ECE 568: Embedded Systems

Spring 2022

Lab 1 – Exercising peripherals (Interrupts, Timers, RTC, GPIO, PWM)

To be done individually; Due by 11:59pm, Sunday, March 6, 2022.

1. Overview

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

2. Flash MicroPython on your ESP32 Feather Board

Follow the instructions (separate PDF file uploaded) to flash MicroPython on your ESP32 board.

3. Lab Exercise

3.1. Hardware Interfacing

Interface the following components to the board:

- One external push button switch as a GPIO (*digital*) input.
- One external LED as a *PWM* (Pulse Width Modulation) output.

3.2. Software Implementation

Your program should implement the following functionality.

- 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.*

```
Year? 2022
Month? 2
Day? 24
Weekday? 3
```

```
Hour? 13
Minute? 30
Second? 00
Microsecond? 000000
```

- Use the real-time clock (RTC) and a hardware timer to print the current *date* and *time* every 10 seconds. 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 and start a PWM signal on the external LED using a *frequency* of 1 Hz and a *duty cycle* of 256. The LED should start blinking at the 1 Hz frequency.
- Detect a switch press using an interrupt/callback. Implement switch debouncing using another timer-based interrupt/callback. The switch press in intended to affect the LED's blink rate (*i.e.,* the PWM frequency). No change should occur in the LED's intensity.
    - When you press the switch for the first time, the LED should start blinking faster at a frequency of 5 Hz.
    - When you press the switch for the second time, the LED should go back to blinking slowly at a frequency of 1 Hz.
    - The third switch press should result in a fast blink (5 Hz), the fourth press should result in a slow blink (2 Hz), and so on...

4. Submission

You need to upload three files on Brightspace.

(a) Your source code. Upload your source code as ***username*_lab1.py** where ***username*** is your Purdue CAREER account login.

(b) A README file named ***username*_lab1_*README.txt***. It should contain a very short (few lines) description of your hardware connections. which pins you used, exactly what they were connected to, etc.).

(c) A short (less than a minute) video clip that shows you demonstrating your working solution. Don't worry too much about a nice-looking video – I just want to see your solution working. For example, you can just use your phone to record a short clip.

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