ECE40862: Software for Embedded Systems

Fall 2020

Lab 3 – Networking with Sleep and Wake-Up with Touch Sensors

To be done individually; Due by 11:59pm, Thursday, October 15, 2020.

1. Overview

This assignment deals with connecting your ESP32 board with the Internet and exploring sleep modes and different wake up sources in esp32. You will also be using the **RTC** module and the on-board **TOUCH Sensor** in this lab. In addition to these peripherals, you will be using **timers** to *implement* different functionalities. 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:

- Two external **LED**s (red and green) as GPIO OUTPUTs.
    - Red LED should indicate if board is in *sleep* mode (OFF) or *awake* (ON).
    - Green LED should be controlled by Touch based input (Table 1).
- Two external **push button switches** as a GPIO (*DIGITAL*) INPUTs**.**
- Two external **Male-Male Jumper wires** connected to two *Touch-enabled* pins (Insert in the Stacking Headers on board). You can select any 2 pins out of the 10 capacitive-touch enabled pins on ESP32 board. This webpage gives more information: http://docs.micropython.org/en/latest/esp32/quickref.html#capacitive-touch.

  NOTE: The pin numbers indicated on the webpage is for the ESP32 microcontroller. All those pins are not available to you on the Feather Board. Specifically, 6 out of the 10 touch-enabled pins are available for use. You must figure out those 6 pins by trial and error (HINT: check for 'ValueError')
    - 1st Touch-enabled wire should be used to wake up the board from sleep.
    - 2nd Touch-enabled wire should control blinking of green LED as indicated in the following sections.

2.2. Software Implementation (main.py)

Your program should implement the following functionalities.

2.2.1.Connect to Internet

- The program should use the *network* module in MicroPython to connect your ESP32 to a Wi-Fi network using '*SSID'* and '*Password*' for that network. You can create a Wi-Fi hotspot in your mobile/laptop and then connect your ESP32 to the hotspot.

  e.g. Your mobile hotspot *SSID*: 'Lenovo-SSID' and *Password*: '12345678', then use this *SSID* and *password* for connecting ESP32 to the internet.

  NOTE: You can also connect to any other Wi-Fi network which works directly with SSID and password. Support for Enterprise networks like PAL3.0 which requires SSID, username, password, certificates, etc. is NOT PRESENT in MicroPython.

- Once the board successfully connects, the program should print out the interface's MAC address and IP address.

```
Oh Yes! Get connected
Connected to Lenovo-SSID
MAC Address: ab:cc:c2:b7:e6:fb
IP Address: 192.168.0.107
```

2.2.2. Display Current Date-Time

- **Getting Time from Internet**: The program should fetch the *current date and time* from the NTP server '**pool.ntp.org**' and use it to initialize the **RTC** (**real time clock**), after converting UTC (Coordinated Universal Time) to local time zone. HINT: check module *ntptime.*

- (**SAME AS LAB 2**) Initialize 1st **hardware timer** and display the current *date* and *time* every **15 seconds**. DO NOT USE *time.sleep()* for this purpose. Use RTC and Timer interrupt/callback.

```
Date: 10/08/2020
Time: 10:00:00 HRS
```

2.2.3. Green LED Control by Touch Input

- Initialize the two Touchpad-enabled pins connected to the jumper wires and perform calibration to observe touchpad pin values when you physically touch the jumper wires.

- Initialize 2nd **hardware timer** and read *2nd touch pin* values every **10 milliseconds** using Timer interrupt/callback and implement the following pattern. Use calibrated values to detect whether wire is touched or not.

  - TouchPin2 NOT touched: Green Led OFF
  - TouchPin2 touched: Green Led ON

2.2.4. DEEP SLEEP and Different Wake Up Sources

- Initialize 3rd **hardware timer** to put the ESP32 into **DEEP SLEEP** every **30 seconds** for a duration of 1 minute. Print out a message on the terminal before going to sleep like:

```
I am awake. Going to sleep for 1 minute
```

  WAKE UP duration: **30 seconds** (3rd hardware timer)

  SLEEP duration: **1 minute** (do not need a timer for this, check this link: https://docs.micropython.org/en/latest/esp32/quickref.html#deep-sleep-mode)

- You will be using three different modes of waking up from deep sleep in this lab. Your program should be checking for all three sources, board should be able to wake up from any of these sources.

2.2.4.1. Wake up Sources in ESP32

NOTE: Whenever the board is awake, the red LED should be on.

- **Timer Wake Up (TIMER_WAKE)**: Wake up the ESP32 after predefined SLEEP duration (1 minute) and print out it's a timer wake-up.
- **Touch-based Wake Up (TOUCHPAD_WAKE)**: Initialize the 1st *touch pin* as a wake-up source for the ESP32. If you touch the jumper wire connected to this touch pin within the 1-minute sleep duration, your board should wake-up and print out that it's a touchpad wake-up. Pressing the 2nd touch pin should have no effect.

  NOTE: If 1-minute passes and no touch is detected, then also your board should wake up which is the Timer Wake Up.

- **External Wake Up Mode 1 (EXT1_WAKE)**: Configure both switches as external wake-up sources. Pressing any one of the two switches should wake up the board within 1-minute sleep duration and print out it's an EXT1 wake-up.

2.3. Overall Program Flow and Sample Output

The board is powered-on for the first time. It connects with the Wi-Fi network, fetches current time from NTP server, displays them after 15s, and blinks green led whenever 2nd touch pin wire is touched. It again displays the date and time at 30s and goes to deep sleep. The program now simultaneously checks for the 1st touch pin wake-up signal as well as both push button wake-up signals. When you press any of the switch, or touch the 1st touch pin, or if 1 minute expires, the board wakes up and starts the overall process again. Table 1 shows the pattern of LED blinking to be followed. The sample output is also provided here for your understanding. Here, we assumed that the program has started at 10:00:00 hrs. EST

Table 1. Sleep and Wake Up Program Flow

Mode	Touchpin1	Touchpin2	Switch1	Switch2	Red Led	Green Led
Sleep	No Press: No Effect	No Press: No Effect	No Press: No Effect	No Press: No Effect	Off	Off
Sleep	Press: Wake-Up	Press: No Effect	Press: Wake-Up	Press: Wake-Up	Off	Off
Awake	No Press: No Effect	No Press: No Effect	No Press: No Effect	No Press: No Effect	On	Off
Awake	Press: No Effect	Press: Detect	Press: No Effect	Press: No Effect	On	On

Sample Output

```
I (200) wifi: wifi driver task: 3ffe2c34, prio:23, stack:3584, core=0
.....
.....
.....
I (22233) network: CONNECTED
I (22883) event: sta ip: 192.168.0.107, mask: 255.255.255.0, gw: 192.168.0.1
I (22883) network: GOT_IP
Oh Yes! Get connected
Connected to Lenovo-SSID
MAC Address: ab:cc:c2:b7:e6:fb
IP Address: 192.168.0.107

Date: 10/08/2020
Time: 10:00:15 HRS

Date: 10/08/2020
Time: 10:00:30 HRS

I am awake, but going to deepsleep

ets Jun  8 2016 00:22:57

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee.....
.....
.....
I (0) cpu_start: Starting scheduler on APP CPU.

Woke up due to Touchpad
I (200) wifi: wifi driver task: 3ffe2c34, prio:23, stack:3584, core=0
///continues
```

Wi-Fi connection messages posted by ESP

Default Messages by ESP on Wakeup and Reset

3. Submission

Make sure you follow these instructions precisely. Points will be deducted for any deviations. You need to turn in your code on Brightspace. Upload your source code as ***username*_lab3.py** where username is your CAREER account login ID. *DO NOT ZIP THE FILE*.

NOTE: Follow the lab document strictly when using different peripherals/modules/packages. Points will be deducted if you fail to follow the lab instructions. If anything is NOT mentioned explicitly, you can use package/module to write your program.

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

[10] Sleep Modes API in C https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/sleep_modes.html

[11] Touch Sensor API in C https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/touch_pad.html

[12] ESP32 specific functionalities in MicroPython
http://docs.micropython.org/en/latest/library/esp32.html