CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Fall 2025 **Microcontroller Lab** **Felix Pinai**
Session 2

LAB 7: Implementation of LCD Panel in a Traffic Light Controller

We will use the traffic light control system designed on Lab #6 that has the following hardware:

1) 4 sets of RGB LEDs with the following assignments:
    a. 1 RGB LED for East/West direction
    b. 1 RGB LED for East/West Left Turn direction
    c. 1 RGB LED for North/South direction
    d. 1 RGB LED for North/South Left Turn direction

2) Four DIP switches used as sensors with the following designations:

    a. EW Switch Pedestrian (EWPED_SW) switch acting to indicate pedestrian present in the East/West direction.
    b. EW Left Turn (EWLT_SW) switch acting as a sensor for cars making left turn on the East/West direction.
    c. NS Switch Pedestrian (NSPED_SW) switch acting to indicate pedestrian present in the North/South direction.
    d. NS Left Turn (NSLT_SW) switch acting as a sensor for cars making left turn on the North/South direction.

3) Light Sensor to define the mode of operation. There will be two modes: Day Mode and Night Mode.

4) A MODE LED to indicate the mode the traffic controller is running (0 for Night and 1 for Day).

5) A SEC_LED to show 1 second pulse.

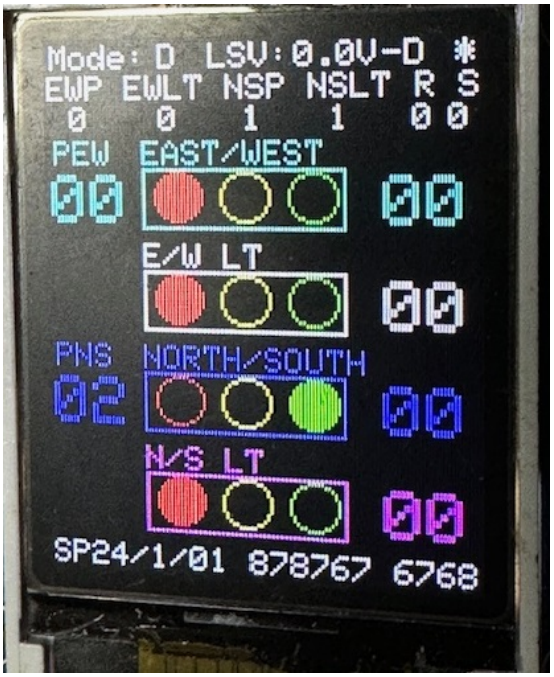
6) 5 LEDS is used to show the number of seconds for pedestrian crossings for each direction.

7) A Buzzer circuit.

From that circuit, we are now going to modify to a new circuit that will have the following:

1) Instead of the two 7-segment displays, a LCD TFT Display is used to show the operation of the controller. A typical screen shot is shown below:

2) Some of the RGB LEDs will have new connections but their functions will remain the same (see attached schematics for new pin assignments).



Here are some descriptions of the information on the above TFT screen:

1) First Line:
    a. Mode: Actual mode the controller is running – A 'D' will indicate the day mode while a 'N' will specify the night mode. Remember this indicator is to show the operational mode being active and not the mode that the light sensor is requiring.
    b. To the right of Mode is 'LSV': Light Sensor Voltage. Next to it is the actual voltage reading to the photoresistor followed by either a 'D' or a '-N' indicating that the voltage is to force a 'Day' mode of operation or a 'Night' mode.
    c. The last element of the first line is a blinking *'. It will be turned on and turned off every 0.5 second.

2) The second line shows the following: 'EWP EWLT NSP NSLT R S'. The numbers below that line show the status of the switches for the East/West PED, EWLT sensor, North/South PED, NSLT sensor and the logic state of the signal 'R' and 'S' that we deal with on the next lab.

3) 'EAST/WEST': The box below this line shows the color of the traffic light in the East/West direction. There are three circles below this box each with the colors Red, Yellow, and Green from left to right. When the circle is not filled, this means that the light is off. When it is filled, the corresponding color is on.

4) The two ‘00’ digits at the right of the ‘EAST/WEST’ are used to display the number of seconds left on this direction when the lights are turned into green and yellow colors.

5) There are three other labels ‘E/W LT’, ‘NORTH/SOUTH’, ‘N/S LT’. They have the same descriptions as for the East/West direction, but each item is for a different direction.

6) The label ‘PEW’ and the number right below show the actual number of seconds left on the Pedestrian Counter in the East/West direction.

7) The label ‘PNS’ and the number right below it shows the actual number of seconds left on the Pedestrian Counter in the North/South direction.

8) The last line (bottom) will show the name of the semester, the session number, and the table number. The next set of numbers shows the delays used for the traffic design.

The operation of the traffic light was already implemented on Lab #7. Both the Night and Day modes should be kept the same.

The TFT Panel uses the hardware interface called SPI Bus for the microcontroller to control the panel to display data on its screen. The basic understanding of the SPI bus will be visited on a later lab, and the hardware explanation will be handled at that time.

In addition, since it would take a good amount of time to implement all the software routines to allow the user to display texts or to generate different types of shapes, library software has been compiled and put together into the file "ST7735_TFT.c".

To develop the basic screen shown on the above screenshot, it will take a good amount of time to do so. We will skip this step. A basic routine is provided to the student that will create the basic framework of the screen. The student should look at the provided routine ‘Initialize_TFT()’ to get the basic ideas to implement the rest of the requirements of this lab.

Here are the steps that you will need to perform to setup the operations for the LCD:

Task #1: Create a new folder for lab 7 just as you have done for the previous labs.

Unzip the provided file ‘ECE3301L_Lab7_Fa25_S2.zip’. Make sure that all the files are saved on the newly created folder.

Task #2: Open the file ‘main.h’ and based on your session number, and table number, modify the values for the following variables by editing the ‘?’:

```
#define Semester          ?     // Type 0 for Sp, 1 for Fa
#define Year              ??    // Type 2 digits year
#define Session_Number    ?     // Type Session Number 1 through 5
#define Table_Number      ??    // Type Table Number from 01 through 14
```

Task #3: Move all the definitions of RGB LEDs from the original main program to this 'main.h'. Make sure to delete the previous definitions. Do the same for the pin definitions of the MODE LED and the SEC LED.

Task #4: Still using the 'main.h', go back to the original program and locate all the numbers of seconds used to delay the GREEN LEDs and the PEDESTRIAN counts. Replace those hard code numbers by the following names:

```
#define PEDESTRIAN_EW_WAIT      ?
#define EW_WAIT                 ?
#define NS_LT_WAIT              ?
#define PEDESTRIAN_NS_WAIT      ?
#define NS_WAIT                 ?
#define EW_LT_WAIT              ?

#define NIGHT_EW_WAIT           ?
#define NIGHT_NS_LT_WAIT        ?
#define NIGHT_NS_WAIT           ?
#define NIGHT_EW_LT_WAIT        ?
```

We have two sets of numbers, one with 6 numbers and one with 4. These two sets are displayed on the bottom line of the screen. They are used to making sure that the delays are correct.

Task #5:
Open the file 'Main_Screen.h' and make sure that the following lines are present:

```
#define EW      0       // Number definition of East/West
#define EWLT    1       // Number definition of East/West Left Turn
#define NS      2       // Number definition of North/South
#define NSLT    3       // Number definition of North/South Left Turn
```

Task #6:
We will need to take the following steps executed to get the LCD screen operating and providing a basic screen with information. Perform the following steps:

Open the file 'ST7735_TFT.h' and look at the schematics to determine the assignment of the following signals:

```
#define TFT_DC      PORT?bits.R??     // Location of TFT D/C
#define TFT_CS      PORT?bits.R??     // Location of TFT Chip Select
#define TFT_RST     PORT?bits.R??     // Location of TFT Reset
```

There is no need to create the definitions for the signal TFT_SCK and TFT_SDO because they are at fixed locations (RC3 and RC5).

Make the updates and save the file.

Task #7:

Once the three bits for the TFT signals are defined in task #6, determine the binary combination to be used based on the location of these three bits in PORT D to create a mask byte used to protect these signals when we will be doing to update the Pedestrian counter (check the function 'void update_LCD_PED_Count(char direction, char count)' in Main_Screen.c file). Set the proper value for the variable 'mask_TFT' in the 'main.h' file. Also, if the LEDs used for the pedestrian count are offset, set the variable 'count' to be the number of bits to be shifted. If none, set it to 0 or remove the shift operation.

Task #8:

Step 1) Create a project as usual

Step 2) Right click the tab **'Header Files'** and select 'Add Existing File'. Go to the folder that contains the files from the zip file and add the file: main.h'.

When completed, repeat the process for the following files:

- Main_Screen.h
- ST7735_TFT.h
- utils.h

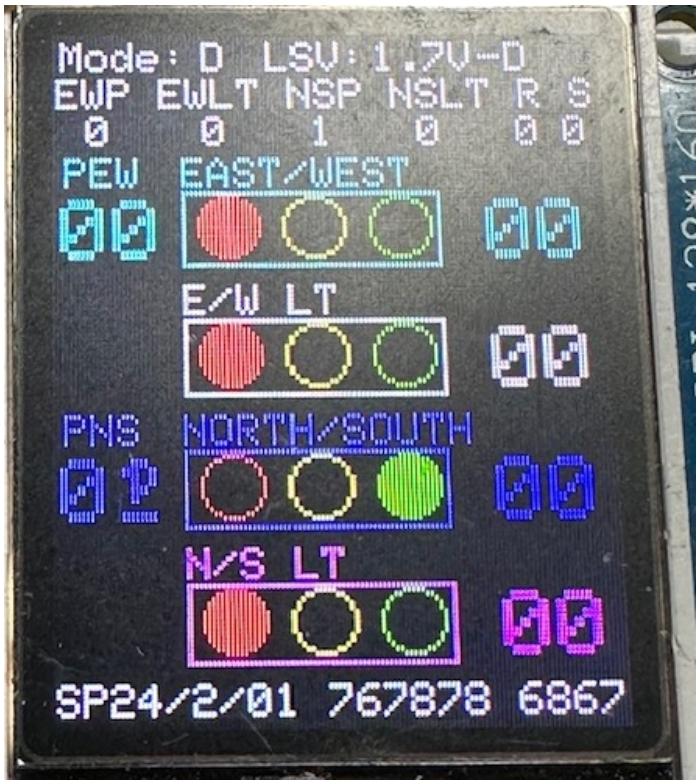
Step 3) Next, add the following files into the **'Source Files'** tab:

- Lab7_S2_sample.c
- Main_Screen.c
- ST7735_TFT.c
- utils.c

Task #9:

Compile the project and correct any error that has occurred. When the project is clean, download it to the board.

The LCD screen will be updated but it will remain static as shown below:



If the correct screen is shown, then the hardware connections to the LCD appear to be correct.

Next, go to the main program of the file ‘Lab7_S2_sample.c’ and comment out the following lines:

```
while (1)
{
    Rcmd2red();
    delay_ms(2);
}
```

That will remove the infinite loop and allow the program to continue.

Task #10:

Open the file ‘Main_Screen.c’ and start to make the following editing:

A) **void update_LCD_color(char direction, char color)**

This routine will change the color of the traffic light on the TFT panel for the specified ‘direction’ with the specified ‘Color’. Fox example, if this function is called with direction=EW and Color=Color_Red, then the RED light in the East/West field should be filled. The other two colors (Yellow and Green) should have only their outline drawn but inside the circle should not be filled.

.

Here is a portion of that routine providing a starting example:

```
void update_LCD_color (char direction, char color)
{
  char Circle_Y;
  Circle_Y = EW_Cir_Y + direction * 30;

  if (color == Color_Off)    //if Color off make all circles black but leave outline
  {
      fillCircle(XRED, Circle_Y, Circle_Size, ST7735_BLACK);
      fillCircle(XYEL, Circle_Y, Circle_Size, ST7735_BLACK);
      fillCircle(XGRN, Circle_Y, Circle_Size, ST7735_BLACK);
      drawCircle(XRED, Circle_Y, Circle_Size, ST7735_RED);
      drawCircle(XYEL, Circle_Y, Circle_Size, ST7735_YELLOW);
      drawCircle(XGRN, Circle_Y, Circle_Size, ST7735_GREEN);
  }
```

// Add code for the remaining colors:

```
  if (color == Color_Red)    //if Color off make all circles black but leave outline
  {

  }
```

Color_Yellow

Color_Green

B) void update_LCD_count(char direction, char count)

This routine will change the value of the Second Counter (right side of the traffic light) for the specified 'direction' with the value indicated by 'count'.

Below is a beginning example of the routine:

```
void update_LCD_count(char direction, char count)
{
        switch (direction)          // update traffic light no ped time
        {
          case EW:
            EW_Count[0] = count/10 + '0';
```

```
            EW_Count[1] = count%10 + '0';
            drawtext(XCNT, EW_Count_Y, EW_Count, EW_Color, ST7735_BLACK, TS_2);
            break;

        // Add code for the other directions (EWLT, NS, NSLT)
          }
}
```

C) void update_LCD_PED_Count(char direction, char count)

This routine will change the value of the Pedestrian Counter (left side of the traffic light) for the specified 'direction' with the value indicated by 'count'.

Below is a beginning example of the routine:

```
void update_LCD_PED_Count(char direction, char count)
{
        switch (direction)
         {
                case EW:
                 PED_EW_Count[0] = count/10  + '0';          // PED count upper digit
                 PED_EW_Count[1] = count%10  + '0';          // PED Lower
                drawtext(PED_Count_X, PED_EW_Count_Y, PED_EW_Count, EW_Color,
        ST7735_BLACK, TS_2);
                break;}

    // Add code for the direction NS
        }
}
```

D) void update_LCD_misc()

This routine will update the various states of the switches and the photo sensor. Below is an example of that routine:

```
void update_LCD_misc()
{
   int nStep = get_full_ADC();         // calculates the # of steps for analog conversion
   float volt = Read_Volt (0);         // Read light sensor
   Light_Sensor = volt < 2.5 ? 1:0;  // Mode = 1, Day_mode, Mode = 0 Night_mode
   Light_Sensor_Voltage_Txt[0] = ((int) volt + '0');
```

```
    Light_Sensor_Voltage_Txt[2] = ((((int) (volt*10))%10) + '0');
    if (Light_Sensor == 0) Light_Sensor_Mode_Txt[1] = 'N'; else Light_Sensor_Mode_Txt[1] = 'D';

    SW_EWPED = EW_PED_SW;
    SW_EWLT = EW_LT_SW;
    SW_NSPED = NS_PED_SW;
    SW_NSLT = NS_LT_SW;

    if (SW_EWPED == 0) SW_EWPED_Txt[0] = '0'; else SW_EWPED_Txt[0] = '1';

     // put code here for SW_EWLT, SW_NSPED, SW_NSLT, SW_MODE

}
```

Task #11:

Now, we will use the file 'Lab7_S2_sample.c'.

Use the content of lab#7 and its routines. Move them into the routines shown in this file with the expression '// add code here'

When complete with the move, do the following:

:

a. In the 'void Set_EW(char color)' routine, add at the beginning of the routine the following lines:

```
direction = EW;
update_LCD_color (direction, color);
```

b. Repeat the same modifications for the other routines Set_EWLT(), Set_NS() and Set_NSLT()

c. In the 'PED_Control(char direction, char count)' routine, remove all the instructions to output to the PORTB and PORTD and replace them with the use of the newly created routine 'void update_LCD_PED_Count(char direction, char count)'. Simply, use a FOR loop to do the countdown using a call the routine 'update_LCD_PED_Count (direction, I)' with 'I' being the countdown number. Don't forget to add the extra delays after the FOR loop.

d. Go to the routine 'Wait_One_Second()'. Add the line 'update_LCD_misc();' at the end. Do the same for the routine 'Wait_One_Second_With_Beep()'.

e. Go to the routine 'Wait_N_Second()'. Add the line 'update_LCD_count(direction, I);' in the FOR loop. Then, add the line 'update_LCD_count(direction, 0);' at the end of the routine.

f. Copy both the ‘Night_Mode’ and ‘Day_Mode’ implementations from lab #7.

g. Replace all the hard coded wait numbers used for Pedestrian and Green on both Day-Mode and Night-Mode by the variable names specified in Task #3

Task #12:

When all the tasks are completed, compile and download it to the board. The operation should run like on lab #6 but now with more interactive information displayed on the LCD.