Microprocessor Final Report

62011145 Kritapas Chutiwanichyakul

62011278 Thawat Jarupenpat

Topic: Traffic Control Light System Model

Introduction:

A traffic light signal is a signaling device that uses a sensor to indicate when it is safe to drive, bike, or walk down a main road or across a pedestrian crossing. LED traffic lights come in three colors: red, yellow, and green. The car must stop and the pedestrian can cross the crosswalk if the light is red, the car must slow down to prepare to stop if the light is yellow, and the car can move but the pedestrian cannot cross the crosswalk if the signal is green.

All road users have benefited from the traffic lights. Aside from reducing the number of accidents, it improved traffic flow and may have saved individuals time.

Objective:

The model of the traffic light control system will be used to control vehicle traffic on the main road. Its primary goal is to reduce traffic congestion and accidents. It even aids in making traffic flow more smoothly and safely. The safety of pedestrians will be a significant consideration in this system. It will also help drivers anticipate when a person may cross the road, preventing more collisions and accidents.

Specification:

The maximum operational duration delay should not be more than 1 second. **The LED traffic light can show green, yellow or red light.**

- **Green Light LED** → indicate that the vehicles can move or pass.

- **Yellow Light LED** → indicates that the vehicles must slow down and show signs of red light turning to green light.

- **Red Light LED** → indicates that the vehicles should not move or pass.

Buttons will be used to detect pedestrians of the model. When it is pushed by an object (pedestrian) waiting to cross the road, the green light duration on the main road will be reduced, and the **pedestrians' lights switch on**. When the number of waiting vehicles exceeds the **defined point, which we mark on the route** to signal that there is heavy traffic on this road, **the countdown timer** will begin counting in seconds format, and reduce the red light duration.

The sensors should detect any object in the range within at least 1 cm of the model.

Hardware

These are the lists of hardwares we used in this traffic light control model.

- **LPCXpresso1679 Board** -- used for main board and processor

- **Traffic Light LED (Red, Yellow, Green)** -- used for showing traffic signal light

- **LED (Pedestrian)** -- used for signaling for pedestrians to cross the road.

- **Buttons** -- used for changing light by pedestrians when crossing the road.

- **LCD Monitors** -- used to show outputs of countdown timer and ultrasonic sensor and changing state of LED lights.

- **Ultrasonic Sensors** -- used to detect pedestrians and mark the point of waiting vehicles on the route.

Project Model Design

The main component or hardware that we used as a main processor is LPCXpresso 1769.

- How does Traffic light LED works

The traffic light LED consists of Red light, yellow light and green light. The traffic light LED ground was connected to the ground breadboard. The Red, yellow and green were then connected to the pin on the LPC board. The traffic light will show the green light with the delay of 15 seconds and after that the yellow light will show up with the delay of 3 seconds then after that, the red light will then show with the delay of 20 seconds. These LED time delays can be adjusted by users anytime they want.

- *How does Button works*

The buttons were placed on 2 different sides, the left and right side (opposite side of the road). Buttons were connected with the power (VCC) and the output pin in the breadboard and then into the pin of LPCXpresso. If a pedestrian wishes to cross the crosswalk and the traffic light LED green is still turn on, the pedestrian can press the button and the countdown segment of traffic green light will be reduced to 9 seconds when it finishes countdown, it will change to red light and the pedestrian can cross the crosswalk.

- How does LED (Pedestrian) works

The LED light for pedestrians will turn on when the traffic red light LED turn on, which indicates that the pedestrians can cross the road now, but when the traffic green light LED is still turn on, the LED light for pedestrian will be turn off since the pedestrian can’t cross when the vehicles can still pass.

- *How does 7-Segment work*

7- Segment type is Anode, pin a,b,c,d,e,f,g of the 7 Segments were connected to the LPCxpresso board. 7- Segment is used for countdown delay for LED traffic lights. For red light traffic LED, the delay of it is 20 seconds, the segment will display the “-” sign which means it still 2 digits, but when it comes to 1 digits, it will start display number “9” until it reaches “1” then it change color to green LED light which have the countdown delay of 15 seconds, after segment finish display number 10 to 1, it will then changes to yellow with the countdown delay of 3 seconds, after segment finish display number “3” “2” “1” finishes. It will change to a red LED traffic light.

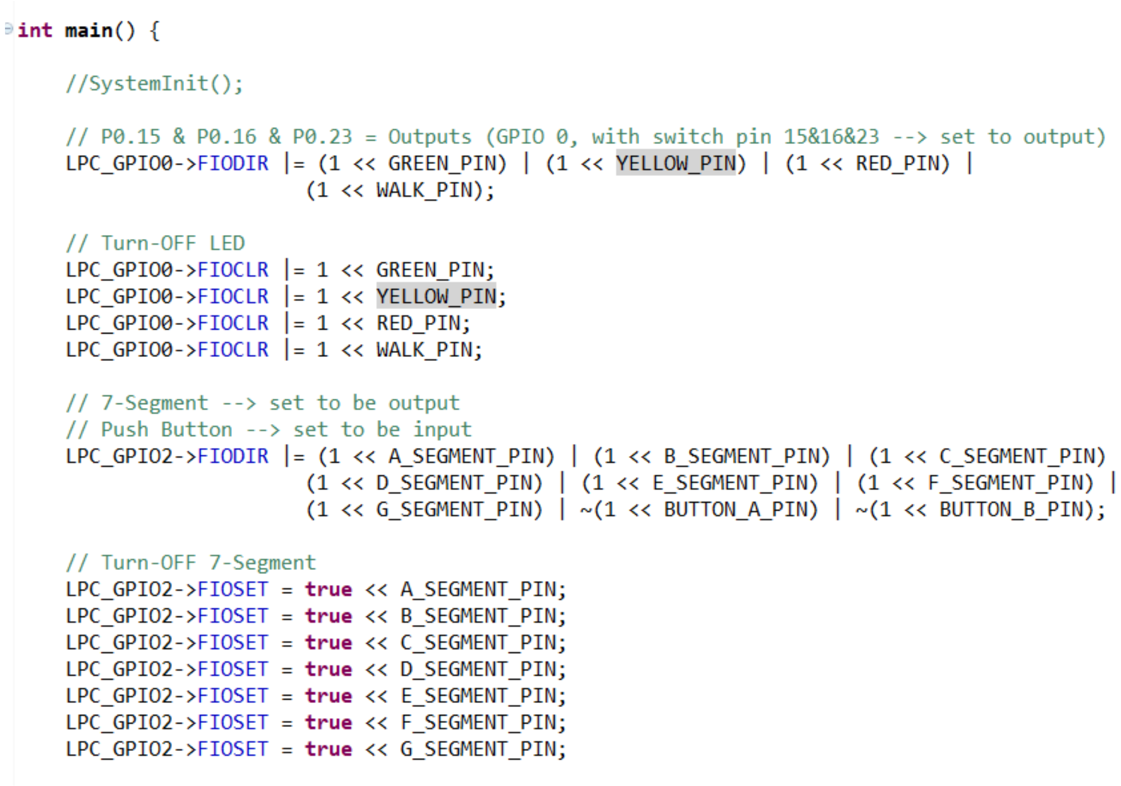
- *How does LCD Monitor work*

LCD Monitor will show the output of the program. It shows the countdown segment of each LED light(Red, Yellow, Green). LCD pins were connected through a breadboard, and output distance of the ultrasonic sensor. It acts as a monitor for an admin.

- *How does Ultrasonic sensor work*

Ultrasonic sensor pins were connected through a breadboard. It will detect a vehicle's distance and duration of waiting. If the vehicles were detected by the sensor as we marked the sensor at that point for a maximum of 5 seconds, the traffic red light segment countdown will be reduced.

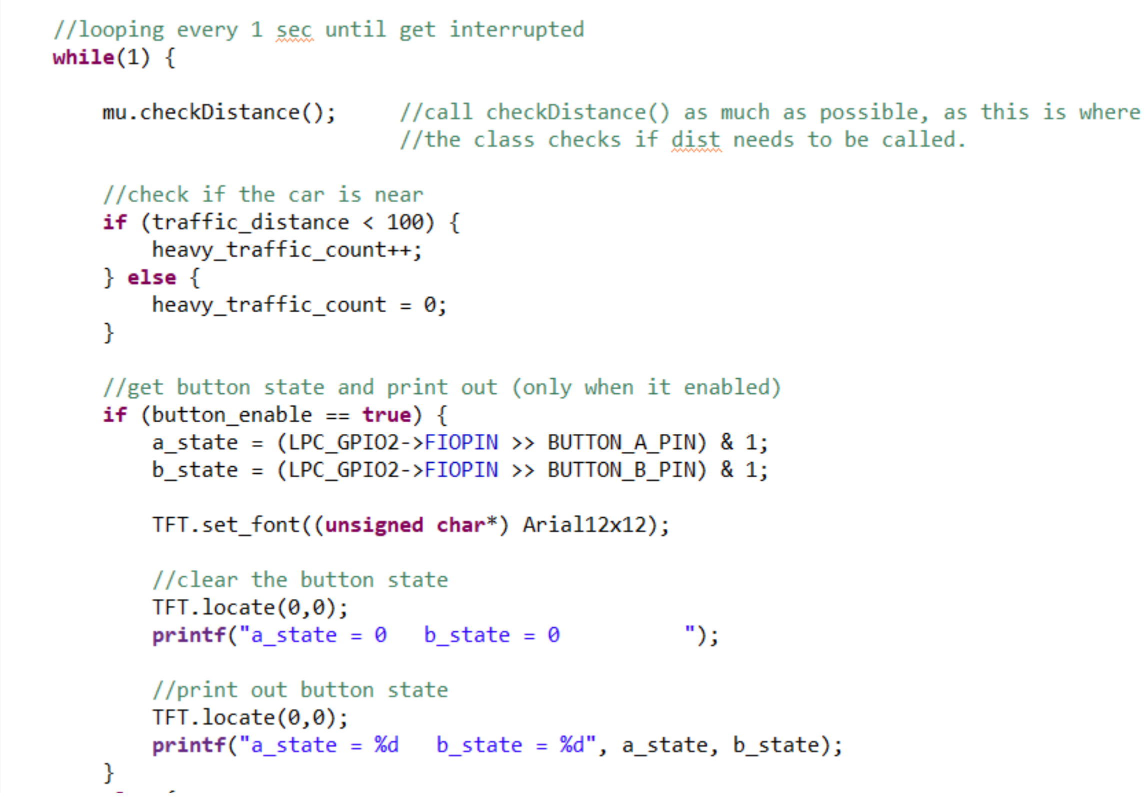
Implementation of Code



```
int main() {

    //SystemInit();

    // P0.15 & P0.16 & P0.23 = Outputs (GPIO 0, with switch pin 15&16&23 --> set to output)
    LPC_GPIO0->FIODIR |= (1 << GREEN_PIN) | (1 << YELLOW_PIN) | (1 << RED_PIN) |
                         (1 << WALK_PIN);

    // Turn-OFF LED
    LPC_GPIO0->FIOCLR |= 1 << GREEN_PIN;
    LPC_GPIO0->FIOCLR |= 1 << YELLOW_PIN;
    LPC_GPIO0->FIOCLR |= 1 << RED_PIN;
    LPC_GPIO0->FIOCLR |= 1 << WALK_PIN;

    // 7-Segment --> set to be output
    // Push Button --> set to be input
    LPC_GPIO2->FIODIR |= (1 << A_SEGMENT_PIN) | (1 << B_SEGMENT_PIN) | (1 << C_SEGMENT_PIN) |
                         (1 << D_SEGMENT_PIN) | (1 << E_SEGMENT_PIN) | (1 << F_SEGMENT_PIN) |
                         (1 << G_SEGMENT_PIN) | ~(1 << BUTTON_A_PIN) | ~(1 << BUTTON_B_PIN);

    // Turn-OFF 7-Segment
    LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << G_SEGMENT_PIN;
```

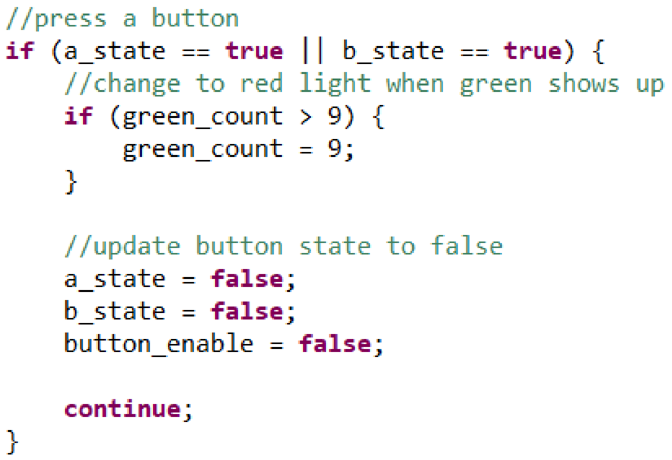
We use “LPC_GPIO0->FIODIR” to set which pin is used by the LED, and “LPC_GPIO2->FIODIR” to set the pin used by 7-Segment.

FIOCLR is used to clear the LED, and FIOSET is used to turn-on the LED. This logic will be swapped when used at 7-Segment (Common Anode).



```
//looping every 1 sec until get interrupted
while(1) {

    mu.checkDistance();     //call checkDistance() as much as possible, as this is where
                            //the class checks if dist needs to be called.

    //check if the car is near
    if (traffic_distance < 100) {
        heavy_traffic_count++;
    } else {
        heavy_traffic_count = 0;
    }

    //get button state and print out (only when it enabled)
    if (button_enable == true) {
        a_state = (LPC_GPIO2->FIOPIN >> BUTTON_A_PIN) & 1;
        b_state = (LPC_GPIO2->FIOPIN >> BUTTON_B_PIN) & 1;

        TFT.set_font((unsigned char*) Arial12x12);

        //clear the button state
        TFT.locate(0,0);
        printf("a_state = 0   b_state = 0          ");

        //print out button state
        TFT.locate(0,0);
        printf("a_state = %d   b_state = %d", a_state, b_state);
    }
```

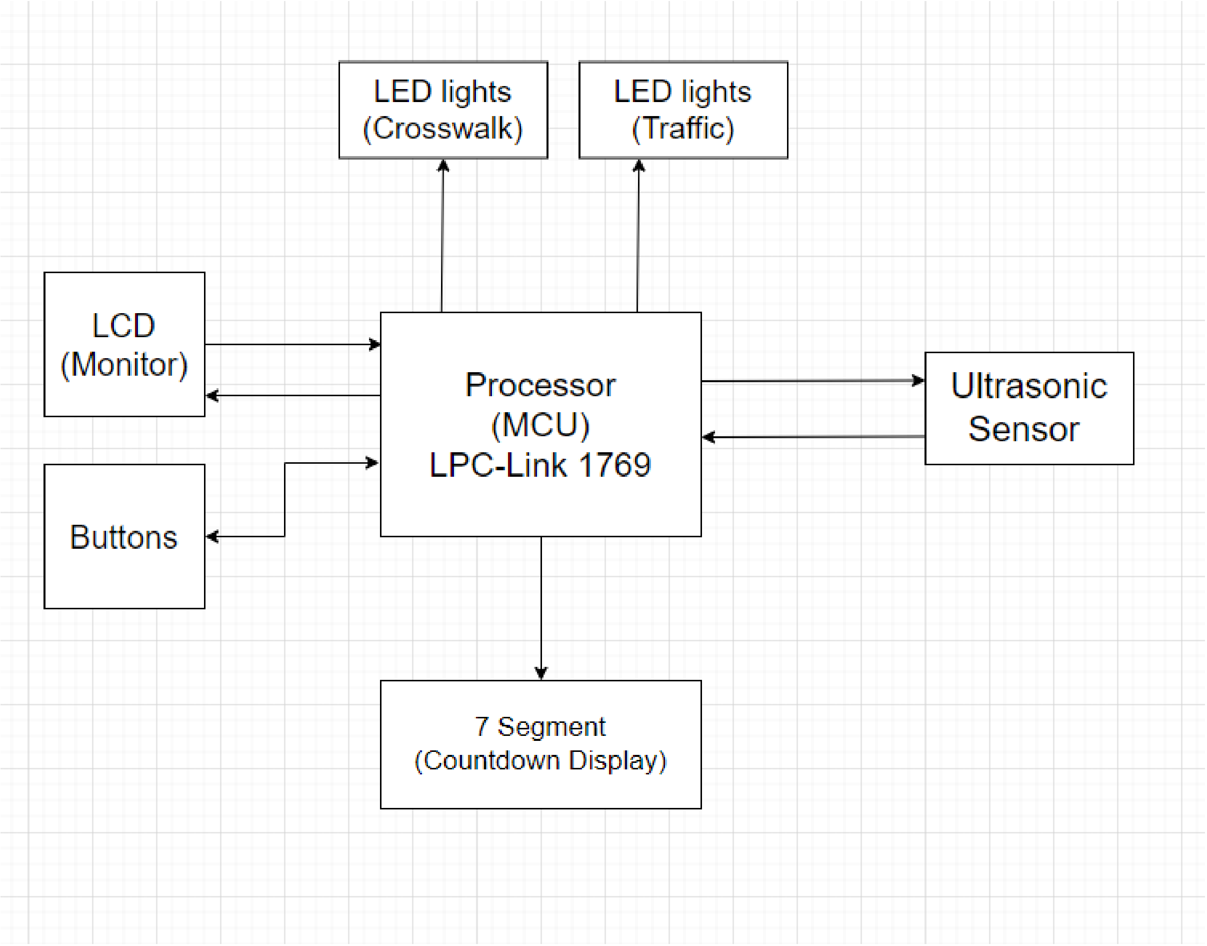
We run the LED counter with while loop. Everytime run, it will decrease the countdown by 1 and update the ultrasonic sensor distance.



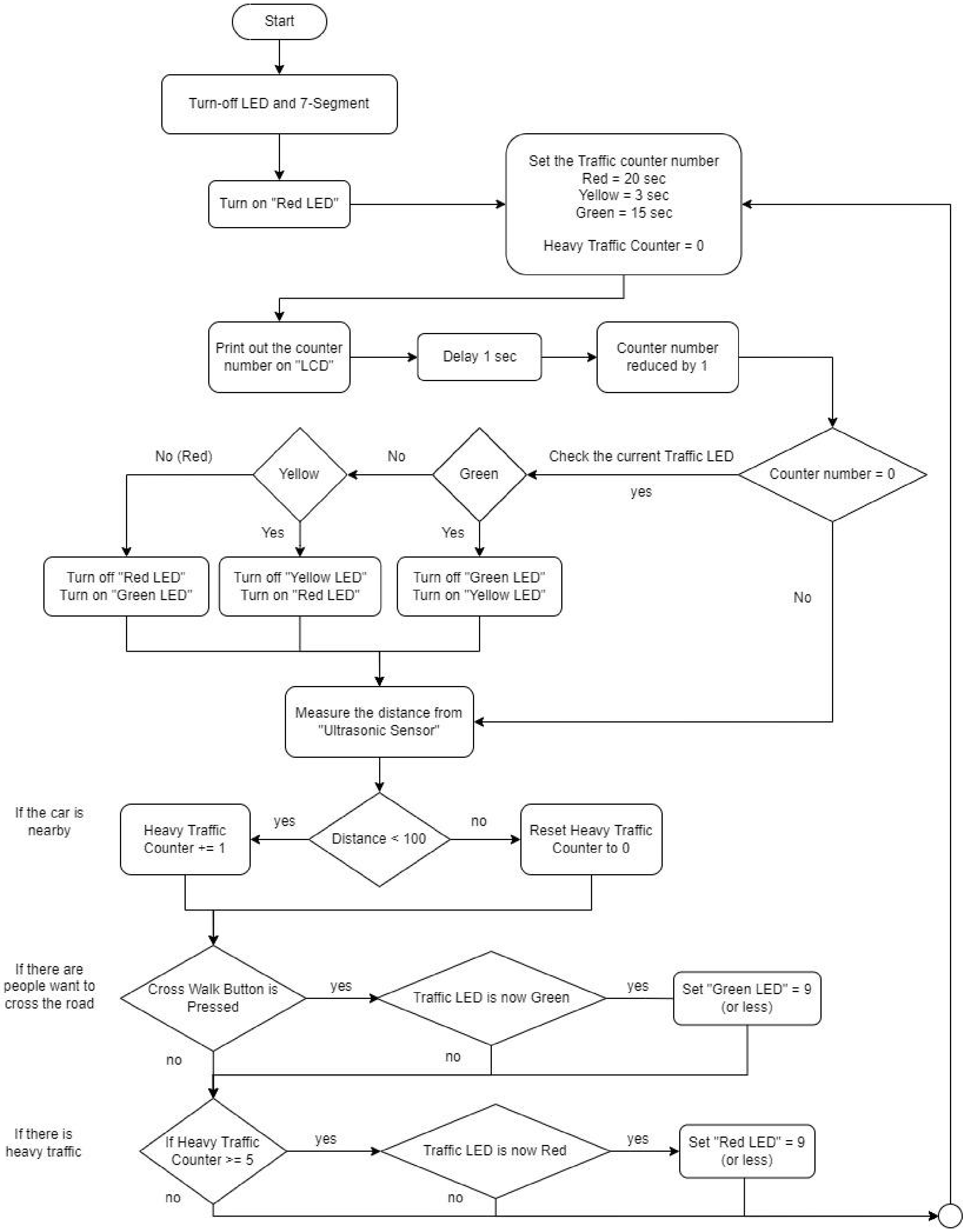
```
//press a button
if (a_state == true || b_state == true) {
    //change to red light when green shows up
    if (green_count > 9) {
        green_count = 9;
    }

    //update button state to false
    a_state = false;
    b_state = false;
    button_enable = false;

    continue;
}
```

If the button is pressed, the green counter will be reduced.

Functional Block Diagram:



Flowchart:


Start
Turn-off LED and 7-Segment
Turn on "Red LED"
Set the Traffic counter number
Red = 20 sec
Yellow = 3 sec
Green = 15 sec
Heavy Traffic Counter = 0
Print out the counter number on "LCD"
Delay 1 sec
Counter number reduced by 1
Counter number = 0
yes
Check the current Traffic LED
No
Green
No
Yellow
No (Red)
Yes
Yes
Turn off "Red LED" Turn on "Green LED"
Turn off "Yellow LED" Turn on "Red LED"
Turn off "Green LED" Turn on "Yellow LED"
Measure the distance from "Ultrasonic Sensor"
If the car is nearby
Heavy Traffic Counter += 1
yes
Distance < 100
no
Reset Heavy Traffic Counter to 0
If there are people want to cross the road
Cross Walk Button is Pressed
yes
Traffic LED is now Green
yes
Set "Green LED" = 9 (or less)
no
no
If there is heavy traffic
If Heavy Traffic Counter >= 5
yes
Traffic LED is now Red
yes
Set "Red LED" = 9 (or less)
no
no

Summary

Traffic light control systems are widely used to monitor and control the flow of automobiles through the junction of many roads. They aim to realize smooth motion of cars in the transportation routes.

This control traffic management aims for the pedestrian to cross the crosswalk safely. If the pedestrian wishes to cross the road, they can press the button and the green light countdown will be reduced and will change to red light after it is done countdown. The LED light pedestrian will be turned on when the traffic light turns to red which indicates that the pedestrian can cross.

If there is heavy traffic, there will be an ultrasonic sensor placed in a pinpoint on a road, if the amount of vehicles waiting in a line reach the sensor, and that vehicle hasn’t been moved for more than 5 seconds, the red light countdown will be reduced.

Problem Face

- The LCD can be connected to the LPC sometimes. If the wire moves too much, it will be disconnected. We think this is a kind of hardware limitation which we have to avoid touching on the wire when we work on that.

  Solution: It may be caused by the wire inside of this LCD or the wire connected to the LPC board. To have a stable output, we guess that the other module may be better in this kind of work.

- Buttons should be separated on each side. In other words, the left side and the right side of the road should contain each 1 button. The problem is in this kind of hardware, we can’t connect it to the wire directly. So, all we can do is connect them via the breadboard.

  Solution: Having another button module may help. Since it is not properly designed for the jump wire connection purpose.

Source Code

```
/* LED */
#include "lpc17xx.h"
#include <stdio.h>

/* LCD */
#include "mbed.h"
#include "SPI_TFT_ILI9341.h"
#include <string>
#include "Arial12x12.h"
#include "Arial24x23.h"
#include "Arial28x28.h"
#include "font_big.h"

#include "ultrasonic.h"


//LCD Pin
DigitalOut LCD_LED(P0_2); // the Watterott display has a backlight switch

SPI_TFT_ILI9341 TFT(P0_9, P0_8, P0_7, P0_6, P0_0, P0_1,"TFT"); // mosi, miso, sclk, cs, reset, dc


/* LED pin with 0.xx */

#define RED_PIN         15              //Red pin = 0.15
#define YELLOW_PIN      16              //Yellow pin = 0.16
#define GREEN_PIN       23              //Green pin = 0.23
#define WALK_PIN  4             //Walk pin = 0.4
```

```
/* LED waiting time */

#define RED_SEC              20
#define YELLOW_SEC           3
#define GREEN_SEC            15

/* 7-Segment pin with 2.xx */

#define A_SEGMENT_PIN        7                //A pin = 2.7
#define B_SEGMENT_PIN        6                //B pin = 2.6
#define C_SEGMENT_PIN        5                //C pin = 2.5
#define D_SEGMENT_PIN        4                //D pin = 2.4
#define E_SEGMENT_PIN        3                //E pin = 2.3
#define    F_SEGMENT_PIN     2                //F pin = 2.2
#define G_SEGMENT_PIN        1                //G pin = 2.1

/* Switch Pin with 2.xx */

#define BUTTON_A_PIN         11               //Button A = 2.11
#define BUTTON_B_PIN         10               //Button B = 2.10

// Locate LCD

#define NUM_POS_X            30
#define NUM_POS_Y            100

// Distance

int traffic_distance = 0;

void delay() {
        static int j;
        int count = 0;

        //make a delay with 1 sec
        for(j = 12000000; j > 0; j--)
                count++;
}

void countdown_segment(int num) {

        switch (num){
        case 0:
                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << E_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << F_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << G_SEGMENT_PIN;
                break;
        case 1:
                LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << G_SEGMENT_PIN;
                break;
        case 2:
                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << E_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;

                break;

        case 3:

                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;

                break;

        case 4:

                LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << F_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;

                break;

        case 5:

                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;

                LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;

                LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << F_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;
                break;
        case 6:
                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = false << C_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << E_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << F_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;
                break;
        case 7:
                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << G_SEGMENT_PIN;
                break;
        case 8:
                LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << E_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << F_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;
                break;
```

```
    case 9:
        LPC_GPIO2->FIOCLR = 1 << A_SEGMENT_PIN;
        LPC_GPIO2->FIOCLR = 1 << B_SEGMENT_PIN;
        LPC_GPIO2->FIOCLR = 1 << C_SEGMENT_PIN;
        LPC_GPIO2->FIOCLR = 1 << D_SEGMENT_PIN;
        LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
        LPC_GPIO2->FIOCLR = 1 << F_SEGMENT_PIN;
        LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;
        break;
    }
}

void dist(int distance)
{
    TFT.set_font((unsigned char*) Arial12x12);

    //print out the distance
    //put code here to execute when the distance has changed
    TFT.locate(0, 225);
    TFT.printf("Distance %d mm\r   \n", distance);

    //Save the traffic distance
    traffic_distance = distance;
}

//Set the trigger pin to p6 and the echo pin to p7
//have updates every .1 seconds and a timeout after 1
//second, and call dist when the distance changes
ultrasonic mu(P0_10, P0_11, .1, 1, &dist);

int main() {
    //SystemInit();

    // P0.15 & P0.16 & P0.23 = Outputs (GPIO 0, with switch pin 15&16&23 --> set to output)
    LPC_GPIO0->FIODIR |= (1 << GREEN_PIN) | (1 << YELLOW_PIN) | (1 << RED_PIN) |
                                        (1 << WALK_PIN);

    // Turn-OFF LED
    LPC_GPIO0->FIOCLR |= 1 << GREEN_PIN;
    LPC_GPIO0->FIOCLR |= 1 << YELLOW_PIN;
    LPC_GPIO0->FIOCLR |= 1 << RED_PIN;
    LPC_GPIO0->FIOCLR |= 1 << WALK_PIN;

    // 7-Segment --> set to be output
    // Push Button --> set to be input
    LPC_GPIO2->FIODIR |= (1 << A_SEGMENT_PIN) | (1 << B_SEGMENT_PIN) | (1 << C_SEGMENT_PIN) |
                                        (1 << D_SEGMENT_PIN) | (1 << E_SEGMENT_PIN) | (1 << F_SEGMENT_PIN) |
                                        (1 << G_SEGMENT_PIN) | ~(1 << BUTTON_A_PIN) | ~(1 << BUTTON_B_PIN);

    // Turn-OFF 7-Segment
    LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
    LPC_GPIO2->FIOSET = true << G_SEGMENT_PIN;


    /* ---------------------------------------------------------------- */
```

```
          // Setup LCD

          LCD_LED = 1;  // backlight on


          TFT.claim(stdout);      // send stdout to the TFT display

          TFT.set_orientation(1);

          TFT.background(Black);    // set background to black

          TFT.foreground(White);    // set chars to white

          TFT.cls();                // clear the screen


          TFT.set_font((unsigned char*) Arial28x28);

          TFT.locate(NUM_POS_X, NUM_POS_Y);


          /* ---------------------------------------------------------------- */


          //start measuring the distance

    mu.startUpdates();


          //declare counting number

          int green_count = 0;

          int yellow_count = 0;

          int red_count = 0;


          //status of the counting led

          //0 --> first started counting

          //1 --> counting green

          //2 --> counting yellow

          //3 --> counting red

          int status = 0;


          //button state
```

```
bool a_state = false, b_state = false;
bool button_enable = false;

int heavy_traffic_count = 0;

//looping every 1 sec until get interrupted
while(1) {

    mu.checkDistance();   //call checkDistance() as much as possible, as this is where
                                                //the class checks if dist needs to be called.

    //check if the car is near
    if (traffic_distance < 100) {
        heavy_traffic_count++;
    } else {
        heavy_traffic_count = 0;
    }

    //get button state and print out (only when it enabled)
    if (button_enable == true) {
        a_state = (LPC_GPIO2->FIOPIN >> BUTTON_A_PIN) & 1;
        b_state = (LPC_GPIO2->FIOPIN >> BUTTON_B_PIN) & 1;

        TFT.set_font((unsigned char*) Arial12x12);

        //clear the button state
        TFT.locate(0,0);
        printf("a_state = 0   b_state = 0           ");

        //print out button state
        TFT.locate(0,0);
```

```
            printf("a_state = %d   b_state = %d", a_state, b_state);
        }
        else {
            TFT.set_font((unsigned char*) Arial12x12);
            TFT.locate(0,0);
            printf("a_state = false   b_state = false");
        }


        //press a button
        if (a_state == true || b_state == true) {
            //change to red light when green shows up
            if (green_count > 9) {
                green_count = 9;
            }

            //update button state to false
            a_state = false;
            b_state = false;
            button_enable = false;

            continue;
        }

        //first time count --> assign red to the counter
        else if (status == 0) {
            red_count = RED_SEC;                                    // set the time to be
countdown

            status = 3;

            LPC_GPIO0->FIOSET = 1 << RED_PIN;               // Turn-On Red
        //turn off 7-segment
        LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;
        LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;
        LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;
        LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
        LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
        LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
        LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;

        LPC_GPIO0->FIOSET = 1 << WALK_PIN;                // Turn-On Walk Way LED
    }

    else {
        //countdown the green and assign yellow when it reaches 0
        if (status == 1) {
            if (green_count == 0){
                yellow_count = YELLOW_SEC;
                status = 2;

                LPC_GPIO0->FIOCLR = 1 << GREEN_PIN;              // Clear Green
                LPC_GPIO0->FIOSET = 1 << YELLOW_PIN;     // Turn-ON Yellow

                //turn off 7-segment
                LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
                LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
                LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;
```

```
                    //clear the number

                    TFT.set_font((unsigned char*) Arial28x28);

                    TFT.locate(NUM_POS_X, NUM_POS_Y);

                    TFT.printf("Yellow Light:      ");


                    button_enable = false;

                    continue;

                }


                countdown_segment(green_count);                                        // show the
number on 7-Segment


                //print out green light count

                TFT.set_font((unsigned char*) Arial28x28);

                TFT.locate(NUM_POS_X, NUM_POS_Y);

                TFT.printf("Green Light: %d  ", green_count);


                delay();


                green_count--;

            }


            // countdown the yellow and assign red when it reaches 0

            if (status == 2) {

                if (yellow_count == 0) {

                    red_count = RED_SEC;

                    status = 3;


                    LPC_GPIO0->FIOCLR = 1 << YELLOW_PIN;        // Clear Yellow

                    LPC_GPIO0->FIOSET = 1 << RED_PIN;                // Turn-On Red


                    //turn off 7-segment
            }

                    LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;

                    LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;

                    LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;

                    LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;

                    LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;

                    LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;

                    LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;


                    //clear the number

                    TFT.set_font((unsigned char*) Arial28x28);

                    TFT.locate(NUM_POS_X, NUM_POS_Y);

                    TFT.printf("Red Light:     ");


                    LPC_GPIO0->FIOSET = 1 << WALK_PIN;                  // Turn-ON
Cross Way LED

                    continue;

                }


                countdown_segment(yellow_count);                        // show the number on
7-Segment


                //print out yellow light count

                TFT.set_font((unsigned char*) Arial28x28);

                TFT.locate(NUM_POS_X, NUM_POS_Y);

                TFT.printf("Yellow Light: %d  ", yellow_count);


                delay();


                yellow_count--;

            }


            //countdown the red and assign green when it reaches 0
```

```
                                        if (status == 3) {
                                                if (red_count == 0) {
                                                        green_count = GREEN_SEC;
                                                        status = 1;

                                                        LPC_GPIO0->FIOCLR = 1 << RED_PIN;                       // Clear Red
                                                        LPC_GPIO0->FIOSET = 1 << GREEN_PIN;                     // Turn-ON
Green

                                                        //turn off 7-segment
                                                        LPC_GPIO2->FIOSET = true << A_SEGMENT_PIN;
                                                        LPC_GPIO2->FIOSET = true << B_SEGMENT_PIN;
                                                        LPC_GPIO2->FIOSET = true << C_SEGMENT_PIN;
                                                        LPC_GPIO2->FIOSET = true << D_SEGMENT_PIN;
                                                        LPC_GPIO2->FIOSET = true << E_SEGMENT_PIN;
                                                        LPC_GPIO2->FIOSET = true << F_SEGMENT_PIN;
                                                        LPC_GPIO2->FIOCLR = 1 << G_SEGMENT_PIN;

                                                        //Clear the number
                                                        TFT.set_font((unsigned char*) Arial28x28);
                                                        TFT.locate(NUM_POS_X, NUM_POS_Y);
                                                        TFT.printf("Green Light:     ");

                                                        button_enable = true;
                                                        LPC_GPIO0->FIOCLR = 1 << WALK_PIN;                      // Clear Walk
Way LED
                                                        continue;
                                                }

                                                //reduce red light time if >= 9
                                                if (heavy_traffic_count >= 5) {
                                                        if (red_count > 9) {
```

```
                        red_count = 9;
                    }
                }

                countdown_segment(red_count);                                   // show the
number on 7-Segment

                //print out red light count
                TFT.set_font((unsigned char*) Arial28x28);
                TFT.locate(NUM_POS_X, NUM_POS_Y);
                TFT.printf("Red Light: %d  ", red_count);

                delay();

                red_count--;
            }
        }
    }

    return 0;
}
```

Reference

https://www.keil.com/pack/doc/cmsis/Core/html/group__system__init__gr.html

https://os.mbed.com/components/HC-SR04/