



System-on-chip Architecture

Lab 3: STM32F0-Discovery and Silica Branca Wi-Fi

Andrea Floridia, 224906

Luca Mocerino, 229006

Robert Margelli, 224854

Contents

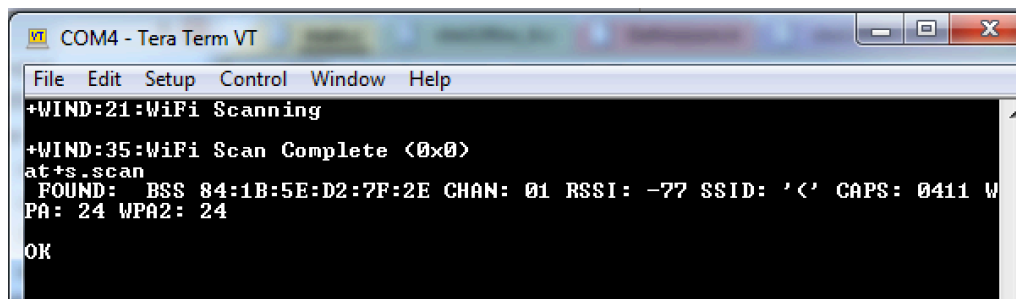
1. Purpose of the lab session	2
2. The Silica Branca board.....	2
3. The main.c	3
4. Conclusions.....	6

1. Purpose of the lab session

The purpose of this lab session was to run and modify a sample IoT application in which the user can send commands via the browser to control the Discovery board.

2. The Silica Branca board

The Silica Branca is a WiFi module used for prototyping IoT applications. We first tested it by sending commands via TeraTerm and connecting it to an available WiFi connection. The module supports AT commands, which enables a simple way to serially send and receive data to it.



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
+WIND:21:WiFi Scanning
+WIND:35:WiFi Scan Complete <0x0>
at+s.scan
FOUND: BSS 84:1B:5E:D2:7F:2E CHAN: 01 RSSI: -77 SSID: '<' CAPS: 0411 W
PA: 24 WPA2: 24
OK
```

Figure 1 - Example of the AT+S.SCAN command and its response

The second step was connecting it to the STM Discovery board and running the provided sample program. The Branca board has both a Flash and RAM which can be used to store files, in our case html pages that in this application are used to interact with the system:

1. Index.html: this is the landing page and can be accessed by going to the module's IP address on the browser (eg. 192.168.0.21);
2. Config.html: lists the configuration (i.e. "hardcoded") variables;
3. Status.html: lists the status (runtime) variables;
4. Led.html: presents the values of the blue and green LEDs of the Discovery board, this is what the user monitors;
5. Cgi_demo.html: a simple form with which the user can send commands to the Branca board which then relays them to the Discovery, this is what the user controls.

In the following we present details about the program that sends and receives data from the WiFi module.

3. The main.c

The code is divided in several parts:

- **Initialization:** in this part all the used peripherals are initialized, such as: USART with GPIOs and the NVIC structure related to this functionality; user button (PA0) to begin the connection to the net; PA2 as USART Tx; PA3 as USART Rx; LEDs (green and blue) used to monitor the status of the connection.
- **Wifi initialization:** with function `uint8_t ConfigureWiFi(void)` the Discovery Board, through USART, sends all the necessary configurations to setup a connection using specific credentials. The SILICA Board returns acknowledgements to notify that the connection is in progress and then succeeds. The most meaningful functions in this phase are described below:

- `uint8_t ConfigureWiFi(void)`: it is the function that allows connection to AP. The following steps are necessary:
 - 1) Send Router Name;
 - 2) Send Router Password;
 - 3) Send Router Protection Mode;
 - 4) Send Router Radio in STA Mode;
 - 5) Send Router DHCP Client;
 - 6) Send Router Save Settings;
 - 7) Send Router Soft Reset;
 - 8) Test the WiFi Connection checking the `:WiFi Up:` response;
 - 9) Prepare the LED.HTML page to be loaded on the STM WiFi;
 - 10) Prepare the upload of the LED.HTML page;
 - 11) Upload the LED.HTML page;
 - 12) Clear the RxBuffer.

Between each operation a check for OK response is needed.

- **Support functions:** functions used for parsing data, to choose correct html pages and so on. The most meaningful are described in the next lines:

- `void LoadAppropite_LedPage(void)` : according to the type of command or status of LEDs, this function chooses the correct html page to load. The following steps are necessary:
 - 1) Delete the led.html page;
 - 2) Clear the RxBuffer;
 - 3) Prepare a blank led.html page;
 - 4) Prepare to upload the led.html page;
 - 5) Test the value of LedG and LedB for upload the appropriate led.html page;
 - 6) Test the OK answer;
 - 7) Clear the RxBuffer.

- `uint8_t Search_B2inB1 (...)` : it is used to search a predefined string in the buffer.

3.1 Point 1: Implement AP scanning

We managed to implement this command by using the predefined AT command `at+s.scan` in order to scan the nearby access points nearby. One more command was to be added, we called it `SCAN`. By typing that command on the `Cgi_demo.html` page, this string is sent through the net to the SILICA Board and through serial connection to the Discovery Board. Here that string is recognized and the `at+s.scan` command is sent serially and follows the inverse path of the one described above. After that, we can see the scanning of APs using HTERM.

3.2 Point 2: Print IP address on HTERM

Because the firmware within our SILICA Board contains some bugs, we could only partially implement the requested behaviour. The command issued by the user through the web browser has been named `post_ip`. The equivalent command issued by the Discovery Board to the SILICA Board is `at+s.sts=ip_ipaddr`. In a bug-free board this command yields the board IP address printed on the serial terminal. Due to the bug, the whole list of status variables is printed instead.

3.3 Point 3: Print IP address on led.html

Regarding this request, we managed to implement this command resorting to the same buggy AT command `at+s.sts`. We leveraged the fact that both the PC (HTerm) and the Discovery board receive whatever the SILICA Board on the UART Tx line sends. In particular, the full list of variables produced by the above command is transmitted and stored in a buffer (an array of char) within the MCU RAM. After performing some parsing on the string, it is possible to extract the IP address and build the new HTML page. The post-processing consists in exploiting `string.h` functions available in C, namely `strtok`, `strstr` and `strcat`. First, the string is split into tokens by `strtok` using as delimiter “=”. Using a while-loop, we check whether each token has `ip_ipaddr` as part of it (done by the `strstr` function). Once the token (token 1 in Figure 2) is found, due to the format of the string, we have to advance to token 2, hence we call again `strtok`. At this point `strtok` is called again, but using # as delimiter. In order to have the final IP address, another final call to `strtok` needs to be done by using as delimiter `\n`. Doing so, we finally obtain the IP address.

To build the final HTML page, we used `strcat` for merging the IP address with the strings already stored in the RAM containing the full HTML page.

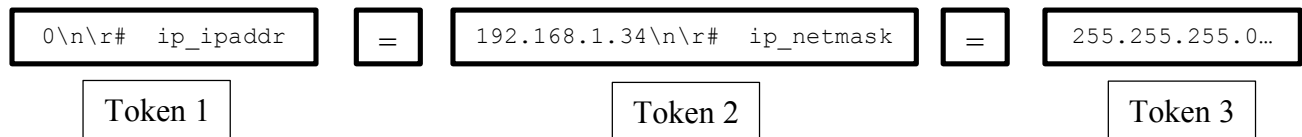


Figure 2 - Format of the transmitted list. Between each token, the delimiter is highlighted.

4. Conclusions

In this lab session we learned how to easily deploy an IoT solution. This is of course a starting base for a more complex system. Additions from previous labs can be easily integrated in the program, especially the low power state management.

In a “real world” application the board, along with sensors and the Wifi module, would probably be in a remote location and running solely on a battery so resorting to low power states is crucial to provide a longer lasting service.