

ECE 56800: Embedded Systems

Spring 2025

Lab 2 - LED control using ADC, PWM, Timers and Interrupts

To be done individually; Due by 11:59pm, Tuesday, March 4, 2025.

1. Overview

This assignment deals with 3 different peripherals, namely the PWM, ADC, and RTC on the ESP32 V2 Feather Board. In addition to these peripherals, you will also be using timers and interrupts to *implement LED control using a potentiometer and switch*. The hardware and software implementation details for this assignment are described in the following sections.

2. Programming Exercise

2.1. Hardware Interfacing

Interface the following components to the board:

- The on-board **push button switch** as a GPIO (*digital*) input (**nothing to interface here!**)
- An external **potentiometer** as an *analog* input.
- One external **LED** as a *PWM* (Pulse Width Modulation) output.

2.2. Software Implementation

Your program should implement the following functionalities.

- The program should start by asking the user to input the current date and time as shown in the following format. Use the user inputs to initialize the RTC (real time clock). Use the current time in the EDT time zone for entering the time.

NOTE: *Weekdays: Enter 0 for Monday, 1 for Tuesday, 6 for Sunday.*

```
>>> python ghosh37_lab2.py
Year? 2022
Month? 9
Day? 27
Weekday? 2
Hour? 7
Minute? 30
```

```
Second? 00  
Microsecond? 000000
```

- Use the real-time clock (RTC) and a hardware timer to display the current *date* and *time* every 30 seconds. Display these times on 2 separate lines and ensure that both dates and times are immediately recognizable with a formatted print statement. Do not use *time.sleep()*. Use the **RTC** and a **timer interrupt/callback** instead. See this URL for more information on callbacks & interrupts in MicroPython.
<https://docs.micropython.org/en/latest/library/machine.html#machine-callbacks>
- Initialize another hardware timer and read the *analog input* (potentiometer/pot values) every 100ms using the **timer interrupt/callback** and **ADC**. **IMPORTANT: Connect the potentiometer only to a pin associated with ADC1 and not ADC2.**
- Initialize and start a **PWM** signal on the external LED using a *frequency* of 10 Hz and a *duty cycle* of 512 (50%). The LED should start blinking at the defined frequency.
- In the beginning, rotating the pot should have no effect on the LED. The LED should continue blinking at predefined intensity and frequency.
- Detect a **switch press** using an **interrupt/callback**. Implement switch debouncing using another timer-based interrupt/callback.
 - Alternate switch presses should direct control towards either the frequency or the duty cycle of the LED.
 - The LED's PWM signal **frequency** and **duty cycle** should be controlled by the pot readings.
 - When you press the switch for the first time, the pot should control the LED's PWM **frequency**. Now, if you rotate the pot, your LED should blink **faster or slower**, as the frequency of PWM changes. **No change should occur in the LED's intensity.**
 - When you press the switch for the second time, the pot should control the LED's PWM **duty cycle**. Now, if you rotate the pot, your LED's **intensity** should be **higher or lower**, as the duty cycle of PWM changes. **No change in the LED's blinking frequency.**
 - The third switch press should revert the control back to the frequency, the fourth press should give control to the duty cycle and the process should continue forever.

3. Submission

Make sure you follow these instructions precisely. You need to turn in your code on Brightspace. Please create a directory named **username_lab2**, where username is your CAREER account login ID. This directory should contain only the following files, i.e., no executables, no temporary files, etc.

1. **lab2.py**: your program
2. **A README file** named **username_lab2_README.txt** that contains **a description of your hardware connections** (which pins you used, exactly what they were connected to, etc.) and **a YouTube link** that shows you demonstrating your working solutions (explain your hardware connections briefly before demonstrating the working solution).

Zip the directory and name it **username_lab2.zip** and upload the .zip file to Brightspace.

Note: Do not directly upload your video on Brightspace!! Upload your video on YouTube and copy paste the link in the README file.

4. Grading

We will use manual code inspection for evaluation. If your code is unusually complex or different from expected solutions, you may be asked to attend an office hour and explain your code.

REFERENCES

- [1] Getting started with MicroPython on the ESP32
<https://docs.micropython.org/en/latest/esp32/tutorial/intro.html>
- [2] ESP32 PICO MINI 02 Datasheet
https://www.espressif.com/sites/default/files/documentation/esp32-pico-mini-02_datasheet_en.pdf
- [3] ESP32 Technical Reference Manual
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
- [4] Adafruit HUZZAH32 – ESP32 V2 Feather Online Manual
<https://learn.adafruit.com/adafruit-esp32-feather-v2>
- [5] Adafruit ESP32 Feather Pinouts: <https://learn.adafruit.com/adafruit-esp32-feather-v2/pinouts>