ECE 40862: Software for Embedded Systems

Fall 2023

Lab 1 – Introduction to Python and – Digital Input and Output using GPIOs

To be done individually; Due by 11:59pm, Sunday, September 24, 2023.

1. Overview

This assignment will familiarize you with the Python programming language and the Thonny Python IDE editor, which we will use throughout the semester. Python is an easy-to-learn but powerful programming language and is one of the fastest growing programming languages for embedded systems. In this course, we will use MicroPython to program the ESP32 board. MicroPython is a lean and efficient implementation of the Python3 programming language. Before working with MicroPython on your board, you should acquaint yourself with Python.

2. Getting started with Python and MicroPython

The following sections describe various software installation procedures.

2.1. Install Thonny IDE on your own PC

Thonny is a Python IDE meant for learning programming. You can find more details about this IDE and download it at https://thonny.org. **Install the latest version (4.1.2)**. Installation instructions for Windows, Linux, and Mac OS are provided on the above webpage. By default, Thonny comes with Python 3.10 built in, so a separate installation of Python is not needed.

2.2. Install USB-Serial Driver on your host PC (Windows and Mac only)

Follow the guide available at https://learn.adafruit.com/how-to-install-drivers-for-wch-usb-to-serial-chips to install the drivers for the USB to serial converter chip that the board uses to communicate with your host PC. There are drivers for Windows and Mac computers available. If you're on Linux, the drivers are built in.

2.3. Flashing MicroPython on ESP32 (using Thonny)

The latest version of Thonny (4.1.2) comes preinstalled with the package 'esptool', which is a Python-based, open source, platform independent, utility to communicate with the ROM bootloader in ESP32. Esptool can be used to load MicroPython on your ESP32 board.

2.3.1. Acquiring MicroPython Firmware

The esptool utility will help you download the latest ESP32 firmware.

2.3.2. Thonny Setup and Flashing

- Plug in your ESP32 board to your computer using a USB-C cable, launch Thonny and navigate to "Run > Configure Interpreter".
- Under the "Which interpreter or device should Thonny use for running your code?" label, select "MicroPython (ESP32)".
- Select the COM port to which your ESP32 is connected under "Port" (see Figure 1).

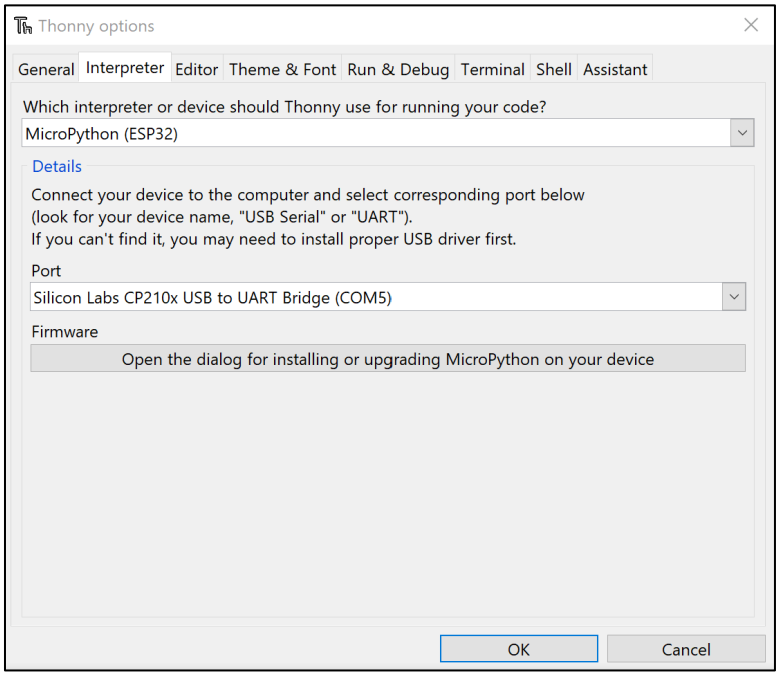

Figure 1. Set the interpreter and port

- After setting the interpreter and the port, click on the link at the bottom right: "Install or Update MicroPython". Select your device's port again, then select ESP32 as the MicroPython family, Espressif – ESP32 / WROOM as the variant, and 1.20.0 as the version. If you selected the options correctly, the information field should show https://micropython.org/download/ESP32_GENERIC/ corresponding to the **v1.20.0 (2023-04-26) .bin** firmware**.** If you are flashing MicroPython on your ESP32 board for

the first time, then make sure that “Erase flash before installing” is checked, then click “Install”, as shown in Figure 2.

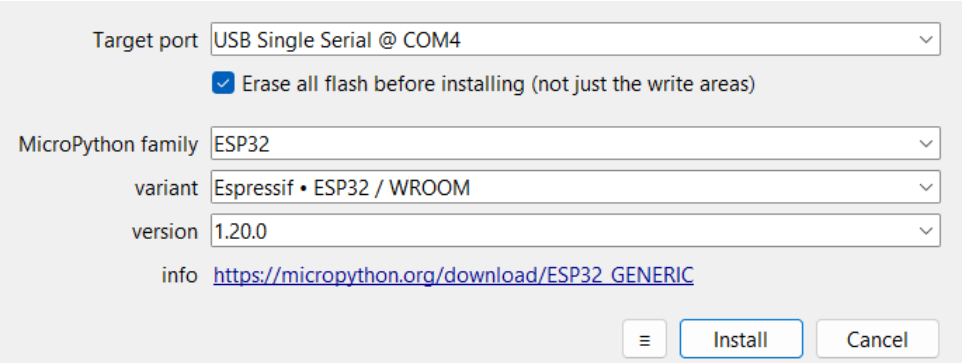

Figure 2. Select the port and downloaded MicroPython firmware

Thonny will now erase any existing firmware, and then flash the new MicroPython firmware on your ESP32, as you can see in Figure 3 and 4. This may take a few minutes. After the firmware has finished installing, you may close all the open dialogs.

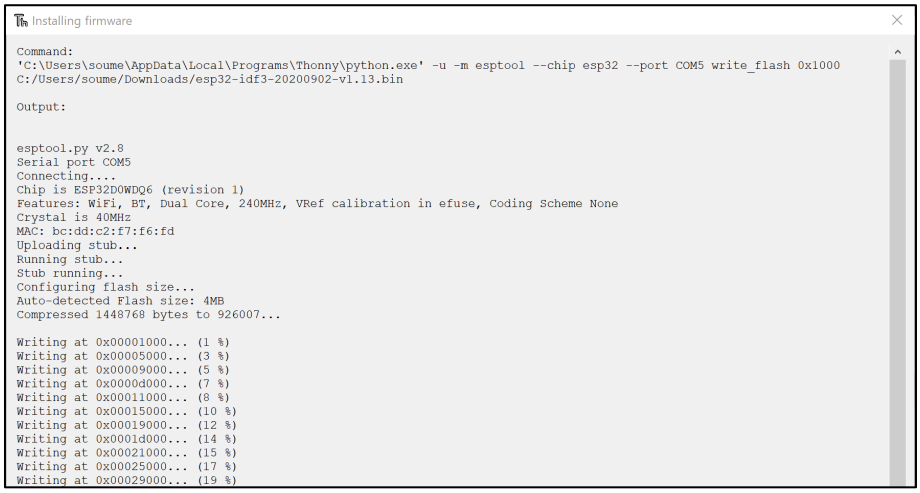

Figure 3. Flashing progress

Figure 4. Thonny has completed flashing the firmware on the microcontroller.

- At the bottom of the Thonny window in the tab labeled “Shell”, you should see text that resembles Figure 5.

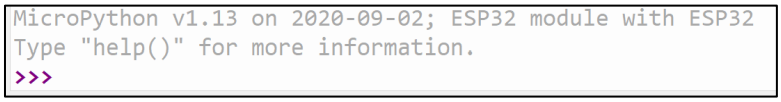

Figure 5. MicroPython REPL ready for input

Your ESP32 board is now ready for programming.

3. Python warm up programming exercises

You DO NOT need to use ESP32 for this part. You can use the default python interpreter.

3.1. Character Input (Upload as program1.py)

Write a program that asks the user to enter name and age. Print out a message that tells the year the user will turn 100 years old.

Hint: Use **input** command to get user input

```
>>> python program1.py
What is your name? Xavier
How old are you? 25
Xavier will be 100 years old in the year 2094
```

3.2. Lists and Conditional Statements (Upload as program2.py)

Write a program to initiate a list of 10 numbers, print the list, and ask the user for a number and return a list that contains only elements from the original list that are smaller than the number given by the user.

```
>>> python program2.py
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
Enter number: 25
The new list is [1, 1, 2, 3, 5, 8, 13, 21]
```

3.3. Loops

3.3.1. While loop: (Upload as program3a.py)

Write a program to get the Fibonacci series between 0 to a **user input number** using a **while** loop.

```
>>> python program3a.py
How many Fibonacci numbers would you like to generate? 10
The Fibonacci Sequence is: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

3.3.2. For loop: (Upload as program3b.py)

Write a program that generates a random number (0-10) and asks the user to guess it within three chances. If user guesses correctly, print 'You win!', otherwise print 'You lose!'

```
>>> python program3b.py
Enter your guess:3
Enter your guess:2
Enter your guess:10
You win!
```

3.4. Dictionary (Upload as program4.py)

Write a program to create a dictionary of 5 names and birthdays. Upon execution, it should ask the user to enter a name, and return the birthday of that person back to them.

```
>>> python program4.py
```

```
Welcome to the birthday dictionary. We know the birthdays of:
Albert Einstein
Benjamin Franklin
Ada Lovelace
Whose birthday do you want to look up?
Benjamin Franklin
Benjamin Franklin's birthday is 01/17/1706.
```

3.5. Class and Functions (Upload as program5.py)

Write a program to create a Python class to find a pair of elements (indices of the two numbers) from a given list of numbers whose sum equals a specific target number.

Use this list in your program: [10, 20, 10, 40, 50, 60, 70]

Hint: You might create dictionaries of array index as *keys* and array item as *values* as you scan through the array. Use conditional statements, loops, etc.

```
>>> python program5.py
What is your target number? 60
index1=1, index2=3
```

4. Digital input/output using GPIOs

This section explains how to get started with the **Adafruit HUZZAH32 - ESP32 V2 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-PICO-MINI-02 module on the V2 Feather Board has 53 pins, many of which are GPIO (general-purpose input/output) pins. However, the feather board has only 28 pins and has its own ad-hoc 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-PICO-MINI-02 datasheet and ESP32 Feather V2 board manuals and the board's Pinouts to figure out the mapping between physical chip pins and board logical pins.

4.1. Programming ESP32 using Thonny IDE

Before proceeding further, you should have completed installing any necessary software (explained in Section 2) and flashed MicroPython on the ESP32 board. The implementation procedure explained in this section demonstrates a typical procedure to run any program on the board.

4.2. Connect the Board to PC

Connect the board to your PC using the USB-C cable (just plug into the jack) and the Feather will regulate the 5V USB down to 3.3V which will power the 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.

4.3. 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 (ESP32).**
2. Select the device serial port

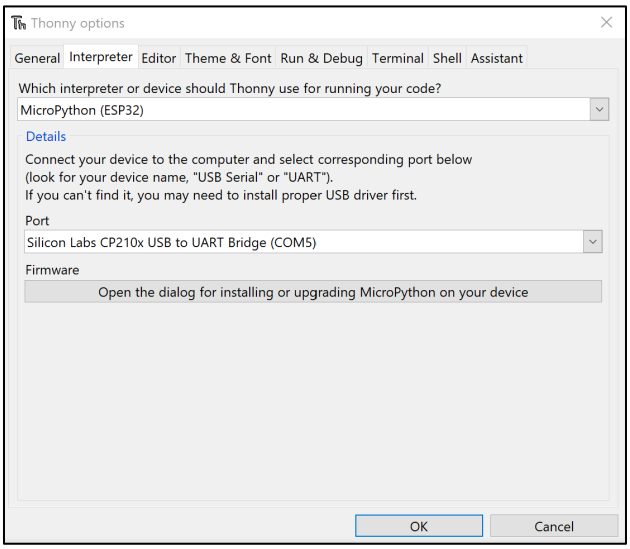

Figure 6. Thonny Interpreter and Port selection

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 8.

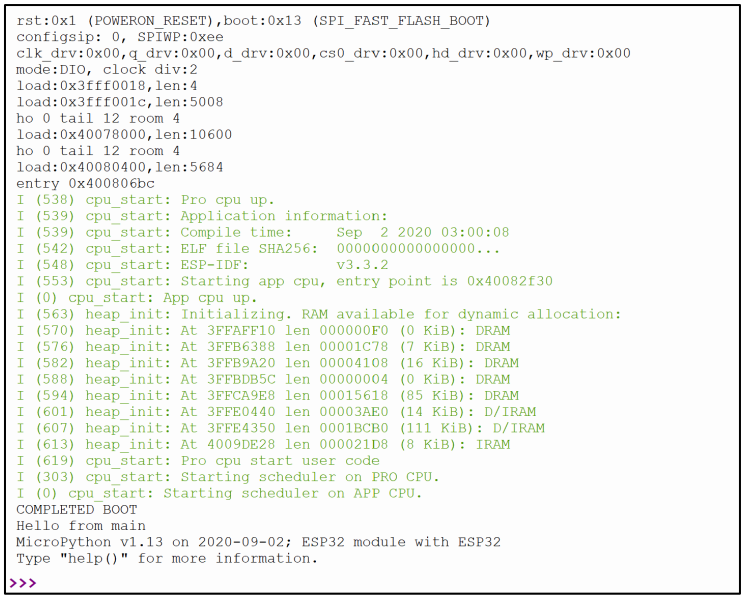

Figure 7. MicroPython startup on ESP32 in Thonny IDE

4.4. 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).

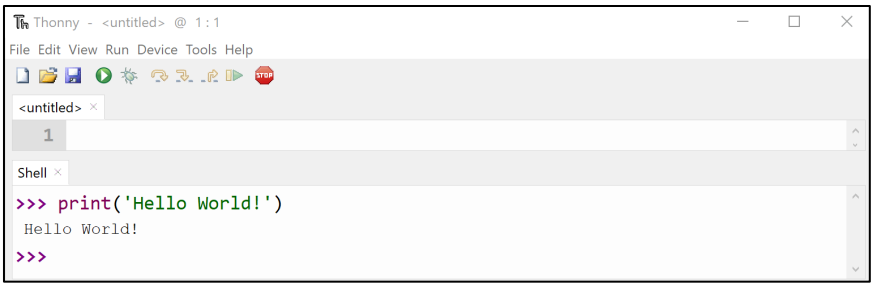

Figure 8. Experimenting with REPL

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

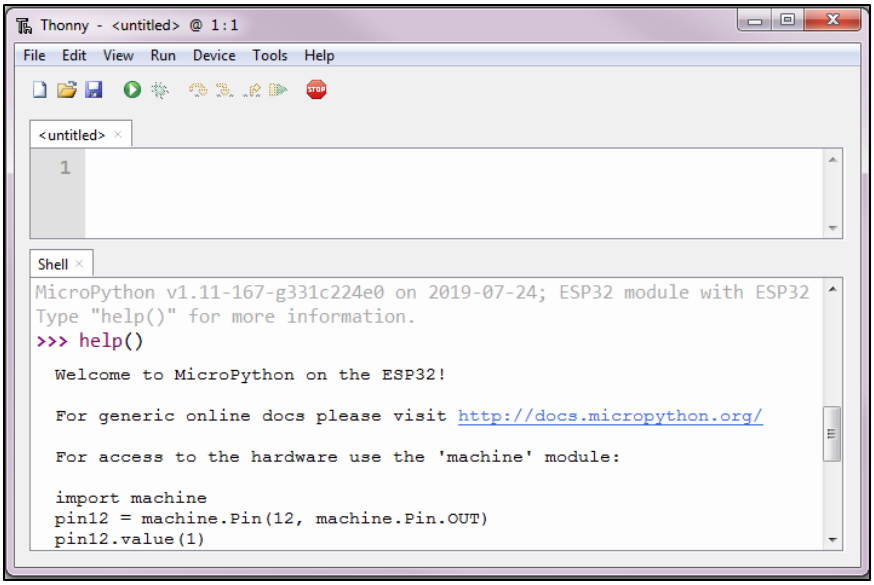

Figure 9. Testing the REPL

4.5. 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 11. 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 12. 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 10. Saving file on ESP32 board

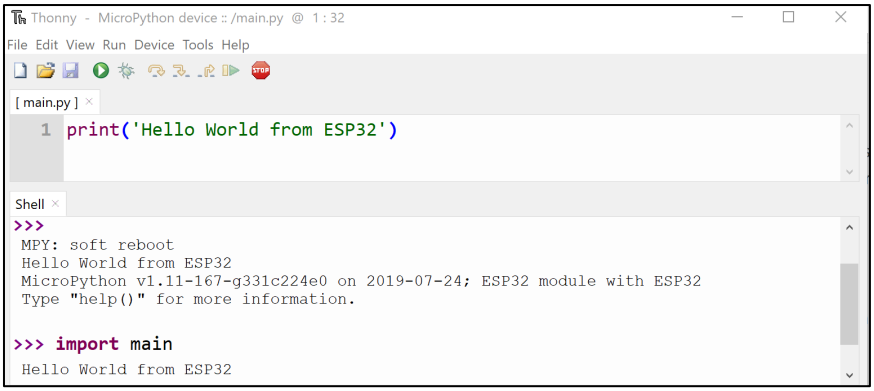

Figure 11. Reboot and Running main.py

4.6. The internal filesystem

Since ESP32 Feather has 8MB 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.

5. ESP32 Programming Exercises

5.1. Simple Blink Program (main.py)

The example source code below blinks the Red LED on the ESP32 V2 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 V2 board **Pinouts** 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")
```

5.2. NeoPixel Color Changes based on Switch Presses (switchpress.py)

Now that you have gone through the warm-up program above, you can finally get to the actual problem statement for Lab 1. You will use the onboard user push button and the onboard NeoPixel LED for this lab. The **switch** is considered to be ON when it is pushed down and OFF when it is released. Start your program by lighting up the NeoPixel to red. When the switch is

ON (pressed), the NeoPixel should turn green. When the switch is OFF (released), the NeoPixel should go back to being red.

In addition, you must determine the **number of times the switch is pressed** from the start of the program. If the switch is **pressed 5 times**, when it is released, the NeoPixel should turn OFF and your program should exit with a message on the MicroPython shell '**You have successfully implemented LAB1!**' and exit to the shell.

6. Submission

Make sure you follow these instructions precisely. Points will be deducted for any deviations. You need to turn in your code on Brightspace. Please create a directory named *username*_lab1, where username is your CAREER account login ID. Inside this directory, create two more directories, *part1* and *part2*.

- Put all the files created in section 3 (Python programming exercise) in *part1*.
- Put all the files created in section 5 in *part2*.

NOTE: *These directories should contain only the source code files, i.e., no executables, no temporary files, etc.*

- Zip the parent directory (i.e., *username*_lab1) and name it as *username*_lab1.zip and upload the .zip file to Brightspace.

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

[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