After the basic unit tests described in the study doc now my objective is to develop a neowayGsmClient like class which will have all necessary functions related in order to develop a client like class

Now lets make a rough list of the functions that we need

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sim800.h** |  |  |  |  |
| Sim800() |  | STATE\_NONE |  |  |
| bool init(SoftwareSerial \*modemPort, byte onOffPin); |  |  |  |  |
| void switchOn(); |  | STATE\_ON |  |  |
| void switchOff(); |  | STATE\_OFF |  |  |
| void switchModem(); |  |  |  |  |
| bool init\_onceautobaud(); | AT | STATE\_INITIALIZED | AT |  |
| bool init\_autobaud(); |  | STATE\_NONE |  |  |
| bool init\_fixbaud(); | AT  AT+IPR (set baudrate)  AT&F (set current parameter to factory configuration)  AT&W (store current current configuration) | STATE\_INITIALIZED | AT  AT+IPR=0  AT&W | OK  OK  OK |
| bool setup(); | AT&F  ATE0 (turn off echo)  AT+CREG? (network registration status) |  | ATE0  AT+CREG? | OK  +CREG: 0,1 OK |
| bool startNetwork(const char \*apn, const char \*user, const char \*pwd );st | AT+CIPMUX=0 (start single IP connection)  AT+SAPBR (bearer settings [APN, user , pass ])  AT+SAPBR | STATE\_REGISTERED | AT+CGDCONT=1,”IP”,”gpinternet” | OK |
| bool stopNetwork(); | AT+SAPBR=0 (close bearer) |  |  |  |
| bool checkNetwork(); | AT+SAPBR=2 (query bearer) |  |  |  |
| bool GetMyIP(char\*ip); | AT+CIFSR (get local IP address) |  | AT+XIIC? |  |
| bool getIMEI(char \*imei); | AT+GSN (serial identification number) |  | AT+CGSN | +CGSN: "860218037564108" OK |
| bool getSignalQualityReport(int\*rssi,int\*ber); | AT+CSQ (signal quality report) |  | AT+CSQ | +CSQ: 31,0 OK |
| bool httpGET(const char\* server, int port, const char\* path, char\* result, int resultlength); | AT+HTTP | STATE\_HTTPINITIALIZED |  |  |
| bool isOn(); |  |  |  |  |
| bool isInitialized(); |  |  |  |  |
| bool isRegistered(); |  |  |  |  |
| bool isHttpInitialized(); |  |  |  |  |
| void send(const char \*buf); |  |  |  |  |
| void cleanInput(); |  |  |  |  |
| byte receive(char \*buf); |  |  |  |  |
| byte receive(char \*buf, uint16\_t timeout); |  |  |  |  |
| bool receive(char \*buf, uint16\_t timeout, char const \*checkok, char const \*checkerror); |  |  |  |  |
| bool receivelen(char \*buf, uint16\_t timeout, unsigned int datalen); |  |  |  |  |
| bool ATcommand(const char \*command, char \*buf); |  |  |  |  |
| bool ATcommand(const char \*command, char \*buf, char const \*checkok, char const \*checkerror, unsigned long timeout); |  |  |  |  |
| time\_t RTCget(); |  |  |  |  |
| uint8\_t RTCread(tmElements\_t &tm); | AT+CCLK (read clock) |  | AT+CCLK? | +CCLK: "12/06/01,10:11:29" OK |
| uint8\_t RTCset(time\_t t); |  |  |  |  |
| uint8\_t RTCwrite(tmElements\_t &tm) | AT+CCLK (read Clock) |  | AT+CCLK= |  |
| bool TCPstart(const char \*apn, const char \*user, const char \*pwd ); | AT+CGATT? (attach /detach from gprs)  AT+CIPMODE (select TCPIP application Mode)  AT+CIPCCFG (configure transparent mode)  AT+CSTT (set APN user name password)  AT+CIICR (bring up wireless connection) | STATE\_REGISTERED | AT+CPIN?  AT+CCID  AT+CSQ  AT+CREG?  AT+XISP=0  AT+CGATT?  AT+XIIC=1  AT+XIIC?  AT+TCPTRANS="mqtt.opensensors.io",1883 | +CPIN: READY OK [means no password]  +CCID: 8988010908804282473F OK  +CSQ: 31,0 OK  +CREG: 0,1 OK [means registered to local network]  OK [use internal protocol stack]  +CGATT: 1 OK [ 1 means GPRS is attached]  OK [the module is required to setup a PPP connection]  +XIIC: 1, 10.110.73.174 OK [PPP connection is set up successfully]  OK +TCPTRANS:OK |
| bool TCPconnect(const char\* server, int port); | AT+CIPSTART start TCP connection | STATE\_HTTPINITIALIZED |  |  |
| bool TCPGetMyIP(char\*ip); | AT+CIFSR get Local IP Address |  | AT+XIIC? |  |
| bool TCPstop(); | AT+CIPSHUT deactivate gprs  AT+CGATT? Gprs query  AT+CGATT=0 detach gprs |  | AT+CGATT?  AT+CGATT=0 | GPRS DISCONNECTION OK |
| **Sim800Client.h** |  | |  |  |
|  |
|  |
| Transparentescape() | +++ (switch from data mode to command mode) |  | +++ | OK [means you can send at commands now] |
| Transparent() | ATO (switch from command mode to data mode) |  | ATO | CONNECT [means you have now switched back to data mode] |
| Stop() | AT+CIPCLOSE=0 (close TCP connection) |  | AT+ AT+IPSTATUS=0 [querying TCP connection status]  TCPCLOSE=0 | +IPSTATUS:0,CONNECT,TCP,2047 [A TCP connection has been setup on socket 0 & the buffer size is 2047 bytes]  +TCPCLOSE: 0,OK |
| connect() |  |  |  |  |
| connected() |  |  |  |  |
| available() | Calls serial.avaialble() |  |  |  |
| read() | Calls serial.read() |  |  |  |
| readBytes() |  |  |  |  |
| setTimeout() |  |  |  |  |
| Write() | Calls serial.write() |  |  |  |
| Flush() |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Green function are called by mqtt class

Blue functions are called by the green functions

Pink functions are called by main sketch file

Yellow Functions are called by pink functions

Silver function are called by yellow functions