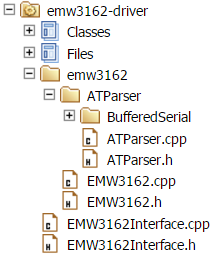
**Porting guide: Adding WIFI module (EMW3162) support to NSAPI**

The network-socket API (NSAPI: <https://docs.mbed.com/docs/mbed-os-api-reference/en/5.2/APIs/communication/network_sockets/>) provides a common interface for using sockets on network devices. It’s a class-based interface, which should be familiar to users experienced with other socket APIs. Here’s the porting guide that provides you an example to add WIFI module (EMW3162) support to NSAPI.

**1. Framework Reference**

Here we use the framework inheriting from the ESP8266 driver (https://github.com/armmbed/esp8266-driver), both of which support the AT command via UART to implement the communication between module and board, shown as below:



* BufferSerial dir
  + Software Buffer, mainly extends mbed Serial functionallity adding irq driven TX and RX.
* ATParser.cpp / ATParser.h
  + Parser for the AT command syntax
* EMW3162.cpp / EMW3162.h
  + EMW3162Interface class, provide an interface to a EMW3162 radio.
* EMW3162Interface.cpp / EMW3162Interface.h
  + EMW3162Interface class, implementation of the NetworkStack for the EMW3162

Usually, when add a new WIFI module using the above framework, we just need rewrite the EMW3162 interface file (including EMW3162.cpp / EMW3162.h and EMW3162Interface.cpp / EMW3162Interface.h) and might modify the AT parser file (including ATParser.cpp / ATParser.h) slightly according to the format difference of AT command RX & TX in different WIFI modules.

**2. API Implementation**

Here we just port the functions on client side.

2.1. EMW3162Interface.cpp / EMW3162Interface.h

In these two files, we need port the APIs both in WIFI Connection and Socket Implementation (including TCP and UDP connection).

WIFI Connection APIs

* EMW3162Interface(PinName tx, PinName rx, bool debug = false): define the EMW3162 Interface class.
* int connect( const char \*ssid, const char \*pass, nsapi\_security\_t security = API\_SECURITY\_NONE ): start the interface, attempts to connect to a WiFi network.
* int disconnect(): Stop the interface.
* const char \*get\_ip\_address():get the internally stored IP address.
* const char \*get\_mac\_address():get the internally stored MAC address.

Socket Implementation APIs

* int socket\_open(void \*\*handle, nsapi\_protocol\_t proto): open a socket.
* int socket\_close(void \*handle): close a socket.
* int socket\_connect(void \*handle, const SocketAddress &address): connect the TCP socket to a server with the specified socket address.
* int socket\_send(void \*handle, const void \*data, unsigned size): send data to the remote host when in TCP connection.
* int socket\_recv(void \*handle, void \*data, unsigned size): receive data from the remote host when in TCP connection.
* int socket\_sendto(void \*handle, const SocketAddress &address, const void \*data, unsigned size): send data to a remote host with the specified address when in UDP connection.
* int socket\_recvfrom(void \*handle, SocketAddress \*address, void \*buffer, unsigned size): receive data from a remote host with the specified address when in UDP connection.
* void socket\_attach(void \*handle, void (\*callback)(void \*), void \*data): register a callback on state change of the socket.

2.2. EMW3162.cpp / EMW3162.h

These two files mainly implement all the detail functions for the above APIs using the parser interface from ATParser module. It is the most key part in the porting process as details implementation of different WIFI module are quite different, thus nearly all the code need be rewrite.

2.3. ATParser.cpp / ATParser.h

In EMW 3162, format of data to be sent is a bit different from ESP8266 (<CR> need be added to the end in EMW 3162 while ESP 8266 needn’t). We add the following code to “ATParser::vsend” function after all the data have been sent out.

```

// Finish with <CRs>

char s[] = "\x0d";

for (int i = 0; s[i]; i++){

if(putc(s[i]) < 0)

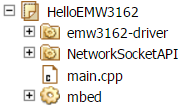
printf("send <CR> error\n");

}

```

**3. Target Setting / Module Using**

When using the emw3162 module, the project (HelloEMW3162) must include the emw3162-driver library, NetworkSocketAPI library, mbed library and main.cpp:



In main.cpp, we need add the following code to define the WIFI interface:

```

#include "EMW3162Interface.h"

EMW3162Interface wifi(D1, D0); // D1 / D0 here refers to the UART TX / RX interface on the board

```

Thus the EMW3162 driver should work!