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** | **neowayM590.h** | **Purpose of the neoway function** |
| Sim800() | neowayM590() | State = STATE\_NONE |
| bool init(SoftwareSerial \*modemPort, byte onOffPin); | Init() | Check AT, power up , check AT, power off, power up, check AT return  If returning true state = STATE\_INITIALIZED |
| void switchOn(); | powerUp() | Power up the module state = STATE\_ON |
| void switchOff(); | powerOff() | Power down the module state = STATE\_OFF |
| void switchModem(); | sendAtCommand() | Func to check whether an AT command is executed |
|  | findSubString() | Check whether sub string is in the string |
| bool init\_onceautobaud();  bool init\_autobaud();  bool init\_fixbaud(); | **Auto feature** |  |
| bool setup();  bool startNetwork(const char \*apn, const char \*user, const char \*pwd );  bool stopNetwork();  bool checkNetwork();  bool GetMyIP(char\*ip); // ip no less 15 char + terminator  bool getIMEI(char \*imei); // imei no less 15 char + terminator  bool getSignalQualityReport(int\*rssi,int\*ber);  bool httpGET(const char\* server, int port, const char\* path, char\* result, int resultlength);  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);  bool TCPstart(const char \*apn, const char \*user, const char \*pwd );  bool TCPconnect(const char\* server, int port);  bool TCPGetMyIP(char\*ip);  bool TCPstop();  uint8\_t RTCset(time\_t t);  uint8\_t RTCwrite(tmElements\_t &tm); |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Sim800() |  |  |  |
| bool init(SoftwareSerial \*modemPort, byte onOffPin); |  |  |  |
| void switchOn(); |  | STATE\_ON |  |
| void switchOff(); |  |  |  |
| void switchModem(); |  | STATE\_INITIALIZED |  |
| bool init\_onceautobaud(); | AT |  | 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 |
| bool setup(); | AT&F  ATE0 (turn off echo)  AT+CREG? (network registration status) |  | ATE0  AT+CREG? |
| bool startNetwork(const char \*apn, const char \*user, const char \*pwd ); | AT+CIPMUX=0 (start single IP connection)  AT+SAPBR (bearer settings [APN, user , pass ])  AT+SAPBR | STATE\_REGISTERED | AT+CGDCONT=1,”IP”,”gpinternet” |
| 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 |
| bool getSignalQualityReport(int\*rssi,int\*ber); | AT+CSQ (signal quality report) |  | AT+CSQ |
| 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? |
| 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=ip,port |
| 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 |
|  |  |  |  |
| Transparentescape() | +++ (switch from data mode to command mode) |  |  |
| Transparent() | ATO (switch from command mode to data mode) |  |  |
| stop | AT+CIPCLOSE=0 (close TCP connection) |  | AT+IPSTATUS=0  AT+TCPCLOSE=0 |