**Course Project**   
DeVry University   
College of Engineering and Information Sciences

**Course Number: CEIS114**

**Final Project: Multiple Traffic Light Controller with Cross Walk, Buzzer, and LCD Display with IOT Access**

# Background

## The development of modern cities and urban areas has brought about an ever-increasing need for efficient traffic management systems. Ensuring the smooth flow of vehicles and pedestrian traffic at intersections is crucial not only for reducing congestion but also for enhancing safety and minimizing environmental impacts. In response to these challenges, the design and implementation of intelligent traffic light systems have become a key focus in the field of transportation engineering. This project aims to provide students with hands-on experience in creating a Multi-Intersection Traffic Light System using Wokwi Lab, offering them a valuable opportunity to explore the intersection of electronics, microcontroller programming, and traffic engineering. By delving into this project, students will gain insights into the principles behind traffic signal control, as well as practical skills that can contribute to more efficient and sustainable urban transportation systems.

## **Introduction**

Imagine a world where your things, like lights, coffee makers, and even plants, can talk to each other through the internet and do cool stuff. For example, your lights could change colors based on your mood, your coffee maker could start brewing when you wake up, and your plants could send you messages when they need water. This amazing world is called the Internet of Things (IoT).

But don't worry if all these big words sound complicated. You don't need to be a super skilled programmer to make your own IoT system now. This final step of your multi-intersection traffic light system will help you enter the world of connected devices and teach you how to create your own simple IoT system without having to deal with difficult code.

We'll be using something called the MQTT protocol, which is like an easy-to-understand language that your devices can use to talk to each other. It's like giving them a common way to chat and share information without any trouble.

# Deliverables

* Complete the Course Project PowerPoint Deliverable
* Screenshot of your Circuit Simulation from Wokwi
* One picture should show LCD with message “DoNot Walk!”.
* The other should show LCD with message “Emergency”
* Screenshot of Code Simulation from Wokwi
* Screenshot of output in Serial Monitor from your computer

# Parts List

* ESP32 Board in Wokwi
* Two sets of Colored LEDs: Red, Yellow and Green in Wokwi
* Breadboard in Wokwi
* Button in Wokwi
* Wires in Wokwi
* LCD Unit
* Buzzer
* One Blue LED in Wokwi

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

# Procedures

## **Start with Wokwi**

1. Open Wokwi.com – Login In – Select **My Projects** under Account on the top right corner.
2. Open your **Module 6 project** and click on the **Save Dropdown** to **Save a Copy** to start the Final Project – Save it as – **Final Project – YourFirstNameLastName**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Install LiquidCrystal Library**

1. Here we will install the **pubsubclient library** and the **WIFI library** in Wokwi
2. Click the Library Manager in Wokwi Dashboard  
     
   A screenshot of a computer

   Description automatically generated
   1. Click the A blue circle with a plus symbol

      Description automatically generated Add Button
   2. Type **pubsubclient** and select it to install the library
   3. Type WiFi and select it to install the library
      1. You should have 3 libraries installed (see below)

A screenshot of a computer

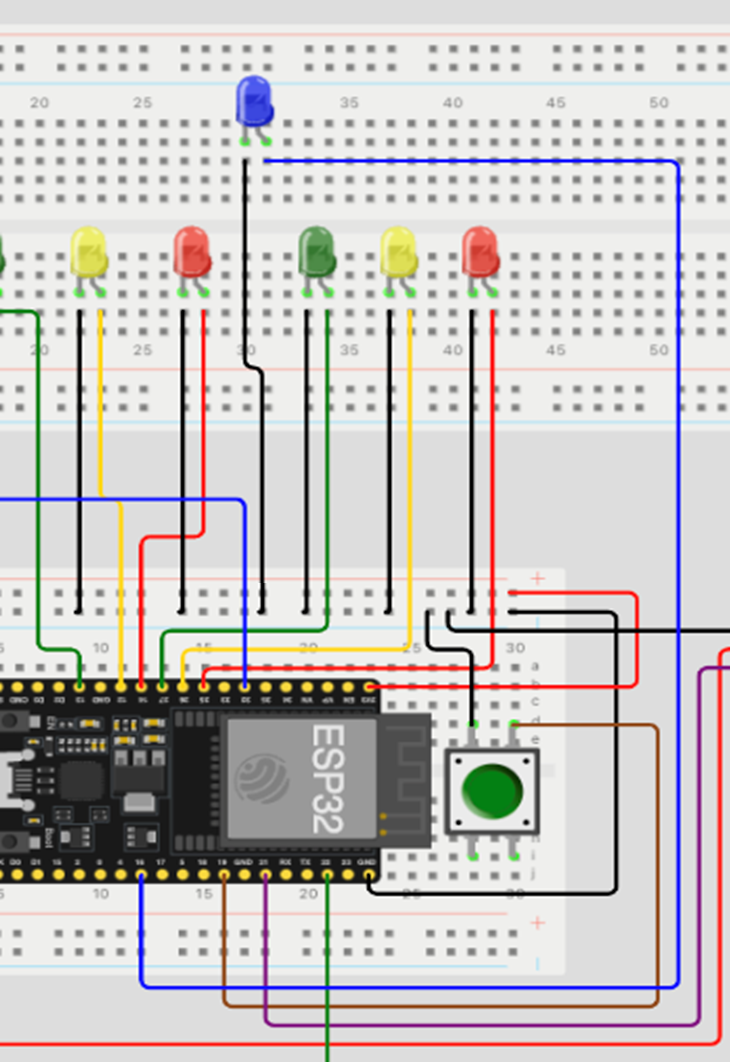
Description automatically generated

* 1. Return to the code editor by clicking on **sketch.ino**

**Add components**

1. Here we will add a blue LED to our multi-intersection traffic light.
2. Click the A blue circle with a white cross

   Description automatically generated in the Wokwi Dashboard
   1. Type LED
3. Move the LED to the Full Breadboard as seen in the image below and change the color to Blue using the color chart.
4. Connect the blue LED to the digital pins on the ESP32 Pin 16 and – (GND) ground on the breadboard.

**Installing IOT MQTT Panel**

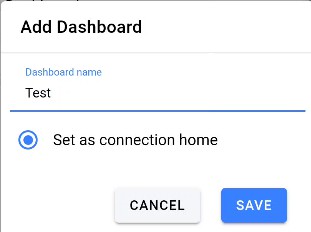
1. Download app IoT MQTT Panel
2. Open your browser on your smartphone or computer
3. Copy and paste the following link: [https://napkforpc.com/apk/snr.lab.iotmqttpanel.prod/](about:blank)
4. Scroll down and choose the platform you're using (Android, iOS, or PC).
5. Follow the installation instructions for your chosen platform

**Setting up IOT MQTT Panel**

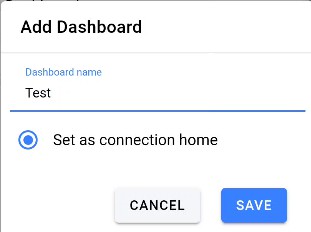
1. Open the IoT MQTT Panel app.

A screenshot of a computer

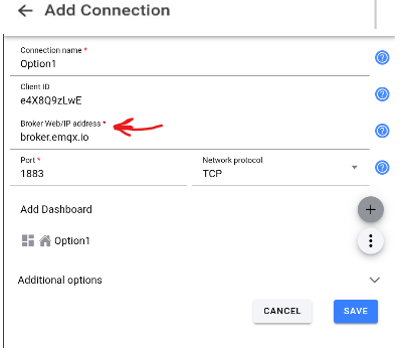
Description automatically generated

1. Connect to your MQTT broker:
   1. Click “Setup Connection
2. Click the **Connection name field**. Type a name for your connection.
   1. For example: Demo
3. Click the **Broker Web/IP address** field.

Enter the following URL:  **broker.emqx.io**

1. Click the **+** icon to **add a dashboard**.
   1. A window will pop up allowing you to give your dashboard a name. For example: **Test**
   2. To complete the creation of the dashboard, click the **SAVE** button.

To add the connection, click the **CREATE** button



# A screenshot of a computer Description automatically generatedAfter a short while, and if everything was done properly, you should see what looks like a cloud with a check mark on it. A screenshot of a computer Description automatically generatedClick Test (Demo) to add a panel to the dashboard.

# Adding a couple buttons to turn on and off the LED

# Click the Button icon A screenshot of a device Description automatically generated

# In the Panel name field, type Emergency On

# In the Topic field, type LED

# In the Payload field, type ON

# Press Create to save the panel A screenshot of a computer Description automatically generated

* + 1. **Press the + button** again, repeat the same process.
       1. This time, you will name the Panel: **Emergency OFF.**
       2. The **topic** will be **LED**, and the **Payload** will be **OFF**.
       3. Press the **CREATE** button  
            
          A screenshot of a phone

          Description automatically generated

# Your dashboard should be similar to the image below.

# A white background with red and black text Description automatically generated

If you see the message **“Failed to connect”** at the bottom of the dashboard, ignore it.

# Code

1. Now, that we have all the components connected time to add the code to test our Multi Intersection Traffic light with the Crosswalk, Buzzer, LCD display, and LED.
2. Delete the code currently in the Wokwi Code Editor.
3. [Copy the code](#_CODE) from the [next page](#_CODE) into the Wokwi Code Editor.

**Note:** If you did not use the pins configuration suggested in the instructions you will need to modify the code to use the GPIO pins you've connected the components to.

# Run the simulation

* 1. Click the “Start the Simulation” A green and white play button

     Description automatically generated button in Wokwi to start the simulation.
  2. Observe the traffic lights changing based on your code's logic.
  3. Observe the output in the LCD Panel.
  4. Press the button Emergency ON in the IoT MQTT Panel App
     + If everything was successful, you should now be able to trigger an emergency traffic light pattern.
  5. Press the button Emergency OFF.
     + You should also be able to switch to the normal operation of the traffic light
  6. Click the crosswalk button.
     + Does not need to be pushed for an extended period.
     + The controller will record your request and switch to the crossing pattern once the current traffic cycle completes.

Congratulations. You have just succeeded in building your first IoT system. You are using your phone to control a device over an internet connection.

**A screenshot of a computer

Description automatically generatedFigure 2 – Simulation of Multi-Intersection Traffic Light with Crosswalk, buzzer, and LCD display and IoT MQTT**

## **Code for Multi-Intersection Traffic Light with Crosswalk, buzzer, and LCD display and IoT MQTT**

## **CODE**

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

// Final Project Component, Option 1

#include <WiFi.h> // WiFi header file

#include <PubSubClient.h> // MQTT publish and subscribe header file

#include <Wire.h> // I2C header file

#include <LiquidCrystal\_I2C.h> // I2C lcd header file

const char\* ssid = "Wokwi-GUEST"; // This is the access point to your wireless network.

const char\* password = ""; // This is the password to the SSID. For the smart mini router

const char\* mqttServer = "**broker.emqx.io**"; // This is the free MQTT broker we will use.

int port = 1883; // MQTT brokers listen to port 1883 by default

String stMac; // C string used for convenience of comparisons.

char mac[50]; // C char array used to hold the MAC address of your ESP32 microconroller

char clientId[50]; // This client ID is used to identify the user accessing the MQTT broker.

// For our test.mosquitto.org broker, we just generate a random user client ID

WiFiClient espClient; // instantiate the WiFi client object

PubSubClient client(espClient); // instantiate the publish subscribe client object

LiquidCrystal\_I2C lcd(0x27,16,2); //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

const int redLightNorthSouth = 14; // The red LED NS is wired to ESP32 board pin GPIO 14

const int yellowLightNorthSouth = 12; // The yellow LED NS is wired to ESP32 board pin GPIO 12

const int greenLightNorthSouth = 13; // The green LED NS is wired to ESP32 board pin GPIO 13

const int redLightEastWest = 25; // The red LED EW is wired to ESP32 pin GPIO 25

const int yellowLightEastWest = 26; // The yellow LED EW is wired to ESP32 board pin GPIO 26

const int greenLightEastWest = 27; // The green LED EW is wired to ESP32 board pin GPIO 27

int crossWalkButtonState = 1 ; // Variable will store the state of the crosswalk button

const int crossWalkButton = 19; // Cross Walk button pin is GPIO 19

const int emergencyBlueLED = 16; // The blue LED is wired to ESP32 board pin GPIO 16

const int buzzerPin = 32; // Active Buzzer pin is GPIO 32

int loopCount; // Variable will keep count of the number of times the light pattern repeats

int secondsLeft; // counter to keep track of number of seconds left for crossing intersection

int iotControl = 0; // Variable will be used to switch between emergency and normal operations of

// traffic controller

void setup() {

**Serial**.begin(115200); // set baud rate of serial monitor to 115200 bits per second

randomSeed(analogRead(0)); // seed the random() function

delay(10); // wait 10 milliseconds

**Serial**.println(); // start by skipping a line on the serial monitor

**Serial**.print("Connecting to "); // display message connection to

**Serial**.println(ssid); // display SSID name

wifiConnect(); // initiate connection to WiFi access point

**Serial**.println(""); // Skip a line on serial monitor

**Serial**.println("WiFi connected"); // display message WiFi connected

**Serial**.println("IP address: "); // display message IP Address

**Serial**.println(WiFi.localIP()); // display local IP address

**Serial**.println(WiFi.macAddress()); // display MAC address

stMac = WiFi.macAddress(); // copy MAC address

stMac.replace(":", "\_"); // replace the underscores with colons

**Serial**.println(stMac); // display the cleaned up MAC address

client.setServer(mqttServer, port); // Connect to MQTT server test.mosquitto.org

client.setCallback(callback); // set callback function for handling MQTT messages

pinMode(crossWalkButton, INPUT\_PULLUP); // 0 = button is pressed, 1 = button not pressed

attachInterrupt(crossWalkButton, buttonPressed, FALLING); // set up crosswalk button to trigger

// interrupts on falling edges

pinMode(redLightNorthSouth, OUTPUT); // initialize digital pin 14 (Red LED NS) as an output.

pinMode(yellowLightNorthSouth, OUTPUT); // initialize digital pin 12 (Yellow LED NS) as an output.

pinMode(greenLightNorthSouth, OUTPUT); // initialize digital pin 13 (green LED NS) as an output.

pinMode(redLightEastWest, OUTPUT); // initialize digital pin 25 (Red LED EW) as an output.

pinMode(yellowLightEastWest, OUTPUT); // initialize digital pin 26 (Yellow LED EW) as an output.

pinMode(greenLightEastWest, OUTPUT); // initialize digital pin 27 (Green LED EW) as an output.

pinMode(emergencyBlueLED, OUTPUT); // initialize digital pin 16 (Blue emergency LED) as an output.

pinMode(buzzerPin, OUTPUT); // initialize digital pin 32 (Active buzzer) as an output.

lcd.init(); // initialize the lcd

lcd.backlight(); // turn on backlight of lcd

lcd.setCursor(0,0); // set cursor to column # 1 and Row # 1

lcd.print("=== CEIS-114 ==="); // display this message

}

void wifiConnect() {

WiFi.mode(WIFI\_STA); // set WiFi mode to STA

WiFi.begin(ssid, password); // connect WiFi using SSID and password

while (WiFi.status() != WL\_CONNECTED) { // As long as WiFi connection is not established

delay(500); // wait half a second

**Serial**.print("."); // display period on serial monitor

}

}

void mqttReconnect() { // If connection to MQTT broker is lost. Call this function

while (!client.connected()) { // As long as connection is not established

**Serial**.print("Attempting MQTT connection..."); // Display message attempting MQTT connection

long r = random(1000); // Generate a long integer

sprintf(clientId, "clientId-%ld", r); // display client ID

if (client.connect(clientId)) { // Is connection to MQTT broker is established

**Serial**.print(clientId); // Display client ID on serial monitor

**Serial**.println(" connected"); // Display connected message on serial monitor

client.subscribe("LED"); // Subscribe to topic LED.

}

else { // If connection to MQTT broker has failed

**Serial**.print("failed, rc="); // Display failed

**Serial**.print(client.state()); // Display client ID for failed connection

**Serial**.println(" try again in 5 seconds"); // Display message try again in 5 seconds

delay(5000); // Wait 5 seconds

}

}

}

void callback(char\* topic, byte\* message, unsigned int length) { // Callback function for handling MQTT

//messages

String stMessage; // Create C string object

for (int i = 0; i < length; i++) { // Run a loop that will process all the characters of the MQTT message

stMessage += (char)message[i]; // Add character to the C string one at a time

}

if (String(topic) == "LED") { // Check if the topic received is LED

if(stMessage == "ON"){ // Check if the message is ON

iotControl = 1; // Set iotControl variable to high

}

else if(stMessage == "OFF"){ // if the message received is OFF

iotControl = 0; // Set iotControl variable to low

}

}

}

void loop() { // loop() function

if (!client.connected()) { // If the client is not connected to MQTT broker

mqttReconnect(); // Try reconnecting

}

while( iotControl == 0 ) { // As long as variable iotControl is low. Run the normal traffic light

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

lcd.print("= Do Not Walk! ="); // display Walk tmessage to the LCD.

**Serial**.println(" == Do Not Walk == "); // display Do Not Walk message on serial monitor

digitalWrite(emergencyBlueLED, LOW); // This should turn off the Remergency blue LED

if (crossWalkButtonState == 0 ){ // if crosswalk button is pressed

digitalWrite(yellowLightNorthSouth , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightNorthSouth, LOW); // This should turn off the Green LED EW

digitalWrite(yellowLightEastWest , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightEastWest, LOW); // This should turn off the Yellow LED EW

// display Walk message on the LCD

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

lcd.print(" == Walk! == "); // display Walk tmessage to the LCD.

secondsLeft = 15; // Set the delay for crossing the intersection at 15 seconds.

while( secondsLeft > 0 ){ // As long as the timer hasn’t run up yet

if( secondsLeft >= 10){ // If the timer has a value with 2 digits

**Serial**.print(" == Walk == "); // display message Walk on serial monitor

**Serial**.println(secondsLeft); // display timer value

lcd.setCursor(14 ,1); // set the cursor to column 15, line 2

lcd.print(secondsLeft); // display number of seconds left 2

}

else {

**Serial**.print(" == Walk == "); // If timer has a single digit. Line up the numbers

**Serial**.println(secondsLeft); // Display the timer value

lcd.setCursor(14,1); // set the cursor to column 15, line 2

lcd.print(" "); // clear anything on the last 2 columns

lcd.setCursor(15,1); // set the cursor to column 15, line 2

lcd.print(secondsLeft); // display number of seconds left

}

digitalWrite(redLightNorthSouth, HIGH); // This should turn on the RED LED NS

digitalWrite(redLightEastWest, HIGH); // This should turn on the RED Yellow LED NS

delay(500); // wait 0.5 seconds

digitalWrite(redLightNorthSouth, LOW); // This should turn off the Red LED NS

digitalWrite(redLightEastWest, LOW); // This should turn off the Red LED EW

delay(500); // wait 0.5 seconds

secondsLeft -= 1; // Decrement timer count by one second

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

} // end of while() loop

crossWalkButtonState = 1 ; // Set the crosswalk variable high. This should get us out of the cross

// walk routine

} // end of if statement

else {

// The next three lines of code turn on the red LED NS

digitalWrite(redLightNorthSouth, HIGH); // This should turn on the Red LED NS

digitalWrite(yellowLightNorthSouth , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightNorthSouth, LOW); // This should turn off the Green LED NS

digitalWrite(redLightEastWest, HIGH); // This should turn on the Red LED EW

digitalWrite(yellowLightEastWest , LOW); // This should turn off the Yellow LED EW

digitalWrite(greenLightEastWest, LOW); // This should turn off the Green LED EW

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

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

// The next three lines of code turn on the red Yellow LED NS

digitalWrite(redLightEastWest, LOW); // This should turn off the Red LED EW

digitalWrite(yellowLightEastWest , LOW); // This should turn off the Yellow LED EW

digitalWrite(greenLightEastWest, HIGH); // This should turn on the Green LED EW

delay(2000); // wait for 2 seconds

// The next three lines of client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

// code to turn on the red LED NS

digitalWrite(redLightNorthSouth, HIGH); // This should turn on the Red LED NS

digitalWrite(yellowLightNorthSouth , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightNorthSouth, LOW); // This should turn off the Green LED NS

// The next three lines of code turn on the red Yellow LED NS

digitalWrite(redLightEastWest, LOW); // This should turn off the Red LED EW

digitalWrite(yellowLightEastWest , HIGH); // This should turn on the Yellow LED EW

digitalWrite(greenLightEastWest, LOW); // This should turn off the Green LED EW

delay(2000); // wait for 2 seconds

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

// The next three lines of code turn on the red Yellow LED NS

digitalWrite(redLightEastWest, HIGH); // This should turn on the Red LED NS

digitalWrite(yellowLightEastWest , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightEastWest, LOW); // This should turn off the Green LED NS

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

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

// The next three lines of code turn on the yellow LED NS

digitalWrite(redLightNorthSouth, LOW); //This should turn off the Red LED NS

digitalWrite(yellowLightNorthSouth , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightNorthSouth, HIGH); // This should turn on the Green LED NS

delay(2000); // wait for 2 seconds

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

// The next three lines of code turn on the yellow LED NS

digitalWrite(redLightNorthSouth, LOW); // This should turn off the Red LED NS

digitalWrite(yellowLightNorthSouth , HIGH); // This should turn on the Yellow LED NS

digitalWrite(greenLightNorthSouth, LOW); // This should turn off the Green LED NS

// The next three lines of code turn on the red Yellow LED NS

digitalWrite(redLightEastWest, HIGH); // This should turn on the Red LED EW

digitalWrite(yellowLightEastWest , LOW); // This should turn off the Yellow LED EW

digitalWrite(greenLightEastWest, LOW); // This should turn off the Green LED EW

delay(2000); // wait for 2 seconds

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1){ // If iotControl variable is high.

break; // Break out of the while() loop

}

}

} // end of while() loop

while( iotControl == 1){

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

lcd.println("= Emergency! =");

**Serial**.println("= Emergency! =");

digitalWrite(yellowLightNorthSouth , LOW); // This should turn off the Yellow LED NS

digitalWrite(greenLightNorthSouth, LOW); // This should turn off the Green LED NS

digitalWrite(yellowLightEastWest , LOW); // This should turn off the Yellow LED EW

digitalWrite(greenLightEastWest, LOW); // This should turn off the Yellow LED EW

digitalWrite(redLightNorthSouth, HIGH); // This should turn on the RED LED NS

digitalWrite(redLightEastWest, HIGH); // This should turn on the RED Yellow LED NS

digitalWrite(emergencyBlueLED, HIGH); // This should turn on the Remergency blue LED

digitalWrite(buzzerPin, HIGH); // This should turn on the active buzzer

delay(500);

digitalWrite(emergencyBlueLED, LOW); // This should turn off the Remergency blue LED

digitalWrite(buzzerPin, LOW); // This should turn off the active buzzer

delay(500);

client.loop();

client.loop(); // Check if new MQTT messages have been published

if( iotControl == 1)

{ // If iotControl variable is high.

break; // Break out of the while() loop

}

}

client.loop(); // Check if new MQTT messages have been published one last time

}

void buttonPressed() // Function to handle any cross walk button presses

{

static unsigned long lastInterruptTime = 0; // Create variable for holding interrupt time

unsigned long interruptTime = millis(); // Store current time in milli-seconds

// If interrupts come faster than 200ms, assume it's a bounce and ignore

if (interruptTime - lastInterruptTime > 20)

{

crossWalkButtonState = 0; // Change the value of the variable that holds the state of the button

}

lastInterruptTime = interruptTime; // Save the timer

}