**Course Project – Module 6**

**College of Engineering and Information Sciences**

**Course Number: CEIS114**

# **Multiple Traffic Light Controller with Cross Walk, Buzzer, and LCD Display**

# Objective

Setup ESP32 Board with Two Traffic Lights and a Cross Walk Option along with Buzzer

# PARTS LIST

* PC running Arduino IDE
* ESP 32 Board
* Two sets of Colored LEDs: Red, Yellow and Green
* Button
* LCD Unit
* Buzzer
* Wires
* Breadboard

**Software Library:** LiquidCrystal\_I2C-master – for LCD Display

Deliverables

* Complete the Course Project PowerPoint Deliverable
* Include a picture of your circuit
* Screenshot of Arduino IDE code from your computer
* Screenshot of output in Serial Monitor from your computer

# PROCEDURE

1. Install the **LiquidCrystal\_I2C**-master Library in the Arduino IDE using the procedure identified below:
   1. Download the LiquidCrystal\_I2C-master Library. The file will be provided in the project page.
   2. Under Sketch, Include Library, Add.Zip Library as shown in Figure 1.

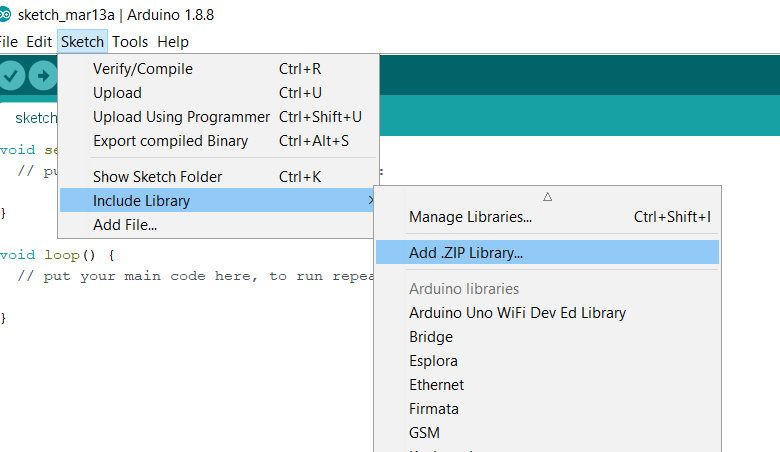


Figure 1

* 1. Select our Library location as shown in figure 2.

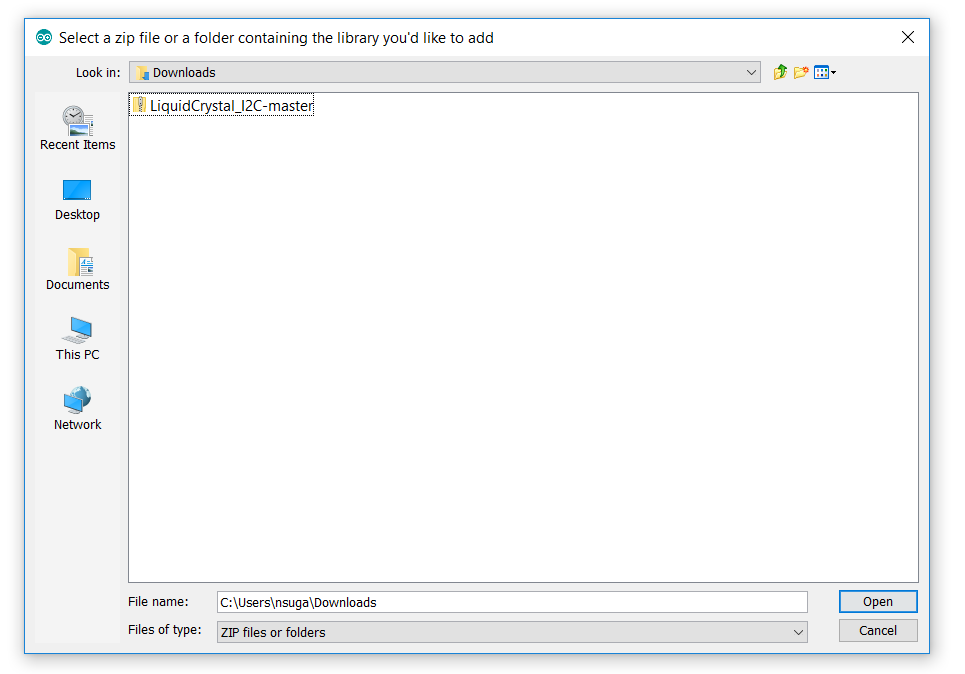


Figure 2

* 1. Select Open to add the Library. You should see the “*Library Added*” message in the bottom of the screen.

**Note: Please see additional information on setting up LCD panels in Appendix A.**

1. Assemble the circuit as shown in Figure 3
2. Connect all the wires as shown in Figure 3. Make sure to connect the Red, Yellow, and Green wires to the appropriate pins in the ESP32 Board.
3. Open the Arduino IDE and load the code shown in Figure 4
4. Connect the ESP32 Board to your computer and Check if the board is recognized. Make sure that the port and the board are correctly selected. Also, make sure the upload speed is set to 115200 in the Tools menu
5. Run the code and observe the output in the serial monitor
6. Observe the output in the LCD Panel and Buzzer
7. Press the wire for activate the cross walk and observe the output

**Figure 3: Multiple Traffic Lights with Crosswalk, buzzer, and LCD display**

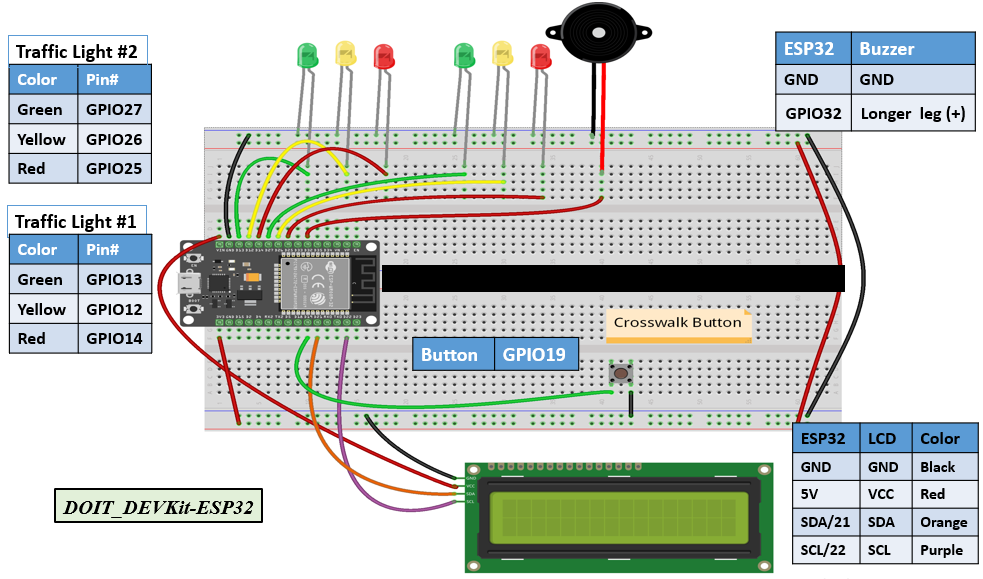


Figure 4. Code for Two Traffic Lights, Crosswalk, buzzer, and LCD display

***// === Replace this text with your Name ====***

***// 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.*

*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 == ");*

*lcd.setCursor(0,1); // set the cursor to column 1, line 2*

*// lcd.clear(); // clears the display to print new message*

*lcd.print(" ");*

*lcd.setCursor(0,1); // set the cursor to column 1, line 2*

*lcd.print(" == Walk == "); //* ***Walk*** *characters to the LCD.*

*lcd.print(i); // Print the count to the LCD*

*digitalWrite(red\_LED1, HIGH); // This should turn on the RED LED1*

*digitalWrite(red\_LED2, HIGH); // This should turn on the RED LED2*

*digitalWrite(bzr, HIGH);*

*delay(500);*

*digitalWrite(red\_LED1, LOW); // This should turn off the RED LED1*

*digitalWrite(red\_LED2, LOW); // This should turn off the RED LED2*

*digitalWrite(bzr, LOW);*

*delay(500);*

*} // End of counter*

*// clears the display to print new message*

*// ===== lcd.clear();*

*lcd.setCursor(0,1); // set the cursor to column 1, line 2*

*lcd.print(" ");*

*} //*

*else //* ***No Emergency ===***

*{*

*lcd.setCursor(0,1); // set the cursor to column 1, line 2*

*lcd.print(" = Do Not Walk ="); //* ***Do Not Walk*** *characters to the LCD.*

*Serial.println(" == Do Not Walk == ");*

*// 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 red LED2*

*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 1 second*

*// 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 red LED2*

*digitalWrite(red\_LED2, LOW); // This should turn on the RED LED2*

*digitalWrite(yellow\_LED2 , HIGH); // This should turn off the YELLOW LED2*

*digitalWrite(green\_LED2, LOW); // This should turn off the GREEN LED2*

*delay(2000); // wait for 1 second*

*// The next three lines of code turn on the red LED2*

*digitalWrite(red\_LED2, HIGH); // This should turn on the RED LED2*

*digitalWrite(yellow\_LED2 , LOW); // This should turn off the YELLOW LED2*

*digitalWrite(green\_LED2, LOW); // This should turn off the GREEN LED2*

*delay(1000); //Extended time for Red light#2 before the Green of the other side turns ON*

*// 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 , LOW); // This should turn off the YELLOW LED1*

*digitalWrite(green\_LED1, HIGH); // This should turn off the GREEN LED1*

*delay(2000); // wait for 1 second*

*// 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*

*// The next three lines of code turn on the red LED2*

*digitalWrite(red\_LED2, HIGH); // This should turn on the RED LED2*

*digitalWrite(yellow\_LED2 , LOW); // This should turn off the YELLOW LED2*

*digitalWrite(green\_LED2, LOW); // This should turn off the GREEN LED2*

*delay(2000); // wait for 1 second*

*}// Emergency Button closing ============*

*}*

# **Appendix A**

Setting up the I2C-LCD in Arduino IDE

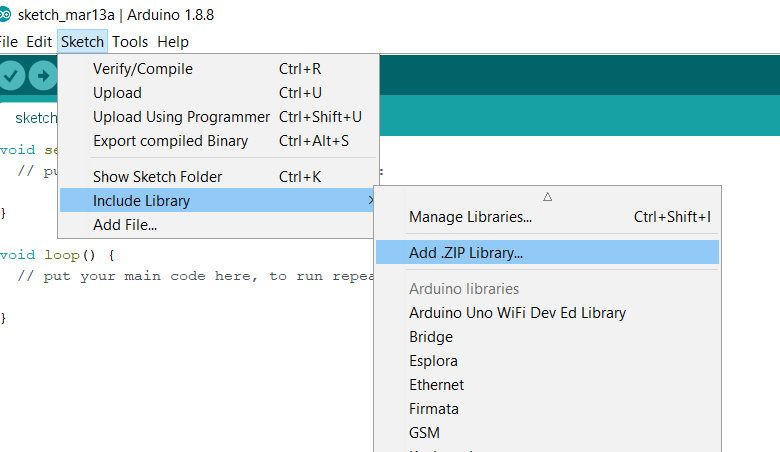
**I2C LCD library installation**

Before we can use the I2C LCD display with the ESP32, we need to add the I2C LCD library to the Arduino library. This library has many built-in functions that make programming the LCD quite easy. Make sure that you have this exact library installed and delete any other libraries that have the same name.

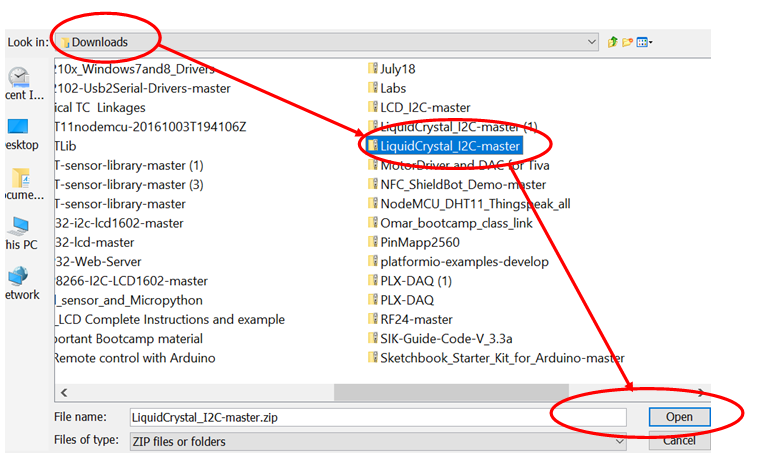
To install the LiquidCrystal\_I2C-master Library in the Arduino IDE use the procedure below:

1. From Canvas, **Download** the I2C master library (LiquidCrystal\_I2C-master):
2. From the Arduino’s IDE go to:

Sketch ==> Include Library ==> Add .ZIP Library



1. Now browse to where you downloaded the zipped library and select it (**see next image**)



1. Press Open and the library will be added.

# **Appendix B**

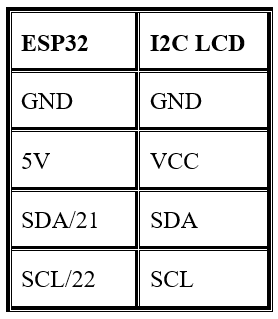
**Interfacing the I2C LCD to the ESP32 board**

The I2C LCD consist of 2 parts usually a controller **HD44780** and display 16×2 LCD and an I2C backpack which connects to the LCD exposing the standard power and I2C pins. See the pictures below



*A circuit board

Description automatically generated*



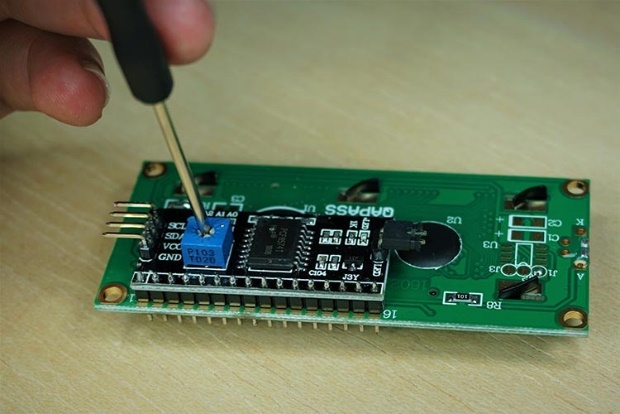
Use the table shown to make the connections:

## Appendix C

## Troubleshooting your LCD

## If the LCD does not display sent messages then check the following troubleshooting options

1. Double check your LCD signal wirings
2. The LCD comes with a built-in potentiometer that you can use to adjust the contrast between the background and the characters on the LCD (see picture below)

**

1. Try a different I2C address as indicated in the code above. If not successful, copy, paste and upload the I2C scanner code to display the exact I2C address in the serial monitor (see code below)

*// This sketch tests the standard 7-bit addresses*

*#include <Wire.h>*

*void setup()*

*{*

*Wire.begin();*

*Serial.begin(9600);*

*Serial.println("\nI2C Scanner");*

*}*

*void loop()*

*{*

*byte error, address;*

*int nDevices;*

*Serial.println("Scanning...");*

*nDevices = 0;*

*for(address = 1; address < 127; address++ )*

*{*

*// The i2c\_scanner uses the return value of the Write.endTransmisstion*

*// to see if a device did acknowledge to the address.*

*Wire.beginTransmission(address);*

*error = Wire.endTransmission();*

*if (error == 0)*

*{*

*Serial.print("I2C device found at address 0x");*

*if (address<16)*

*Serial.print("0");*

*Serial.print(address,HEX);*

*Serial.println(" !");*

*nDevices++;*

*}*

*else if (error==4)*

*{*

*Serial.print("Unknow error at address 0x");*

*if (address<16)*

*Serial.print("0");*

*Serial.println(address,HEX);*

*}*

*}*

*if (nDevices == 0)*

*Serial.println("No I2C devices found\n");*

*else*

*Serial.println("done\n");*

*delay(5000); // wait 5 seconds for next scan*

*}*