CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Spring 2024 Microcontroller Lab Felix Pinai
Session 1

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
- An alarm function is available to activate a multicolor LED when the alarm set time is reached
- A remote control used to setup the actual time, the alarm time and the set temperature for a heater control (represented by the fan).
- A push-button switch used to enable the Alarm function

The hardware schematics is provided on a separate pdf file.

Here is the screenshot of the main screen:

Here are the descriptions of the fields:

- ECE3301L Finssyy/s/tt where your team will have to replace the following:
  - SS : Use FA or SP for Fall or Spring semester
  - yy : Academic Year
  - s : ECE3301L Session number (1,2,3,4,5)
  - tt : Table Number (01 to 14)
- '28C/82F' : Actual Temperature display in degree C and degree F
- '08:18:25' : Time display
- '01/01/00' : Date display
- 'Alarm Time' : Time set for the Alarm
- 'Alarm Sw' : Switch to turn On or Off the Alarm
- '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 '0' to enable/disable the fan speed control mode based on the temperature
- 'SW' : Switch to turn ON or Off the Fan
- 'DC': Actual level of Duty Cycle

- ‘RM’ : RTC Match status used for Alarm (active high)
- ‘Volt’: Actual Readout of the voltage of the light sensor
- ‘RPM’: Actual RPM of the fan

Three 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 set.

2) Alarm Setup Screen:

This screen will allow the setting of the alarm time

3) Fan Temp Setup:

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

A) Setup Operations

Three ‘Setup’ buttons on the remote control are used to select one of three different setup screens. From the main screen, the following three buttons will force the system to go to different setup screens:

1) ‘ 1 ‘ : Time Setup Screen
2) ‘ 2 ’ : Alarm Setup Screen
3) ‘ 3 ’ : 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 move 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 use for this purpose.

Alarm Setup Mode:

This mode is similar to the Time Setup Mode but it only changes three time fields for the alarm function. It affects the three time registers of the alarm function of the DS3231. The same buttons of the Time Setup Mode are used in this screen.

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 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 – 16);
       duty_cycle = diff_temp * 5;
       if duty_cycle > 100 then force it to be 100
       return the value of duty_cycle

}
```

The set temperature cannot go lower than 16C and cannot go higher than 30C.

Since there is only one field, only 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

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

When the FAN SW is in the 'OFF' mode, the FAN_EN signal must be set to 0 to disable the FET controlling the fan and the duty cycle 'DC' must be set to 0. The FANEN is also turned off.

When the fan operation is switched to 'ON' mode, the 'fan_set_semp' variable must be compared against the ambient temperature. If the ambient temperature is less than the set temperature, then the fan must be treated as in the 'OFF' condition; both the FAN_EN and the FANEN_LED signals must be turned off.

When the ambient temperature is lower than the set temperature, use the newly defined function 'int get_duty_cycle' to obtain the value of the duty cycle to be applied to the fan. Call the function 'do_update_pwm()' to apply that duty cycle to the fan. Measure the RPM of the fan and show it on the display. Both the FAN_EN and FANEN_LED signals must be turned on to activate the fan.

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) **Alarm Operation**

The Alarm can be enabled or disabled by the Alarm Mode push-button switch. The status of the Alarm operation can be seen on the LCD screen under the line 'Alarm SW' with the display of either 'OFF' or 'ON'. When the Alarm mode is enabled, the time set under the 'Alarm Time' will be programmed into the DS3231 chip and the Alarm interrupt operation must be enabled in the DS3231. When the alarm times matches with the alarm set time, this signal 'RTC_ALARM#' will be asserted by the DS3231 chip to the logic 0. When that event happens, the buzzer will be activated and the RGB LED D3 will change to a different color every second using the following sequence:

a) No color
b) RED
c) GREEN
d) YELLOW
e) BLUE
f) PURPLE
g) CYAN
h) WHITE

In addition, the value under 'RM' on the LCD Display should show the value of 1 when the time matches (be aware that the signal RTC_ALARM# is active low while the variable 'RM' is active high).

The buzzer and the RGB LED D3 will stay active until either one of the following two cases happen:

a) the Alarm is turned off
b) when the light sensor is blocked momentary causing its voltage to go above a 2.5V threshold and therefore will clear the match and terminate the buzzer sound and the activities on the RGB LED D3. **This later event will not disable the Alarm meaning that the alarm mode remains on.**

Guidance:

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

Here is the list of files in the sample code:

- 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_Alarm_Time.c : all the functions to support the setup of the alarm time are handled in this file
- Setup_Fan_Temp.c : the functions to allow the setup of the temperature for the fan are in this file
- Interrupt.c : this is where the interrupt handler and the code for the IR remote control function
- I2C_Soft.c : this is the original library file for the basic functions to handle the I2C protocol.
- I2C_Support.c : this is all the functions needed to interface to the DS1621 and DS3231 ICs
- Fan_Support.c : all the functions to support the fan operations should be included in this file
- ST7735_TFT.c : this is the original library file to support the LCD
- utils.c : utility functions are included in this files

2) The file 'Main.h' contains all the assignments for the pin signals. Modify them based on the schematics before testing the board.

3) The 'Setup_Time.c' is provided as a startup example to modify the time. The routine will start to read the actual time and uses it to setup 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 detail information will be discussed on the lecture on the implementation.

4) The other two files 'Setup_Alarm_Time.c' and 'Setup_Fan_Temp.c' are as their names indicate to change the time for the alarm and the set temperature for the heater function.

5) The main routine has the following tasks in its 'while (1)' loop:

    a. Check for the time change on 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() where :
            1. The duty cycle is always calculated based on the condition of the ambient temperature and the set temperature.
            2. The variable FAN is checked. If 0, then turn off the fan. Else, turn on the fan
        v. Check if the Clock Alarm function by calling the function 'Test_Alarm()' (see below)
        vi. Output the information on TeraTerm
        vii. Update the information on the LCD screen

b. Check if the INT0_flag is set to detect that the ALARMEN push-button switch is pressed
c. 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_Alarm_Time()
        3. 'CH+' for Setup_Fan_Temperature()
        4. 'Play/Pause' for Toggle_Fan_Monitor()
        5. '-' to decrease the duty cycle in the Fan Manual mode
        6. '+' to increase the duty cycle in the Fan Manual mode
        7. '0' to toggle the Fan Auto/Manual mode

6) The Clock Alarm allows the user to set a desired time for the alarm to sound and to generate a continuous light changing sequence on a RGB LED. When the Alarm switch is activated by pushing on INT0 on the main page, the unit will program the RTC chip to compare between the actual time and the alarm time. When the time matches, the signal RTC_ALARM# will be set to 0. The buzzer will be activated and a RGB LED will be changing its color every second. To reset the alarm, either the ALARM SW is toggled or the light to the photo sensor is blocked off.

   The routine 'Test_Alarm()' in main.c program is the place to implement this function. Follow the basic steps provided in that routine.

    1) The Alarm mode can be toggled by pushing on the INT0 push-button. When INT0 is detected to be 1, it indicates that the push-button has been pressed. This will change the logic state of a variable ALARMEN (initially at logic 0).
    2) Need to have a variable called alarm_mode to retain the actual state of the alarm operation.
    3) We will have to handle the following situations:
        a. When 'alarm_mode' is 0 and ALARMEN = 1. This means that no alarm operation was on and now it is activated. Need to call the function 'DS3231_Turn_On_Alarm();' to do the activation and set 'alarm_mode' to 1
        b. When 'alarm_mode' is 1 and ALARMEN = 0. This forces the alarm mode to be disabled. Need to call 'DS3231_Turn_Off_Alarm();' to turn off the alarm. Next, set 'alarm_mode' to 0, clear out any RGB LED and deactivate the buzzer
        c. When 'alarm_mode' is 1 and ALARMEN = 1. This is when the program is waiting for the alarm time to match with the actual time. The signal 'RTC_ALARM#' must be checked to be at logic 0 to indicate that the time matches. When that happens, activate the buzzer

and start the RGB LED to show the event. To keep the buzzer going and the RBG LED changing the color for every second, use a variable ‘MATCHED’ to show that the matching event did happen. This will continue until the voltage sampled from the light sensor is detected to be above 3.0V because that is when the photo sensor is blocked. At that time, ‘MATCHED’ will be cleared and the buzzer will be deactivated as well as the RGB LED will be turned off.