UC Berkeley, ME100, Fall 2020
Lab 1: ESP32 Familiarization

Plan to complete these tasks during the week of February 1st
Checkoff video due on Gradescope by Monday February 8, 11:59pm

Notes:

- We know that those of you who are having your kits shipped to you might not have received your kit by the start of this week. The lab signoff is not due until February 8, but we will give you an extension beyond that if you find you need more time to complete the lab.
- If you need help, you have several options. During the scheduled lab times, you can join the virtual queue for assistance from instructors at https://me100.not.cs61a.org/. Instructors *might* be online on that site at other times, but it's not guaranteed. At other times, you can frequent office hours, use Piazza, or e-mail the instructors.

Introduction and objectives

The purpose of this lab is for you to get the ESP32 and your own computer set up and ready to start IoT app development, and become comfortable with the software involved. You'll:

- Install Python on your computer, as well as the `shell49` software, which allows you to control the ESP32 directly from your computer;
- Complete a 'boot script' to connect the ESP32 to the internet automatically;
- Start and test a REPL running on the ESP32;
- Complete and test a simple program to flash an LED;

Step 1: check your kit

First, we would like to know if anything is missing from your kit. Please go to https://microkit.berkeley.edu/, click on 'Parts List' in the menu on the left, and then go down the list of components (one webpage per item, starting with 'microcontroller' and ending with 'motor mounting bracket'). Check that each item is contained in your kit box. There are photos on the webpages to help you identify the components. If anything appears to be missing, please note this in item 1 of the Gradescope checkoff assignment. (Please note that some of the motors are being supplied with the encoder board already connected to the back of them, not in a separate package.)

Also, please make sure that you have received the mini oscilloscope that is being loaned to you for ME 100.

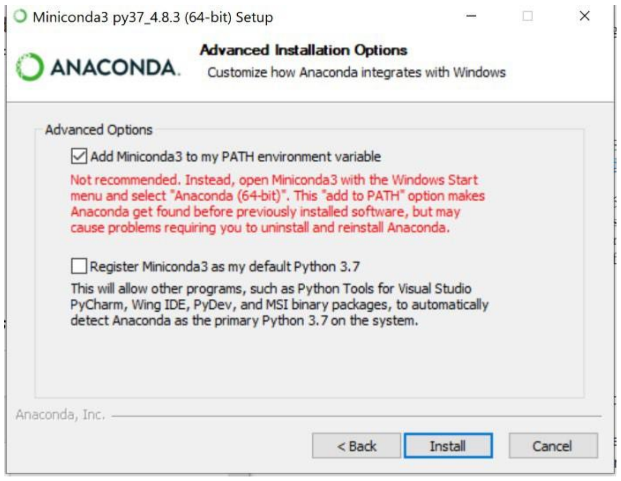
Step 2: install software and MicroPython firmware

Please carefully follow the instructions here, originally written by Professor Boser, which will lead you through the following steps. For each step we provide below some additional hints and troubleshooting notes based on our experience of going through this process:

1. **Install the CP210X USB driver, which will allow your computer to recognize the ESP32 development board when you plug it into a USB port.**

2. **Install Miniconda**

- Note that the instructions linked above refer to Python v3.6. You can install the latest version (Python v3.8) at https://docs.conda.io/en/latest/miniconda.html. Some students have faced some minor errors (relating to temporary file cleanup) when flashing their ESP32 using `shell49` and the latest version of miniconda, but these errors do not appear to prevent successful flashing, or operation of `shell49`. If you want to avoid those minor errors, we recommend installing Miniconda3-4.5.11 .7 from the archive instead (be sure to pick the correct version of the .exe for your system noting whether it is 32 or 64-bit).
- When installing, note the importance of **selecting** the option to "Add Miniconda3 to my PATH environment variable":


Miniconda3 py37_4.8.3 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☑ Add Miniconda3 to my PATH environment variable

Not recommended. Instead, open Miniconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☐ Register Miniconda3 as my default Python 3.7

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.

  Although this option is not recommended by the Setup utility, it will help you to run `shell49` more easily. Otherwise, you may not be able to run `shell49` seamlessly from your command prompt.
- If a version of Python 3.* is already installed on your computer you may want to give it a try. But be aware that all testing has been done with the Miniconda distribution and subtle (or not so subtle) problems have been encountered with other versions. `shell49` does not run on Python 2.
- **System command prompt, a.k.a. 'console' or 'terminal'**. To launch Python 3.* or `shell49` you will need to first to get a system command prompt on your screen. In Windows, for example, you can do this by opening the start menu, typing `cmd`, and opening "Command Prompt". Instructions for the Mac are here.
- Once you have installed Python, try launching it by typing `python` at the command prompt. Once you have obtained the Python prompt (`>>>`), try some simple commands like `a=1` or examples from Lecture 2.

3. **Install** `shell49`

- After following the `shell49` installation instructions at Prof. Boser's link above, confirm that you have the latest version installed. Type:

```
shell49 -V
```

  and you should get a result of 0.1.10.
- If for any reason you have an earlier version than 0.1.10, then apply the following patch[1] to improve the reliability with which you will be able to connect your computer to the ESP32:

[1] Courtesy of Jared Porter

    - Download shell49.zip from this link or bCourses in Files > Labs > Lab 1
    - Extract to a folder.
    - In a terminal/cmd window, navigate to that folder.
    - Run the command:

      ```
      python setup.py install --user
      ```
    - Confirm that you have the latest version:

      ```
      shell49 -V
      ```

      and you should get a result of 0.1.10.
  - There are some vagaries in the way `shell49` handles directory paths. Therefore, we strongly suggest that you navigate to the folder where the .py file(s) for your current lab are *before* starting `shell49`. (You can use the `cd` command in your computer system's shell to move to the relevant directory.) This way, you will not need to change the directory on your computer from within `shell49`.

4. **'Flash' the MicroPython firmware to the Huzzah32 board.**
   - Before connecting the Huzzah32 ESP32 board to your computer, you will need to remove it from its packaging. Be careful not to touch any conductive surfaces (e.g. metals) with the board when powered, to prevent electrical damage to the board. The headers (i.e. the rows of connection pins along each side of the board) have been pre-soldered for you. Connect the board to your computer using the supplied micro-USB cable. After the board receives power through the USB connection, a yellow LED will start flashing rapidly. This is normal.
   - If, having followed the instructions for flashing the MicroPython firmware to the board, `shell49` is unable to connect to the board with the `connect serial` command (i.e. if the `shell49` prompt appears as "`NO BOARD:.>`", try the following:
     - Try typing `connect serial` again
     - Exit `shell49` using the command Ctrl-C, re-enter `shell49`, enter `flash -e`, and try `connect serial` again.
     - Disconnect the USB cable from your computer, reconnect, and try again. (This is very often required immediately after flashing the board.)
     - For Windows users: in some cases, the following works: download the community (free) version of PyCharm (https://www.jetbrains.com/pycharm/), install, then create a new Environment and select the conda interpreter. Click on the terminal link at the bottom of the window, and use that instead of the Windows command prompt window.
     - For Windows users: ensure your copy of Windows 10 is fully updated.
     - Contact your GSI for assistance.
   - Once the connection has been successfully made, the shell49 prompt will appear as "`nameless board:.>`" or "`py.>`"
5. **Install a text editor, such as Atom, on your computer if you do not already have one.**

Step 3: Hello World, IoT-style

Once you have connected your computer to the ESP32, it's time to run our first program on the ESP32. Create a Python script `blink.py` with the following content:

```
from board import LED
from machine import Pin
import time
```

```
#Initialize built in LED pin
led  =  Pin(LED,  mode=Pin.OUT)
while True:
    led(1)
    time.sleep(1)
    led(0)
    time.sleep(1)
```

Navigate to the folder or directory where the `blink.py` file is found on your computer and start `shell49`. It is very important that you are in the folder where `blink.py` is found; otherwise you will not be able to execute the command through `shell49` on your microcontroller. Once you type `connect serial` and you are within `shell49`, type `ls` (to 'list' the contents of the current directory) and verify that the file `blink.py` is present. Then at the `shell49` prompt type:

```
> run blink.py
```

You have completed this part of the lab correctly once you see a red light turning on and off every second.

Note that Ctrl-C does not stop the `blink.py` program (why not?[2]). You will need to disconnect the power from the ESP32 to stop the program from executing.

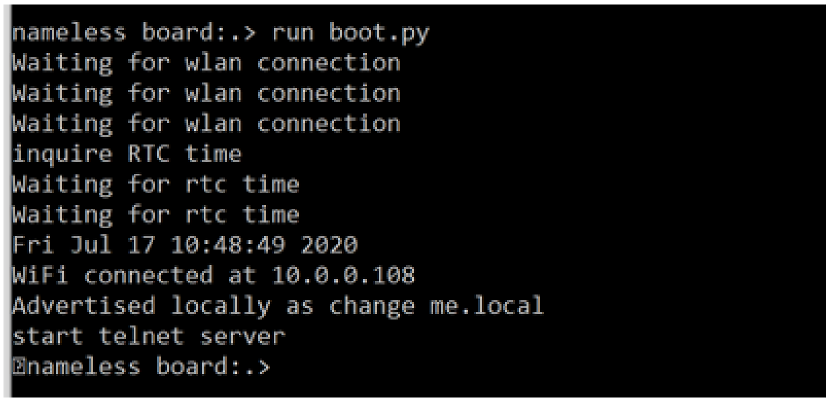
Step 4: Connect to the Internet

Now it is time to connect your ESP32 to the Internet!

1. Download the file `boot.py` from bCourses: Files > Labs > Lab 1.
2. Change the following items in the script:
    - Change `<SSID>` to the SSID of a WiFi network to which you have access (e.g. your home network, or a cellphone hotspot).
    - Change `<Password>` to your WiFi password.
    - Change `my_hostname` to a hostname of your choice for the ESP32, e.g. based on your own name.
    - Change `<telnet_username>` to a suitable username for a telnet account on the ESP32.
    - Change `<telnet_password>` to a password of your choice (do not re-use a password you use for anything else).
3. Read through `boot.py` and make sure that you understand lines 1–20 in particular. Consult with course staff if you have any questions.
4. Once again, to run and test the script, you must navigate to the directory where `boot.py` is located, enter `shell49`, and run the `boot.py` script as modified by you. Verify that it prints the

[2] The `run` command in `shell49` actually starts a REPL on the ESP32 and sends all the lines of the .py file to the ESP32 in one go. The REPL then goes into an infinite `while` loop as instructed by the code, and stops looking for further inputs on the serial connection.

correct time and date fetched from the Internet, and note the IP address. Your command prompt should output something similar to the following:



```
nameless board:.> run boot.py
Waiting for wlan connection
Waiting for wlan connection
Waiting for wlan connection
inquire RTC time
Waiting for rtc time
Waiting for rtc time
Fri Jul 17 10:48:49 2020
WiFi connected at 10.0.0.108
Advertised locally as change me.local
start telnet server
nameless board:.>
```

Step 5: Connecting to the Internet without a USB connection

Up to this point, every program that we have run on our microcontroller has relied on having a USB connection with our computer. Real IoT devices need to operate without being physically tethered to a computer, so we would like a way to communicate with the ESP32 over a WiFi connection.

1. With the ESP32 still connected to your computer via USB, copy your modified `boot.py` script onto the flash memory of your ESP32 with the following command:

   ```
   cp boot.py /flash/boot.py
   ```

2. Modify `blink.py` to terminate after, say, 10 or 20 flashes. Then copy it onto your ESP32 with the following command:

   ```
   cp blink.py /flash/main.py
   ```

   Note that when the ESP32 is powered up, it will first run `boot.py`, then `main.py`, and will be available for a remote connection once `main.py` has completed. Therefore, the LED flashing program needs to be named `main.py` when it is placed on the board, which the above command accomplishes.

3. Disconnect your ESP32 board from the USB port of your computer. Since the USB cable was also providing power to the ESP32 board, we will need to provide an alternate source of power for it to operate. This can come from the lithium ion battery in your kit, which you can connect to the black 2-pin socket on the Huzzah32 board (if necessary, first charge the battery for a while by having both the battery and the USB cable connected). If you do not have the battery to hand, you could also use the USB cable for power only by connecting it to, e.g., a USB phone charger.
4. Once the ESP32 board is disconnected from your computer and has an independent power source attached, press the reset button on the board, and wait a few seconds. The LED should start flashing every second, as it did in Part 3. Take a video of this for the check-off.
5. Once the flashing stops, you can try to connect remotely to the ESP32.
   - If you are using Windows, download and install PuTTY, a simple, free terminal emulator program.

- Ensure your computer is connected to the same WiFi network as you have specified in `boot.py` for the ESP32 to connect to.
- Launch PuTTY, select the “telnet” connection type with the appropriate radio button, and under “Host Name”, enter the hostname of your ESP32 (the text you replaced `my_hostname` with in `boot.py`), followed immediately by “`.local`”.
- Press the ‘Open’ button in PuTTY. A terminal window will open and you will then be prompted to enter the telnet username and password that you specified in `boot.py`.
- If you are using a Mac or Linux, the telnet command is available at the command prompt: `telnet my_hostname.local 23.` You may need to install `telnet` first with `pip install telnet.`
- Once you successfully make the connection, you will see a MicroPython REPL prompt (`>>>`) in the terminal window. Commands you execute here are run on the ESP32.
- Note that it is very important that both your computer and ESP32 are connected to the same WiFi network for the hostname to work. In my experience it may not work if they are connected to different nodes of a mesh router, even if the SSID is the same. Also, the ESP32 can only connect to 2.4 GHz networks, not 5 GHz networks. If you are unable to connect using your hostname, connecting with the specific IP address of the ESP32 is usually much more reliable. To find out the IP address, reset the board while connected serially via shell49, and the IP address will print to the screen when the connection is made. The IP address may vary each time the board restarts, so if you use this approach and want to operate the ESP32 independently of your computer, you may need to connect your battery, restart the board, note the IP address while the USB cable is plugged in, and then disconnect the USB cable and connect via telnet.
- Some people have reported being unable to establish a telnet connection from a computer to which the board is also connected with the USB cable. In this case, disconnect the USB cable from your computer and either power the board from the battery or connect it to a USB power source that is separate from your computer.

Check-off on Gradescope

Check-off will be done via the Gradescope “Lab 1” assignment. This is an online form that we ask you to complete. There are fields to upload the following:

- A video of your `blink.py` script running on your ESP32 board — i.e. untethered from your computer,
- A video or static screen shot(s) showing a telnet connection being made to your ESP32 without a direct USB connection, and running some simple commands (e.g. a=1) in the REPL on the ESP32 via the telnet connection.
- Your `blink.py` script after modification to flash only a certain number of times.

If you worked with a partner, only one person needs to submit the assignment. There is a field in the assignment where we ask you to note the name of the person you worked with, and the contributions made by each person. We will manually link the submission to both students and assign the same credit to each person.

Further reading

The ESP32 chip datasheet and the Huzzah32 development board datasheet are on bCourses under Files > Labs > Lab 1. You may want to read through these to familiarize yourself with the features and pin connections of the system.

Student Technology Equity Program (STEP)

If you find you do not have a suitable WiFi router or phone hotspot to connect your ESP32, you may be able to obtain a WiFi hotspot through Berkeley's Student Technology Equity Program (STEP https://technology.berkeley.edu/STEP. We have not verified whether the hotspots being distributed through this program will enable all the functionality described in this handout, but we would be interested to learn if anyone has experiences of using a campus-loaned hotspot.

Alternative connection technique

The new version of `shell49` (v0.1.10) described above appears to resolve most if not all connection issues. However, if you still encounter glitches, here is an alternative way of interacting with your ESP32 board:

- First use `shell49` to flash the firmware to the board (i.e. get as far as the `flash -e` command in Prof Boser's instructions).
- Then, at the system prompt of your computer, type

  ```
  pip install adafruit-ampy
  ```

- You can then use commands in the following form at your system prompt (replace COM3 with the appropriate COM port as needed):

  ```
  ampy --port COM3 ls
  ```

  lists the contents of the flash memory on your ESP32

  ```
  ampy --port COM3 put main.py
  ```

  places `main.py` in the flash memory of the ESP32 (equivalent to a `cp` command in `shell49`)

  ```
  ampy --port COM3 run blink1.py
  ```

  runs the code contained in `blink1.py` (located in the current directory on your computer) on your ESP32 board (equivalent to a `run` command in `shell49`)

- See the Adafruit page about this tool for more information

- To access a REPL on the ESP32 board via a serial connection, you can do the following:

  - For Windows, use PuTTY with the "Serial" radio button selected, COM3 (or the appropriate port number) in the "Serial line" field, and 115200 in the "Speed" field.
  - For Mac/Linux, install picocom and use the command `picocom /dev/ttyUSB0 -b115200`

- To access a REPL via `telnet`, use the methods described in Step 5, item 5 above.

Revision history

Date	**Revision**
September 1	Amended text of Step 1 to include "(Please note that some of the motors are being supplied with the encoder board already connected to the back of them, not in a separate package.)"
September 1	Added notes on troubleshooting serial connection
September 1	Added notes on WiFi connection/telnet: using IP address
September 2	At the start of Part 3, change "from time import sleep" to "import time"
September 3	Modified Part 2 to include new fix for serial connection problems in shell49 (highlighted in blue)
September 3	Modified Part 5 to stay that serial connection and telnet may not work together on the same computer – use separate power supply if necessary.
September 3	Added a note to Section 5 to say that the ESP32 can only connect to 2.4 GHz WiFi
September 3	Added a Further Reading section to refer to the Huzzah32 and ESP32 datasheets.
September 3	Added reminder that `telnet` may need to be `pip installed` in MacOS or Linux
September 3	Added "Alternative connection technique" section
September 3	Added section on obtaining hotspots from STEP
September 3	Added note on possible use of earlier version of Miniconda (Miniconda3-4.5.11 .7) to solve minor errors.
September 7	Extended deadline to September 28 to accommodate students waiting for their kits to arrive.
September 7	The latest version of `shell49` (0.1.10) is now registered on PyPi, so the manual patch is no longer needed and you can get the latest version by typing `pip install shell49`. Modified Step 2 item 3 to reflect this, and removed blue highlighting.