ECE40862: Software for Embedded Systems

Fall 2020

Lab 4 – HTTP Web Server and Client using ESP32

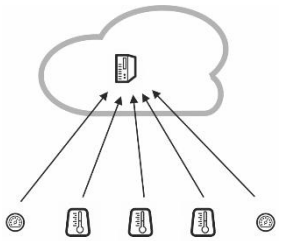
To be done individually; Due by 11:59pm, Sunday, October 25, 2020.

1. Overview

The goal of this assignment is to familiarize you with **Internet of Things (IoT)**. The IoT is a network of *'smart'* embedded devices that connect and communicate via the Internet. The key to the IoT is the *interconnectivity* of devices, which collect and exchange information through embedded software, cameras, and sensors which sense things like light, sound, distance, movement, temperature, and other environmental parameters. Smart embedded devices either operate automatically or are controlled and monitored remotely. In this assignment, you will be using your ESP32 as the *'smart'* embedded device operating automatically as well as communicating with the Internet.

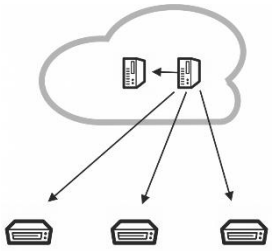
The assignment is subdivided into two sections. In both sections, you will be implementing **client-server communication** using **socket APIs** and **HTTP** protocol. To review, network socket APIs are used to transfer *messages* between two processes over the network. HTTP is the application layer protocol which defines different parameters of these *messages*, viz., type, format, length, authentication, error handling, etc.

1.1. Device as Client



A typical IoT system has a single cloud server and multiple client devices. Data flows from the IoT-devices to the cloud-server, as shown in this image. In the first section, 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 in the internet. Specifically, you will be using the ThingSpeak IoT Platform as the cloud server, collecting sensor data transmitted by the ESP.

1.2. Device as Server



In some scenarios/configuration, e.g. when you have a single IoT device and a local PC or a phone, the IoT device can also be setup as the 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*, as well as monitor *status of different GPIO pins*, and **SEND** these data to the client ONLY WHEN THE CLIENT REQUESTS. You will be using your local PC or phone's web browser (e.g. Chrome, Firefox, etc.) as the client. Additionally, you will also **RECEIVE** inputs from the client, i.e. the web browser and control GPIO pins on the board.

NOTE: ESP as client (1.1) only SENDS data to cloud server. ESP as server (1.2) both SENDS data to client and RECEIVES data from client.

2. Programming Exercises

2.1. Hardware Interfacing

Interface the following components to the board:

- Two external **LED**s (red and green) as GPIO OUTPUTs.

2.2. Software Implementation - ESP32 HTTP Client

2.2.1. Setup ThingSpeak

ThingSpeak is a free IoT Platform for gathering, 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. You need to perform the following steps to configure it as your cloud server before you can start sending data from the ESP32.

- Create an account on ThingSpeak website.
- Create a new channel and enable two fields to receive data from the ESP32.
    - Field1: Temperature Sensor
    - Field2: Hall Sensor

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.

2.2.2. ESP32 Program (Upload as espclient.py)

Implement the following functionalities in your program.

- Connect to Internet (same as **Lab 3**).
- Initialize a **hardware timer** with timer period of **15 seconds**. Perform the following operations:
    - Measure the onboard *temperature sensor data* and *hall sensor data*
    - Display both the measured data on the terminal/REPL.
    - **SEND** these data to ThingSpeak cloud server using **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.

2.2.3. Sample Result

Your ThingSpeak account should show the visualizations for both sensor data, as shown in Fig. 1. **Run your program for 5 minutes**, so that there are **30 data points** on both graphs. Fig. 1 only shows 4 data points. Upload a screenshot from the ThingSpeak website like Fig. 1 with 30 data points.

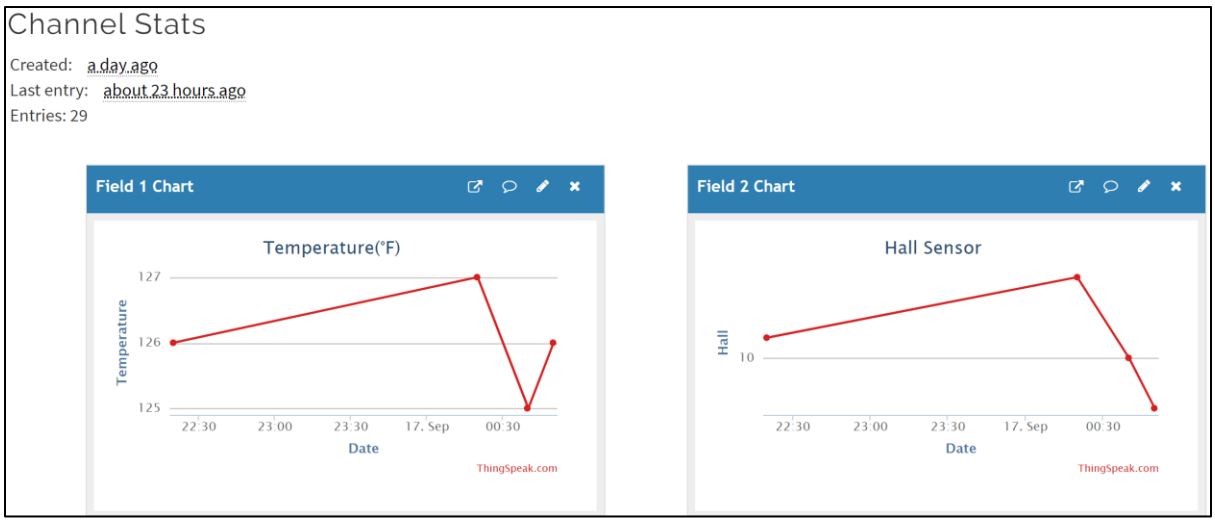


Figure 1. Screenshot from ThingSpeak showing Temperature and Hall Senor data

2.3. Software Implementation - ESP32 HTTP Server

2.3.1. ESP32 Program (Upload as espserver.py)

Implement the following functionalities in your program. You can update the **espserver.py** file which has been provided to you.

- Connect to Internet (same as Lab 3 Section 2.2.1).
- Measure the onboard *temperature sensor data* and *hall sensor data.*
- Measure the values of different GPIO pins, viz., *two LEDs.*
- Use the function web_page (present in provided file) to create an HTML text to build the webpage with 2 sensor data and 2 LED pin values. *The HTML text has been provided in the espserver.py*. You can replace the variable names (*temp, hall, red_led_state, green_led_state*) in the provided file with your own variables measuring the sensor and pin values.
- **Create an HTTP server** using **socket API** to listen for incoming requests from any client (browser on your local PC or phone connected to same Wi-Fi network).
- **Use infinite loop**: Whenever your ESP server receives any client request, it should use the function web_page to update the HTML text with the current sensor and LED pin values, SEND necessary HTTP headers and finally SEND the HTML text as response to the client.

2.3.2. 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 states.

NOTE: Pressing any buttons on the webpage counts as one CLIENT REQUEST. So, every time you press any button, the *temperature*, *hall*, and *led states* should update. e.g. if you press the **RED ON** button, red led connected to your board should light up, and the **RED LED Current State** should display ON as shown in Fig. 2. Now if you press the GREEN ON button, green led connected to your board should light up, and the GREEN LED Current State should display ON, as shown in Fig. 3.

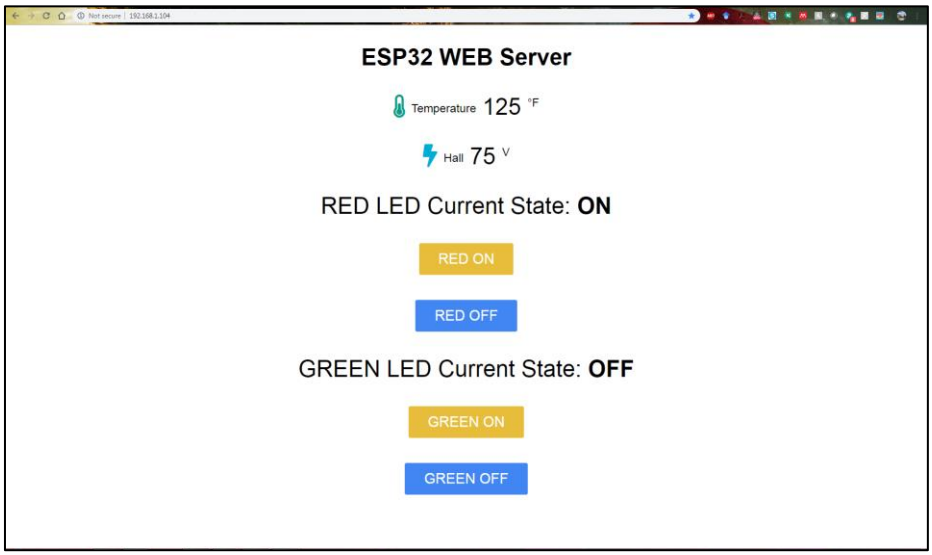


Figure 2. Webpage on pressing RED ON button

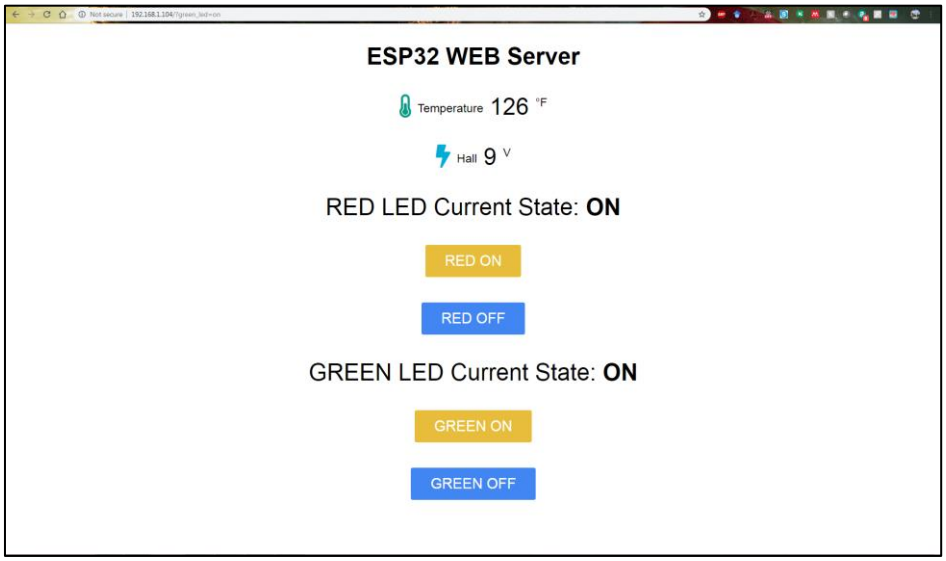


Figure 3. Webpage on pressing GREEN ON button

3. 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*_lab4**, where username is your CAREER account login ID. *This directory should contain only the following files, i.e., no executables, no temporary files, etc.*

1. ***espclient.py***: your program for ESP32 HTTP Client (**2.2.2**)
2. ***ThingSpeak screenshot***: Screenshot obtained from ThingSpeak website showing Temperature and Hall Senor data, each with 30 entries, as shown in Fig. 1.
3. ***espserver.py***: your program for ESP32 HTTP Web Server (**2.3.1**)

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

4. Bonus Exercise (10% of the total)

- Interface two external **push button switches** with your ESP32 board.
- Update the HTML text given in the espserver.py file to display the switch values, i.e. pressed or not. You don't need to create any buttons for the two switches. Your webpage should display the following in addition to the sensors and LEDs information:
    - Either SWITCH1 Current State: ON or SWITCH1 Current State: OFF depending on switch1 state.
    - Either SWITCH2 Current State: ON or SWITCH2 Current State: OFF depending on switch2 state.

NOTE: Follow the lab document strictly when using different peripherals/modules/packages. Points will be deducted if you fail to follow the lab instructions. If anything is NOT mentioned explicitly, you can use package/module to write your program.

REFERENCES

[1] Getting started with MicroPython on the ESP32
https://docs.micropython.org/en/latest/esp32/tutorial/intro.html

[2] ESP32 WROOM-32 Datasheet
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_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 Feather Online Manual
https://learn.adafruit.com/adafruit-huzzah32-esp32-feather

[5] Adafruit ESP32 Feather Schematics https://cdn-learn.adafruit.com/assets/assets/000/041/630/original/feather_schem.png?1494449413

[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 usocket and socket:
https://docs.micropython.org/en/latest/library/usocket.html