CEIS 114 Final Project

By Brady Sisk

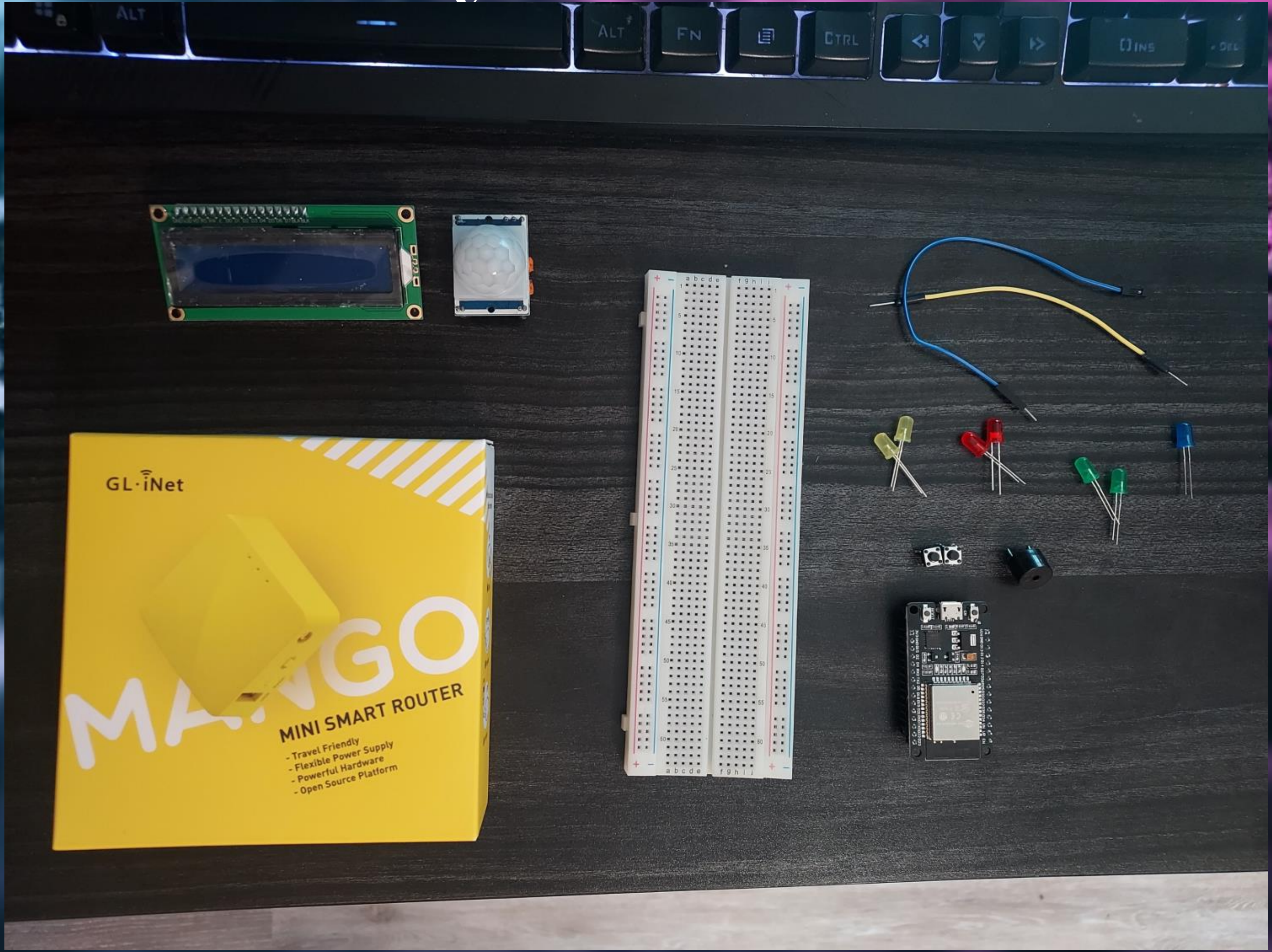
Introduction

- Building and programming of two way traffic controller with a pedestrian crossing and an emergency signal
- The final step is to secure the system so that is could be controlled remotely via web browser
- The alternative final is a non internet connection using a motion sensor to allow traffic to continuously flow on major street and only switches to slow traffic street when motion is detected

Project Preparation

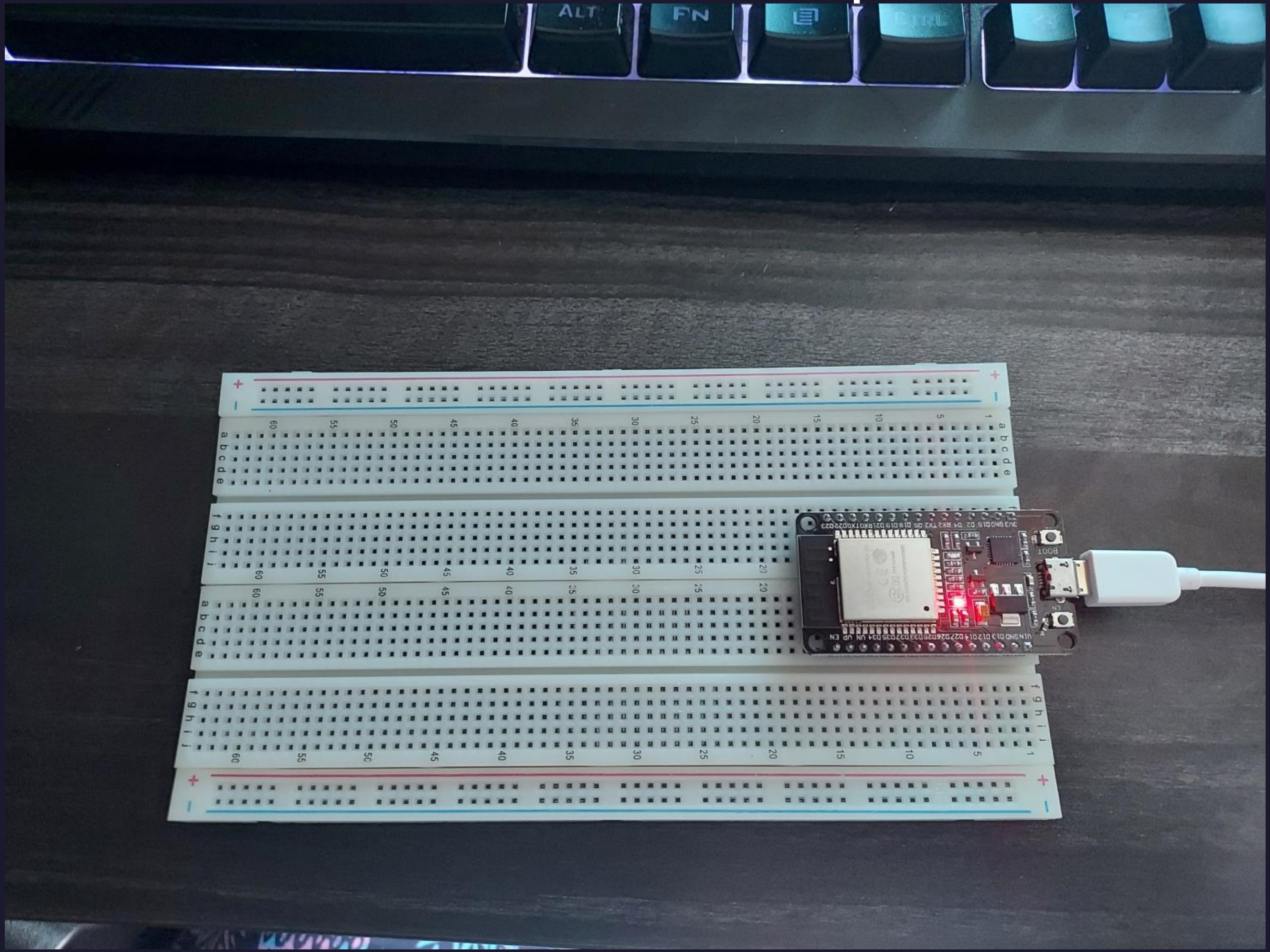
Inventory

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



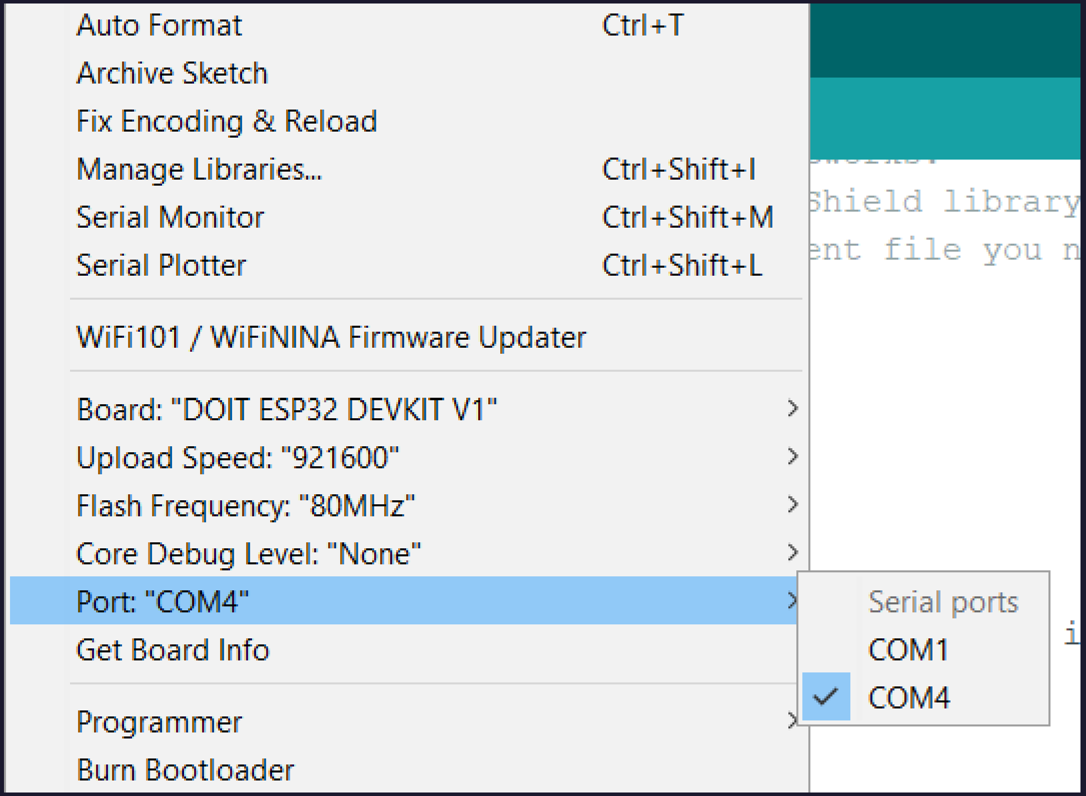
ESP32

Microcontroller mounted and powered ON

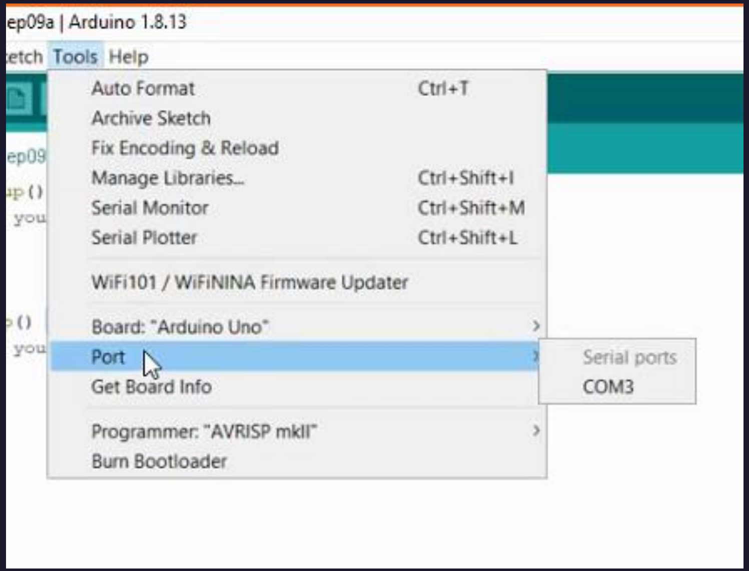


Installation of Arduino IDE

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


Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM1
COM4

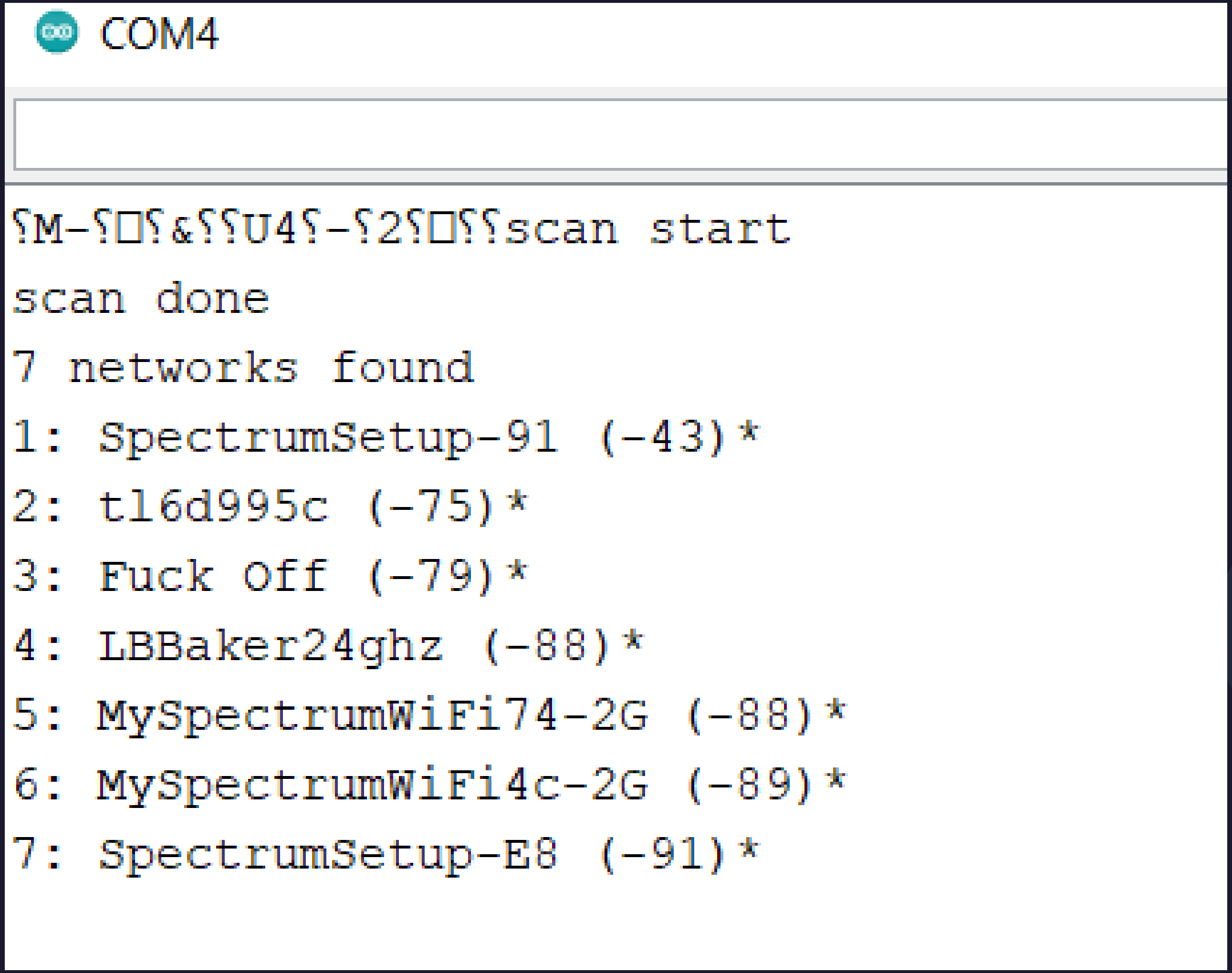
Board: "DOIT ESP32 DEVKIT V1"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Core Debug Level: "None"
Port: "COM4"
✓ COM4



Board: "Arduino Uno"
Programmer: "AVRISP mkII"

ESP32 WiFi Scan

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

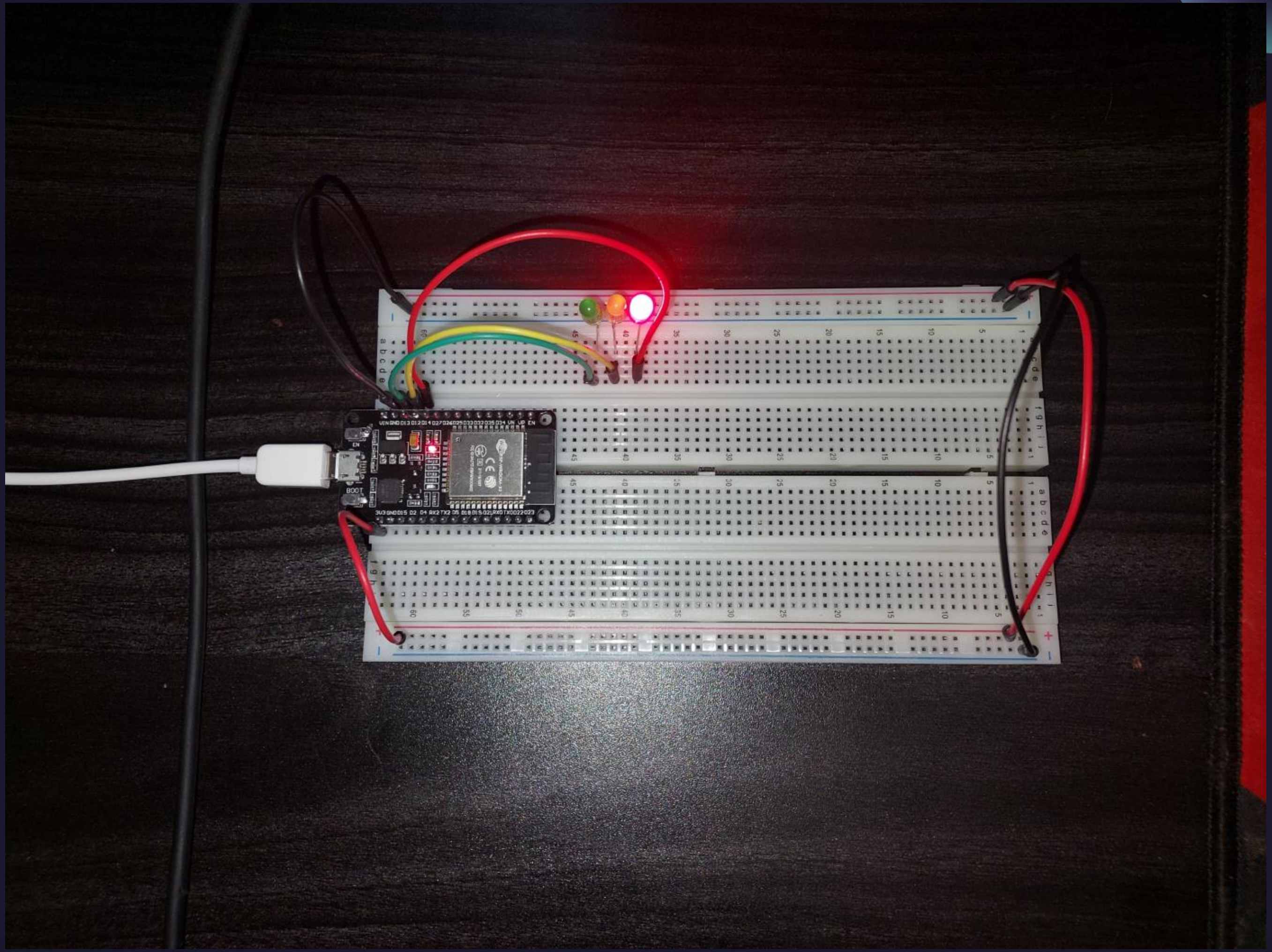

COM4

```
⸮M-⸮□⸮&⸮⸮U4⸮-⸮2⸮□⸮⸮scan start
scan done
7 networks found
1: SpectrumSetup-91 (-43)*
2: tl6d995c (-75)*
3: Fuck Off (-79)*
4: LBBaker24ghz (-88)*
5: MySpectrumWiFi74-2G (-88)*
6: MySpectrumWiFi4c-2G (-89)*
7: SpectrumSetup-E8 (-91)*
```

Creating Basic Traffic Controller

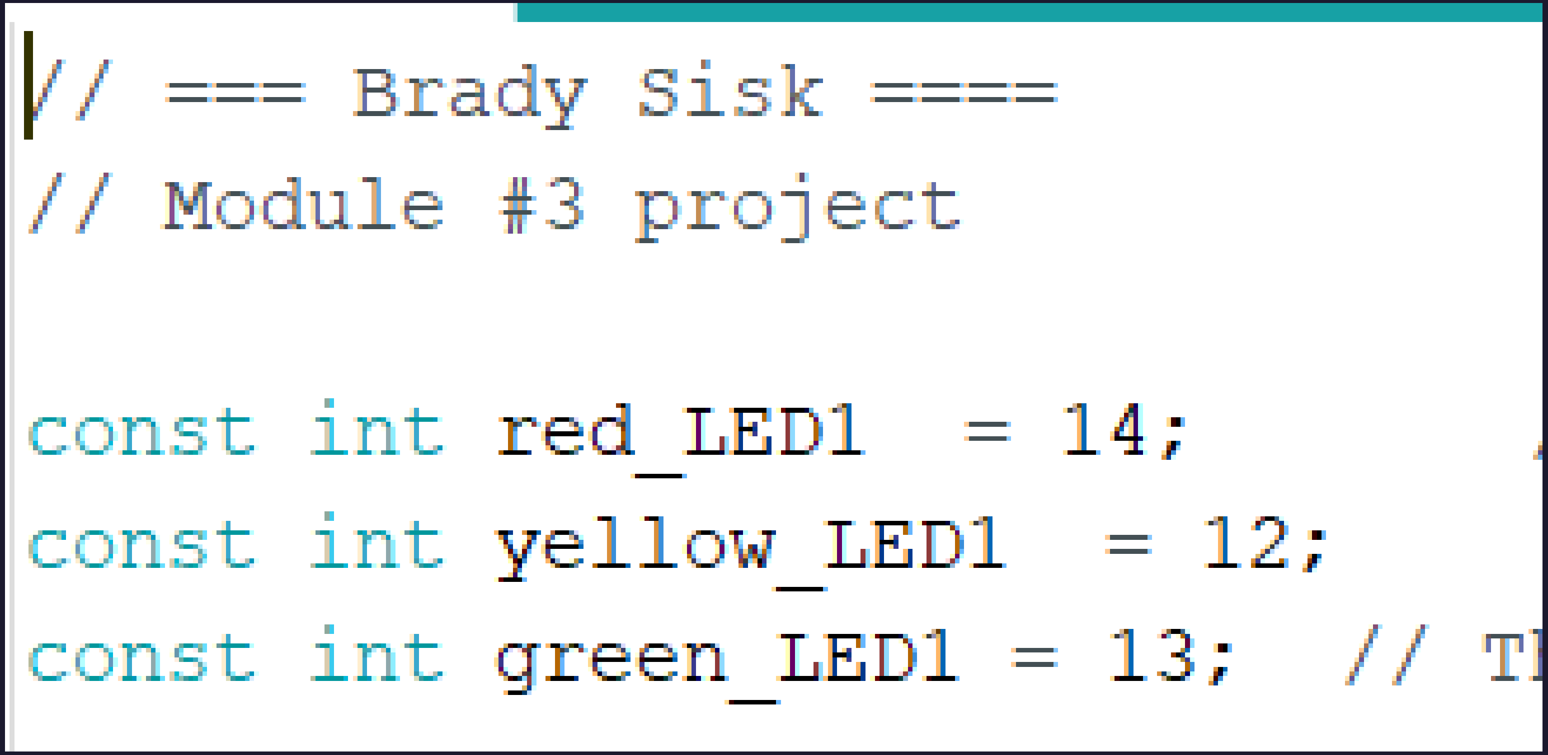
Picture of circuit with working LEDs

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



Screenshot of code in Arduino IDE

Screenshot of code in Arduino IDE



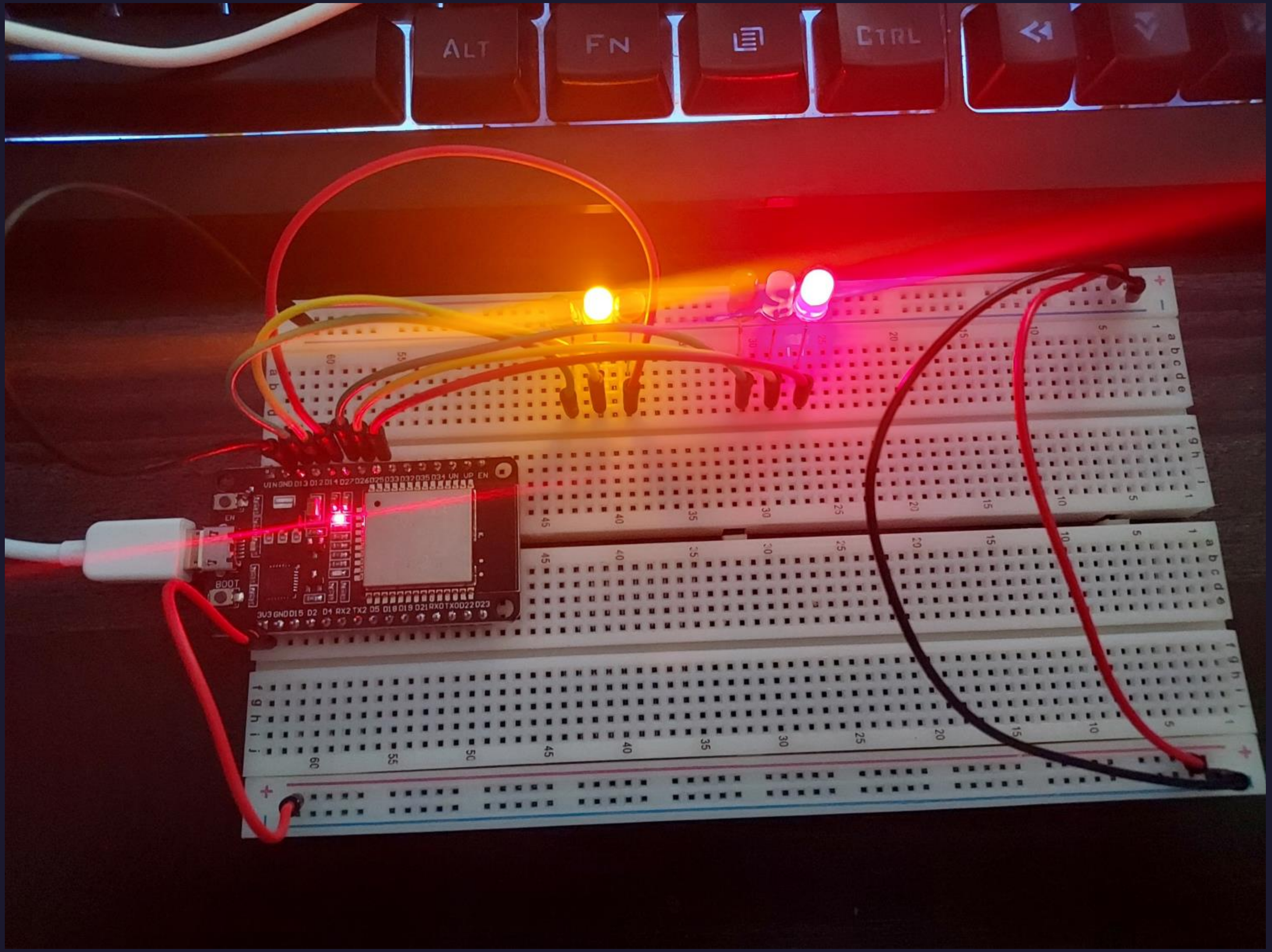
```
// === Brady Sisk ====
// Module #3 project


const int red_LED1   = 14;
const int yellow_LED1  = 12;
const int green_LED1 = 13;  // T
```

Multiple Traffic Lights

Picture of circuit with working LEDs

Picture of circuit with working LEDs

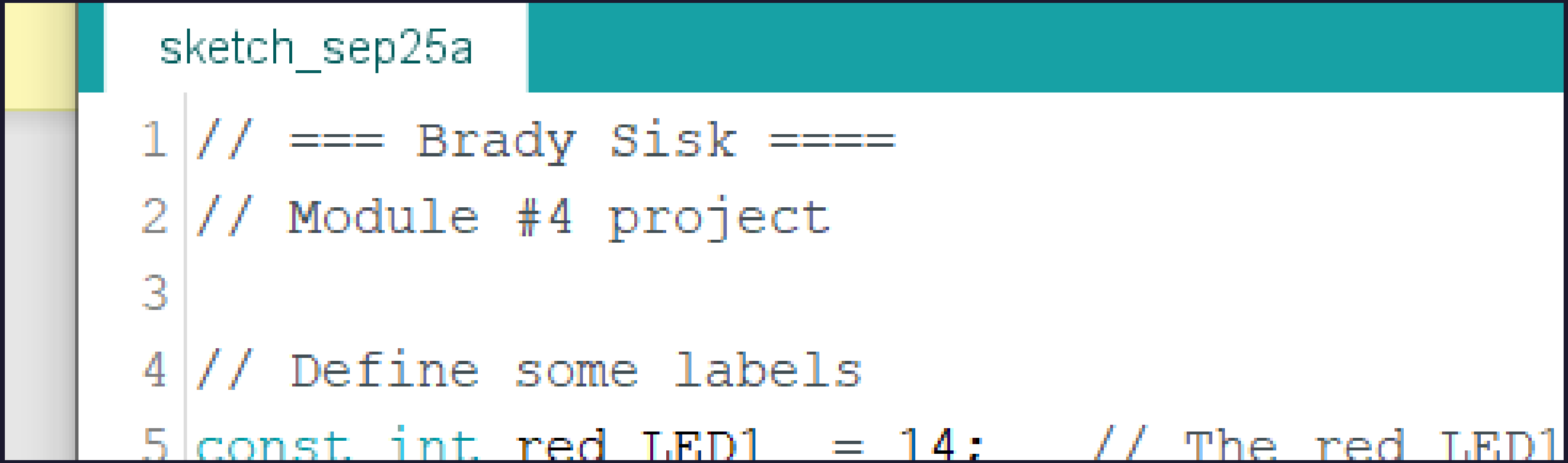
ESP 32 Board
Colored LEDs: Red, Yellow and Green (two sets)
220 Ohm Resistors (optional)
Wires
Breadboard



Screenshot of code in Arduino IDE

Screenshot of code in Arduino IDE

Screenshot of code in Arduino IDE



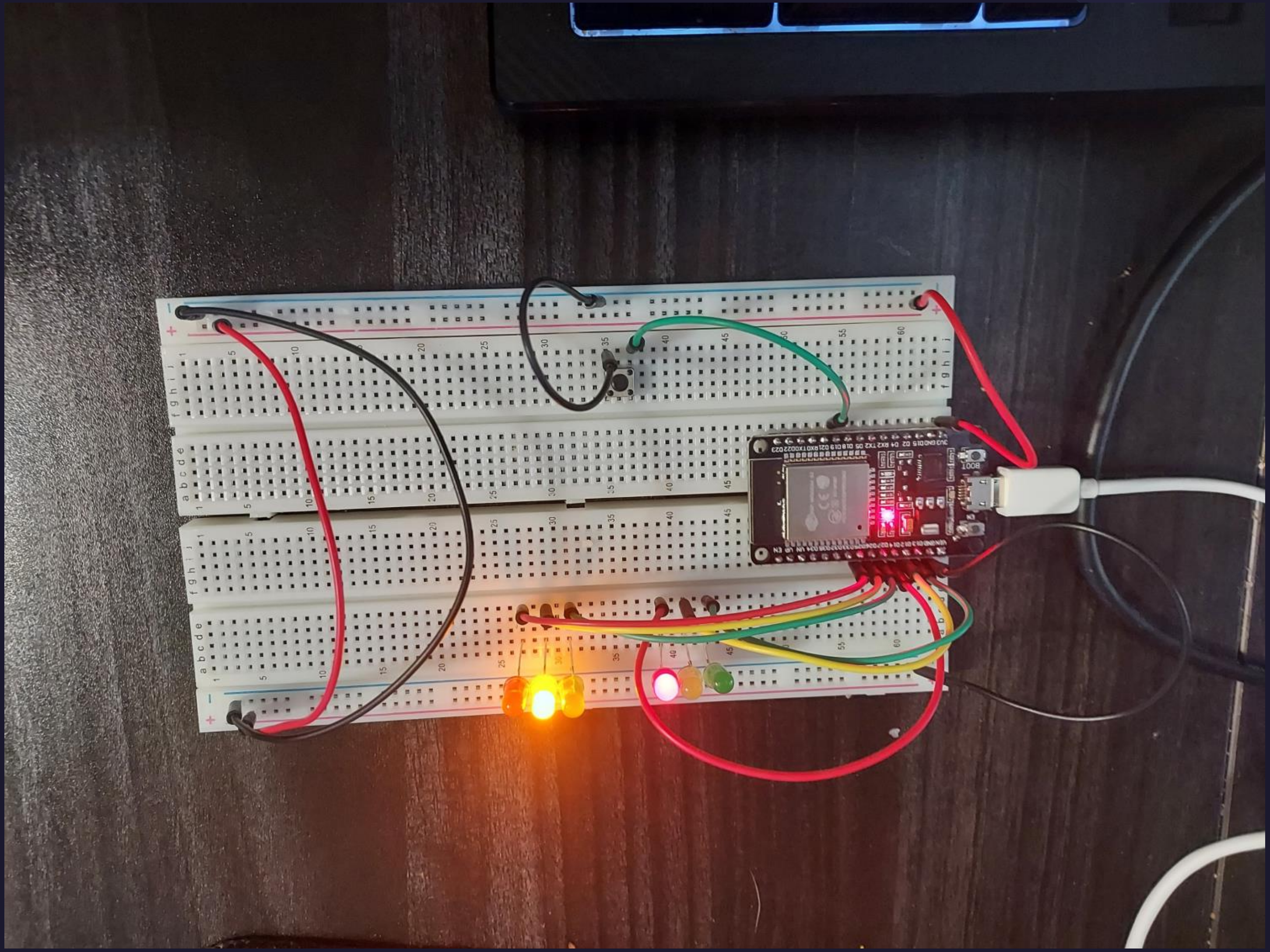
```
// === Brady Sisk ====
// Module #4 project

// Define some labels
const int red LED1  = 14;    // The red LED1
```

Adding Crosswalk Button

Picture of circuit with working LEDs

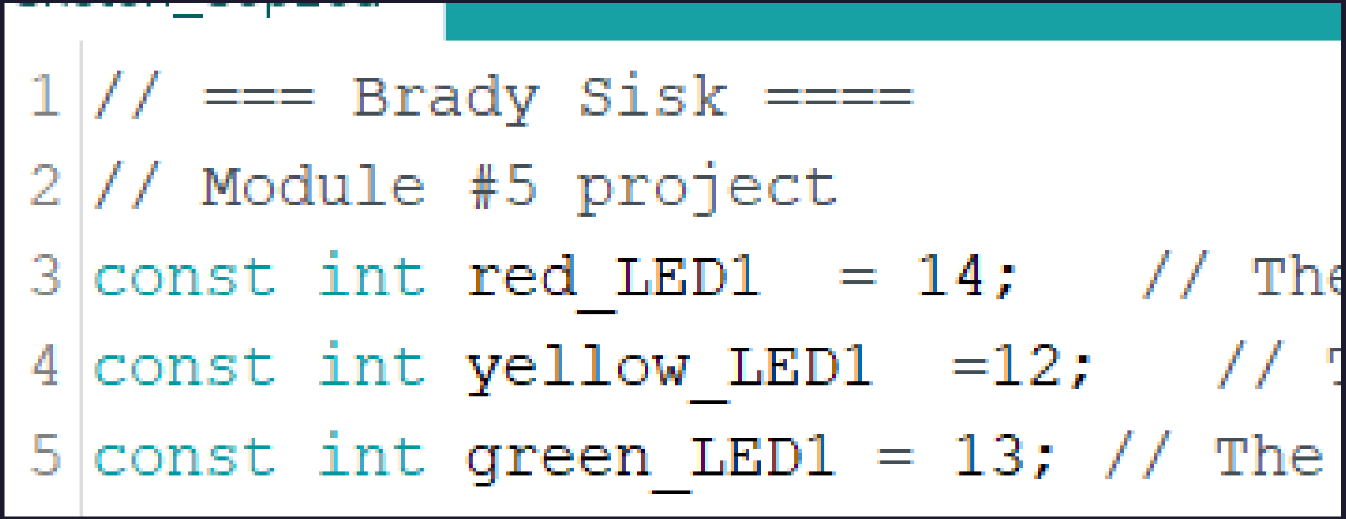
Picture of circuit with working LEDs

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

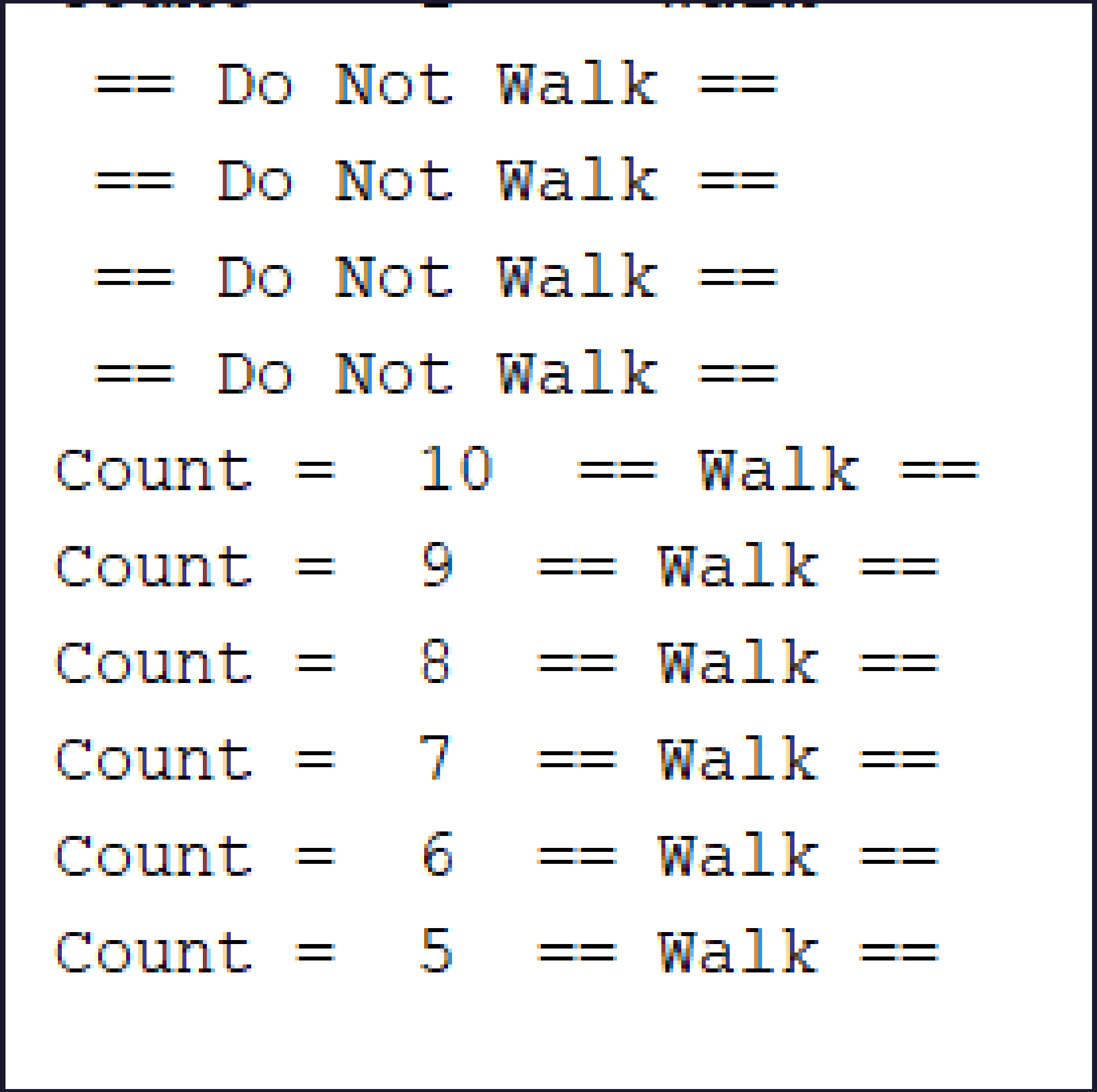
Screenshot of code in Arduino IDE showing **your name in the comment**



```
// === Brady Sisk ====
// Module #5 project
const int red_LED1  = 14;    // The
const int yellow_LED1  =12;    // T
const int green_LED1 = 13; // The
```

Screenshot of Serial Monitor in Arduino IDE

Screenshot of output in Serial Monitor

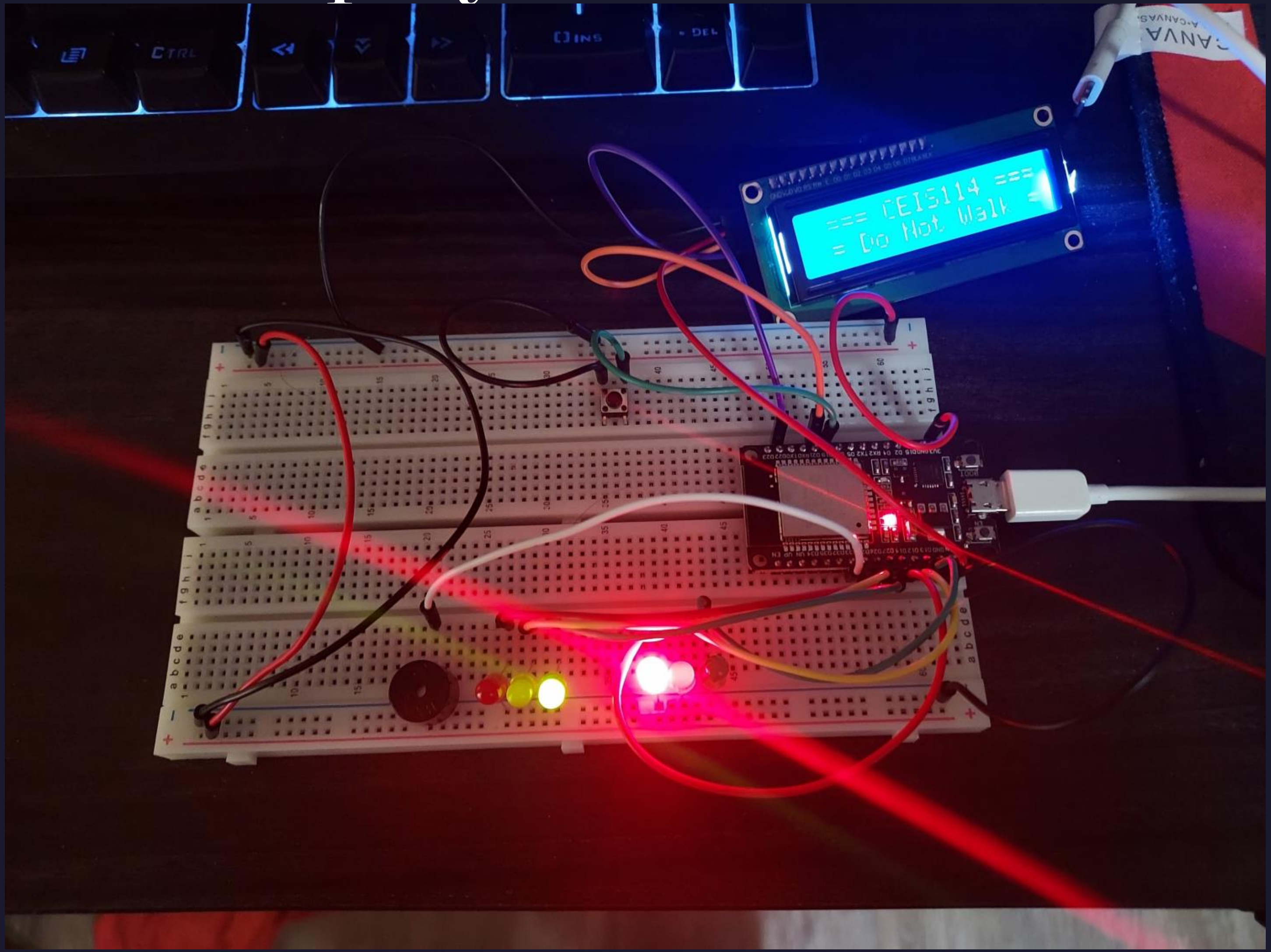


```
 == Do Not Walk ==
 == Do Not Walk ==
 == Do Not Walk ==
 == Do Not Walk ==
Count =  10  == Walk ==
Count =  9  == Walk ==
Count =  8  == Walk ==
Count =  7  == Walk ==
Count =  6  == Walk ==
Count =  5  == Walk ==
```

Adding Buzzer and LCD Display

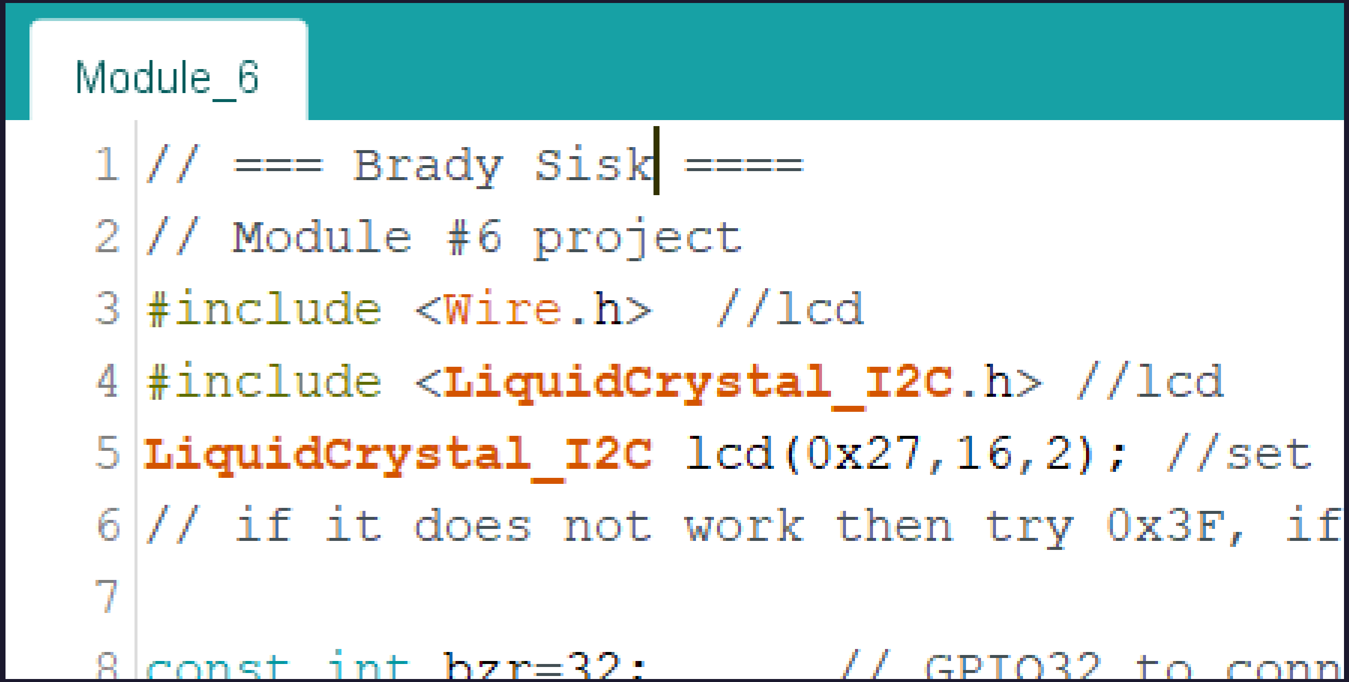
Picture of circuit with working LEDs and LCD display

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 Arduino IDE

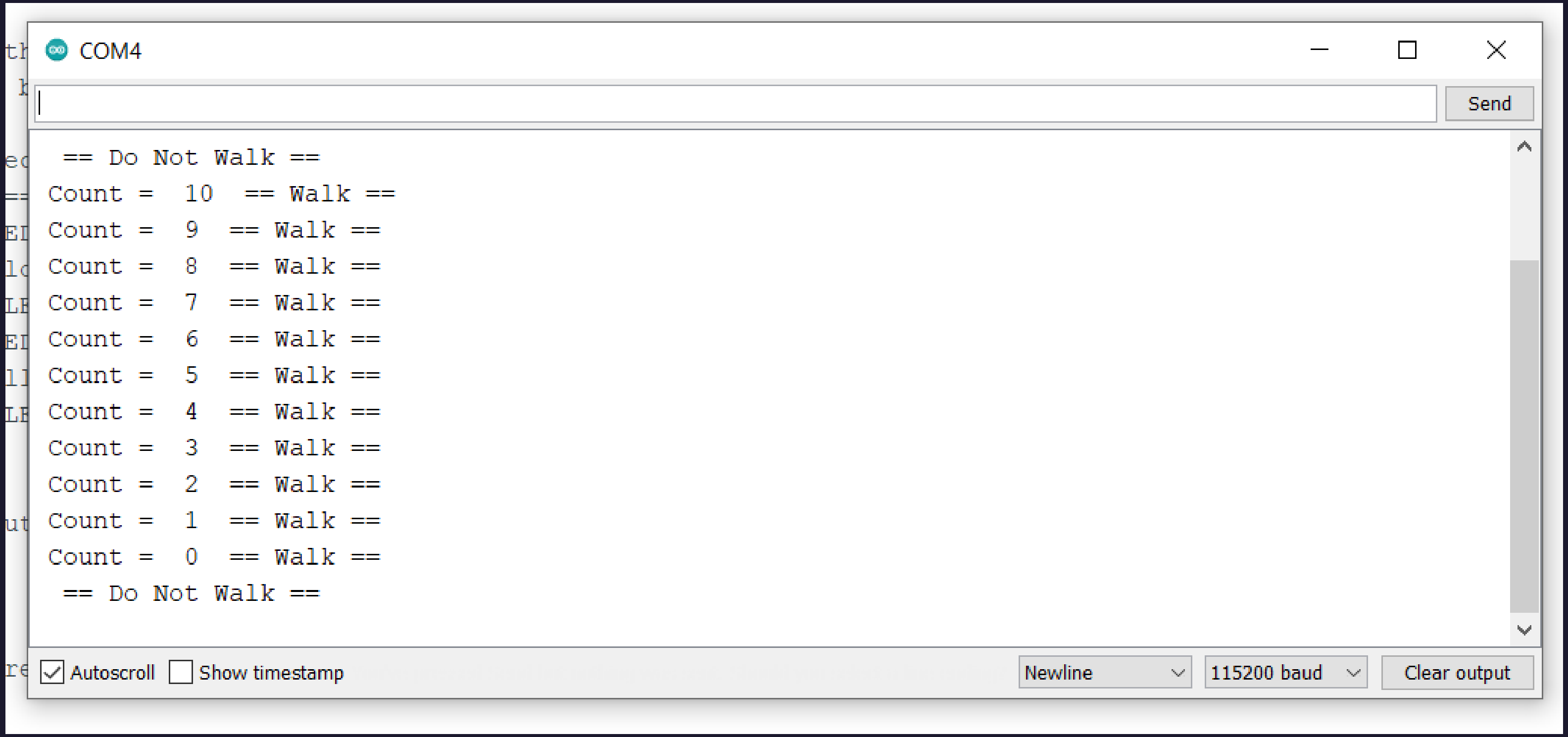
Screenshot of code in Arduino IDE



```
// === Brady Sisk ====
// Module #6 project
#include <Wire.h>  //lcd
#include <LiquidCrystal_I2C.h> //lcd
LiquidCrystal_I2C lcd(0x27,16,2); //set
// if it does not work then try 0x3F, if

const int bzr=32;       // GPIO32 to conn
```

Screenshot of Serial Monitor in Arduino IDE

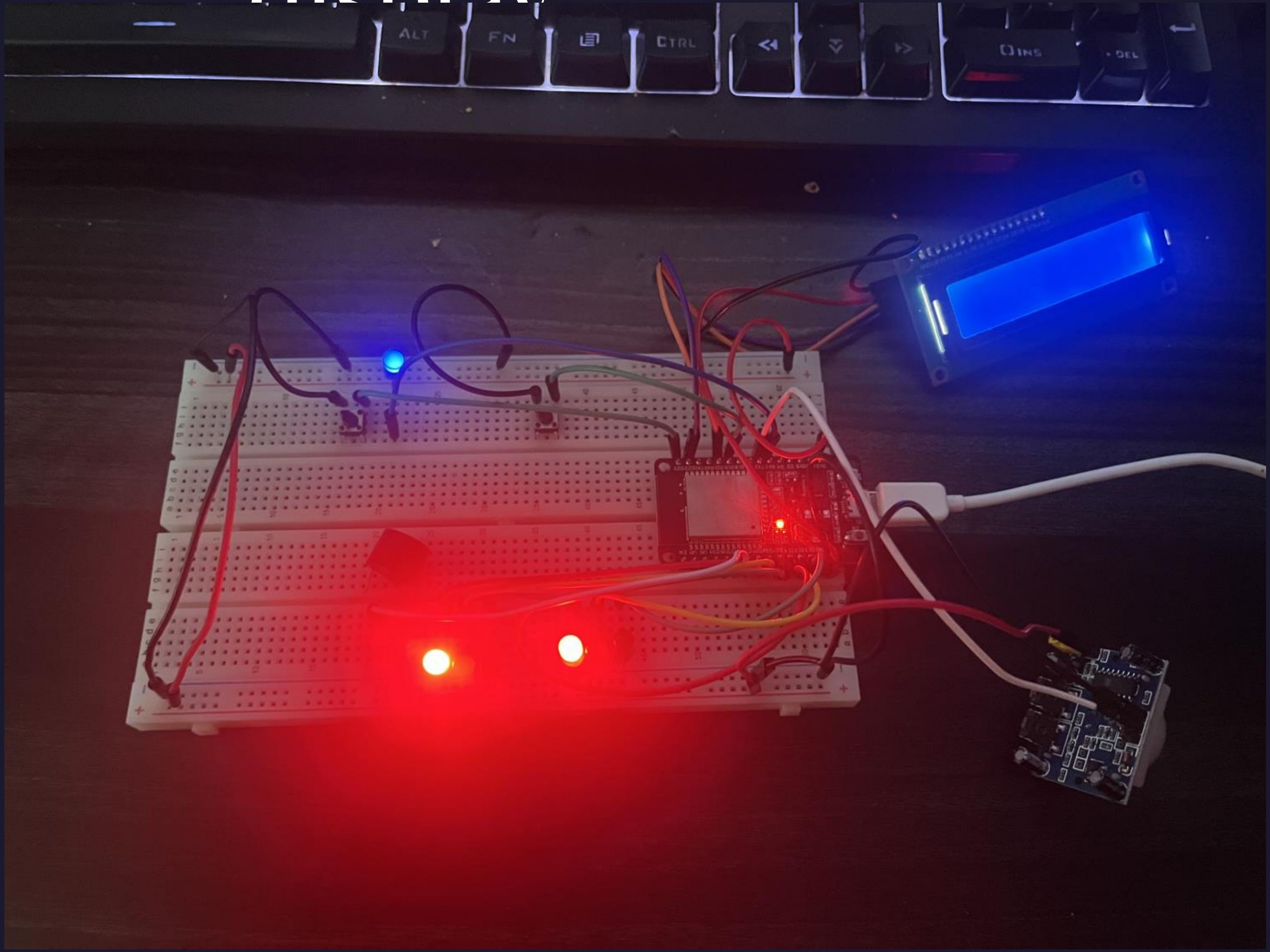
Screenshot of output in Serial Monitor



```
 == Do Not Walk ==
Count =  10  == Walk ==
Count =  9  == Walk ==
Count =  8  == Walk ==
Count =  7  == Walk ==
Count =  6  == Walk ==
Count =  5  == Walk ==
Count =  4  == Walk ==
Count =  3  == Walk ==
Count =  2  == Walk ==
Count =  1  == Walk ==
Count =  0  == Walk ==
 == Do Not Walk ==
```

Adding Remote Emergency Control

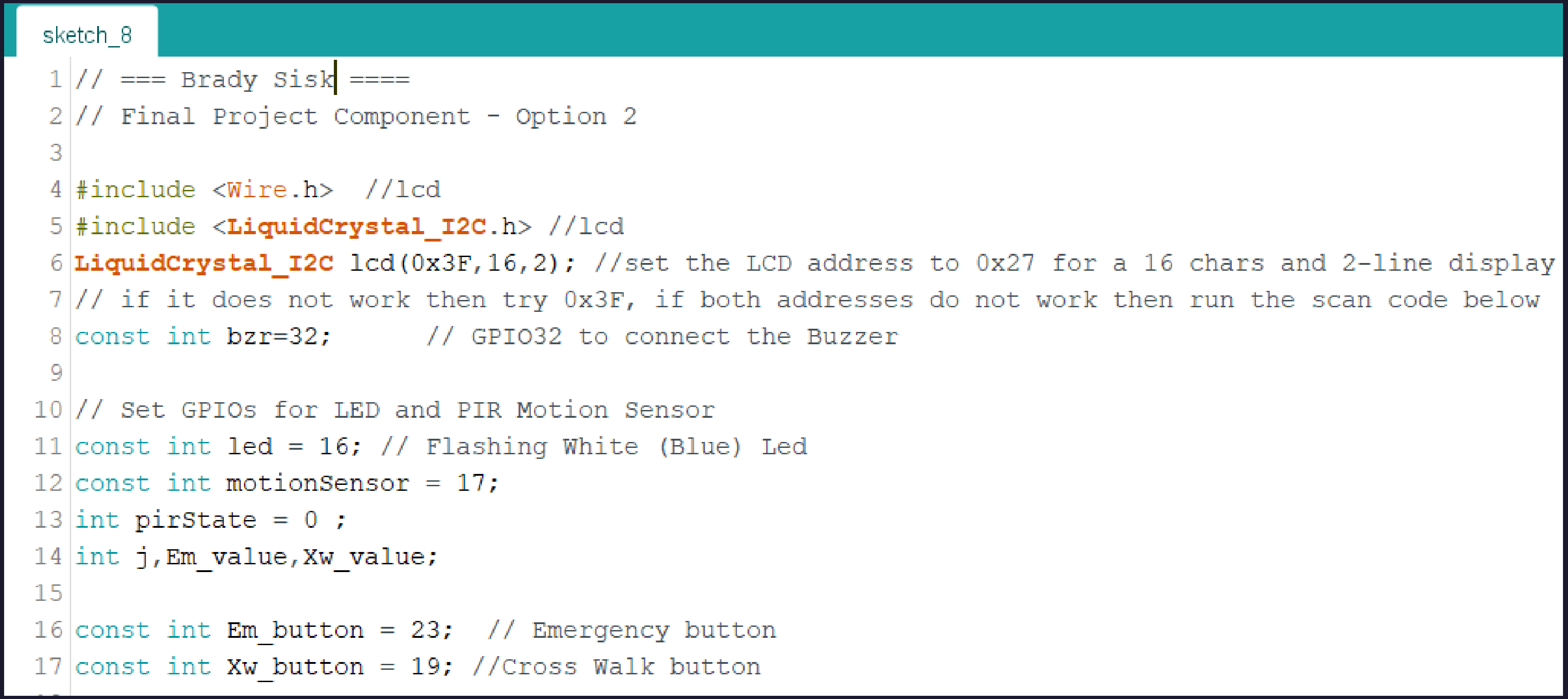
Picture of circuit with working LEDs and LCD display



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

Screenshot of code in Arduino IDE

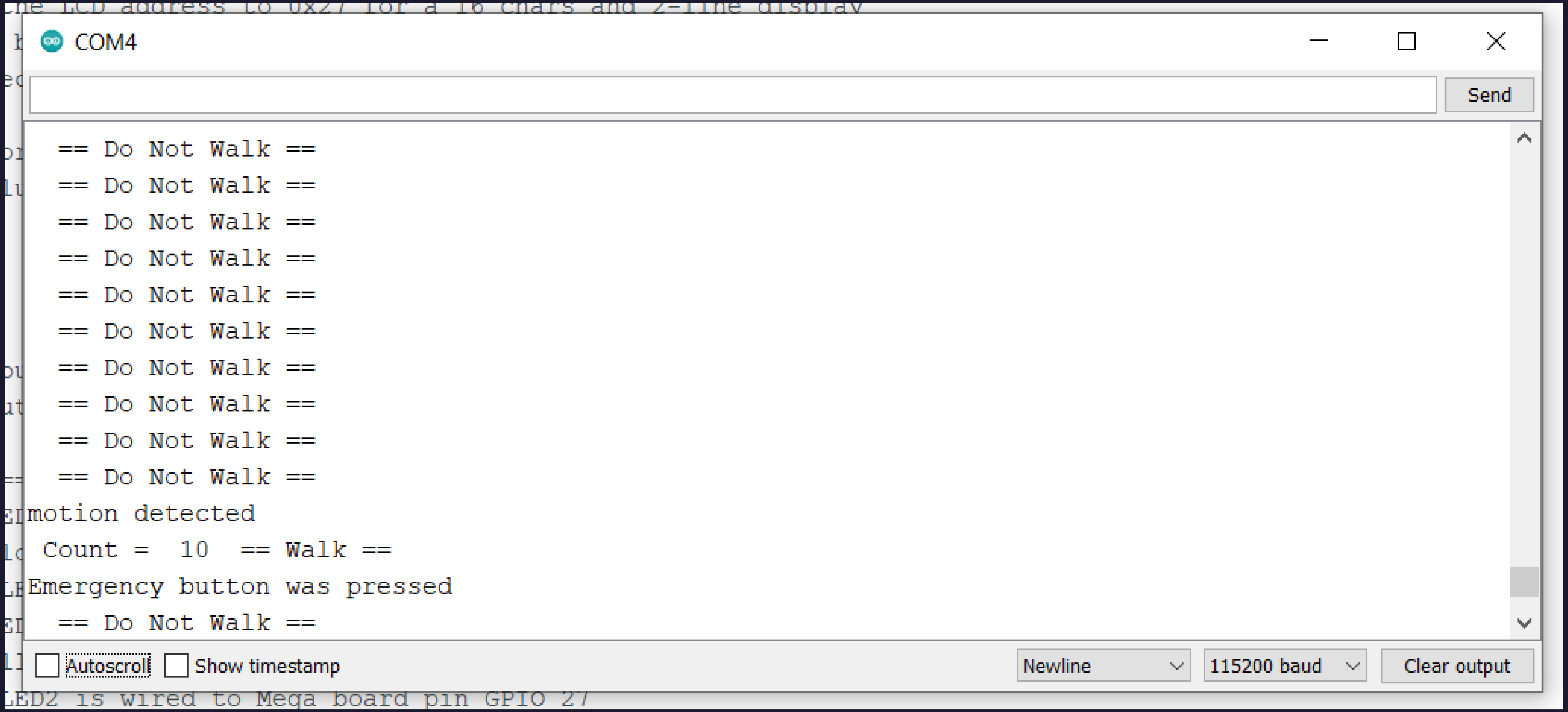
Screenshot of code in Arduino IDE



```
// === Brady Sisk ====
// Final Project Component - Option 2

#include <Wire.h>  //lcd
#include <LiquidCrystal_I2C.h> //lcd
LiquidCrystal_I2C lcd(0x3F,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 below
const int bzr=32;      // GPIO32 to connect the Buzzer

// Set GPIOs for LED and PIR Motion Sensor
const int led = 16; // Flashing White (Blue) Led
const int motionSensor = 17;
int pirState = 0 ;
int j,Em_value,Xw_value;

const int Em_button = 23;  // Emergency button
const int Xw_button = 19; //Cross Walk button
```

Screenshot of Serial Monitor in Arduino IDE

Screenshot of output in Serial Monitor

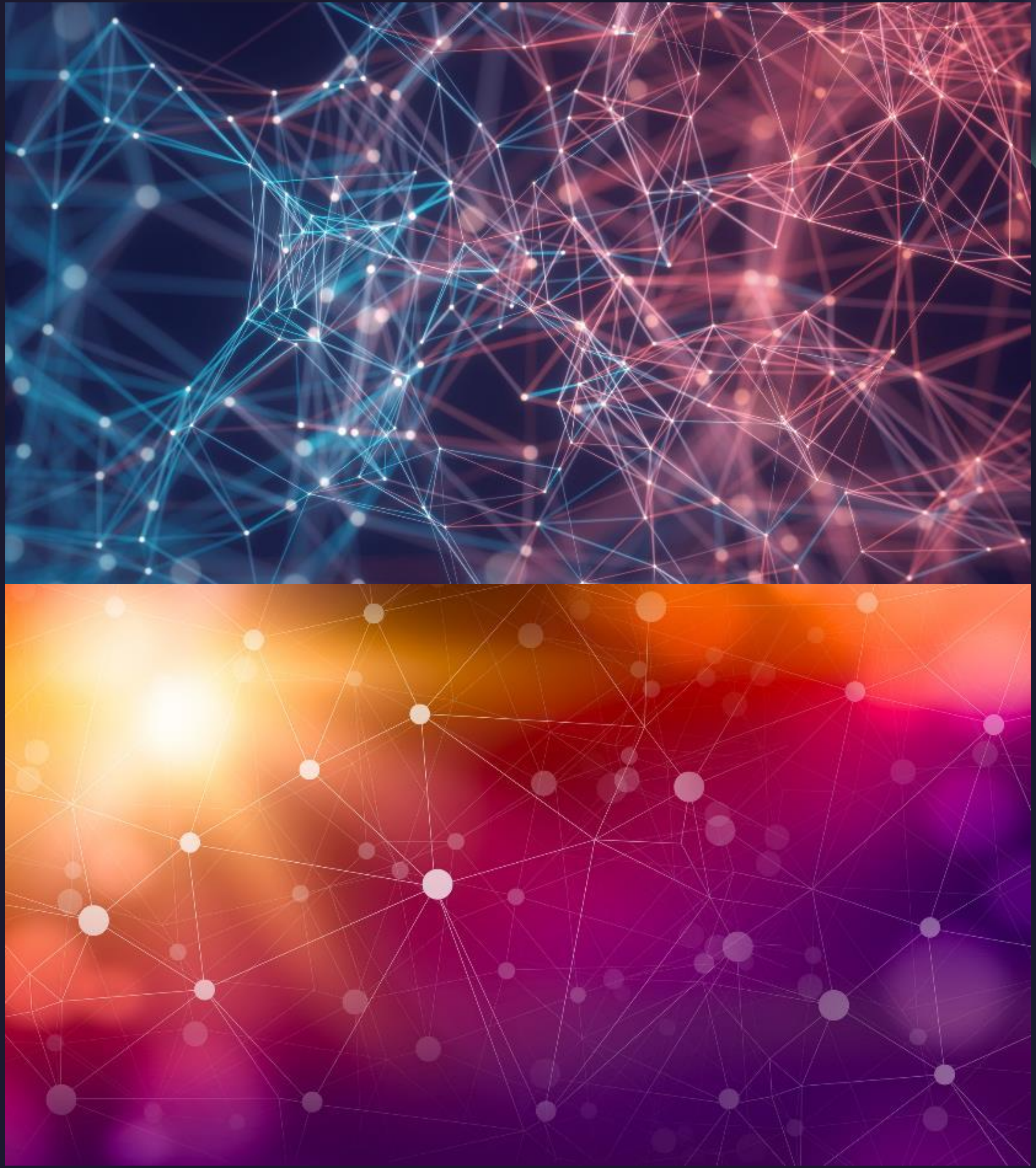


```
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
  == Do Not Walk ==
motion detected
 Count =  10   == Walk ==
Emergency button was pressed
  == Do Not Walk ==
```

Newline | 115200 baud

Challenges

- Forgetting to hold the boot button
- Making sure the right board and port was selected



Conclusion

This project encompasses the many areas of the IoT

Will help prepare me for this emerging industry with lots of exciting prospects