ECE40862: Software for Embedded Systems

Fall 2019

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

To be done individually; Due by 11:59pm, Tuesday, September 10, 2019.

1. Overview

This assignment deals with 3 different peripherals, viz., **PWM, ADC, and RTC** on the ESP32 Feather Board. In addition to these peripherals, you will also be using **timers** and **interrupts** to *implement LED control using 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:

- An external **push button switch** as a GPIO (*DIGITAL*) INPUT**.**
- An external **potentiometer** as an *ANALOG* INPUT.
- Two external **LED**s (red and green) as *PWM* (Pulse Width Modulation) OUTPUTs.

2.2. Software Implementation (Upload as *username*_lab2_adc_pwm.py)

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

```
>>> python ghosh37_lab2_adc_pwm.py
Year? 2019
Month? 9
Day? 3
Weekday? 1
Hour? 7
Minute? 30
Second? 00
Microsecond? 000000
```

- Initialize a **hardware timer** and display the current *date* and *time* every **30 seconds**. DO NOT USE *time.sleep()* for this purpose. Use RTC and Timer interrupt/callback. See this webpage for more information on callbacks and interrupts in MicroPython. https://docs.micropython.org/en/latest/library/machine.html#machine-callbacks
- Initialize a **software timer** and read *analog input* (potentiometer/pot values) every **100ms** using Timer interrupt/callback and ADC.
- Initialize and start PWM signals on the external LEDs using a *frequency* of 10 Hz and a *duty cycle* of 256. The LEDs should start blinking at the defined frequency.
- In the beginning, **rotating the pot should have NO EFFECT**. Both LEDs should continue blinking at predefined intensity and frequency.
- Detect **switch press** using **interrupt** (*this interrupt is separate from the initialized timers*)
    - Alternate switch presses should direct control towards either of the LEDs.
    - The red LED's PWM signal **frequency** should be controlled by the pot readings.
    - The green LED's PWM signal **duty cycle** should be controlled by the pot readings.
    - When you press the switch for the first time, the pot should control the red LED's PWM **frequency**. Now, if you rotate the pot, your red LED should blink **faster or slower**, as the frequency of PWM changes. No change in green LED.
    - When you press the switch for the second time, the pot should control the green LED's PWM **duty cycle**. Now, if you rotate the pot, your green LED's **intensity** should be **higher or lower**, as the duty cycle of PWM changes. No change in red LED.
    - The third switch press should revert the control back to the red LED, the fourth press should give control to the green LED and the process should continue.

3. Submission

Make sure you follow these instructions precisely. Points will be deducted for any deviations. You need to turn in your code on Blackboard. Upload your source code as ***username*_lab2_adc_pwm.py** where username is your CAREER account login ID. *DO NOT ZIP THE FILE.*

4. Grading

15 students will be randomly selected for lab demo, they will be informed on Wednesday, 11th September, 10 AM via the course mailing list. If you are selected for the demonstration, you will need to login to any of the computers in the EE217 lab, download your submitted source codes from Blackboard, copy them to your ESP32 board *using rshell* and show the TA that your code works correctly. You will also be asked to show the TA your code listing and answer conceptual questions about your code. Marks are only awarded if you satisfy all requirements.

In addition, automatic evaluation scripts will be executed on your uploaded source codes on Blackboard and you will be graded according to the evaluation results.

REFERENCES

[1] Getting started with MicroPython on the ESP32
https://docs.micropython.org/en/latest/esp32/tutorial/intro.html

[2] ESP32 WROOM-32 Datasheet
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_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 Feather Online Manual https://learn.adafruit.com/adafruit-huzzah32-esp32-feather

[5] Adafruit ESP32 Feather Schematics https://cdn-learn.adafruit.com/assets/assets/000/041/630/original/feather_schem.png?1494449413

[6] MicroPython GitHub https://github.com/micropython/micropython

[7] REPL http://docs.micropython.org/en/latest/esp8266/tutorial/repl.html

[8] Thonny IDE https://thonny.org

[9] Rshell GitHub https://github.com/dhylands/rshell