Lab #5
ESP8266 WiFi Module

Ver 1.12 w/software serial

ESP8266 Background[1]:

The ESP8266 module is a TTL "Serial to Wireless Internet" device. This allows us to connect the ESP8266 to any microcontroller with a UART.

Please refer to previous labs and class notes on UART.

Since it is a serial device, we need a way to send commands to the device, but the device needs to be able to differentiate commands from data. To do this, the ESP8266 uses an old idea of the "AT" command.

AT commands were first used in ~1981 to send commands to Hayes Dial Up Modems. Commands were proceeded by the keyword AT (standing for ATTENTION).

The ESP8266 uses such commands.

Device Setup Procedure[2]:

First we need to setup the device to connect to our WiFi network.

To do this connect the device as follows:

Arduino	ES98266		Arduino
3.3v	3.3v	RX	RX (0)
nc	RST	IO0	Nc
3.3v	EN	IO2	Nc
TX (1)	TX	GND	GND

Also connect the Arduino GND and RESET pins (this will make the Arduino act like a USB to SERIAL Converter, putting the Atmel328P in a constant reset mode and allow us to send commands directly from the PC to the ESP8266 without the need to code).

We are going to use the Arduino IDE's dumb terminal instead of putty for this experiment. To open:

- Open Arduino IDE
- TOOLS | PORT (Set port to board port)
- TOOLS | Serial Monitor
- Setup "Both NL & CR" and "115200 baud"

[1] https://www.instructables.com/id/Using-the-ESP8266-module/
[2] Same as above

You should now be able to communicate with the ESP8266. To verify this, type AT into the box at the top on the screen at either press enter or click on the SEND button. You should see “OK” as a response. If not, it is possible the ESP8266 had its com settings change, so change the terminal setting to 9600 baud and try again. If this did not work, ask the instructor for help.

Next we will reset the ESP8266 with the following command:

AT+RST

You will see some data returned about the board. Then send the following command:

AT+CWMODE=3

This sets the module as both a client and an access point. You should get an OK back.

Now we need to check what WiFi networks are available (very similar to seeing the networks within range on your mobile device). To do this type the command:

AT+CWLAP

Verify that you can see "Sandbox_227 network". If not, ask the instructor for help.

If you did see the network, you are now ready to set up the network SSID and WEP Password on the ESP8266. To do this use the command:

AT+CWJAP="Sandbox_227","PASSWORD"

(*PASSWORD* will be provided by the instructor)

You should get a response back "WIFI Connected" then "WIFI GOT IP".

To see what IP address your device was assigned, you can use the command:

AT+CIFSR

You should get an IP address of 192.168.0.xxx (where xxx is > 100).

Lastly we default the ESP8266 to 9600 BAUD by the command:

AT+UART_DEF=9600,8,1,0,0

[from this point forward the terminal must be set for 9600 baud]

Manually Retrieve a Website

Now that the board is connected to the WiFi network and has an IP address. We can now experiment with retrieving a website. To do this, multiple commands must be sent in quick succession. It is recommended that you type the commands into notepad or notepad++ then copy and past quickly into

the send box on the terminal window (using control+c to copy and control+v to paste) If it is not done quickly enough a timeout will occur and close the connection (and you will have to start again).

The commands to send are as follows:

AT+CIPMUX=1

AT+CIPSTART=4,"TCP","test.dankohn.info",80

AT+CIPSEND=4,16

GET /test.html

This should show you a webpage version of "hello world". Note the 4 in the two lines above are the buffer being used and must be the same value. The 16 is the number of characters to be transmitted (so the number of characters in "GET /test.html" PLUS 2 (for CR and LF).

HTML FORMS Background:

You have probably use form on a website in the past. A html form allows a user to fill in data to be sent to a server to be processes by a separate script.

Goto the website:

http://test.dankohn.info/test_get_form.html

The HTML code for the form is as follows:

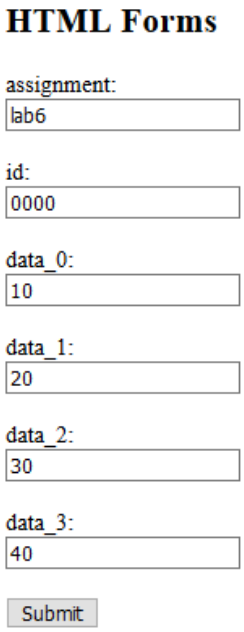
```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form action="/test_get.php" method=get>
  assignment:<br>
  <input type="text" name="assignment">
  <br><br>
  id:<br>
  <input type="text" name="id">
  <br><br>
  data_0:<br>
  <input type="text" name="data_0">
  <br><br>
  data_1:<br>
  <input type="text" name="data_1">
  <br><br>
  data_2:<br>
  <input type="text" name="data_2">
  <br><br>
  data_3:<br>
  <input type="text" name="data_3">
  <br><br>
  <input type="hidden" name="code" value="1234">
  <input type="submit" value="Submit">
</form>


</body>
</html>
```

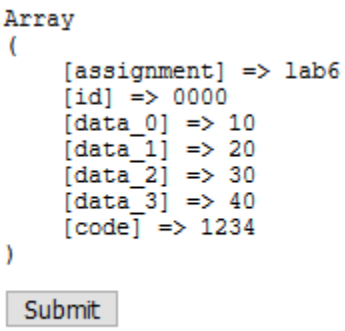
Type into the form the following information:



[Replacing id 0000 with the last 4 digits of YOUR U number]

When you press the submit button, this information is past to a script that will display the information passed to it.

You should see:



Also note the URL in your browser. It should say:

http://test.dankohn.info/test_get.php?assignment=lab5&id=0000&data_0=10&data_1=20&data_2=30&data_3=40&code=1234

This is typical of a web form using the GET method to send data to a script.

Note that each named element in the array displayed is the "name" defined in the HTML code for each textbox.

Also note, that when submit is pressed (on the form webpage) each textbox name becomes a variable and each variable is assigned the value typed into the box. So for the assignment textbox, defined by the HTML code

```
<input type="text" name="assignment">
```

Gets converted to `assignment=lab6` in the URL request for the next page. That page is defined in the form action line `<form action="/test_get.php" method=get>`.

The ? in the URL generated says variables and values to follow, and each variable is separated by the & symbol.

We can accomplish the same thing with the ESP8266 by doing the following commands:

AT+CIPMUX=1

AT+CIPSTART=4,"TCP","test.dankohn.info",80

AT+CIPSEND=4,93

GET /save_get.php?assignment=lab5&id=0000&data_0=10&data_1=20&data_2=30&data_3=40&code=1234

[Note GET to 1234 are on ONE CONINUOUS LINE, the save_get.php send it to the script that stores the information to the final database]

Class Database and Graphing Data:

We will be using the GET method to send data to a script that allows data to be stored and then graphed. The "form data" is as shown above (using the last 4 digits of YOUR U Number for the ID). Note that "code" must be sent with your data and MUST be 1234 for the data to be accepted and placed into the database.

To view your graph: use the URL: http://test.dankohn.info/graph.php?table_name=tbl_lab5_0000 replacing 0000 with the last 4 digits of your U number.

You can also look at the data in table form: use the URL: http://test.dankohn.info/show_data.php?table_name=tbl_lab5_0000 replacing 0000 with the last 4 digits of your U number.

Software Serial:

Since the Arduino Uno only has one UART and it is used for uploading code and for PRINTF statements, we will used a SOFTWARE SERIAL shell to create this program.

Connect the EPS8266 as described in the comments. (IE move RX and TX lines from the previous connection to the new pins - 8 and 2). Also, remove the jumper from GND to RESET.

You should NOT have to modify anything but main() in this program. Lines 77-78 show how to send a string. Sprintf is very similar to printf, except it stores the string to an array (1st parameter in the function call). You can use %i, %c etc in the string and then follow with the variables to print, just like in printf.

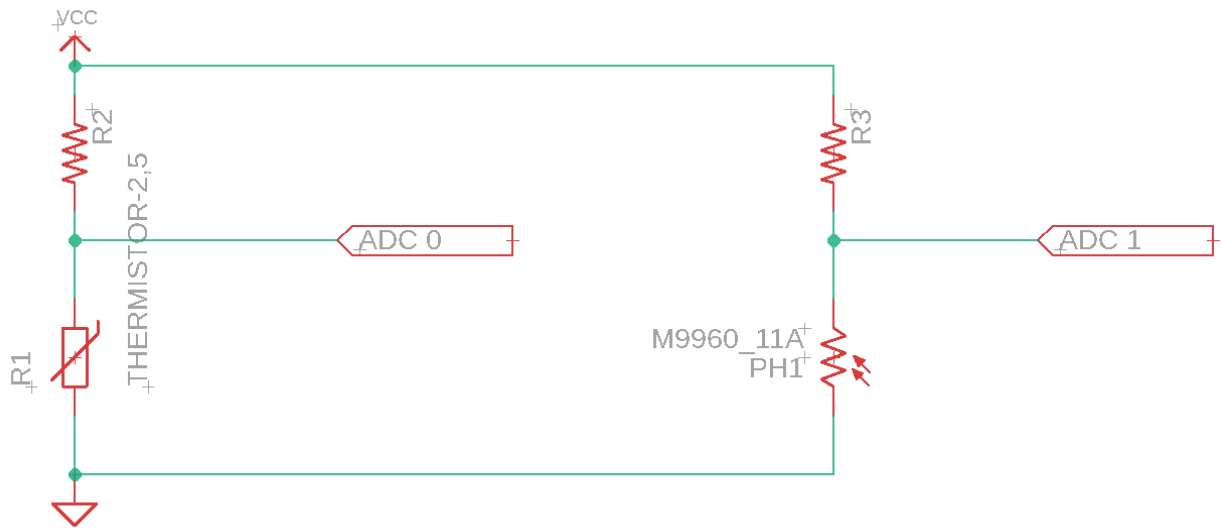
Line 81 reads the response from the connected device and prints it to the Terminal (it does not store the response, but for this program we are not checking the response string anyway).

C Code Hint:

Since you need the length of the string to be sent, you can create the string needed via sprint function and use strlen function to get the length. See C Ref sheet for info.

Lab Assignment:

Connect a thermistor and Light Sensor to the Arduino board as follows:



Send the values read from the ADC (Refer back to TECH 3233 Lab #5 for info on ADC) for the sensors to data_0 and data_1 to the website every 20 seconds (aprox).

Collect valid data for at least 5 minutes. Submit the commented project (.zip) via online submission.