CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Fall 2025 Microcontroller Lab Felix Pinai
Session 2

LAB Final: Final Project

We have covered during this semester the following topics:

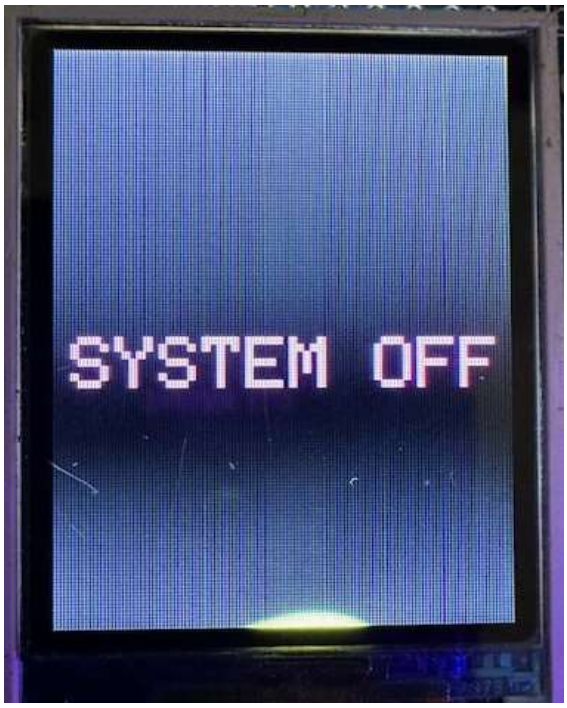
- GPIOs
- A/D Converter
- Temperature sensor
- Light sensor
- PWM – Fan & speaker
- Timer and Counters
- System interrupts
- TFT interface
- I2C bus
- SPI bus
- RTC
- IR Remote Control

The final project will integrate a design that will have the following functions:

- A TFT panel used as a main display.
- An ambient temperature in degree C and F is displayed on the screen
- A digital clock shows the time and date
- A fan support with full function control – On/Off and speed
- An indicator of the duty cycle for the fan control
- A remote control is used to set up the actual time, the alarm time and the set temperature for a fan control
- A push-button switch used to activate the entire system
- A light sensor that also acts for system activation.

The hardware schematics is provided on a separate pdf file.

Here is the screenshot of the main screen when the system is first powered up:

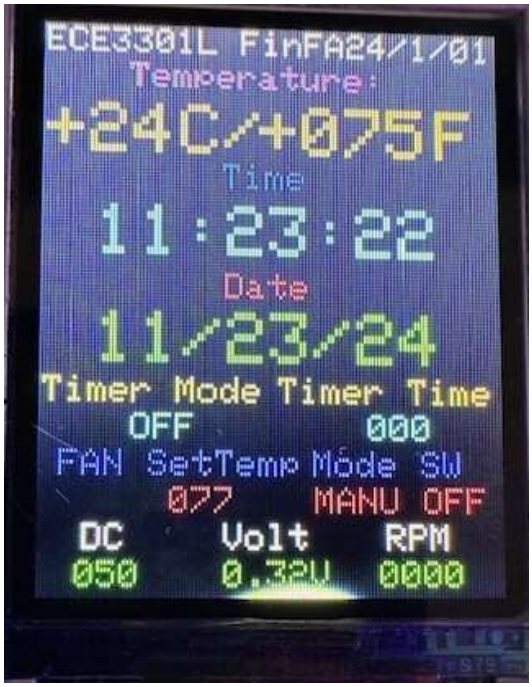


The screen simply shows that the system is not fully activated until either one of the following events happens:

1) The pressing of the push-button on the design
2) The darkening of the light sensor also on the design.

These two mechanisms can also be used to shut down the system when it is active.

Now, when the system is activated, here is the typical screen:

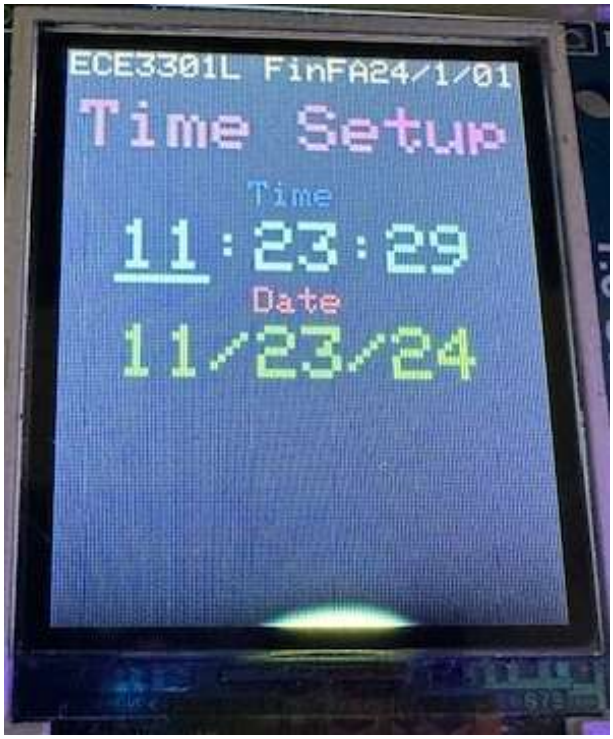


Here are the descriptions of the fields:

- ECE3301L FinSSyy/s/tt where your team will have to replace in the ‘Main.h’ header file the following information:
  - SS: Use 0 for Fall or 1 for Spring semester
  - yy: Academic Year
  - s: ECE3301L Session number (1,2,3,4)
  - tt: Table Number (01 to 14)
- ‘+24C/+77F’: Actual Temperature display in degree C and degree F
- ’11:23:22’: Time display
- ‘11/23/24’: Date display
- ‘Timer Mode’: Timer mode for fan – 4 possible setting – OFF, TM1, TM2, TM3
- ‘Timer Time’: Number of seconds left for fan’s timer
- ‘FAN Set Temp’: Temperature set to the minimum temperature that the Fan Automatic Mode should activate the duty cycle control
- Mode: Auto or Manu. This mode is toggled by the button ‘EQ’ to enable/disable the fan speed control mode based on the temperature
- ‘SW’: Switch to turn ON or Off the Fan – Control by the pressing of the ‘Play/Pause’
- ‘DC’: Actual level of Duty Cycle
- ‘Volt’: Actual Readout of the voltage of the light sensor
- ‘RPM’: Actual RPM of the fan

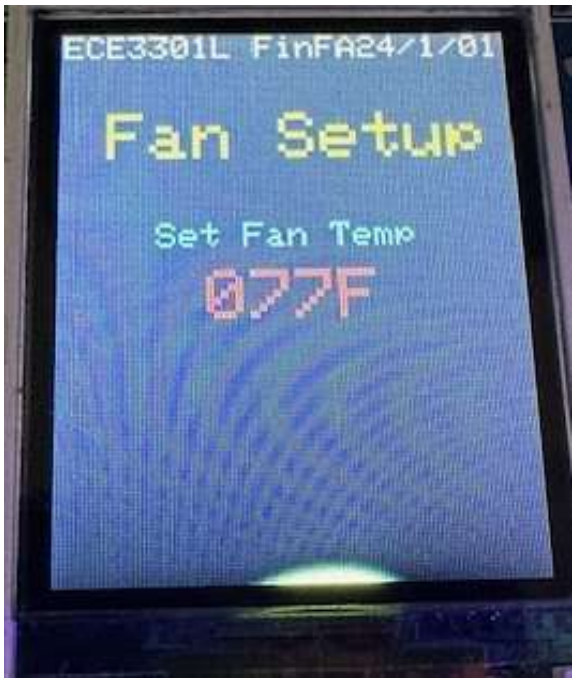
Two setup screens are available to allow changes to the system variables:

1) Time Setup Screen:



This screen will allow the actual time and date to be changed and set.

3) Fan Temp Setup:



This screen sets the minimum temperature for the automatic fan feature.

A) Setup Operations

Two 'Setup' buttons on the remote control are used to select one of two different setup screens:

1) 'CH-': Time Setup Screen
2) 'CH+': Fan Temperature Setup Screen.

Time Setup Mode:

The screen will show the first line as the time and the second line as the date. The time line has three fields 'hour:minute:second' while the date line shows the fields 'month/day/year'. A cursor is shown on the screen to indicate what field is active. A total of 6 fields can be updated. Two buttons are used to move from one field to another:

- '**Prev**' is to move backward. If the cursor is at the first field, then pressing this button will move the cursor to the last field.

- '**Next**' will move the cursor forward. If the cursor is at the last field, then it will be moved to the first field.

To increase or decrease the content of each field, two buttons are used for this purpose.

* Button '+' will do the increase
* Button '-' will decrease the value.

To exit this Time Setup screen without saving the new data, the button '**EQ**' will facilitate this choice.

To exit and save the new time, the button '**Play/Pause**' is to be used for this purpose.

Fan Set Temperature Mode:

This mode is used to program the temperature level that will trigger the fan to be turned on when the ambient temperature is higher than that level. The difference between that set level temperature and the ambient temperature will be handled by a special function to calculate the duty cycle described as follows:

```
int get_duty_cycle(int temp, int set_temp)
{
    a) If temp is less to the set_temp, then duty_cycle is 0
    b) If temp is greater than or equal to set_temp, calculate:

        diff_temp = (temp – set_temp);
        duty_cycle = diff_temp * 3;
        if duty_cycle > 100 then force it to be 100
        else return the value of duty_cycle

}
```

The set temperature cannot go lower than 50F and cannot go higher than 110F

Since there is only one field, the '-' and '+' buttons are used to increase/decrease the temperature level. The 'Prev' and 'Next' are not valid buttons. However, the 'Play/Pause' is still used to save the new data and 'EQ' is to exit without saving.

Standard Operations:

1) Fan Operation – Auto versus Manual

When on the Main screen, the 'Play/Pause' button is used to toggle the fan monitoring operation.

When the FAN SW is switched to the 'OFF' mode, the FAN_EN signal must be set to 0 to disable the FET controlling the fan. The FANON_LED is also turned off.

When the FAN SW is turned on, FAN_EN must be set to 1 as well as FANON_LED. The fan's duty cycle will be programmed to the value set by the variable 'duty_cycle'. This variable can be changed based on two modes of operation: Manual or AUTO. Pressing the 'EQ' button will toggle between these two modes. The default is 'Manual'.

When in manual mode, the fan speed is completely controlled by the setting of the duty cycle. This variable can be changed by pressing the '+' or the '- 'buttons. The former one is to increase the duty cycle while the latter one is to decrease.

In the Auto mode, the speed of the fan is controlled using a temperature sensor to calculate the temperature difference between the actual value against a preset value. The higher the difference, the faster the fan will run. If the preset temperature is lower than the actual temperature, the fan will run at a very nominally low speed. A call to the function:

int get_duty_cycle(int temp, int set_temp)

will calculate the required value of 'duty_cycle'

The two RGB LEDs D1 and D2 must operate just like in lab #12. As a reminder, here are the definitions:

LED D1: void Set_DC_RGB(int duty_cycle)

Set_DC_RGB(int duty_cycle):

a) If duty cycle >= 0 and < 9, color is none
b) If duty cycle >=10 and < 19, color is RED
c) If duty cycle >=20 and < 29, color is GREEN
d) If duty cycle >= 30 and < 39, color is YELLOW
e) If duty cycle >= 40 and < 49, color is BLUE
f) If duty cycle >= 50 and < 59, color is PURPLE
g) If duty cycle >= 60 and < 69, color is CYAN
h) If duty cycle >= 70, color is WHITE

LED D2: void Set_RPM_RGB(int rpm):

a) If rpm == 0, no color to be displayed
b) If rpm > 0 and < 500, color is RED
c) If rpm >= 500 and < 1000, color is GREEN
d) If rpm >= 1000 and < 1500, color is YELLOW
e) If rpm >= 1500 and < 2000, color is BLUE
f) If rpm >= 2000 and < 2500, color is PURPLE
g) If rpm >= 2500 and < 3000, color is CYAN
h) If rpm >= 3000, color is WHITE

2) Fan Timer Operation

A timer operation is available with the design. When the timer is off, there will be no timeout, and the fan will operate until it is manually shut off. The timer can be selected to have three timeout periods to be called TM1, TM2 and TM3. Using the 'Prev' and 'Next' buttons, we can change between those three modes along with the 'OFF' modes. A time value will be displayed on the 'Timer Time' field. If the fan is not turned on, then that value will stay static. When the fan is turned on, then the timeout operation will start. When the time expires, the fan will be automatically shut off.

The RGB LED D3 will show the status of the Timer Mode by showing four different color settings:

- When off, D3 will be off
- When in TM1, D3 will be RED
- When in TM2, D3 will show the GREEN color
- When in TM3, D3 will be YELLOW

Guidance:

1) A basic sample of the code is provided to the student to allow shorter development time for the project.
2)

Here is the list of files in the zip file of the sample codes:

Source files:

- Main.c: this is where the main program resides
- Main_Screen.c: this is the file that will generate the main screen on the LCD
- Setup_Time.c: this is where all the functions to support the setup of the time
- Setup_Fan_Temp.c: the functions to allow the setup of the temperature for the fan are in this file

- Fan_Support.c: all the functions to support the fan operations should be included in this file
- Interrupt.c: this is where the interrupt handler and the code for the IR remote control function
- utils.c: utility functions are included in this file

Header files:

- Main.h: Used this new file and make sure to edit the location of your KEY_PRESSED button
- Main_Screen.h: Used as is
- Interrupt.h: Used as it
- Setup_Time.h: Used as is
- Setup_Fan_Temp.h: Used as is
- Fan_Support.h: Used this new one and make sure to edit the pin assignments
- utils.h: Used as is

You will need to take the following files from lab #12 and add them to this new project:

Source files:

- I2C_Soft.c
- I2C_Support.c
- ST7735_TFT.c

Header files:

- I2C.h
- I2C_Support.h
- ST7735_TFT.h

3) Go through all the files and look for places that indicate // add code here' to supply your own codes.

4) The 'Setup_Time.c' is provided as a startup example to modify the time. The routine will start to read the actual time and use it to set up the startup screen. It will then stay in a while loop to receive incoming buttons from the remote. Based on the inputs, it will call appropriate functions to execute the commands. More detailed information will be discussed at the lecture on the implementation.

5) The file 'Setup_Fan_Temp.c' is to change the set temperature for the fan function.

6) The main routine has the following tasks in its 'while (1)' loop:
   a. Check for the time change every second. When that happens, it will:

i. Measure the fan's rpm
ii. Measure the ambient temperature
iii. Measure the voltage of the light sensor
iv. Call the function Monitor_Fan() (located in the 'Fan_Support.c') where:
    1. First, check if the FANMODE is set to 1 (Automatic fan control). If so, the duty cycle is calculated based on the condition of the ambient temperature and the set temperature.
    2. Next, the variable FAN is checked. If 0, then turn off the fan. Else, turn on the fan and check if the 'FAN_TIMER_MODE' variable is not 0. If not, then we are in timer mode. Decrement the count in the FAN_TIMER_TIME. If the count has reached 0, then set FAN to 0 and turn off the fan.
v. Output the information on TeraTerm
vi. Update the information on the LCD screen

b. Check if a button on the remote control is received. If that happens, it will:
    i. Check if button is valid. If not, generate a bad beep tone. If yes, then generate good beep tone and process the button. Here are the buttons that are allowed during the main operation:

        1. 'CH-' for Setup_Time()
        2. 'CH+' for Setup_Fan_Temperature()
        3. 'Prev' to decrease the fan timer's time
        4. 'Next' to increase fan timer's time
        5. 'Play/Pause' for Toggle_Fan_On/Off
        6. '-'to decrease the duty cycle in the Fan Manual mode
        7. '+' to increase the duty cycle in the Fan Manual mode
        8. 'EQ' to toggle the Fan Auto/Manual mode