ECE40862: Software for Embedded Systems

Fall 2019

Lab 1 – Digital Input and Output using GPIOs, LEDs, Switches

To be done individually; Due by 11:59pm, Tuesday, September 3, 2019.

1. Overview

This assignment explains how to get started with the **Adafruit HUZZAH32 - ESP32 Feather** board and how to program the board using MicroPython and Thonny IDE. The goal of this experiment is to walk you through the *embedded software development* flow for the ESP32, where you will learn how to use microcontroller I/O, the inputs and outputs, to sense and actuate devices in the world outside the board.

The assignment is subdivided into two sections. In the first section, you will learn to implement a simple LED blink program that will give you a high-level idea of how you can accomplish the remaining labs. In the second section, you will interface two **LEDs** (Red and Green) and 2 **Push Button Switches** with the board and implement a simple pattern function using them.

The ESP32-WROOM-32 module on the Feather Board has 38 pins, many of which are GPIO (general-purpose input/output) pins. However, the feather board has only 28 pins and has its own adhoc pin numbering (marked e.g. A0, A1...) with many peripheral pins (like TX, RX, SCL, SDA...) indicated on the board itself. You need to explore the ESP32-WROOM-32 datasheet and Feather board manuals and Schematics to figure out the mapping between physical chip pins and board logical pins, and interface LEDs and Switches with the board pins.

2. Programming ESP32 using Thonny IDE

After successfully installing necessary software outlined in *Lab0 (section 2.1 to 2.5)*, you need to flash MicroPython on the ESP32 board before you can do any programming. The implementation procedure explained in this section demonstrates a typical procedure to run any program on the board.

2.1. Connect the Board to PC and Flash MicroPython

Connect the board to your PC using the Micro USB cable (just plug into the jack) and the Feather will regulate the 5V USB down to 3.3V which will power the board. In addition, you

can also connect the Lithium ion polymer (Lipo/Lipoly) battery to the JST jack (if you have ordered it), which will get continuously charged by the USB Power (OPTIONAL).

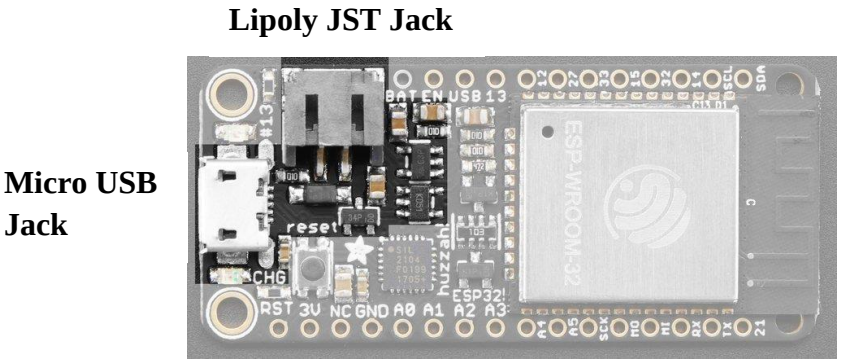

Figure 1. Board Power and PC Connections

Follow the instructions from *Lab0, section 2.6* to install MicroPython on your board. You can get more information on starting MicroPython on your ESP32 board on this webpage: http://docs.micropython.org/en/latest/esp32/tutorial/intro.html#esp32-intro.

NOTE: If you are not able to flash MicroPython on your ESP32 board using your own PC, you can use the EE217 Lab machines to perform the flashing.

2.2. Testing Thonny IDE Installation

Important: Before testing the installation, your ESP32 board needs to be flashed with MicroPython firmware. Connect the board to your PC and open Thonny IDE. To test the installation, you need to tell Thonny that you want to run MicroPython Interpreter and select the board you are using.

1. Go to **Tools** > **Options** and select the **Interpreter** tab. Select **MicroPython(generic).**
2. Select the device serial port: **Silicon Labs CP210x USB to UART Bridge (COMPort)**

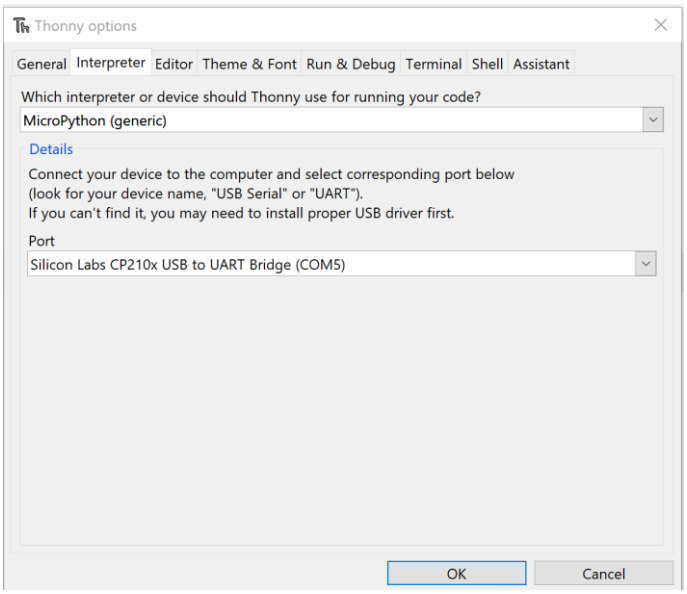

Figure 2. Thonny Interpreter and Port selection

Thonny IDE should now be connected to your board. The board will start and run the files '**boot.py**' and '**main.py**' (if any) automatically and provide you a MicroPython REPL shell. A read–eval–print-loop (REPL), also termed interactive top-level or language shell, is a simple, interactive computer programming environment that takes single user inputs (i.e., single expressions), evaluates (executes) them, and returns the result to the user, shown in Figure 3.

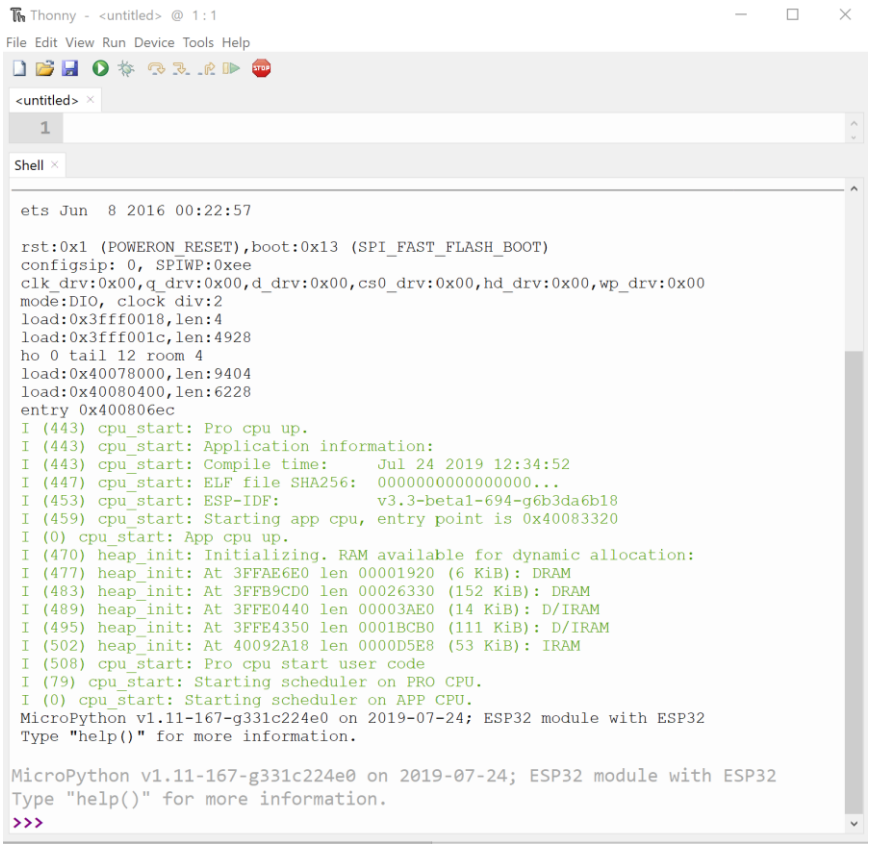

Figure 3. MicroPython startup on ESP32 in Thonny IDE

2.3. Using the REPL

Once you have a prompt, you can start experimenting! Anything you type at the prompt will be executed after you press the *Enter* key. MicroPython will run the code on the board that you enter and print the result (if there is one). If there is an error with the text that you enter, then an error message is printed. You can find more information on REPL at http://docs.micropython.org/en/latest/esp8266/tutorial/repl.html (MicroPython execution is similar in ESP8266 and ESP32).

Figure 4. Experimenting with REPL

Type the command **help()** in the Shell and see if it responds back. If it responded back, everything is working fine.

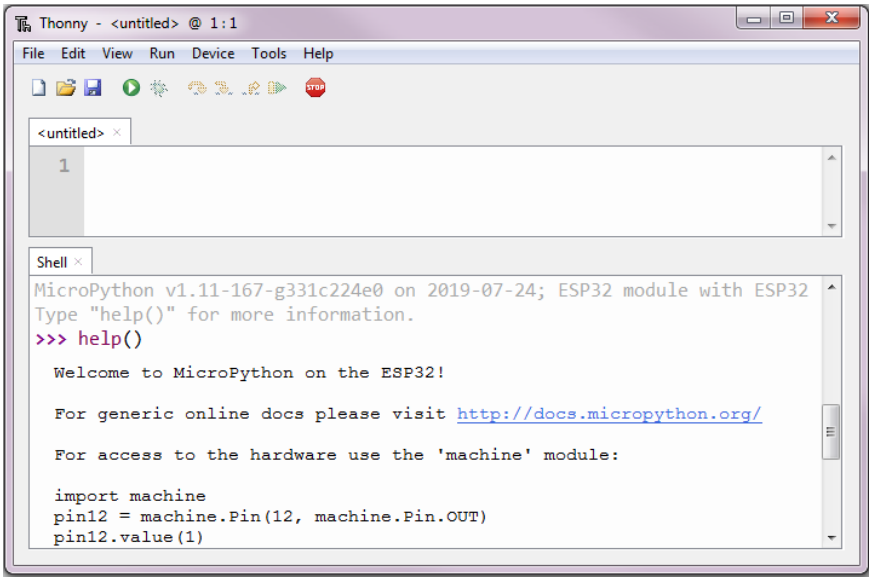

Figure 5. Testing the REPL

2.4. First Program on the ESP32

In the EDITOR of the Thonny IDE, write your first ***'Hello World'*** program and save it on the board as **main.py**. The IDE asks for location to save the file, **Select** *MicroPython device* to save the file on the flash memory on the board, as shown in Figure 6. Pressing **Ctrl+D** or selecting Run > Send EOF/Soft Reboot in Thonny will perform a soft reboot of the board and as mentioned earlier, it will run **main.py** and you will see the output on the MicroPython shell. You can also run the program **main.py** from the shell by calling the following command, which will run the code inside main.py and give you output on the shell, both of which are shown in Figure 7. Similarly, any code you write for your labs, you can save them on the board and import your program module in MicroPython shell to run the code.

```
import main
```

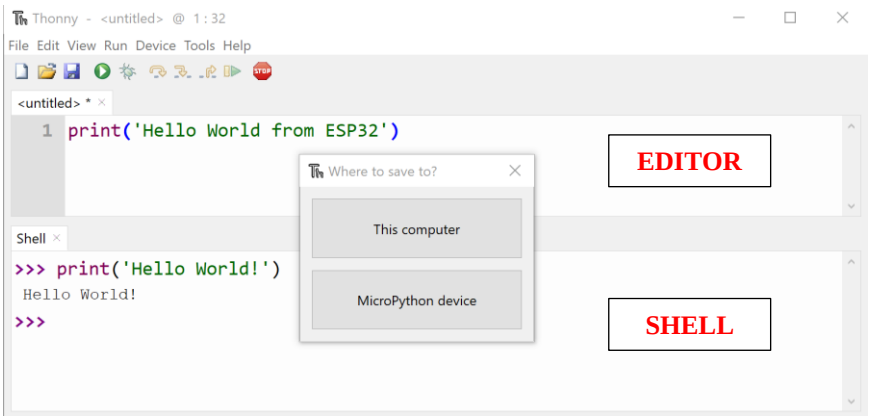

Figure 6. Saving file on ESP32 board

Figure 7. Reboot and Running main.py

2.5. The internal filesystem

Since ESP32 Feather has 4MB flash memory, a default filesystem will be generated upon first boot, which uses the FAT format and is stored in the flash after the MicroPython firmware. You can open any file already stored inside the board using Thonny IDE, modify it and save it back on the board. More details on working with the internal filesystem are available at this webpage: http://docs.micropython.org/en/latest/esp8266/tutorial/filesystem.html.

3. Rshell Basics

To use **rshell** to communicate with your ESP32 board, connect your board to the PC, and type the following command. You can also specify other arguments like *--editor* EDITOR_NAME (gedit/vim/…), *--quiet*, etc.

```
>> rshell -port /dev/ttyUSB0
```

For the ESP32 board, you can only reference its directory by using the board name ***pyboard***. To copy your source codes from your local directory to the board, use the following command.

```
>> cp SOURCE_FILE1 SOURCE_FILE2 /pyboard/
```

NOTE: Although your board is NOT pyboard, rshell uses the **same parent folder name** for all ESP based boards. You can however create multiple folders inside **/pyboard/** to organize files for different labs.

4. Programming Exercises

4.1. Simple Blink Program (main.py)

The example source code below blinks the Red LED on the ESP32 Feather board 5 times with the interval of roughly 0.5 seconds. Use this source code and save as **main.py** file on your board and perform a *soft reboot*. Use the Feather board **schematics** to figure out the pin X connected to the red LED on the board.

```
from machine import Pin
from time import sleep

# Onboard RED LED is connected to IO_X
# FIND OUT X FROM SCHEMATICS AND DATASHEET
# Create output pin on GPIO_X
led_board = Pin(X, Pin.OUT)

# Toggle LED 5 times
for i in range(10):
    # Change pin value from its current value, value can be 1/0
    led_board.value(not led_board.value())
    sleep(0.5)    # 0.5 seconds delay

print("Led blinked 5 times")
```

4.2. LED Pattern Display using Switches (pattern.py)

Now that you have gone through the warm-up program above, you can finally get to the actual problem statement for Lab 1. Implement a simple pattern function by interfacing two **push button switches** and two external **LEDs** (red and green) with the ESP32 Feather board that performs the operation given in Table 1. The **switches** are considered to be ON when they are pushed down and OFF when they are released.

Table 1. LED Pattern for Lab 1

Switch1	Switch2	Red LED	Green LED
OFF	OFF	OFF	OFF
OFF	ON	OFF	ON
ON	OFF	ON	OFF
ON	ON	OFF	OFF

In addition, you have to determine the **number of times Switch1 and Switch2 are pressed** from the start of the program. If any one of the switches is **pressed 10 times**, the pattern in Table 1 should be stopped and the two **LEDs** should start blinking alternatively. This should continue until your program detects a **switch press** on the **other switch** (i.e., NOT the switch which was pressed 10 times, pressing it again should not have any effect).

Sample Program Flow:

- The user starts *testing* the pattern function; the program starts turning the LEDs ON and OFF depending upon switch presses.
- Let's assume that throughout the *testing*, **Switch1** is pressed **9** times and **Switch2** is pressed **5** times. On the next **Switch1** press (the 10th press), the LEDs start blinking alternatively. The user presses **Switch2**, both LEDs turn OFF and the program displays a message on the MicroPython shell '**You have successfully implemented LAB1 DEMO!!!**' and exits to the shell.

5. Submission

Make sure you follow these instructions precisely. Points will be deducted for any deviations. You need to turn in your code on Blackboard. Please create a directory named *username*_lab1, 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. *main.py*: simple blink program (**3.1**)
2. *pattern.py*: your implementation of the pattern system (**3.2**)

Zip the file and name it as *username*_lab1.zip and upload the .zip file to Blackboard.

6. Grading

15 students will be randomly selected for lab demo, they will be informed on Wednesday, 4th 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