Table of Contents

1. AT commands controlled WiFi module 2

1. AT commands controlled WiFi module

The STM32WB55 microcontroller does not have the WiFi module built-in. There isn't also any additional WiFi-enabling element on the Nucleo board. The laboratory stand is equipped with a module based on ESP32-C3 SoC - super mini. It works as a WiFi modem connected by a UART port controlled with AT commands. It is connected to the hardware serial port lines RxD and TxD available at pins PC_0 and PC_1 respectively. To use it the HardwaSerial class object must be created.

```
// Serial port configuration (Constructor uses STM port numbering)
HardwareSerial WiFiSerial(PC_0, PC_1, NC, NC);
```



HardwareSerial class object constructor uses STM-like pin numbering, not Arduino numbering. Pins for RxD and TxD lines are specified. Because no hardware flow control is used the lines RTS and CTS remain unconnected.

In the setup function, the serial port should be opened and the bitrate specified.

```
// WiFiSerial is a hardware port for ESP32-C3 WiFi modem
WiFiSerial.begin(115200);
```



ESP32-C3 working as a WiFi modem uses 115200 bps and 8N1 frame format (8 bits of data, no parity control, one stop bit).

The WiFi modem can connect to the wireless network by using AT commands. The list of all supported commands is available on the Espessif website [1)]. It is important to finish every command with both CR and LF characters (13 and 10 decimal). Fortunately the Arduino function println() sends proper characters automaticaly. If the command is sent properly the modem responds according to the command and finishes the response with “OK”. For example, sending the command “AT” should result in an “OK” response.

```
AT

OK
```

If the command is wrong, contains errors, or is not allowed in the actual mode of the modem's operation the response is “ERROR”. A possible situation is that the modem has the WiFi or MQTT configuration stored in the EEPROM memory, and establishes connections automatically after the power-on. If a user tries to send the configuration data while connection is established the modem's response is “ERROR”. The execution time of commands can be quite long, up to a few seconds, especially if commands rely on other devices, eg. establishing the connection to the WiFi Access Point or receiving data from a remote server. It is always a good way to wait for “OK” (and possible “ERROR”) before taking further action. An example of the code that sends an “AT” command and waits for “OK” is shown below.

```
  String response;  //String object to store the response from modem
  String compOK;    //String object to store the "OK" text for comparison

  compOK = "OK";
  WiFiSerial.println("AT");
  do {
    response = WiFiSerial.readStringUntil(0x0A);
  } while (!(response.startsWith(compOK)));
```

The sequence of commands that establishes the connection to the WiFi Access Point contains two of them.

```
AT+CWMODE=1
AT+CWJAP="SSID","password"
```

It is possible to list all available APs by using the command:

```
AT+CWLAP
```

After connecting to a wireless network we can connect to the MQTT broker. ESP32-C3 supports MQTT protocol with some specialized commands. In all MQTT commands, the first parameter is the ID of the connection, it should be 0. Configuring the user data can be done with the following command:

```
AT+MQTTUSERCFG=0,1,"MQTT_node_name","MQTT_user","MQTT_password",0,0,""
```

MQTT_node_name should be the unique name of the device. MQTT_user and MQTT_password are the credentials to connect to the broker if required. Assuming that the broker is available at IP address 192.168.1.100 and uses the default TCP port the connection with the broker can be established with the command:

```
AT+MQTTCONN=0,"192.168.1.100",1883,1
```

After a successful connection to the MQTT broker, we can subscribe to the chosen topic or publish messages.

```
AT+MQTTSUB=0,"MQTT_topic",0
AT+MQTTPUB=0,"MQTT_topic","message",0,0
```

<note> The third parameter of the subscribe command as well as the fourth parameter of the publish command is the QoS. The fifth parameter of the publish command is the retain flag. Please refer to the MQTT lesson for the details.

[1)] https://docs.espressif.com/projects/esp-at/en/latest/esp32c3/AT_Command_Set/Basic_AT_Commands.html

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s2:aih:lab:stm32mqtt?rev=1705498561

Last update: **2024/01/17 14:36**