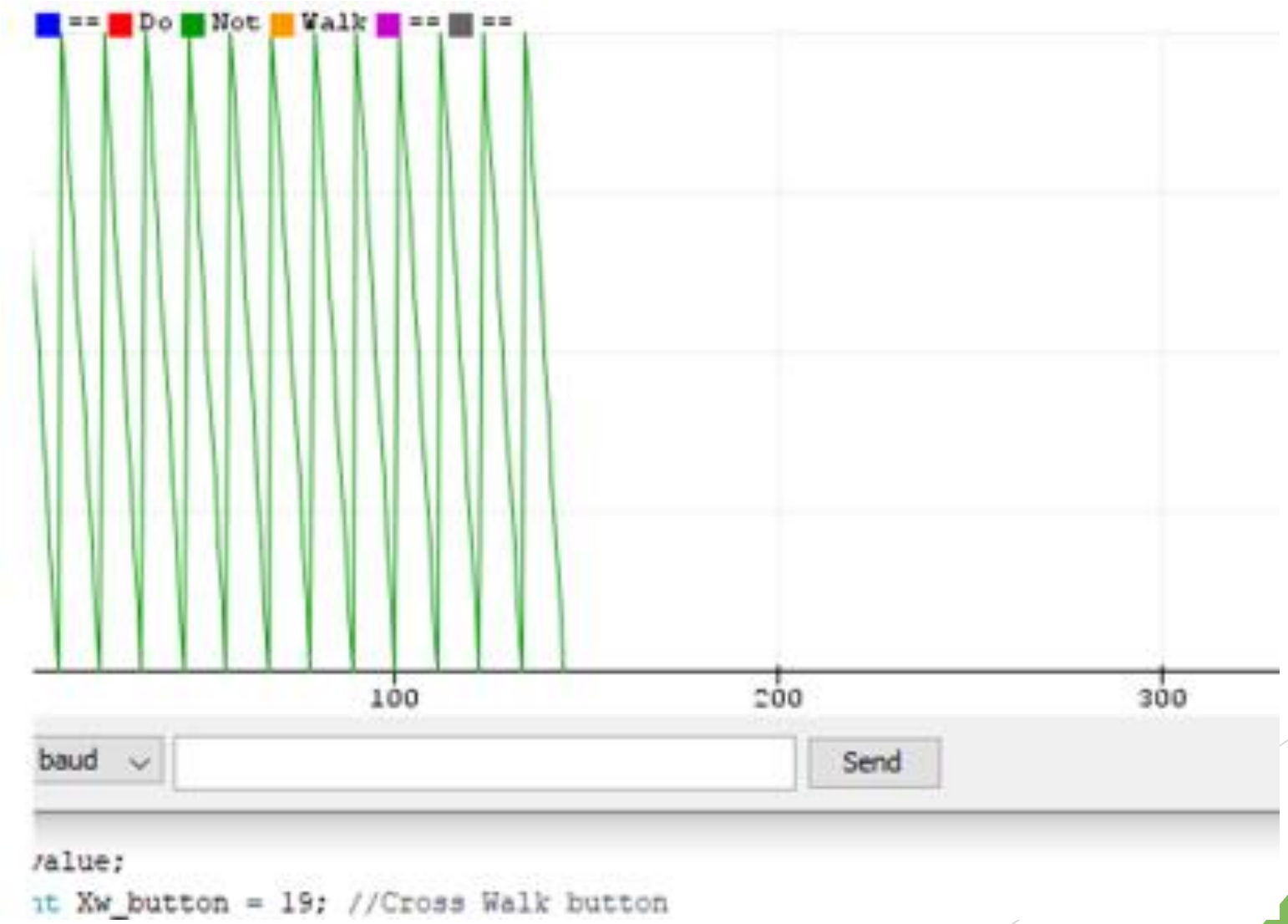


Screenshot of Code in Arduino IDE

sketch_mar13a §

```
// === James Garlie ====
// 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 display
// if it does not work then try 0x3F, if both addresses do not work then run the scan code below

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);
```

Screenshot of the Output in Serial Monitor in Arduino IDE



Creating a Multiple Traffic Light Controller with a Cross Walk and an Emergency Buzzer with secured IoT Control via Web

The next three slides show adding a Secured IoT Control via Web

Picture of circuit with working LEDs and LCD display

ESP 32 Board

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

One Blue LED – Emergency Light

220 Ohm Resistors (optional)

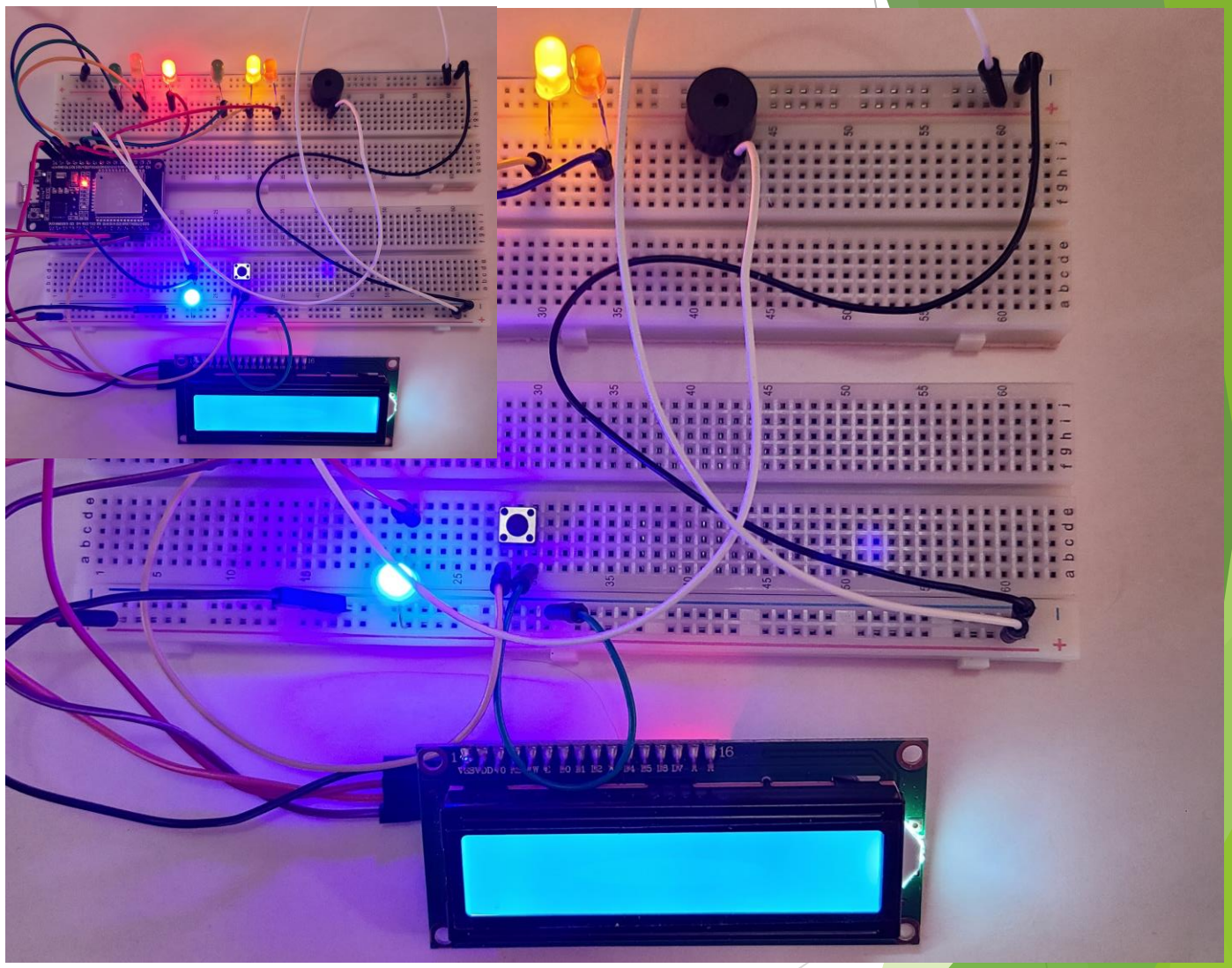
Push Button

LCD Unit

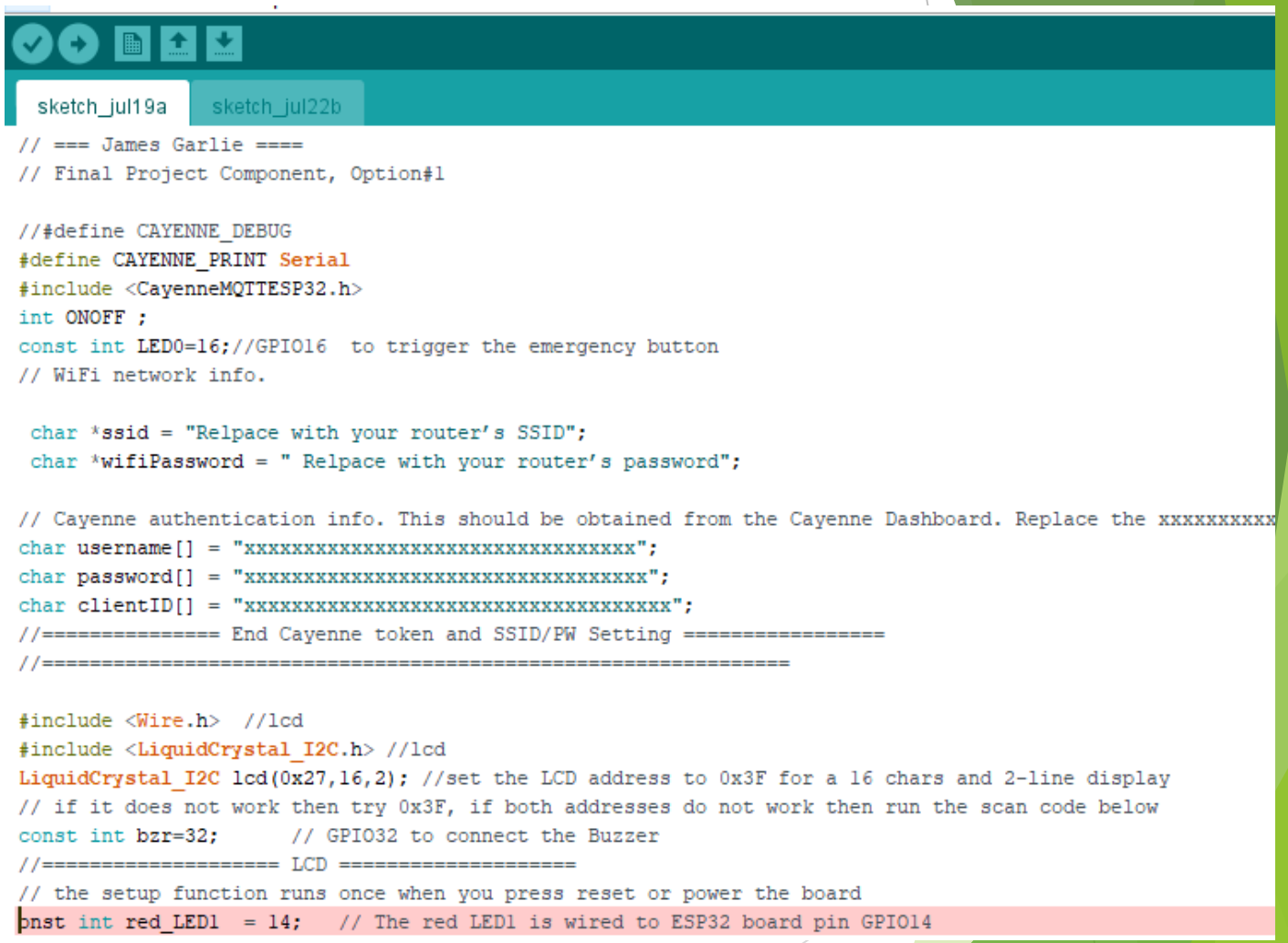
Buzzer

Wires

Breadboard

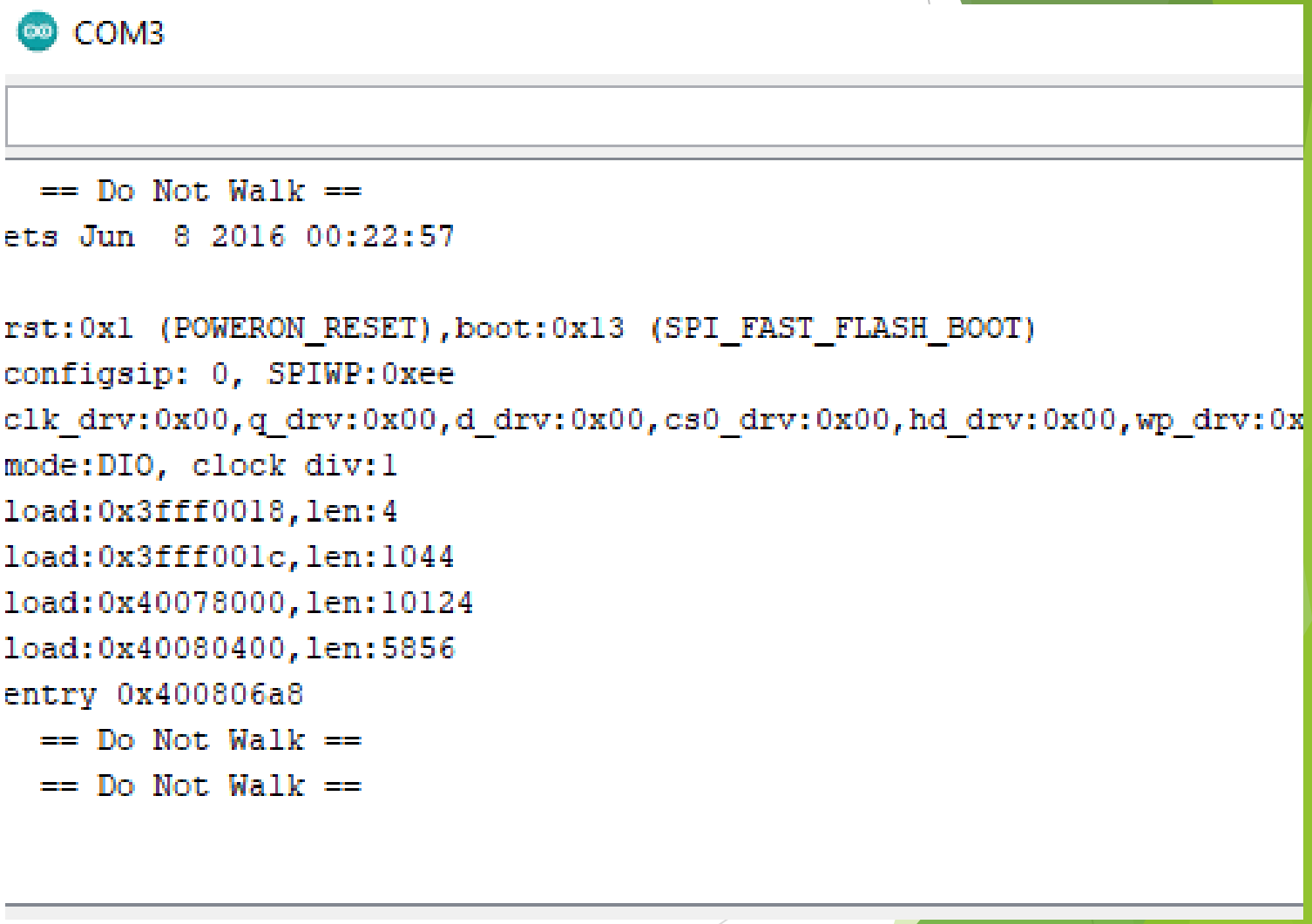


Screenshot of Code in Arduino IDE



Screenshot of Serial Monitor in Arduino IDE

Screenshot of output in Serial Monitor



Challenges

- Challenges included:
  - Identifying the parts needed
  - Learning how to work with Arduino IDE
  - Uploading the program code at each stage
  - Testing the additions at each stage

Career Skills

- I learned how to:
- Create a circuit with Arduino IDE
- Work with Sensors
- How to upload program code into Arduino IDE
- Further developed basic and advanced computer skills

Conclusion

- I found this class; learning how to use Arduino IDE, and the building of the Multiple Traffic Light Controller to be fascinating. Arduino IDE can be very useful when designing a system with physical parts. Arduino IDE is a great program for designing a system when you have the physical parts. The creation of the Multiple Traffic Light Controller with a Cross Walk and an Emergency Buzzer with secured IoT Control via Web, was both educational and inspirational at each stage of the development. I feel this project will help me in the future.