ECE 56800: Software for Embedded Systems

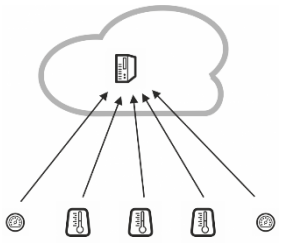
Spring 2025

Lab 4 – A Simple IoT Application using an ESP32-based HTTP Web Server and Client. This lab will use the MicroPython V1.20 Version

To be done individually; Due by 11:59pm, Wednesday, April 16, 2025.

1. Overview

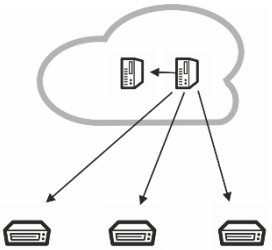
The goal of this assignment is to familiarize you writing simple **Internet of Things (IoT)** applications by using your ESP32 as an IoT device communicating with the Internet. The assignment is divided into three sections. In the first section, you will connect your ESP32 board to the Internet using Wi-Fi. In the subsequent sections, you will implement client-server communication using socket APIs and the HTTP protocol. To review, network socket APIs are used to transfer *messages* between two software processes over a network. HTTP is an application layer protocol that defines different parameters of these *messages*, viz., type, format, length, authentication, error handling, etc.



1.1. ESP32 Device as Client

A typical IoT system has a cloud server and multiple client devices. Data flows from the IoT-devices to the cloud-server, as shown in this image. First, you will configure the ESP32 as an HTTP client, measure the onboard *temperature sensor data* and *hall sensor data*, and send them periodically to an IoT Application Platform hosted on the Internet. Specifically, you will use the ThingSpeak IoT Platform as the server to collect sensor data transmitted by the ESP32.

1.2. ESP32 Device as Server



In some scenarios/configurations, the IoT device can also be setup as a server. In the second section of the assignment, you will configure the ESP32 as a HTTP web server, measure the onboard *temperature sensor data* and *hall sensor data*, monitor the *status of an LED*, and send these data to the client on-demand, only when the client requests it. You will be using your local PC or phone's web browser (e.g., Chrome, Firefox, Safari, etc.) as the client. Additionally, you will also receive input from the client (i.e., the web browser) and control a GPIO pin on the board based on this input.

2. Programming Exercises

2.1. Hardware Interfacing

You will use the onboard red LED, configured as a GPIO ouput.

2.2. Software Implementation – Connect to the Internet over Wi-Fi

Use the *network* module in MicroPython to connect your ESP32 to a Wi-Fi network using the '*SSID'* and '*Password*' for that network. To avoid issues related to complex wireless security configurations, you can just create a Wi-Fi hotspot on your phone/laptop and connect your ESP32 to the hotspot. Refer to the example given in the MicroPython documentation for the network module **Error! Reference source not found.**

e.g., if your mobile hotspot has an *SSID*: 'Lenovo-SSID' and *Password*: '12345678', then use this *SSID* and *password* for connecting the ESP32 to the Internet.

NOTE: You can also connect to any other Wi-Fi network which works directly with an SSID and password. However, enterprise networks such as PAL3.0 which require SSID, username, password, certificates, etc., are unfortunately not supported in MicroPython.

Each time the board successfully connects to Wi-Fi, the program should print the SSID it has connected to and the interface's IP address.

```
Connected to Lenovo-SSID
IP Address: 192.168.0.107
```

2.3. Software Implementation - ESP32 HTTP Client

2.3.1. Setup ThingSpeak

ThingSpeak (https://thingspeak.com) is an IoT Application Platform for collecting, analyzing, and visualizing data from IoT devices. ThingSpeak allows you to publish your sensor readings to their website and display them in a plot with time stamps. It provides a RESTful API for IoT devices (don't worry if you don't know what that means, it's not important for this lab). You need to perform the following steps to configure ThingSpeak as your cloud server before you can start sending data from the ESP32.

- Create a free account on the ThingSpeak website. You only need the free tier for this lab and do not need any of the paid upgrades.
- Create a new channel and enable two fields to receive data from the ESP32.
    - Field1: Temperature Sensor
    - Field2: Hall Sensor [MicroPython V1.20 support]

You'll get an API key for posting data to your channel. This API key should be used for sending data to ThingSpeak from your ESP32.

2.3.2. ESP32 Program (espclient.py)

Implement the following functionality in your program.

- Connect to the Internet and print out the local IP address (same as Section 2.2).
- Initialize a hardware timer with a period of *30 seconds*. When the timer fires, perform the following operations:
    - Measure the onboard *temperature sensor data* and *hall sensor data.*
    - Print both the measured data onto the terminal.
    - Send these data to ThingSpeak cloud server using a socket API and HTTP GET request. You need to use your specific Write API Key to upload data to your channel in your ThingSpeak account.
- Run your program for 5 minutes.

2.3.3. Sample Result

Your ThingSpeak account should show the visualization for both sensor data, as shown in Fig. 1. Since the timer fires every 30 seconds, a maximum of 10 data points (for each sensor) will be uploaded by your client over a duration of 5 minutes. Fig. 1 shows a sample screenshot from the ThingSpeak website showing only 4 data points. Run your program for 5 minutes and take a screenshot (similar to Fig. 1) from the ThingSpeak website showing ~10 data points.

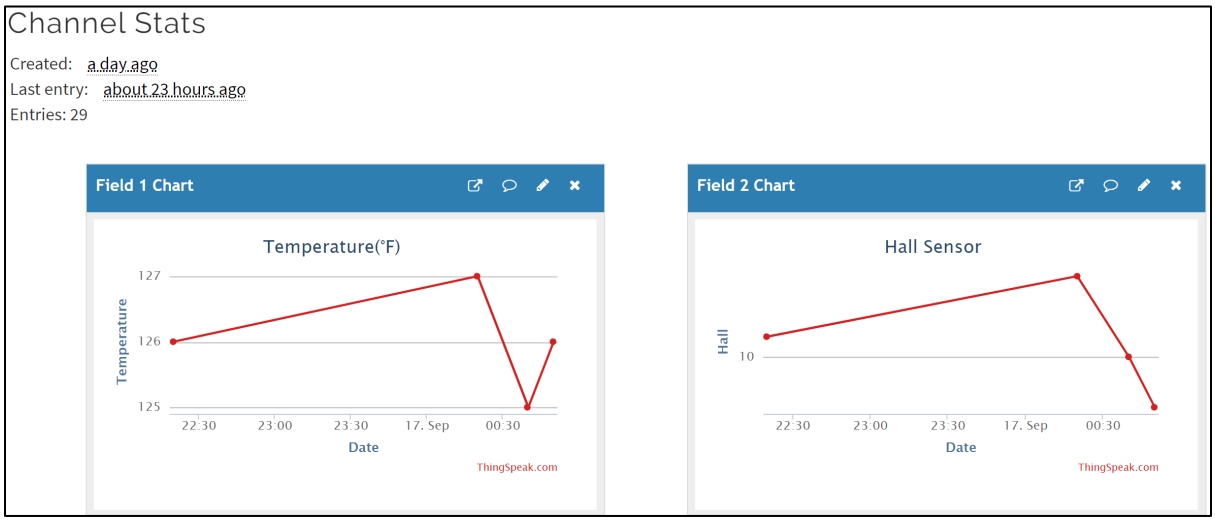


Figure 1. Screenshot from ThingSpeak showing Temperature and Hall Senor data

2.4. Software Implementation - ESP32 HTTP Server

2.4.1. ESP32 Program (espserver.py)

Implement the following functionality in your program. You can just update the **espserver.py** file that has been provided to you.

- Connect to the Internet and print out the local IP address (same as Section 2.2).
- Measure the onboard *temperature sensor data* and *hall sensor data.*
- Measure the state of the *red LED*.
- Use the function web_page (present in the provided file) to create simple HTML text to build the webpage with the sensor data and the LED pin values. *The HTML text has already been provided in the **espserver.py***. You can use the 3 variables defined in the file (*temp, hall, red_led_state*) to measure the sensor and pin values.

- Create a HTTP server using the socket API to listen for incoming requests from any client (browser on your local PC or phone connected to same Wi-Fi network).
- **Use an infinite loop**: Whenever your ESP server receives a client request (e.g., pressing any button on the webpage), it should use the function web_page to update the HTML text with the current sensor values and LED state, send necessary HTTP headers and finally send the updated HTML text as the response to the client.

2.4.2. Sample Result

Open a web browser on your PC or phone and type in the IP Address of your ESP32 server. You should be able to access the web page with the latest sensor readings and GPIO state.

Pressing any button on the webpage counts as one client request. So, every time you press any button, the *temperature*, *hall*, and *LED state* should update. e.g., if you press the *RED ON* button, the red LED connected to your board should light up, and the *RED LED Current State* should display ON as shown in Fig. 2.

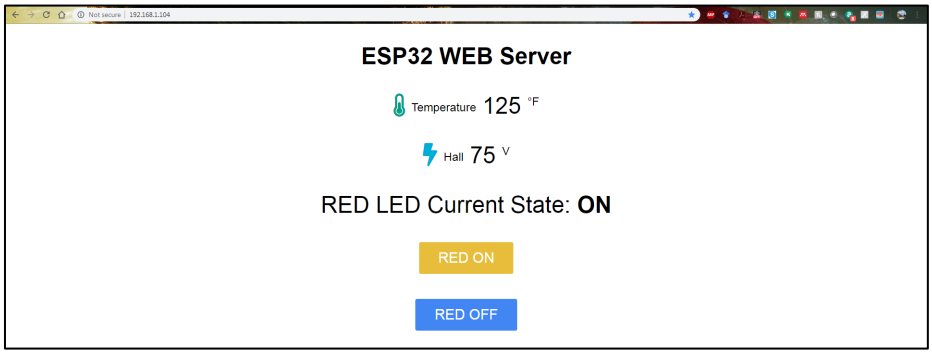


Figure 2. Webpage on pressing RED ON button

3. Video

Create a short video that shows you demonstrating your working solution (explain your hardware connections briefly before demonstrating the working solution). Please do not upload video files directly on Brightspace to save space. Instead, please upload the video to YouTube (or other such video hosting platform) and include the link to the video in your README file mentioned below.

4. Submission

You need to turn in your code on Brightspace. Please create a directory named ***username*_lab4**, where username is your CAREER account login ID. This directory should contain only the following files, i.e., no executables, no temporary files, no sub-directories, etc.

1. ***espclient.py***: your program for ESP32 HTTP Client
2. ***screenshot.jpg (or .png or .gif)***: Screenshot obtained from the ThingSpeak website showing Temperature and Hall Senor data, each with ~10 entries, as shown in Fig. 1.
3. ***espserver.py***: your program for ESP32 HTTP Web Server

4. ***A README file named username_lab4_README***.txt that contains a YouTube link that shows you demonstrating your working solutions.

Zip the files and name it as *username*_lab4.zip and upload the .zip file to Brightspace.

Note: For reference, the directory and file structure of your submission should look something like this. Note the spellings, spaces, etc. in the file/directory names.

```
username_lab4.zip
   |---username_lab4 (directory)
      |---espclient.py
      |---screenshot.jpg
      |---espserver.py
      |---username_lab4_README.txt
```

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] ESP32 specific functionalities in MicroPython
http://docs.micropython.org/en/latest/library/esp32.html

[8] ThingSpeak: https://thingspeak.com/

[9] MicroPython socket module: https://docs.micropython.org/en/latest/library/socket.html