

Lab 8. Arduino and WiFi - IoT applications

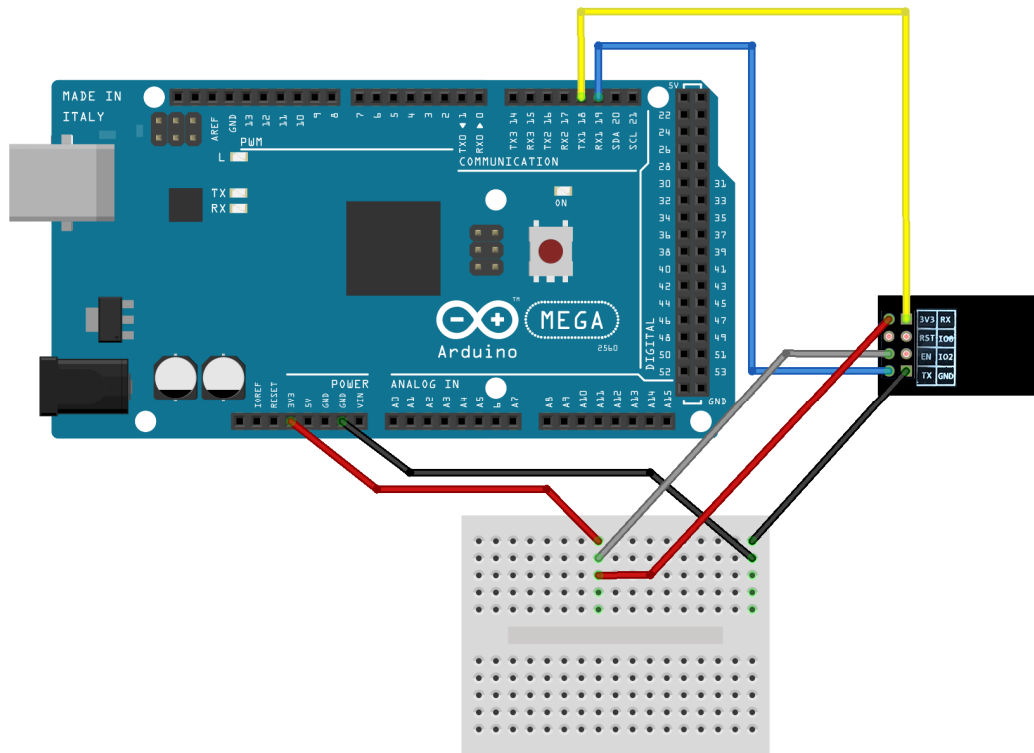
IoT - Internet of Things is a recent trend that refers to connecting smart appliances and electronics such as microcontrollers and sensors to the internet. In this laboratory we will be using an Arduino and an ESP8266 WiFi module. We will control the built-in LED from Arduino directly from a webpage that is hosted on the ESP8266 module. We will also be able to display data from Arduino on the webpage.

ESP 8266 is a standalone WiFi module that can be programmed using the Arduino IDE. It can be programmed via a FTDI programmer, or by using an Arduino board with the ATmega328 chip removed. **In this laboratory we will be configuring the ESP8266 board using the serial communication, by issuing AT commands.**

Connecting the WiFi module

We will connect to the ESP board to Arduino by using the UART interface Serial1. Connect the RX pin of the ESP8266 to **pin 18 (TX1)** on Arduino and the TX pin from ESP8266 to **pin 19 (RX1)** on Arduino. Connect the GND pin to Arduino's GND, and the VCC/3V3 pin to the Arduino **3.3V pin**. You must also connect the **EN pin** of the ESP8266 to the **3.3V power source**.

Consult the schematic below for connecting the WiFi module to Arduino:



The Arduino program must issue AT commands to reset the WiFi module (`AT+RST`). The next step is to configure it as an access point (`AT+CWMODE=2`). After that, we get the IP address:

192.168.4.1 using the command: “AT+CIFSR” which also prints the MAC address of the module. Then we query to get the SSID info (“AT+CWSAP?”): the actual **WiFi network name** and password (default is not set) and then we configure for multiple connections (“AT+CIPMUX=1”) and turn on the web server on port 80 (“AT+CIPSERVER=1,80”). Each AT command must end with carriage return and newline (“\r\n”).

More information regarding the ESP8266 AT commands can be found here: https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf and here: https://github.com/espressif/esp8266_at/wiki.

The commands are sent via the serial interface that was setup from Arduino (“**Serial1**”), by using the “*print*” method. The module’s response to an AT command is read and saved in a string, and then displayed in the serial monitor of the connection between the PC and Arduino (“**Serial**”). Please consult the “**sendData()**” method in the example below.

In the loop we check if data is available on the Serial1 interface, and check if this data is data from the network (it will include the “+IPD” substring). We first read the connection id as it is required when sending data using the command: “AT+CIPSEND”. A simple webpage is constructed as a string and sent to the ESP8266 module. The webpage includes some text to display, two buttons for user input, and below the buttons is another text for displaying data from Arduino. After the AT command for the webpage is sent, we must close the connection by using: “AT+CIPCLOSE”.

The mechanism used for controlling the led on Arduino is built using the buttons on the webpage and the URLs they point to. The first button points to “/0” and the second one to “/1”. By clicking them, the webpage tries to redirect to those addresses and it issues a request on the webserver. On Arduino, we will receive the request response by reading the ESP8266 response (in “*sendData()*” method) and we check if it contains the “/0” or “/1” substrings in the response string (ex: “*response.indexOf("/0") != -1*”).

Data from Arduino is displayed on the web page by adding the result from “*readSensor()*” method to the webpage string and sending it using AT commands to the WiFi module. In the example below the read sensor method will actually display the result of “*millis()*”.

ESP 8266 example code:

```
#define DEBUG true

void setup() {
  Serial.begin(115200);
  Serial1.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);
  sendData("AT+RST\r\n", 2000, false); // reset module
  sendData("AT+CWMODE=2\r\n", 1000, false); // configure as access point
  sendData("AT+CIFSR\r\n", 1000, DEBUG); // get ip address
  sendData("AT+CWSAP?\r\n", 2000, DEBUG); // get SSID info (network name)
```

```

sendData("AT+CIPMUX=1\r\n", 1000, false); // configure for multiple connections
sendData("AT+CIPSERVER=1,80\r\n", 1000, false); // turn on server on port 80
}

void loop() {
  if (Serial1.available()) {
    if (Serial1.find("+IPD,") {
      delay(500);
      int connectionId = Serial1.read() - 48; // read() function returns
      // ASCII decimal value and 0 (the first decimal number) starts at 48
      String webpage = "<h1>Hello World!</h1><a href=\"/I0\"><button>ON</button></a>";
      String cipSend = "AT+CIPSEND=";
      cipSend += connectionId;
      cipSend += ",";
      webpage += "<a href=\"/I1\"><button>OFF</button></a>";

      if (readSensor() > 0) {
        webpage += "<h2>Millis:</h2>";
        webpage += readSensor();
      }

      cipSend += webpage.length();
      cipSend += "\r\n";
      sendData(cipSend, 100, DEBUG);
      sendData(webpage, 150, DEBUG);

      String closeCommand = "AT+CIPCLOSE=";
      closeCommand += connectionId; // append connection id
      closeCommand += "\r\n";
      sendData(closeCommand, 300, DEBUG);
    }
  }
}

String sendData(String command, const int timeout, boolean debug) {
  String response = "";
  Serial1.print(command); // send command to the esp8266
  long int time = millis();
  while ((time + timeout) > millis()) {
    while (Serial1.available()) {
      char c = Serial1.read(); // read next char
      response += c;
    }
  }
  if (response.indexOf("/I0") != -1) {
    digitalWrite(LED_BUILTIN, HIGH);
  }
  if (response.indexOf("/I1") != -1) {
    digitalWrite(LED_BUILTIN, LOW);
  }
}

```

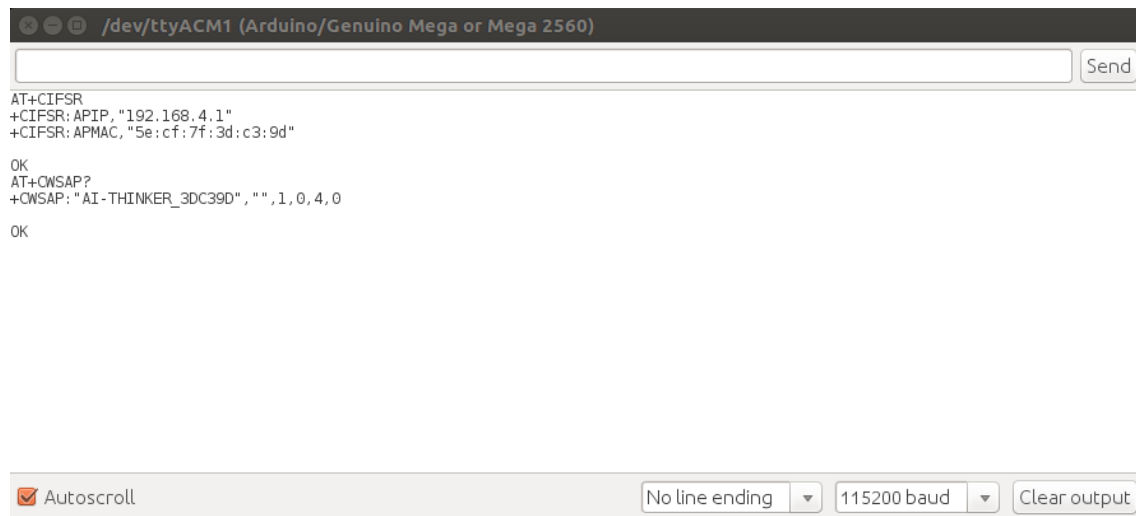
```
}  
if (debug) {  
  Serial.print(response);  
}  
return response;  
}
```

```
unsigned long readSensor() {  
  return millis();  
}
```

Running the program

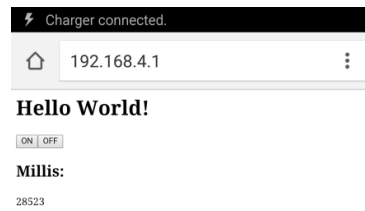
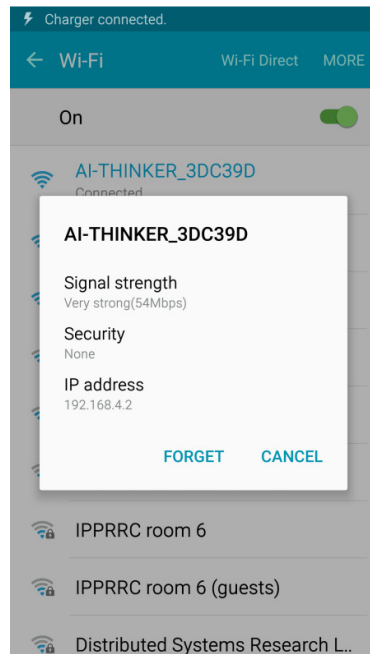
Upload the code on Arduino. Make sure the ESP8266 module is powered on the **3.3V** and that the EN pin is also connected to the **3.3V power source!** **Do not connect any of these pins to 5V, or the adapter will be destroyed!**

For convenience, use your smartphone and scan for WiFi networks. You should find the network's name displayed on the Serial Monitor.



```
AT+CIFSR  
+CIFSR: APIP, "192.168.4.1"  
+CIFSR: APMAC, "5e:cf:7f:3d:c3:9d"  
  
OK  
AT+QWSAP?  
+QWSAP: "AI-THINKER_3DC39D", "", 1, 0, 4, 0  
  
OK
```

Do not forget to change **the baud rate in serial monitor to 115200!** As you can see, the network name for this module is “AT-THINKER_3DC39D” and with no password. The same network should be visible on your mobile device:



Left: WiFi network of the ESP9266 module. Right: The webpage with buttons and displaying the *millis()*.

Note: each individual adapter will have its own unique network name! Please connect only to your own device, not to the devices of the colleagues !

After you connect to the WiFi network, open a browser and enter the IP address of the web server: **192.168.4.1**. See the above screenshot of the resulting webpage on the mobile browser.

Individual work:

1. Run the example. Make sure the connections to the WiFi module are correct, and that the module is powered at 3.3 V.
2. Modify the example to display additional data in the web browser. Connect a sensor to Arduino, and use the webpage buttons to select between displaying the time (*millis()*) and the sensor data. Make sure that the webpage tells the user what kind of information is displayed.
3. Using the WiFi module's manual, change the settings of the wireless network: change the SSID, add a password and an encryption mode.