A logo of a company

Description automatically generated

California Polytechnic State University Pomona

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

# Microcontroller Laboratory

ECE 3301L-01

LAB #13

Final Project

Prepared by

Rayan Alruainy & Kaung Nyi Naing

Presented to

Felix Pinai

December 2, 2024

**Objectives:**

To design and implement a system that integrates various microcontroller functions covered during the semester, including GPIOs, ADC, PWM, interrupts, I2C, and SPI protocols, into a cohesive project and to develop a multifunctional system with features such as fan speed control with automatic and manual modes, temperature and RPM monitoring, time and date display, and remote control-based parameter adjustments, while utilizing a TFT LCD as the main interface for displaying real-time data.

**Summary:**

In terms of physical components, this lab has an additional photo sensor and push button compared to lab 11. Since most input signals are handled through the remote, only one pin is required to do all the complex control mechanisms. Before beginning the lab, we wired all components according to the schematics and double-checked our wiring to eliminate wiring problems while testing our software.

The first step in programming our final lab was to make sure that our power up and power down methods worked well. We have two methods to achieve this. One is to use the push button which will simply toggle between power on and off for every push. Another method is to use light sensors. We simply blocked the sensor a second to toggle the system on or off. We found that if our testing environment was dim, our system would toggle on and off constantly which was solved by increasing the voltage limit for the sensor in the program.

After power up, the LCD screen showed 7 rows of data. The first includes our class and group information. The second row showed the temperature in both Celsius and Fahrenheit along with a plus or minus sign. The third row displayed the time from hour to second. The fourth row showed the date in month, date, year format. The fifth row showed the fan timer mode and a count down time. The sixth row showed the set temperature and fan mode. This row also includes a handy fan on/off indicator which helped a lot during our debugging. The final row the duty cycle of the PWM signal, the light sensor voltage and the RPM of the fan. The light sensor voltage data helps me solve the system toggling problem as well.

The final project also has two extra “screens” which can be accessed using the “CH” and “CH-” buttons on the remote. These screens are used to set up different parameters and add extra complexity to our code since the same remote signals trigger different functions in each setup mode. In the time setup mode, we could update the system's time and date, which were displayed as hour:minute:second on one row and month/day/year on the next. A cursor highlighted the active field, and we used the remote-control buttons to navigate and modify the fields. The “Prev” button moved the cursor backward, wrapping around to the last field if it was at the first, while the “Next” button moved it forward, wrapping around to the first field if it was at the last. To adjust the values of each field, we used the “+” button to increase and the “-“ button to decrease. For the increase and decrease functions, we made sure that the values cannot exceed their logical limits. For example, the minute value cannot be greater than 59 or be less than 0. If these values do become larger or smaller, we simply reset them to zero or one for increasing and reset them to their largest value for decreasing. Once all updates were made, we could save the changes by pressing Play/Pause or exit without saving by pressing “EQ”. This functionality made it easy to adjust the system’s time and date in real time.

Another setup mode is the fan set temperature mode which was used to set the threshold temperature for the fan's automatic operation. If the ambient temperature exceeded the set threshold, the system calculates a duty cycle proportional to the temperature difference, with the fan speed increasing as the temperature rises. The calculation ensured that the duty cycle ranged between 0% and 100%, with no fan activity below the set threshold. In this mode, we used the “+” and “-“ buttons to increase or decrease the set temperature, which was limited to a range of 50°F to 110°F. Since there was only one field to modify, the “Prev” and “Next” buttons were not active. Similar to the Time Setup Mode, pressing Play/Pause saved the new value, while pressing “EQ” exited without saving. These setup modes made the system flexible by designating different screens for different tasks and allowed us to easily adjust parameters during operation.

The normal operating mode of the project focused on controlling the fan and its timer functionality. For the fan operation, we used the Play/Pause button to toggle the fan between ON and OFF states. When the fan was turned off, the FAN variable, FAN\_EN signal and the FANON\_LED were deactivated, ensuring the fan was completely disabled. Turning the fan on reactivated these signals and allowed the fan to operate at a speed determined by the duty\_cycle variable. We could control the fan using two modes: Manual and Auto. Pressing the EQ button switched between these modes. In Manual mode, we had full control over the fan speed and used the “+” and “-” buttons to increase or decrease the duty cycle. In Auto mode, the system adjusted the fan speed automatically based on the temperature difference between our value from the setup mode and the current temperature, with higher differences resulting in faster speeds. The RGB LEDs D1 and D2 displayed the duty cycle and RPM visually. The duty cycle and RPM of the fan are also shown on the LCD screen and the Tera Term Terminal which allows us check if our screen is displaying the correct values.

The project also have a fan timer functionality which added another layer of control. Using the “Prev” and “Next” buttons, I could select from three timer modes TM1 (5 sec), TM2 (10 sec), TM3 (20 sec) or turn the timer off entirely. The remaining time for the active mode was displayed in the Timer Time field on the LCD. When the fan was turned on, the timer began counting down, and once it expired, the fan was automatically shut off. If the fan was off, the timer value stayed static. This feature made it so that we have fan control over time which combined with the fan control over temperature reassembled smart fans that are available in the market.

**Conclusion:**

In conclusion, this final lab served as a culmination of everything we’ve learned throughout the semester. It required us to combine a wide range of microcontroller concepts, including GPIOs, ADC, PWM, I2C, SPI, and interrupts, into a fully integrated project. By designing a system with fan speed control, temperature and RPM monitoring, real-time time and date display, and remote-controlled parameter adjustments, we were able to see how these individual functions come together to create a complex yet cohesive system. The use of the TFT LCD for displaying real-time data further emphasized the importance of presenting information clearly and effectively in embedded systems.

Through careful wiring, and software debugging, we successfully implemented a system that operates smoothly across various modes, including power-up/power-down, setup screens, and normal operation modes. This lab not only tested our understanding of the material but also challenged us to think critically about design, integration, and user interaction. Completing this lab gave us valuable insight into how microcontroller systems are used in real-world applications, such as smart home devices, and solidified our understanding of embedded system design.