## 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.



#### Auto F Archiv WiFiScan /\* \* This sket \* The API i \* the most \*/ finclude "WiF Board:

Serial.be // Set Wi WiFi.mode WiFi.disc delay(100

Serial.pr

#### void loop()

r

Auto Format Ctrl+T   Archive Sketch Fix Encoding & Reload   Fix Encoding & Reload The second of	0013	i i cip								
Archive Sketch   Fix Encoding & Reload   Manage Libraries Ctrl+ Shift+I   Serial Monitor Ctrl+ Shift+M   Serial Plotter Ctrl+ Shift+L   WiFi101 / WiFiNINA Firmware Updater   Board: "DOIT ESP32 DEVKIT V1"   Upload Speed: "115200"   Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Programmer   Burn Bootloader		Auto Format	Ctrl+T							
Fix Encoding & Reload   Manage Libraries   Ctrl+Shift+I   Serial Monitor   Ctrl+Shift+M   Serial Plotter   Ctrl+Shift+L   WiFi101 / WiFiNINA Firmware Updater   Board: "DOIT ESP32 DEVKIT V1"   Upload Speed: "115200"   Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Programmer   Burn Bootloader		Archive Sketch								
Manage Libraries Ctrl+ Shift+1   Serial Monitor Ctrl+ Shift+M   Serial Plotter Ctrl+ Shift+L   WiFi101 / WiFiNINA Firmware Updater   Board: "DOIT ESP32 DEVKIT V1"   Upload Speed: "115200"   Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Programmer   Burn Bootloader		Fix Encoding & Reload								
Serial Monitor Ctrl+Shift+M   Serial Plotter Ctrl+Shift+L   WiFi101 / WiFiNINA Firmware Updater   Board: "DOIT ESP32 DEVKIT V1"   Upload Speed: "115200"   Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Programmer   Burn Bootloader		Manage Libraries	Ctrl+Shift+I							
Serial Plotter       Ctrl+Shift+L       Ineed to include:         WiFi101 / WiFiNINA Firmware Updater       need to include:         Board: "DOIT ESP32 DEVKIT V1"       >         Upload Speed: "115200"       >         Flash Frequency: "80MHz"       >         Core Debug Level: "None"       >         Port: "COM3"       Serial ports         Get Board Info       ✓         Programmer       >         Burn Bootloader       >		Serial Monitor	Ctrl+Shift+M		r					
WiFi101 / WiFiNINA Firmware Updater   Board: "DOIT ESP32 DEVKIT V1"   Upload Speed: "115200"   Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Programmer   Burn Bootloader		Serial Plotter	Ctrl+Shift+L		ne	ed	to inc	luo	de:	
Board: "DOIT ESP32 DEVKIT V1" > Upload Speed: "115200" > Flash Frequency: "80MHz" > Core Debug Level: "None" > Port: "COM3" Serial ports Get Board Info Programmer Sur COM3		WiFi101 / WiFiNINA Firmware Updater								
Upload Speed: "115200"   Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Get Board Info   Programmer   Burn Bootloader		Board: "DOIT ESP32 DEVKIT V1"	>	,						
Flash Frequency: "80MHz"   Core Debug Level: "None"   Port: "COM3"   Get Board Info   Programmer   Burn Bootloader		Upload Speed: "115200"	>	•						
Core Debug Level: "None"		Flash Frequency: "80MHz"	>	•						
Port: "COM3"       Serial ports         Get Board Info       COM3         Programmer       >         Burn Bootloader       >		Core Debug Level: "None"	>	•				_	101.7	
Get Board Info COM3 Programmer Burn Bootloader		Port: "COM3"	;			S	erial ports		лэтү	001
Programmer > Burn Bootloader		Get Board Info			~	С	OM3			
Burn Bootloader		Programmer	>	,						
		Burn Bootloader								

### ESP32 WiFi Scan

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

— —	
	Send
5: BearKave (-85)*	^
6: 611SecondAveWest (-87)*	
7: BearKave eero (-87)*	
8: Brusko (-87)*	
9: DIRECT-68-HP OfficeJet 5200 (-90)*	
scan start	
scan done	
5 networks found	
1: DIRECT-24-HP ENVY 4520 series (-49)*	
2: MyCharterWiFil8-2G (-55)*	
3: BearKave eero (-84)*	
4: BearKave (-84)*	
5: 611SecondAveWest (-91)*	
	×
Autoscroll Show timestamp Newline View 115200 baud View State Stat	Clear output
<pre>Serial.println("Setup done");</pre>	
}	
void loop()	
{	
<pre>Serial.println("scan start");</pre>	
// WiFi.scanNetworks will return the number of networks found	
<pre>int n = WiFi.scanNetworks();</pre>	
<pre>Serial.println("scan done");</pre>	

### 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 GPI014
const int yellow_LED1 = 12; // The yellow LED1 is wired to ESP32 board pin GPI012
const int green_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPI013
```

```
// 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:





### Screenshot of Code in Arduino IDE

File Edit Sketch Tools Help



// Module #4 project

// Define some labels

const int red\_LED1 = 14; // The red LED1 is wired to ESP32 board pin GPI014 const int yellow\_LED1 =12; // The yellow LED1 is wired to ESP32 board pin GPI012 const int green\_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPI013 const int red\_LED2 = 25; // The red LED2 is wired to Mega board pin GPI025 const int yellow\_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPI0 26 const int green\_LED2 = 27; // The green LED2 is wired to Mega board pin GPI0 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 GPIO26 (yellow LED2) as an output.

1

### 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 GPI014 const int yellow\_LED1 =12; // The yellow LED1 is wired to ESP32 board pin GPI012 const int green\_LED1 = 13; // The green LED1 is wired to ESP32 board pin GPI013 const int red\_LED2 = 25; // The red LED2 is wired to Mega board pin GPI025 const int yellow\_LED2 = 26; // The yellow LED2 is wired to Mega board pin GPI0 26 const int green\_LED2 = 27; // The green LED2 is wired to Mega board pin GPI0 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() {

pinHode (Xw\_button, INPOT\_FOLLOP); // 0=pressed, 1 = unpressed button
Serial.begin(115200);
pinHode(red\_LED1, COTFOT); // initialize digital pin 14 (Red LED1) as an output.
pinHode(yellow\_LED1, COTFOT); // initialize digital pin 12 (yellow LED1) as an output.
pinHode(green\_LED1, COTFOT); // initialize digital pin 13 (green LED1) as an output.

pinMode(red\_LED2, COTPUT); // initialize digital pin 25(Red LED2) as an output. pinMode(yellow\_LED2, COTPUT); // initialize digital pin 26 (yellow LED2) as an output. pinMode(green\_LED2, COTPUT); // initialize digital pin 27 (green LED2) as an output.

### Screenshot of Serial Monitor in Arduino IDE

Screenshot of output in Serial Monitor

#### COM3

== 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 ==
== Do Not Walk ==
== Do Not Walk ==
== Do Not Walk ==
== Do Not Walk ==
Autoscroll 🔲 Show timestamp
inMode (Xw_button, INPUT PULLUP); // 0=pressed, 1 = unpressed button
erial.begin(115200);
inMode(red_LED1, OUTPUT); // initialize digital pin 14 (Red LED1) as
inMode(yellow_LED1, OUTPUT); // initialize digital pin 12 (yellow LES
inMode(green_LED1, OUTPUT); // initialize digital pin 13 (green LED

### 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

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);
```



### 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

sketch jul22b sketch jul19a

// === James Garlie ==== // Final Project Component, Option#1

//#define CAYENNE DEBUG #define CAYENNE PRINT Serial #include <CayenneMQTTESP32.h> int ONOFF ; const int LED0=16;//GPI016 to trigger the emergency button // WiFi network info.

char \*ssid = "Relpace with your router's SSID"; char \*wifiPassword = " Relpace with your router's password";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard. Replace the xxxxxxxxx //=------

#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; // GPI032 to connect the Buzzer

//======= LCD =================

// the setup function runs once when you press reset or power the board

bnst int red LED1 = 14; // The red LED1 is wired to ESP32 board pin GPI014

### Screenshot of Serial Monitor in Arduino IDE

Screenshot of output in Serial Monitor 💿 сомз

== Do Not Walk == ets Jun 8 2016 00:22:57

rst:0x1 (POWERON\_RESET),boot:0x13 (SPI\_FAST\_FLASH\_BOOT)
configsip: 0, SPIWP:0xee
clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
== Do Not Walk ==
== Do Not Walk ==

### 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.