Internet of Things - Intelligent and Connected Systems

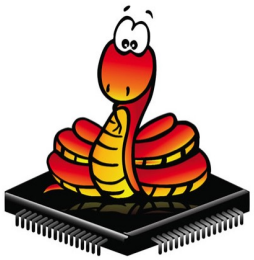
EECS E4764

Columbia University

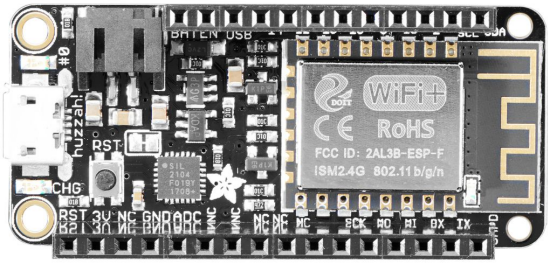
Prof. Xiaofan Jiang

Fall 2022

Lab 4 - Bus Communication: SPI and Interfacing with the World



MicroPython



Introduction

An important concept behind many innovations and advancements is to build upon or improve what already exists. In other words, it is in your best interest to use existing systems that accomplish what you plan to build upon or use as a foundation for your own work. This not only saves you time because you can spend it elsewhere working on your own key contributions rather than reinventing something that has already been done, but it can also make your own products or innovations easier to use or interface with since they build upon existing technologies. This is especially true for IoT systems and devices; one of the primary aims for IoT products is not only to allow for a network of communication between physical devices and systems, but to also do so in a convenient manner by utilizing existing and flexible infrastructures.

In this lab, we will use Google APIs to add some nifty features to our wearable. An API (application programming interface) makes the primary functions and commands available to the user or developer, so that they can utilize the existing technology or functionalities that the API provides. Just as a computer screen shows the user what is available or the functionalities on that computer, an API shows the developer what features of the technology are available to use. While a user can communicate actions and commands to a computer through peripherals, like a mouse and keyboard, a developer can communicate with many APIs through HTTP, an application layer communication protocol, or through the RESTful API, which uses HTTP as its basis.

For our embedded system, we will be using the IP Geolocation API, and the OpenWeatherMap API to track the weather in our area and the Twitter API to send tweets with our wearable.

Part 1: Interfacing with Accelerometer

Task 1

Interface with the accelerometer using SPI. Read the acceleration on three axes continuously and print them out. Tilt the accelerometer to confirm that the values make sense.

Task 2

Use the accelerometer readings and implement text scrolling; if you tilt the accelerometer far enough in one of the four principal directions (up, down, left, right), the text on the screen should begin scrolling in the same direction. For example, if you tilt the accelerometer to the right, the text on the OLED screen scrolls to the right until it goes off screen before it reappears on the left side of the screen.

Checkpoint 1

On top of the combined system from the last lab, implement text scrolling based on accelerometer readings.

Part 2: Geolocation and Weather

The objective of this part is to display weather information onto our smartwatch. To achieve this, we will first use a geolocation API to figure out where is our current location, then sending our location (latitude and longitude) into a weather API to find out what's the weather in the area.

You will find the **`urequests`** and **`ujson`** libraries useful. Also, because they are MicroPython ports for the regular Python **`requests`** and **`json`** library, their usages are roughly the same. Thus, it may be easier for you to test your API codes on your computer with the requests and json libraries, then move the code onto your ESP.

You can also download **Postman** to help you testing different APIs. It can also generate some code for you!

☑ Task 1

Connect your ESP8266 to the Internet. You can follow the documentation here: https://docs.micropython.org/en/latest/esp8266/quickref.html#networking

☑ Task 2

Retrieve the latitude and longitude of the current location. Display the latitude and longitude on the OLED screen. A simple API to use is: http://ip-api.com/json. More information about this API can be found in their documentation: https://ip-api.com/docs/api:json

Checkpoint 2
Interface with a Geolocation API and display your coordinates on the OLED display.

Task 3

Feed the coordinates from the geolocation API into the OpenWeatherMap API to receive weather data for your location. Display the current weather on the OLED display. You should at least display the temperature and description (e.g. mostly cloudy). More information about the OpenWeather API can be found at http://openweathermap.org/apiand http://openweathermap.org/current.

Checkpoint 3

Feed your coordinates into an API to receive information about the weather in your area; display the results on your OLED display.

Part 3: Twitter

Task

Interface with the Twitter API to send tweets from your ESP8266. Make sure to register for a Twitter account if you don't have one and go to https://apps.twitter.com/ to get started.

As a side note: there are many other APIs that allow you to send Twitter messages, other than the Twitter API; you are welcome to use any of them. The tweets you send do not have to be typed out and can be preset. There are links to a few other APIs listed below, that also support Twitter commands.

https://thingspeak.com/apps
https://ifttt.com/twitter

Checkpoint 4

Post a tweet from the ESP8266. An example tweet could be your location coordinates with the weather in that location. Also, open Twitter website or app with your PC or phone to show the tweets.

Lab 4 Checkpoints:

1. On top of the combined system from the last lab, implement text scrolling based on accelerometer readings.

2. Interface with a Geolocation API and display your coordinates on the OLED display.

3. Feed your coordinates into an API to receive information about the weather in your area; display the results on your OLED display.

4. Post a tweet from the ESP8266. An example tweet could be your location coordinates with the weather in that location. Also, open Twitter website or app with your PC or phone to show the tweets.

Submission:

1. **Write a brief description "report" about your understanding of SPI communication used by the accelerometer (checkpoint 1) for this lab.**
    - **How to wire up the accelerometer to the board**
    - **How the read/write is performed**

2. **Submit the file for each checkpoint and writeup in 1) onto courseworks → Assignments → Lab4**

The submitted files should be named like

```
labx_{member1_uni}_{member2_uni}_{member3_uni}_check{X}.py
```

There should be 5 files for this lab.
e.g.

- Lab4_kh3119_sm4928_rl3154_writeup.txt
- Lab4_kh3119_sm4928_rl3154_check1.py
- Lab4_kh3119_sm4928_rl3154_check2.py
- Lab4_kh3119_sm4928_rl3154_check3py
- Lab4_kh3119_sm4928_rl3154_check4.py