Feather HUZZAH setup

Part One: HUZZAH Setup
https://docs.micropython.org/en/latest/pyboard/index.html
https://learn.adafruit.com/building-and-running-micropython-on-the-esp8266/overview
http://micropython.org/download/
https://github.com/themadinventor/esptool
https://github.com/wendlers/mpfshell

Windows:

Step 1: Download the esptool through https://github.com/themadinventor/esptool. Download the Micropython firmware for ESP8266 through http://micropython.org/download/.

Step 2: Flash the Micropython firmware onto the HUZZAH board.

Connect the HUZZAH board to your machine and find out which serial port it is connected to. You can either do this by looking under the Ports tab in Device Manager or through the command line by changing directors to **C:\WINDOWS\system32** and using the "mode" command.

Once you know the serial port, navigate to the folder through the command line which you extracted the esptool files to (should have esptool.py inside). Enter the following command to flash the firmware onto the board.

```
>python esptool.py --port SERIAL_PORT --baud 460800 write_flash -
-flash_size=32m 0 FIRMWARE.bin
```

If there are any issues with the firmware, you can always reprogram the device after erasing the flash with the following command.

```
>esptool.py --port /dev/ttyUSB0 erase_flash
```

Now that the firmware is set up on the HUZZAH, you can now connect to it through serial using a terminal program (mobaXterm, putty, termite, etc.) with the following settings:

```
Baudrate = 115200
Data bits = 8
Stop bits = 1
Parity = None
Flow Control = XON/XOFF
```

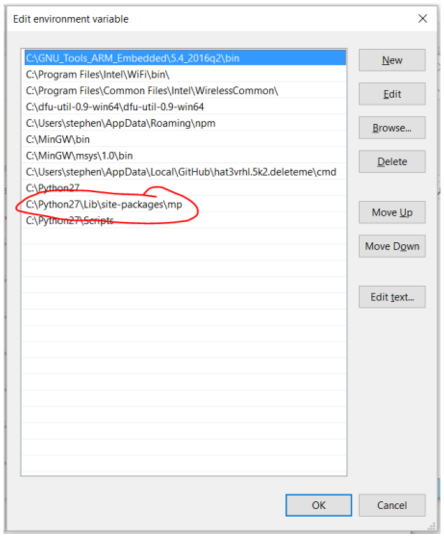
You can now access the python command line on the ESP8266 chip. The Micropython library has some, not all, of the same functionalities as Python 3, as well as some board specific capabilities.   More information can be found by looking through the documentation.

<u>Step 3</u>: Install mpfshell, which will be used to access the internal file system on the ESP8266 and transfer files to and from the board.

Download the files through https://github.com/wendlers/mpfshell.   Run the following command to install mpfshell.

```
>python setup.py install
```

After installation, make sure to include the path to the “mp” folder (includes mpfshell.py) in the user “Path” variable; This folder should be located within the Python installation folder.



```
C:\GNU_Tools_ARM_Embedded\5.4_2016q2\bin
C:\Program Files\Intel\WiFi\bin\
C:\Program Files\Common Files\Intel\WirelessCommon\
C:\dfu-util-0.9-win64\dfu-util-0.9-win64
C:\Users\stephen\AppData\Roaming\npm
C:\MinGW\bin
C:\MinGW\msys\1.0\bin
C:\Users\stephen\AppData\Local\GitHub\hat3vrhl.5k2.deleteme\cmd
C:\Python27
C:\Python27\Lib\site-packages\mp
C:\Python27\Scripts
```

Now you can use the “mpfshell” command in the command line to connect, view, and transfer files to and from the HUZZAH board. **<u>Please make sure you go through the documentation on the link above</u>** about how to use mpfshell for uploading programs to the Huzzah before you start programming.

MAC or Linux:

The process for using the Huzzah is relatively easier for mac and Linux as they come with python. We first install pip (Python package manager) to install dependencies and tools. We then download the micropython firmware and flash it on to the board. We also install a small command line tool which makes it easy to upload programs and interact with the Huzzah board.

Step 1: If you already have pip installed skip to Step 2. Install pip using this script (get-pip.py). Download the linked file to your computer. Open your terminal app and navigate to the folder where you downloaded it and run

```
>python get-pip.py
```

Step 2: Install the esptool (this is used for flashing micropython).

```
>sudo pip install esptool
```

Step 3: Download the micropython firmware and flash it onto the Feather Huzzah board. Download the latest version of micropython here. Download the bin file listed under the ESP8266 section and save it. On your terminal navigate to the location of the bin file you downloaded and run the following commands.

The port for the Huzzah is usually /dev/tty.SLAB_USBtoUART on the **Mac**. To find out the port at which the Huzzah is mounted at, open up terminal and type

```
>ls -l /dev/tty.*
```

This lists all the ports to which devices are connected to. Scroll to the top and note the port which says USB/UART etc.

```
>esptool.py --port /dev/tty.SLAB_USBtoUART erase_flash
>esptool.py --port /dev/tty.SLAB_USBtoUART --baud 460800
write_flash --flash_size=32m -fm dio 0 <name of bin file
downlaoded>.bin
```

This flashes micropython to the board and once the process completes, you are all set.

*Note:

1. If you are using Ubuntu the port should be like /dev/ttyUSB*.
2. If your computer cannot find the HUZZAH, then download the driver located here: https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
And Linux Kernel 4.X.X is already installed with this driver.
3. Regarding flashing the board, see reference here:
https://learn.adafruit.com/building-and-running-micropython-on-the-esp8266/flash-firmware

Step 4: To make things easy for programming the board, we are going to use another tool called mpfshell. Download mpfshell utility here. Follow the Installation Instructions on the page to install mpfshell. Please read the documentation on the link to use mpfshell for uploading programs to the Huzzah.

Part Two: Blink LEDs

Now that you have the environment set up to be able to code and run programs on your HUZZAH, we will now write our first few programs. These exercises hopefully can get you familiar with the ESP8266 environment, which will form the basis of the smartwatch. Specifically:

1. Write a program to blink two separate LEDs simultaneously (you can use some of the on board LEDs too) at two different rates (e.g. blink LED 1 at 100 ms, LED 2 at 500 ms). The stipulation is that you are not allowed to use any interrupts or timers, and that the blinking rates should be discernible to the human eye.

Reading and Processing Analog Sensor Data

The structure behind almost any system is that they take in inputs, process the inputs, and finally create some sort of output based off of the results from the processing.   Especially interesting is the data we can extract from the real-world to process on computers, machines, the cloud, etc.   To be able to process data digitally, we require the use of sensors to detect the changes that are occurring in the world and to produce an output that can be read in by a controller, computer, etc.   In this lab, we will be setting up the ESP8266 board to take in various sensor inputs from the physical world and to generate a response from the board or other components based on our inputs.

The overview of the system we will be building is as follows.   The system is turned on when a user pushes a button, generating an interrupt, that is fed into one of the board’s GPIO pins.   While the system is turned on the ADC will begin sampling from the sensor attached to the board’s only ADC pin.   Based on the values read from the ADC, the system will turn on an LED and begin operating a piezo/vibration motor; the brightness of the LED and the pitch of the vibration motor will change based on the readings from the sensor and are controlled through PWM, a common method for controlling actuators. The sensor that we will be using in this lab is an analog light sensor.

Obtaining and processing sensor data is a vital part of IoT and human-machine interaction; every key pushed on your keyboard or button pressed on your mouse or phone is processed by your computer or laptop to produce an output on your device that people can control and use.   The concepts and processes used in this lab will subsequently be incorporated into the final smartwatch to allow for human-machine interaction.

Part One: Reading in Sensor Data

1. Wire up an LED with an appropriate circuit to a GPIO pin on the ESP8266 and control the brightness of the LED with PWM
2. Do the same with the piezo/vibration motor; The LED and piezo brightness and pitch are what we will vary based on our sensor input.

3. Connect the analog light sensor to the ADC input to the HUZZAH board and sample at 1 Hz (1 sample per second). You will see that values you read out will be an integer number. Try vary the readings from the light sensor by placing your finger on the sensor and removing it. You can see what values you are getting by printing to the serial port.
4. Next, we will combine what we have done so far; Start reading in light sensor values at a rate of at least 10 Hz and based on the readings you get from the sensor, adjust the pitch of the piezo/vibration motor and brightness of the LED accordingly (e.g. increase in light levels to an increase in LED brightness or piezo frequency).

Part Two: Interrupts and Debouncing

1. Next we will work with interrupts by adding an on-off switch. Connect a button to one of the GPIO pins. Use proper hardware and software debouncing so that you can generate stable interrupts with a button press; at most your button should interrupt once when you press the button and once when you release. If it interrupts any more than that, then the debouncing software and circuit is not adequate.
2. Incorporate the button press with the light sensor/LED/piezo system created in part one. When you press and hold the button, your system should generate an interrupt to start sampling the light sensor, changing the brightness of the LED, and changing the pitch of the piezo. When you let go of the button, the system should turn off.

EECS 4764 Lab One Checkpoints:

1. Write a program to blink two separate LEDs simultaneously (you can use some of the on board LEDs too) at two different rates (e.g. blink LED 1 at 100 ms, LED 2 at 500 ms). The stipulation is that you are not allowed to use any interrupts or timers, and that the blinking rates should be discernible to the human eye.
2. Change the brightness of an LED and the pitch of a piezo in accordance to readings from the light sensor. For example, the more light that reaches the sensor, the brighter the LED shines and the higher the pitch from the piezo will be.
3. Debounce a button press. At most, an interrupt should be generated once when the button is pushed and once when the button is let go.
4. Create the combined system; the system is activated when the button is pressed and deactivated when released. While activated, the LED and piezo changes brightness and pitch depending on the light sensor readings.