Final Project
ESP8266 Wi-Fi Communication

Goals:
• Setup the UART communication to the ESP8266 using the MSP430
• Display incoming information from the ESP8266 on the LCD Display. IP address of the ESP8266 must be displayed, all other messages are optional.
• Connect the ESP8266 to the internet and set it up as a server.
• Send commands with your computer to the ESP8266 using the provided program.
• Use the commands to get data from an accelerometer OR to position a servo motor

Deliverables:
• System requirements list +5
• Inputs and Outputs list +5
• Schematic of final circuit +10
• Flow diagram +20
• C source code +20
• Demo on finals week +40

Bonus:
• Use both the accelerometer and servo +10
• System is battery powered +10
• System uses low power modes +10
UART (Universal Asynchronous Receiver Transmitter) Serial Communication

The MSP430 offers 3 different types of serial communication protocols: SPI, I2C and UART. For this project we will be using the Universal Asynchronous Receiver Transmitter (UART). Serial communication as the name implies send bits sequentially, meaning that the bits are shifted into a transmission line 1 bit at a time. Group those bits together and you will have a frame (Look at figure 1) that can be decoded to represent anything (ASCII characters, binary data, images or audio). Asynchronous means that the devices will not be sharing a clock mechanism to synchronize the transmitted/received bits. This will normally cause an issue when using other serial communication protocols, but with UART both ends of the communication agree to use a specific clock speed to send and receive data. This agreed clock speed is what is known as the Baud Rate, and it is important to have the same Baud Rates on both ends. You can also dedicate an extra bit to detect bit frame errors (known as the parity bit), but for this project we will not use it.

For this lab we will be using the UART with the following parameters:

- 8 bit data
- LSB first
- 1 stop bit
- Interrupt Enable for Receiver
- Second stage modulation = 1
- UART Clock = SubMaster Clock running at 16 MHz
- Baud Rate = 115200

You will see that you do not have to worry about setting all of those yourself. If you go to the provided library, you will find a function called UART_Init() which takes care of initializing the UART module.
**ESP8266 Wi-Fi Module**

The ESP8266 Wi-Fi module is a communication device that can function as a station or an access point for internet communication. The ESP8266 uses UART to communicate with an external microcontroller. It can be controlled by sending AT commands to it, which consists of ASCII characters. For example if the ESP8266 receives the `AT+CIPSTATUS` command, the ESP8266 will respond with status data in a series of strings that look like the following:

```
AT+CIPSTATUS
OK
status:GOT IP
id:0
type:TCP
addr:192.168.1.200
port:50
tetype:1
```

We will be using the ESP8266 with the following parameters:

- Connect to an access point and obtain a dynamic IP address
- Enable multiple connections
- Enable server at port 50
- Set as a station
- Wait for data client to transmit data

There are several functions in the provided library to setup the ESP8266 as specified. Find the appropriate functions to connect to an access point, setup a server at port 50, and then wait for a client to send data.
Connect the ESP8266 following figure 2 and table 1

![ESP8266 Pinout](image)

**Figure 2 ESP8266 Pinout**

**Table 1. ESP8266 Connections to the MSP430**

<table>
<thead>
<tr>
<th>ESP8266 Pin</th>
<th>MSP430 Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>VCC</td>
</tr>
<tr>
<td>RESET</td>
<td>P1.0</td>
</tr>
<tr>
<td>CH_EN</td>
<td>VCC</td>
</tr>
<tr>
<td>TX</td>
<td>P1.1</td>
</tr>
<tr>
<td>Ground</td>
<td>GND</td>
</tr>
<tr>
<td>GPIO_0</td>
<td>NC</td>
</tr>
<tr>
<td>GPIO_2</td>
<td>NC</td>
</tr>
<tr>
<td>RX</td>
<td>P1.2</td>
</tr>
</tbody>
</table>
The following is an example on how to initialize the ESP8266 and echo any received data from a client.

```c
#include "lcdLib.h"
#include "ESP8266.h"

char Data[128];

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    ESP8266_Init();
    lcdInit();
    _enable_interrupts();

    lcdSetText("Wait...", 0, 0);
    ESP8266_SoftReset();
    ESP8266_ConnectAP("SSID", "Password");
    ESP8266_ServerInit(50);

    UART_Flush();
    lcdClear();
    lcdSetText("Ready!", 0, 0);

    while(1)
    {
        lcdSetText("Waiting for Data",0,0);
        ESP8266_WaitData(Data);
        ESP8266_SendData(Data, strlen(Data));
        lcdClear();
        lcdSetText("Ready!", 0, 0);
    }
}
```
ESP8266 Windows Client Program

In order to send data to the ESP8266 we need a program that will connect to the ESP8266 using its IP address and port number. An executable file for windows computers is provided so that you can do exactly that. In order to use this program follow these instructions:

1. Open a Command Prompt window

2. Navigate to the folder where you downloaded the ESP8266.exe file (type `cd folderpath`). In my case I downloaded the file on my desktop so I will type `cd desktop` to navigate to the desktop folder.

3. On the command prompt, type `ESP8266 [IP Address] [Port Number] [Data]`, without the square brackets. My ESP8266 has IP address 192.168.1.100 and it is listening at port 50, so I will type the following on the command prompt window:

   **ESP8266 192.168.1.100 50 “Hello World”**

   The ESP8266 will then receive the *Hello World* string. Figure 3 show the commands that you will need to type into the command prompt.

---

**Figure 3 Commands to run ESP8266 Client program**