Remote IoT data monitoring and control using Wi-Fi

1. Project description

In this project, the Arduino board is connected to Wi-Fi module and both act as a server. The Arduino Board is connected to 2 LEDs, 3 sensors (temperature, smoke, motion). The client which is a web browser - running on any computer connected to the same network as the Arduino- communicates to the server and controls it.

The project **must** do the following **2 processes**:

I. The Arduino board must monitor and display sensors data periodically (Temp, Motion, smoke) on LCD every 2 seconds (using timers).
II. The webpage will send a command to the Wi-Fi module to control the Arduino board and switch each of the LEDs on/off.

2. List of hardware requirements:

- Arduino UNO
- Wi-Fi Serial TTL Module ESP-01S (Ram Electronics)
- LCD
- Sensors
- Temperature sensor: LM35
- Smoke sensor MQ2 (Digital/Analog) (Future Electronics)
- PIR Motion sensor module (Adjustable Range) (Future Electronics)
- LEDs, Wires

3. Notes:

1. You need to use Timers to implement the 2 seconds (any timer with any mode).
2. You need to use external interrupts (INT0 or NT1) to detect and read the motion sensor.
3. You need to use ADC functions for reading an analogue value from the sensors (refer to lab 6**).**

Note: **Please keep in mind that this project Can be implemented using the Arduino IDE, and that only Software Serial (UART) and LCD Arduino built-in library are permitted to be used.**

4. Bouns

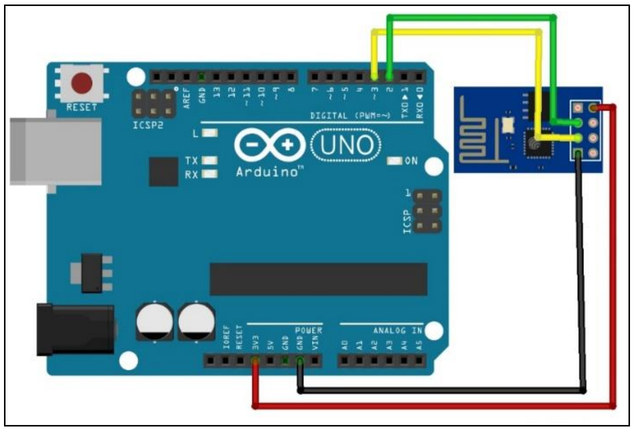
- The Arduino board must detect abnormal conditions and report a notification alert to the web page. The abnormal conditions are:

1. Tempertesure value exceeding 30 degrees.
2. Detecting an existence of smoke.

5. ESP Wifi Arduino connection

The ESP8266 ESP-01 WiFi module and the Arduino microcontroller can be interfaced through UART and with the help of a wide range of AT Commands.

ESP8266	Arduino uno
RX	TX
TX	RX
VCC	3.3V
EN	3.3 V
GND	GND



➢ *Note that to be able to get accurate 3.3 volt from Arduino, you need to use voltage regulator (l7805cv voltage regulator) that will be connected to 5v of Arduino and output 3.3 voltage where VCC and EN pins of ESP 8266 will be connected to it*

6. Detailed procedures for using ESP-01S Wi-Fi module:

I. **Initialization procedure:** To initialize the ESP-01 Wi-Fi module, you should send a sequence of AT commands to do the following:

1. Reset the ESP8266 module.
2. Connect to the Wi-Fi network.
3. Set the ESP8266 Wi-Fi mode to station mode.

4. Show IP Address.
5. Configure Wi-Fi module for multiple connections.
6. Start the communication at port 80, port 80 used to communicate with the web servers through the http requests.

The AT Command is a string message sent through the UART to control the Wi-Fi features of the ESP8266 Module. For helping you in using AT commands, we added ***Appendix A*** *at the end of this documentation.*

II. **Building the web page:** A web page is built and sent using AT commands to be accessed by any web browser having the IP of the Arduino Server. *For helping you with this part, we added Appendix B at the end of this documentation.*

III. **Client – Server communication :** To handle the communication between the client and the server, The Arduino microcontroller next, will continuously read the serial input (UART) and check for any request from the client. *For helping you with this part, we added Appendix C at the end of this documentation.*

Appendix A

ESP8266 - AT Command Reference

ESP8266, in its default configuration, boots up into the serial modem mode. In this mode you can communicate with it using a set of AT commands. Below a reference of all known AT commands that ESP8266 supports, with explanation

- http://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/#AT+CWMODE
- https://www.pridopia.co.uk/pi-doc/ESP8266ATCommandsSet.pdf

ESP Wi-Fi Fundamental AT commands

1. AT+RST

The "AT+RST" command is used to reset the ESP8266 module. This command restarts the module and restores it to its default settings.

When the "AT+RST" command is sent to the module, it will perform a reset and then send a response. The response will typically be:

```
ready
```

2. AT+CWMODE

The ESP8266 can be set to 3 different modes. To set the module as a station, enter AT+CWMODE=1. To set as an access point, enter AT+CWMODE=2. As both, enter AT+CWMODE=3. To check the current mode the ESP8266 is in, use AT+CWMODE?

1. We need the ESP8266 to connect to an existing Wi-Fi network and also act as a server that serves a web page to clients on that same network, then you can set the ESP8266 to Station (STA) mode using the `AT+CWMODE` command with a parameter value of `1`.
2. In this mode, the ESP8266 will function as a Wi-Fi client and connect to the existing network, allowing it to communicate with other devices on the same network. The ESP8266 can then act as a server and serve a web page to clients on that same network.

3. Note that in this mode, the ESP8266 will be assigned an IP address by the router it connects to. You can use the `AT+CIFSR` command to retrieve the assigned IP address and use it to connect to the ESP8266 from a client web browser on the same network.

or when it needs to communicate with other devices on the same network.

3. AT+CWLAP

See the available WiFi networks in your location.

4. AT+CWJAP="WiFi network name","Wifi network password"

Connect to a WiFi network.

Response:

Assuming that the Wi-Fi credentials are correct and the access point is available, the response of the ESP8266 module should be something like:

```
WIFI CONNECTED
WIFI GOT IP
```

5. AT+CIFSR

See ESP-01’s MAC and IP address.

The response might look something like this:

```
+CIFSR:APIP,"192.168.4.1"
+CIFSR:STAIP,"192.168.0.10"
+CIFSR:STAMAC,"5c:cf:7f:00:01:23"
+CIFSR:GATEWAY,"192.168.0.1"
+CIFSR:NETMASK,"255.255.255.0"
```

6. AT+CIPMUX

Enable multiple connections using AT+CIPMUX=1. Disable it by using AT+CIPMUX=0.

Response:

```
If the module is successfully set in multiple connections mode, the response will be:

OK
```

7. AT+CIPSERVER

Start a server using AT+CIPSERVER=1,80. In this command, "1" is the parameter to enable the server mode, and "80" is the port number that the server listens on. The port number 80 is the default port used for HTTP communication, so it is commonly used for web servers.

Response:

```
If the server is already running, the response will be something like:

OK
```

Appendix B

Building the Webpage

1. A web page displaying a title, two buttons for controlling the LEDS, and an alert for showing the abnormal sensor reading should be built using HTML, CSS, JavaScript. For knowing how to write the webpage, **Refer to:** https://www.w3schools.com/html/html_intro.asp
2. Define the HTML content of the webpage as a string variable inside the Arduino sketch. Here is an example of how the HTML code can look like for a webpage with two buttons to control two LEDs:

   Ex:

```
String webpageContent = "<html><head><title>LED
Control</title></head><body><h1>Control the LEDs</h1><button
onclick=\"sendCommand('led1_on')\">LED 1 On</button><button
onclick=\"sendCommand('led1_off')\">LED 1 Off</button><br><br><button
onclick=\"sendCommand('led2_on')\">LED 2 On</button><button
onclick=\"sendCommand('led2_off')\">LED 2 Off</button><script>function
sendCommand(command) {var xhttp = new XMLHttpRequest();xhttp.open('GET', '/'+command,
true);xhttp.send();}</script></body></html>";
```

3. After the web page Content is implemented inside Arduino sketch (as a string variable), this web page content When a client connects to the server, send the web page Content that is implemented inside Arduino sketch (as a string variable), as a response using **the AT+CIPSEND command.**

4. This web page content will be send sent through UART using (AT+CIPSEND) command followed by the webpage content itself (string variable):

   a) Sending AT+CIPSEND AT command:

```
AT_CIPSEND=Param1,Param2
```

- Param1: represents the connection ID, which should be the ID of the client connection that just connected.
- Param2 :represents the length of the HTML content string variable.

  - *To get **Connection ID**, esp6288 will receive network data with prefix +IPD this is happens whenever any client connect to ESP6288 through same local network this data will be available in ESP6288 buffer and by checking the network data which contains IPD keyword we can easily extract the **Channel ID** from it.*
  - *The available network in ESP6288 BUFFER is started with **+IPD** Keyword along with the **channel ID**, data length, network data received.*

  Ex: AT+CIPSEND=0,5 The "0" indicates the channel through which the data is going to be transferred; while "5" represents the number of characters that are going to be sent.

5. Wait for a response from the ESP8266-01S module that indicates that it is ready to receive the HTML content. The response should be ">".

6. Send the HTML content string variable through the serial communication to the ESP8266-01S module.

7. Wait for a response from the ESP8266-01S module to the Arduino board through the serial communication that indicates that it has sent the HTML content successfully. The response should be "SEND OK", this means that the data has been transmitted successfully to the client (browser). However, if nothing appears on the web browser's window. This is because it is required to close the channel first to display the characters. We use the following command to close the channel:

```
AT+CIPCLOSE=ChannelID
```

Channel ID: indicates the channel that is being closed.

Appendix C

ESP Wi-Fi module (client –server) communication

- To handle the communication between the client and the server, The Arduino microcontroller will continuously read the serial input (UART) and check for any HTTP request from the client, as When a client connects to the server(esp8266_arduino), a "+IPD" notification message will be send to the serial port indicating that data has been received. You can read this notification and extract the client's request.
- By reading read the HTTP request sent by the client and extract the command from the request. The command will be the part of the URL that comes after the IP address of the server.

HTTP Request and Response

When the button is clicked, it will generate an HTTP GET request that sends the URL and the value from the button to the Arduino server. The Arduino sketch will reads the HTTP request header and checks for the text **LEDON** and if found, the Arduino microcontroller will turn the LED on.

- To check for a request, the **find** function in Software Serial library is used to search in the message received.

For more info about how to use button as a link refer to: *https://www.w3schools.com/tags/tag_a.asp*

- The following is an example of an HTTP request sent from the Firefox browser to the Arduino server after clicking the button:

```
GET /LEDON HTTP/1.1
Host: 10.0.0.20
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:18.0)
Gecko/20100101 Firefox/18.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-ZA,en-GB;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://10.0.0.20/
Connection: keep-alive
```

For more understanding of URL request parameters **Refer to**: A guide for URL parameter