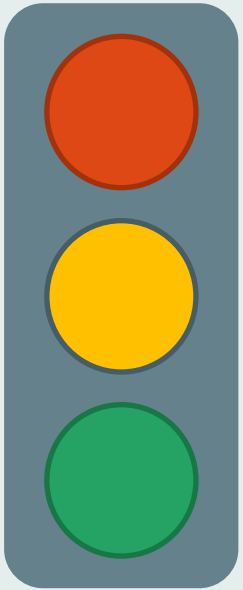
IoT Traffic Controller

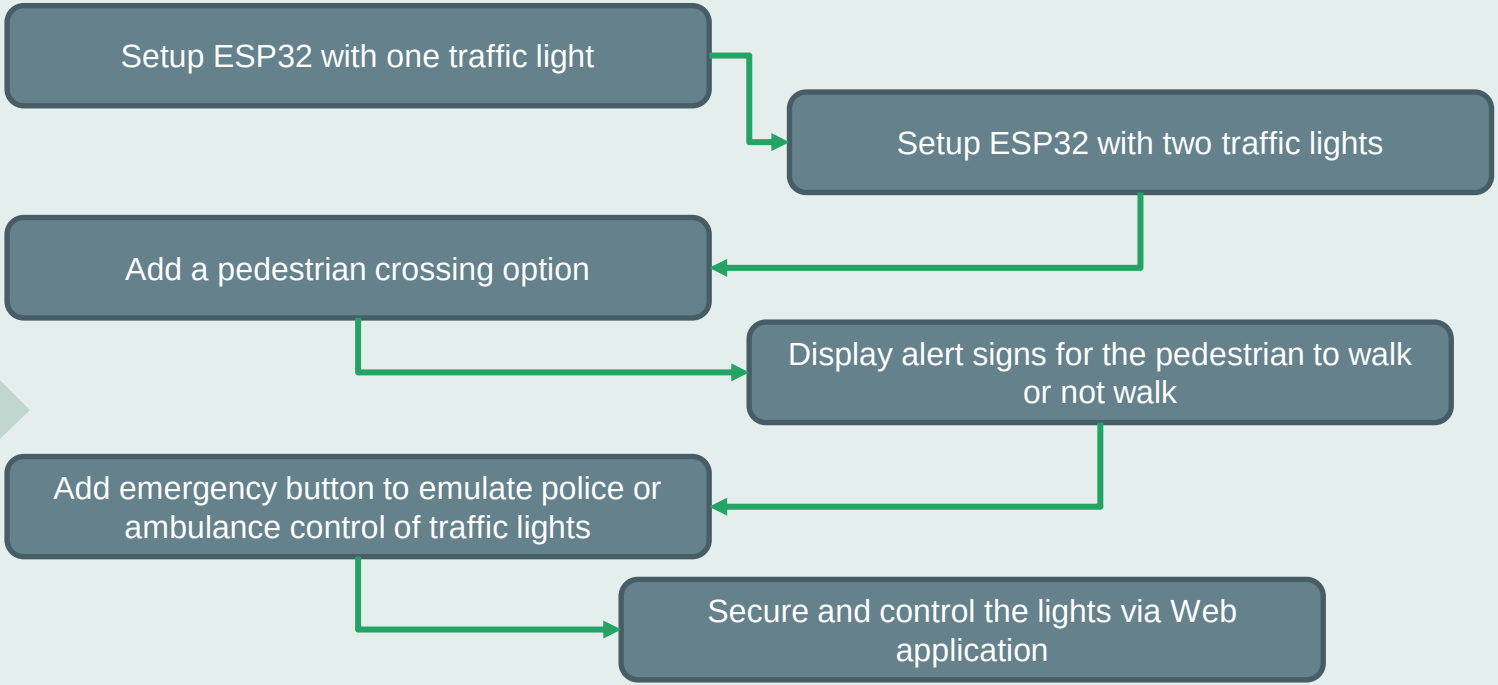


CEIS114
Faith Burnett
DeVry University

INTRODUCTION

In this project I used the ESP32 Microcontroller to implement a two-way traffic light controller. In addition this system includes pedestrian crossing and emergency access through Cayenne.

Project Flow



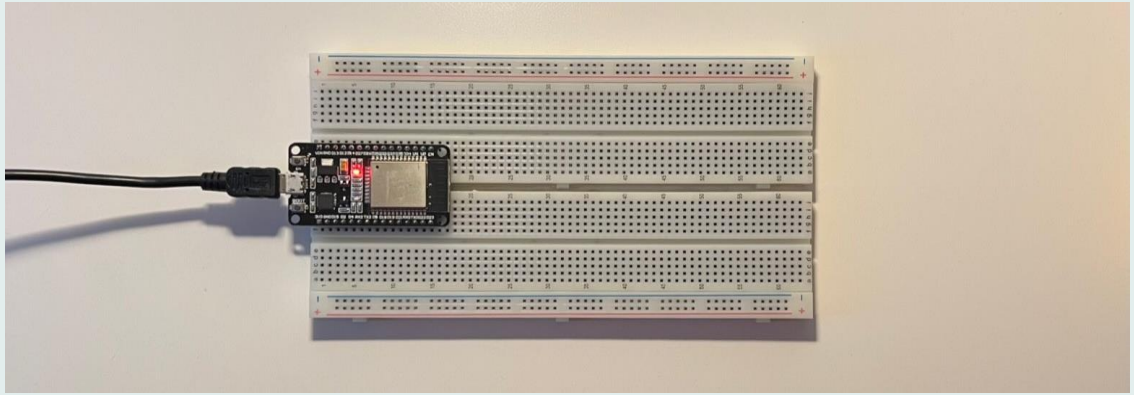
Inventory of project components

Installation & Testing

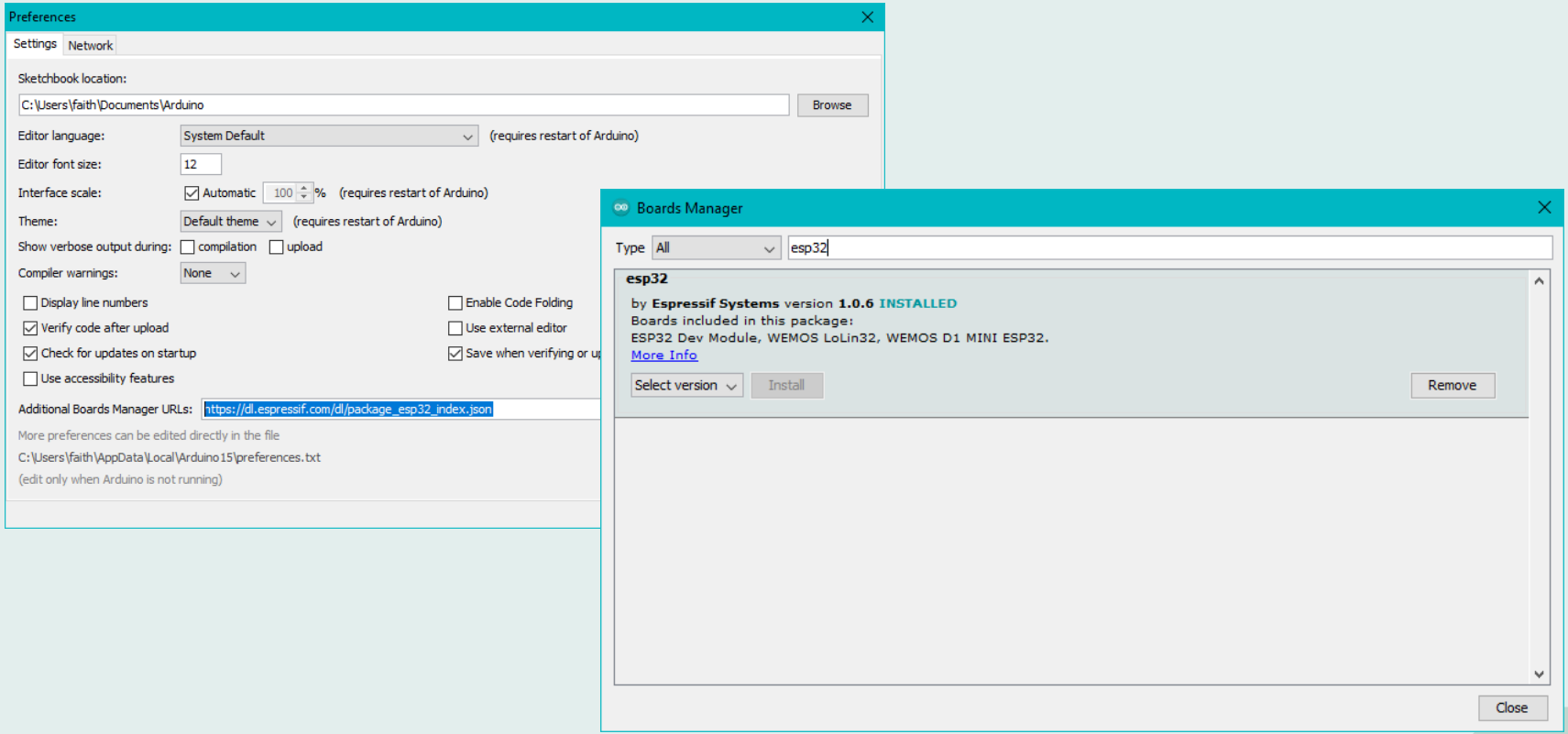
- Installing ESP32 board
- Testing installation
- Selecting port
- Performing a Wi-Fi scan

ESP32

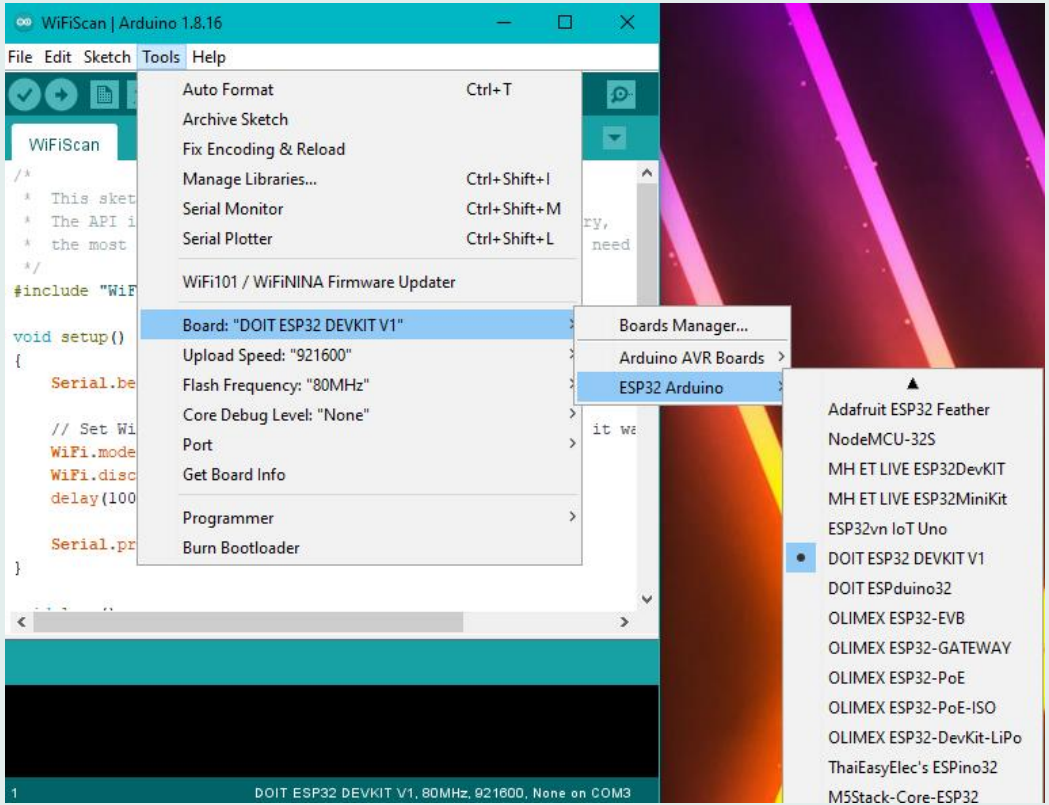
Microcontroller mounted and powered ON



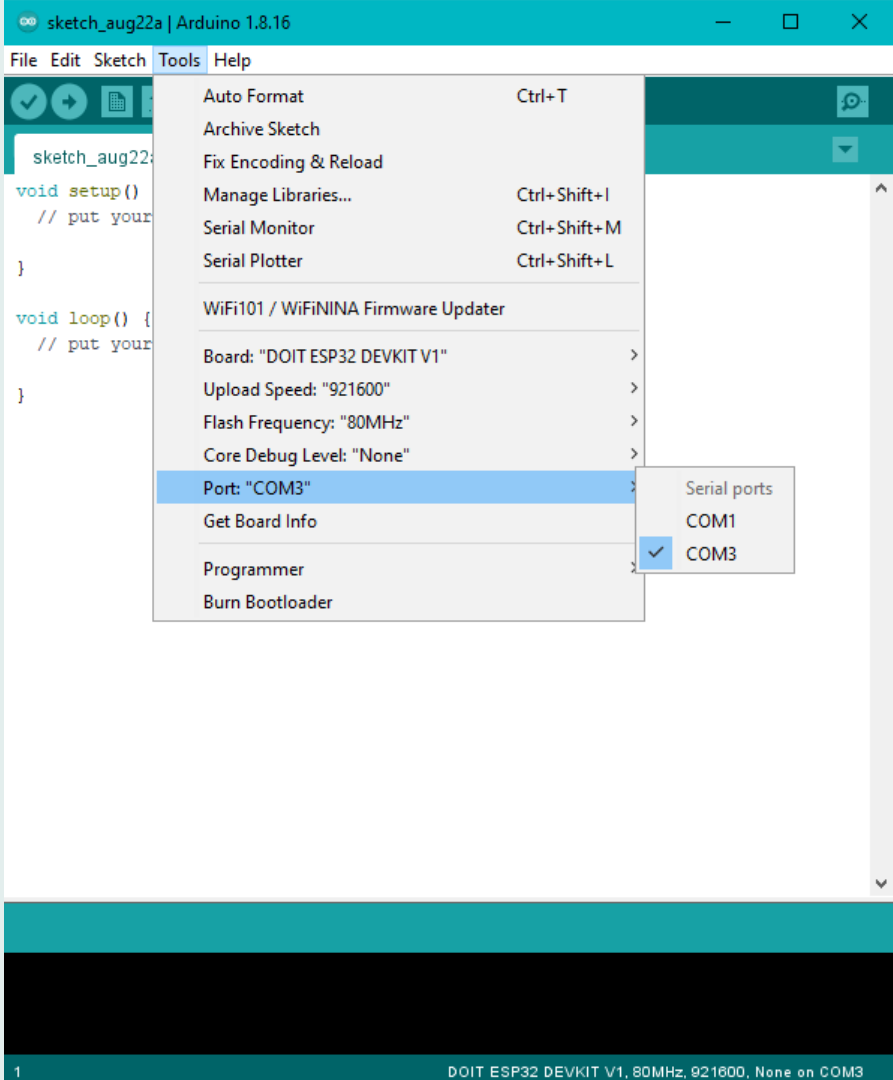
ESP32 Board Installed



ESP32 Installation Test



Port Selected



ESP32 WiFi Scan

COM3

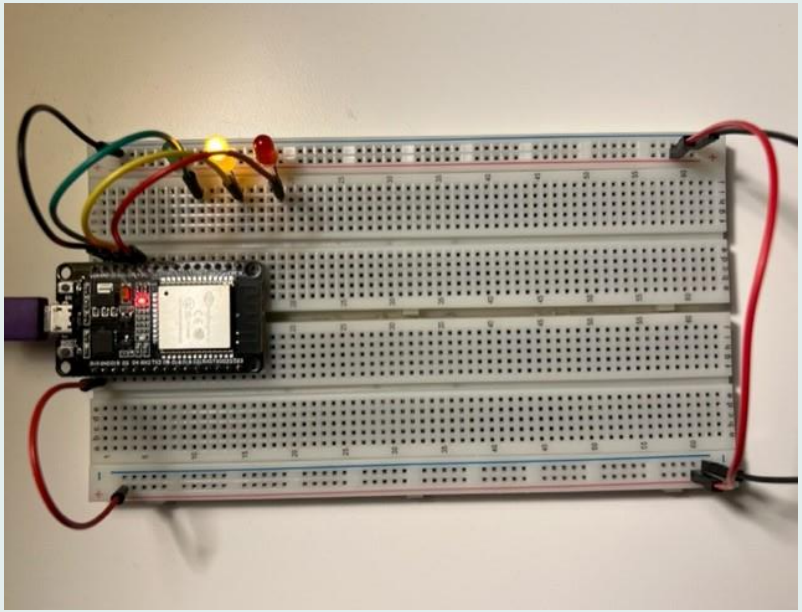
```
scan start
scan done
7 networks found
1: Pretty Fly for a Wi-Fi (-25)*
2: Pretty Fly for a Wi-Fi (-60)*
3: Pretty Fly for a Wi-Fi (-74)*
4: ollies wifi (-83)*
5: SpectrumSetup-00 (-87)*
6: MySpectrumWiFib0-2G (-89)*
7: SpectrumSetup-60 (-90)*

scan start
scan done
7 networks found
1: Pretty Fly for a Wi-Fi (-28)*
2: Pretty Fly for a Wi-Fi (-63)*
3: Pretty Fly for a Wi-Fi (-76)*
4: ollies wifi (-85)*
5: MySpectrumWiFib0-2G (-89)*
6: SpectrumSetup-00 (-89)*
7: SpectrumSetup-60 (-91)*

scan start
scan done
7 networks found
1: Pretty Fly for a Wi-Fi (-27)*
```

First Traffic light installation

- Initializing pins 12, 13, and 14 as output
- Creating a loop that turns each LED on and off in order of red, green, yellow

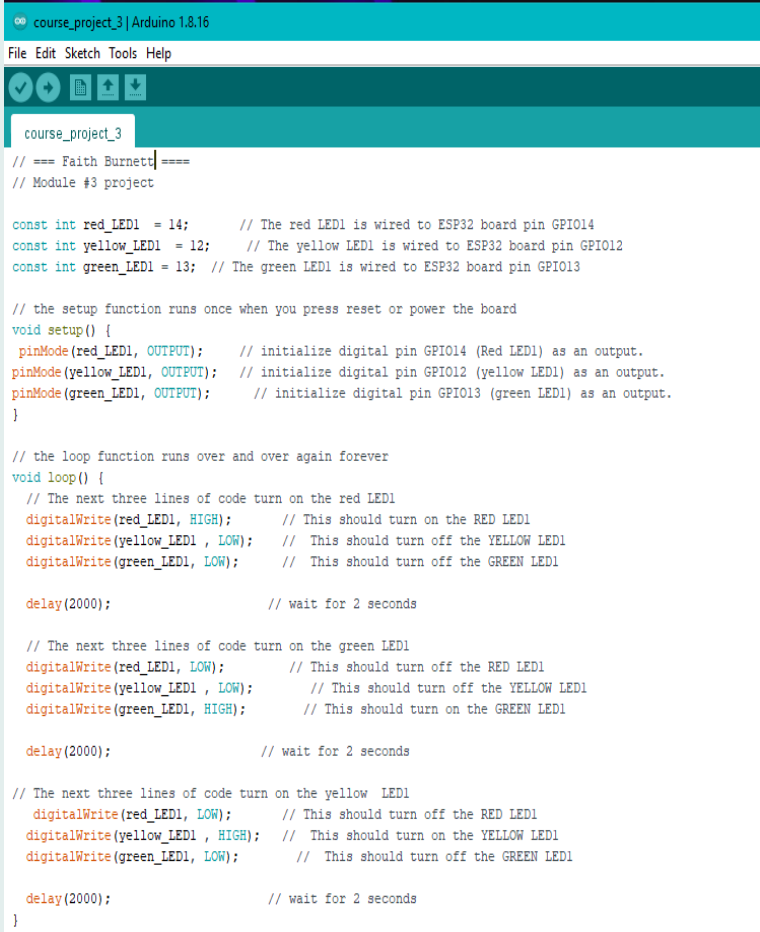
First Traffic Light

- Both sides of the breadboard are connected
- ESP32 is powered on
- Red LED1 is connected to GPIO14, Yellow LED1 connected to GPIO12, and green LED1 connected to GPIO13



First Traffic Light Code

- LED’s assigned board pins
- Digital pins are initialized
- Loop turning LED’s on and off in order red, green, yellow
- Time delay in between turning LED on and off



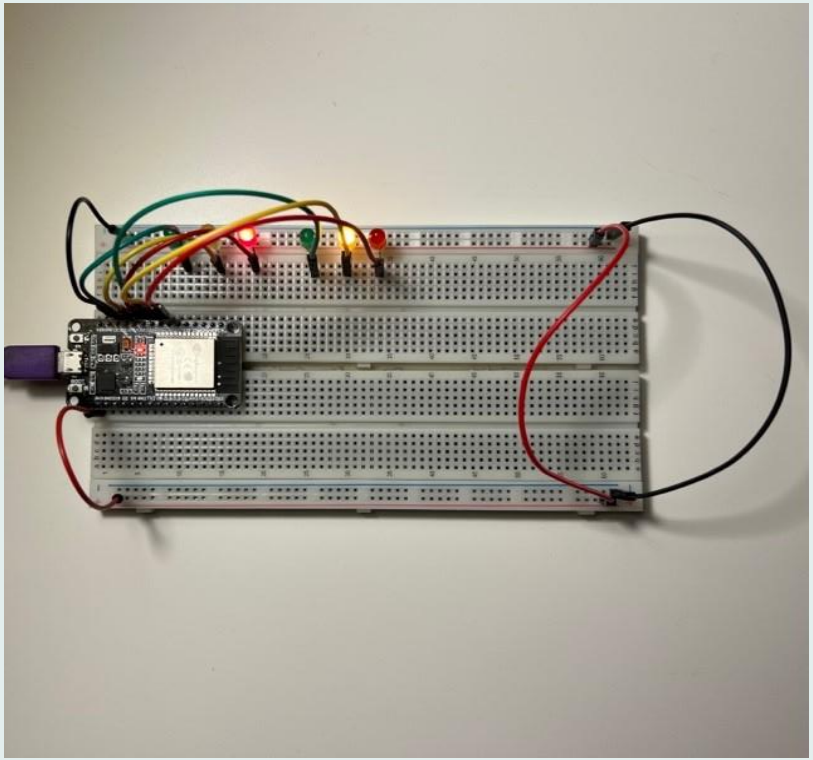

```
// === Faith Burnett ====
// Module #3 project

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
}
```

Second Traffic light installation

- Initializing pins 25, 26, and 27 as output
- Rework loop so that the second set of LED’s cycles through red, green, yellow when the first set of LED’s is on red

Second Traffic Light

- Red LED2 connected to GPIO25, yellow LED2 connected to GPIO26, and green LED2 connected to GPIO27



Adruino_Module_4 | Arduino 1.8.16

File Edit Sketch Tools Help

Adruino_Module_4

```
// === Faith Burnett ====
// 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.
}

// 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
```

Second Traffic Light Code

- Assigned second set of LED's board pins
- Second set of digital pins initialized
- Loop now iterates over the second set of LED's while first set is on red and over the first set when the second set is red
- Time delay in between turning LED's on and off

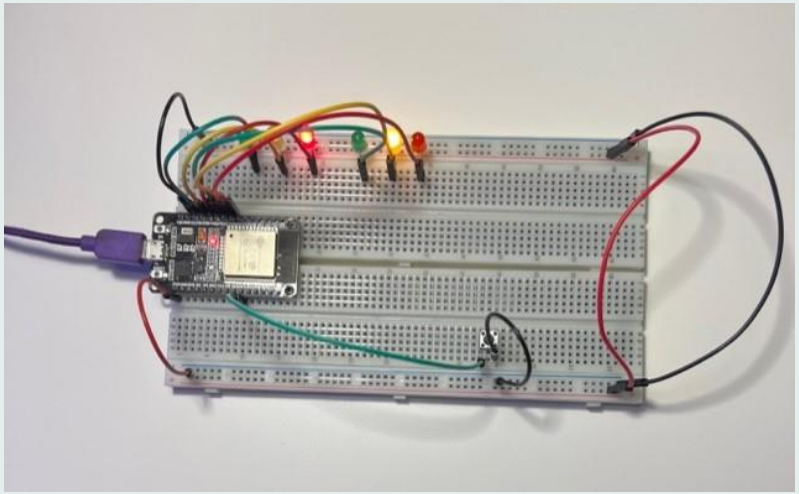
Crosswalk installation

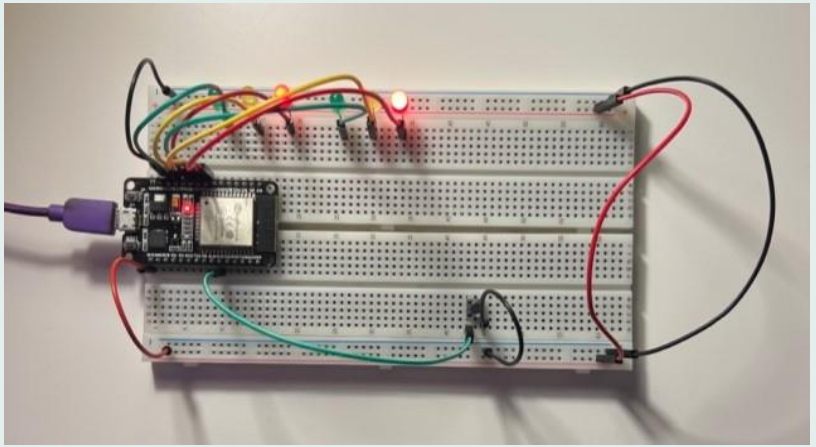
- Initializing pin 19 as input
- Rework loop so that the second set of LED’s cycles through red, green, yellow when the first set of LED’s is on red

Crosswalk button

- Crosswalk button connected to GPIO19

Traffic lights working





Crosswalk lights working

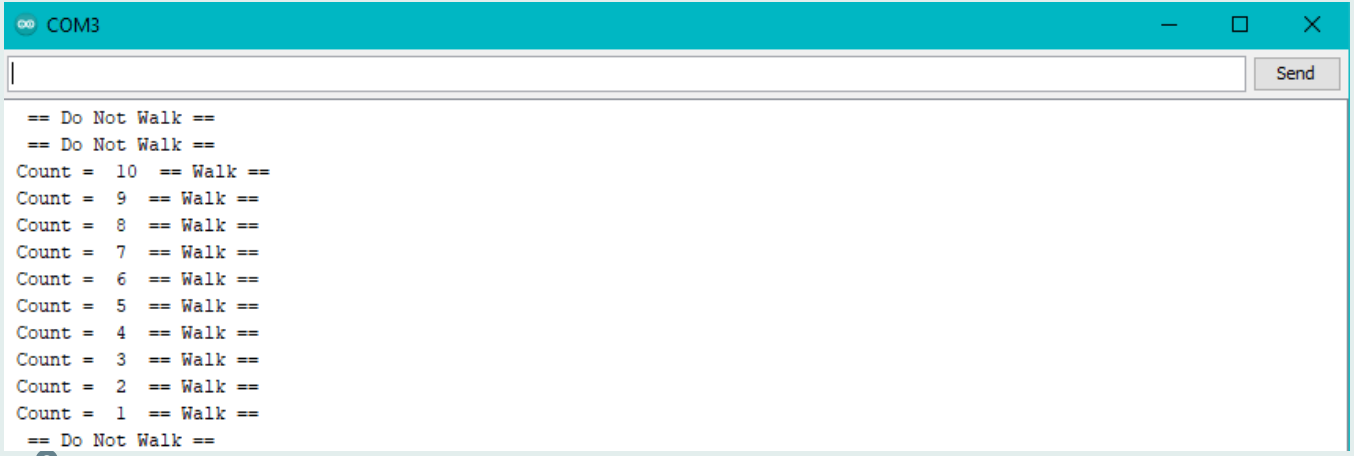
Crosswalk Code

- Assigned crosswalk button board pin
- Initialized crosswalk button as input
- Added statement to loop asking if button was pressed if it was both sets go to red LED and serial monitor will display walk with a countdown 10-0
- Else loop iterates through both sets normally

Adruino_Module5

```
// === Faith Burnett ====
// 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 == 0 ){ // 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
 } // End of counter
```

Crosswalk Serial Monitor

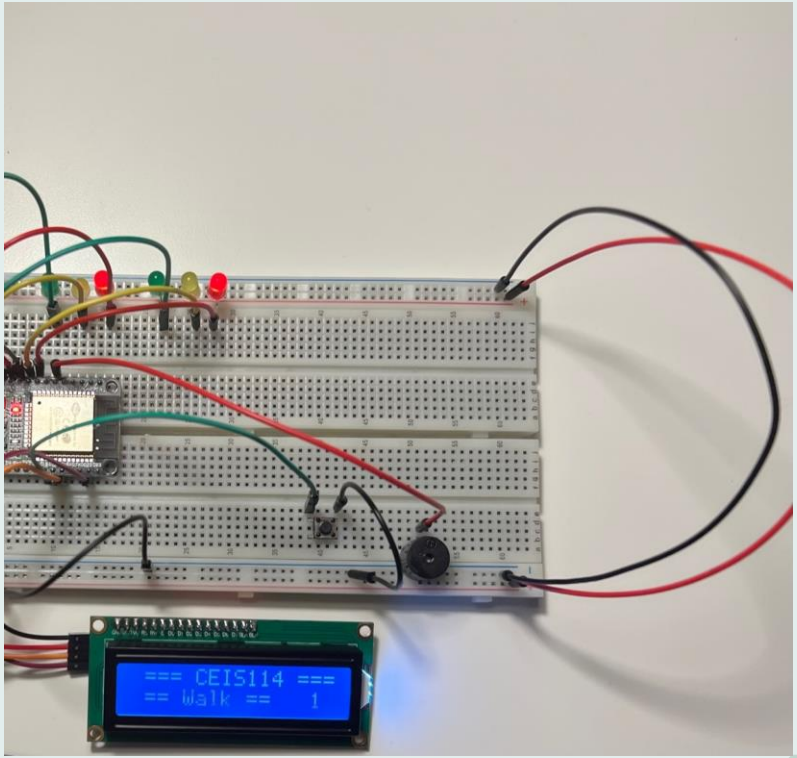


LCD and Buzzer Installation

- Connect LCD, SDA to GPIO21, and SCL to GPIO22
- Connect buzzer to GPIO32
- Display crosswalk output on LCD

LCD Installation

- LCD connected and visible message is displayed
- Crosswalk button active and LEDs are on red while countdown is in progress



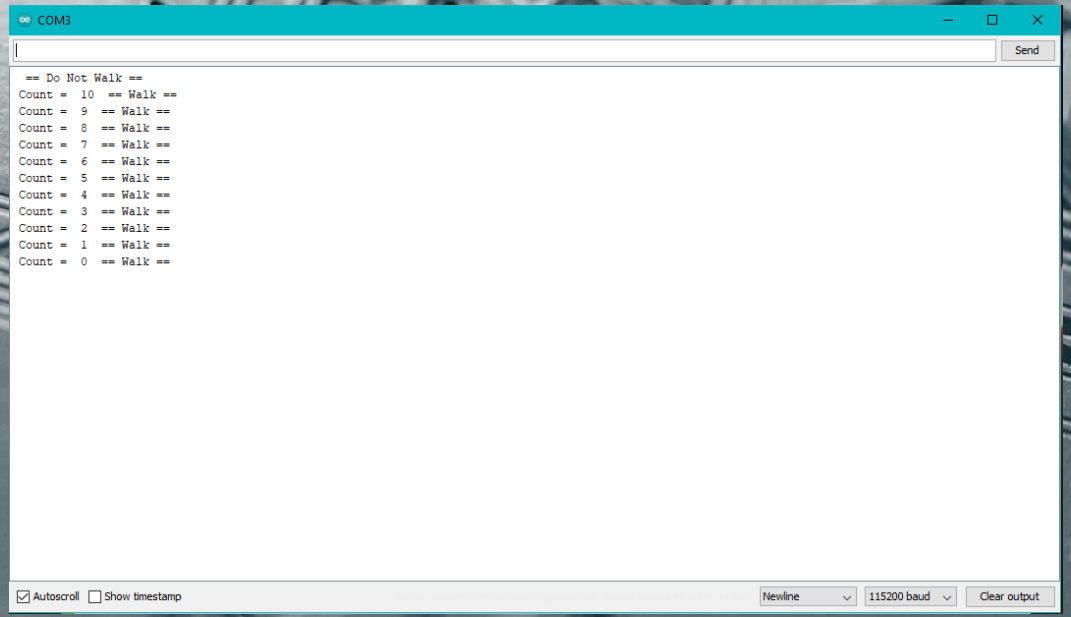
LCD Code

- LCD library LiquidCrystal installed
- Board pin added for buzzer
- Initialized digital pin for buzzer as output
- Altered the loop so that when the crosswalk button is pressed the buzzer sounds and stops once the countdown finishes
- Text now is displayed on the LCD screen

Adruino_Mod6

```
#include <LiquidCrystal_I2C.h>

// === Faith Burnett ====
// 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);

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.
```

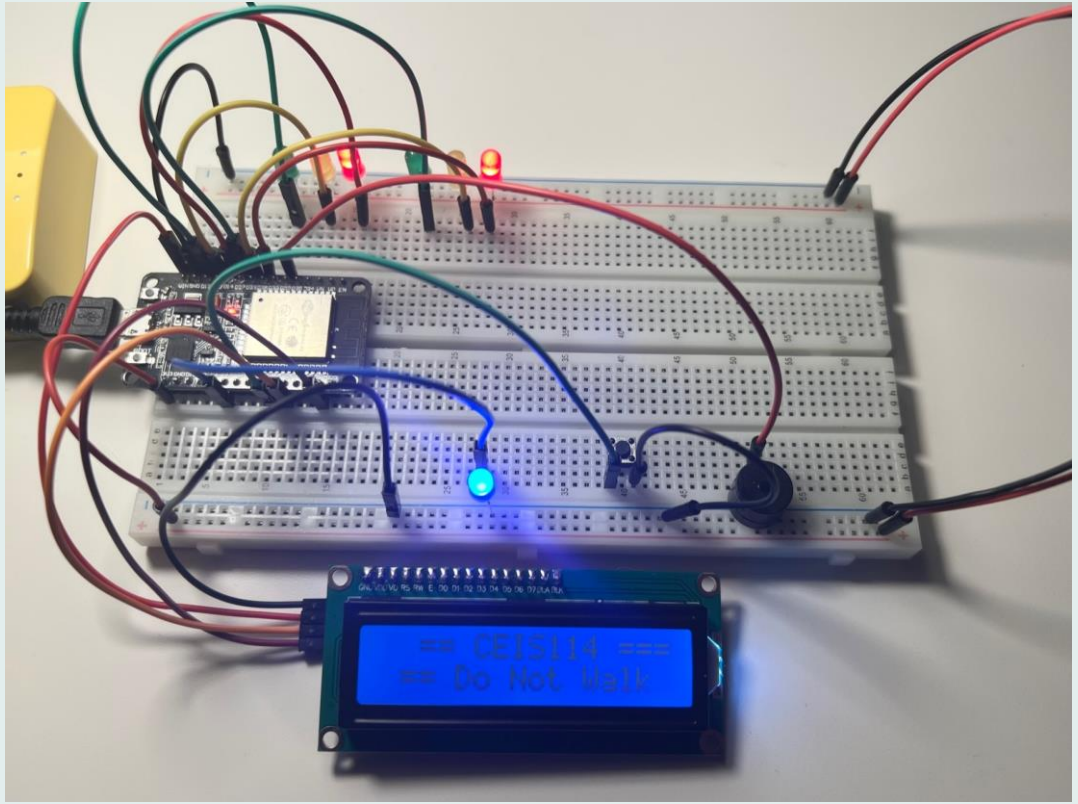
LCD Serial Monitor



Emergency Button Installation

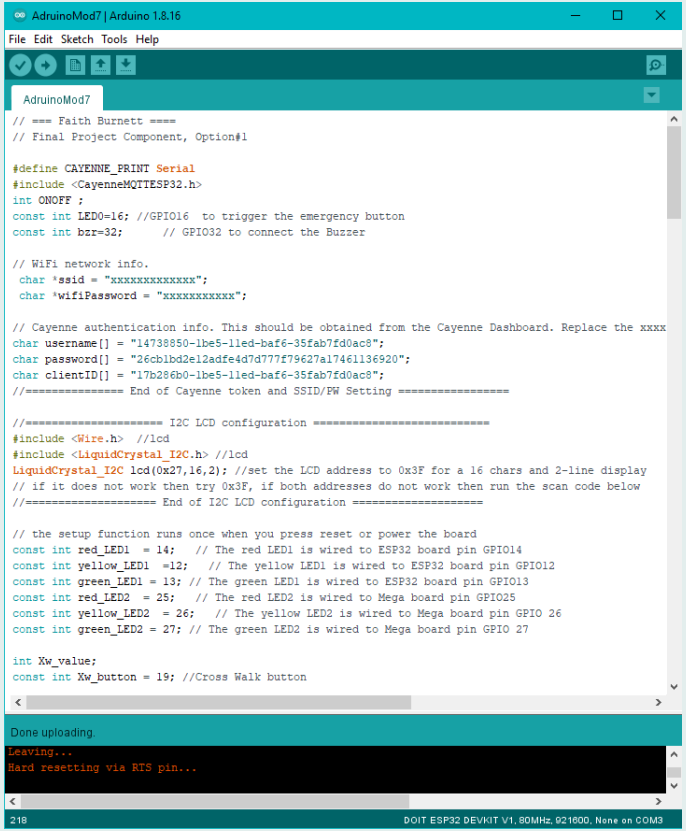
- Install Cayenne-MTQ-ESP library
- Connect emergency LED to GPIO16
- Setup the Mini Smart Router
- Add Wi-Fi network info
- Login to Cayenne and setup your device
- Add your username, password, and clientID to your code

Emergency Button

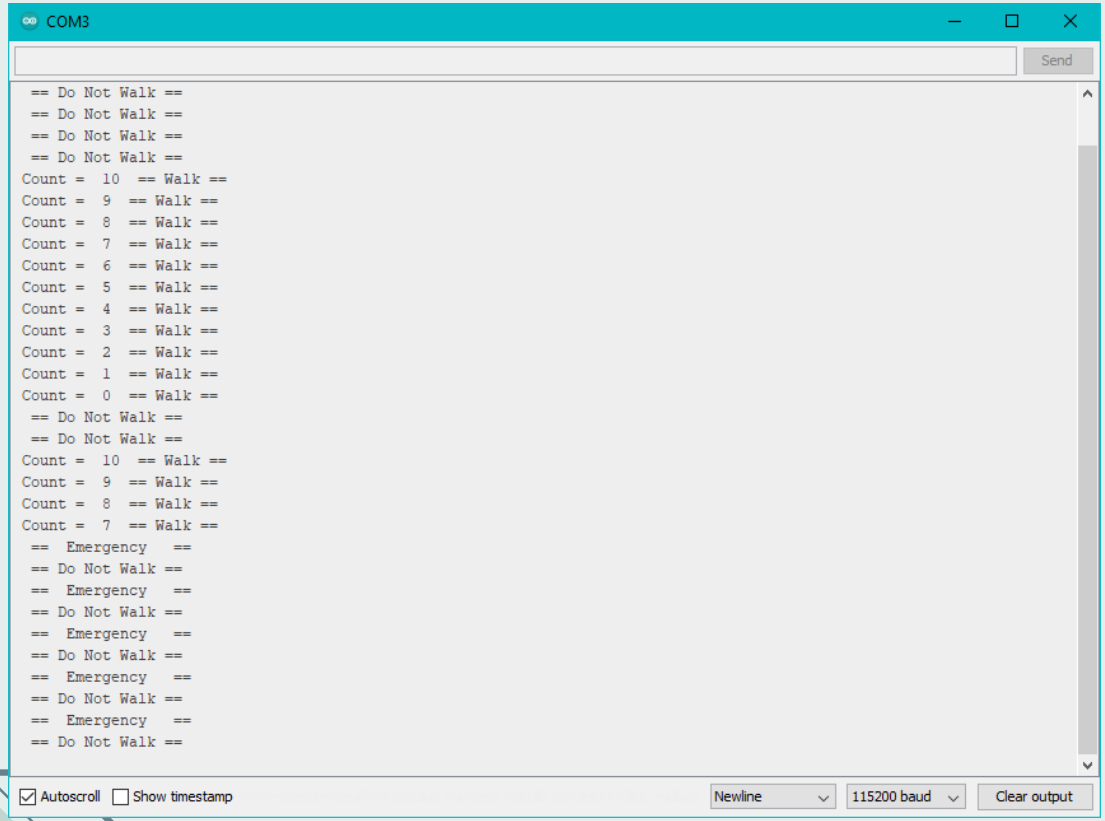


Emergency Button Code

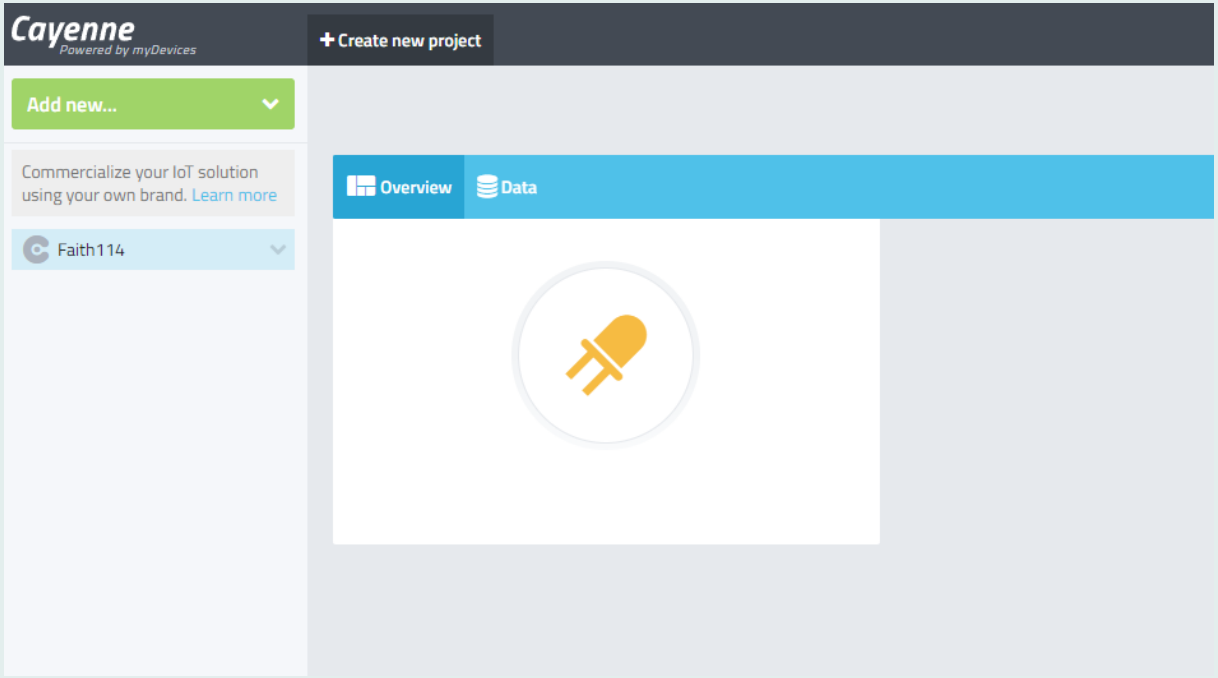
- Cayenne library installed
- Assigned emergency button to board pin
- Wi-Fi setup
- Cayenne authentication completed
- Once emergency button is pressed both sets of LEDs will go to red, the buzzer sounds, the emergency light flashes, and Emergency Do Not Walk is displayed to LCD
- If emergency button is pressed while crosswalk countdown is in progress the countdown will stop and emergency operations proceed



Serial Monitor for Emergency Button



Remote Emergency Button



Conclusion

- Implemented a two-way traffic light controller with pedestrian crossing
- Performed testing to ensure device was functioning properly
- Wrote and altered code to add more functionality
- Setup a remote emergency button to simulate emergency personnel control over device

Challenges

- Making sure that the pin connection was the same on the breadboard as in my code
- I had several LEDs that stopped working in the middle of running my code in several portions of the project
- Setting up the remote connection for the emergency button

Skills Learned

- Problem solving
- Communication
- Components of digital devices
- Basic logic circuit building blocks utilized in digital systems
- Understanding of networking and securing digital devices

THANKS!

Do you have any questions?
faithburnett@outlook.com