# Microprocessor-Based Systems (E155)

**D. Harris and M. Spencer** **Fall 2018**

Lab 6: Internet of Things

## Requirements

*Build an internet accessible device to control two LEDs, measure ambient light intensity, and measure three-axis acceleration. Use an ESP8266 with the provided webserver code to host the webpage, and use the onboard µMudd PIO, ADC and SPI peripherals to toggle two LEDs, read light intensity from a phototransistor and to read three-axis acceleration from an SPI LIS3DH accelerometer. An end-user must be able to blink the LEDs, read phototransistor voltage, and read total acceleration from the webpage.*

*Extra credit is available for improving the website or embedded system in some interesting way. Examples of acceptable improvements include independent control of the LEDs, displaying the current state of the LEDs on the webpage, or implementing another µMudd in a useful way.*

At heart, this lab requires you to directly control the memory mapped peripherals on the µMudd ARM processor, so refrain from consulting easySamIO.h or any other C implementations of the SAM4S peripheral set.

## ESP8266 Web Server

Broadly speaking, everything that you see on the internet is the product of one computer presenting text to another. The text is often formatted in a special, internet-specific, way that includes information about how to display it which is referred to as *hypertext*. (Forgive the early internet engineers this indulgences, I'm sure it sounded really cool at the time.) Hypertext is specified using a compact programming language called hypertext markup language or HTML. It is transferred over the internet based on a predefined set of agreements between all computers which is referred to as the hypertext transfer protocol or HTTP. The latter most of these acronyms should be familiar: whenever you type http:// into a web browser you are informing your computer that you are attempting to retrieve hypertext from the address that follows.

HTTP is complicated and servicing web service requests takes many steps. Fortunately, the tools necessary to do that are very mature. There are two common tools that interact with HTTP: the web browser, which lives on a receiving computer, sends internet requests, and renders the received hypertext, and the web server, which listens for requests from the internet and sends out hypertext in response.

Implementing an HTTP web server on the ARM microcontroller is a non-trivial task. Instead, you will be using an ESP8266, a small WiFi development board which incorporates a TCP/IP stack as well as an onboard WiFi and an integrated antenna. You are provided an Arduino language program which hosts an HTTP web server with an HTML page generated by the µMudd.

To program the ESP8266, you will need to install (as of 12/11/2018):

1) The Arduino IDE[1]
2) Board support for the ESP8266. This requires adding the ESP8266 board manager website[2] to the Arduino Board Manager within the Arduino IDE (File > Preferences), and then adding the ESP8266 board package within (Tools > Boards > Boards Manager).
3) The ESP8266 Exception Decoder[3]. ESP8266 crashes are notoriously difficult to debug, and this tool vastly simplifies the debugging process if you plan to alter the provided program. Note that this is not a required component of the lab.

The programming process is as follows:

1) Open the .ino Arduino program with the Arduino IDE
2) Select the correct serial port (Tools > Port). If you are unsure of the current serial port, you can:
    a. On Windows, open Device Manager, and under the Ports drop-down look for 'Silicon Labs CP210x USB to UART Bridge'
    b. On macOS or Linux, enter 'dmesg | grep tty' in your favorite terminal emulator. This will return a list of all serial port activity.
3) Select the correct board (Tools). We suggest the following settings:
    a. Board: "NodeMCU 1.0 (ESP-12E Module)"
    b. Upload Speed: "115200"
    c. CPU Frequency: "80 MHz"
    d. Flash Size: "4M (1M SPIFFS)"
    e. Debug port: "Serial"
    f. Debug Level: "HTTP_SERVER"
    g. IwIP Variant: "v2 Lower Memory"
    h. VTables: "Flash"
    i. Erase Flash: "Only Sketch"
4) Compile and upload the script with the "Upload" button

On opening the Serial Monitor, you will see a stream of debug messages displaying the current status of the ESP8266.

[1] https://www.arduino.cc/en/Main/Software
[2] http://arduino.esp8266.com/stable/package_esp8266com_index.json
[3] https://github.com/me-no-dev/EspExceptionDecoder

# ESP8266-µMudd Interface:

The ESP8266 program only allows the µMudd to communicate with the outside world. The µMudd must supply a webpage to the ESP8266, and must interpret any client requests to the ESP8266. The devices interface through a 9600 baud serial connection, hosted on pins D8 (TX) and D5 (RX) of the ESP8266. The protocol is as follows:

1) When the ESP8266 updates the webpage from the µMudd, it sends the most recent request from the client, within '<' … '>'. For example, a user accessing the page http://<server_address>/ledon would result in the request '<ledon>' send to the microcontroller. A user accessing the root webpage of the server, http://<server_address>/ would result in the request '<>'
2) The µMudd then transmits the entire web page to the ESP8266. The ESP8266 expects a webpage encoded as an HTML file. Therefore the webpage must start with '<!DOCTYPE html><html>' and end with '</html>'

Note that the ESP8266 does not request a webpage from the µMudd immediately on client request. Instead, it caches a copy of the webpage, and updates the webpage from the µMudd when the server is not processing a client request. At maximum, it will take 10 seconds after the client request to update the webpage, after which the student will have to reload the page in their web browser.

# µMudd Hardware and the Internet of Things:

The last component of this lab is to write a program that parses a request from the ESP8266, toggle LED states as necessary, read from the SPI LIS3DH accelerometer and phototransistor, and use this data to generate a webpage that is transmitted to the ESP8266.

We suggest using the LPT2023 phototransistor to measure ambient light intensity. You will have to design a circuit composed of the LPT2023, a power supply, and a resistor to generate an output voltage that varies with ambient light intensity. We recommend reading the LPT2023 datasheet before starting this design.

You will need to write an HTML webpage that displays dynamic acceleration and voltage data as well as creating requests to change the state of the LEDs. There are many ways to do this, but we suggest the following resources for information on HTML formatting and interactive elements:

http://www.w3schools.com/html/default.asp
http://www.w3schools.com/html/html_forms.asp

The final product of this lab is a simple member of an emerging class of devices called the Internet of Things. Proponents of these devices argue that everything—from your washing machine to your car to giant factories—should be connected to the internet so that the shared data can be used to optimize and improve societal functions. Internet-controlled lighting, and internet-accessible sensors are two promising domains for the field, and are exemplified in this lab.

# Credits:

This lab was original developed in 2015 by Alex Alves ’16, and redesigned for the µMudd Mark 5.1 by Kaveh Pezeshki ’21 and Christopher Ferrarin ’20.