Activité : Communication par modem GPRS/GSM

Thierry Vaira <tvaira@free.fr>

01/02/2016 (rev. 1)

Table des matières

Communication par modem GPRS/GSM

Notions de base

Le GPRS (*General Packet Radio Service*) est une norme (protocole réseau) pour la téléphonie mobile dérivée du GSM et complémentaire de celui-ci, permettant un débit de données plus élevé.

Le GPRS est une extension du protocole GSM : il ajoute par rapport à ce dernier la transmission par paquets. Cette méthode est plus adaptée à la transmission des données. En effet, les ressources ne sont allouées que lorsque des données sont échangées, contrairement au mode « circuit » en GSM où un circuit est établi pour toute la durée de la communication. Le GPRS a ensuite évolué au début des années 2000 vers la norme Edge également optimisée pour transférer des données et qui utilise les mêmes antennes et les mêmes fréquences radio.

Le GPRS permet de fournir une connectivité IP constamment disponible à une station mobile (MS), mais les ressources radio sont allouées uniquement quand des données doivent être transférées, ce qui permet une économie de la ressource radio. Les utilisateurs ont donc un accès bon marché, et les opérateurs économisent la ressource radio. De plus, aucun délai de numérotation n'est nécessaire.

Terminaux GSM/GPRS

Ce moyen de transmission nécessite un terminal disposant d'un modem GSM/GPRS ou 3G/UMTS, ainsi que d'une carte SIM de n'importe quel opérateur avec un forfait "données" (Data) adapté. Le terminal nécessite d'être sous couverture GSM/GPRS pour pouvoir envoyer les données vers la plateforme de traitement. Ce type de terminal est utilisé lorsque l'objet ou la personne à géolocaliser reste dans une zone bien couverte par les réseaux GSM/GPRS.

Remarque : Les forfaits GSM/GPRS sont économiquement plus avantageux que les forfaits satellite lorsque l'on souhaite remonter les positions à une fréquence élevée. Ils sont donc à privilégier si les zones où l'équipement se déplace restent bien couvertes par les réseaux GSM/GPRS.

Commandes AT

La firme Hayes, fabricant de modems, a développé un protocole pour la commande d'un modem externe à partir d'un ordinateur. Le protocole définit diverses commandes permettant par exemple :

– de composer un numéro de téléphone
– de commander le raccordement du modem à la ligne (l'équivalent de décrocher le téléphone)

– de connaître l'état de la ligne : tonalité d'invitation à transmettre, ligne occupée ...
– de spécifier le type de transmission et le protocole de liaison à utiliser
– de régler le volume sonore du haut-parleur interne du modem
– d'envoyer les caractères transmis simultanément vers l'écran
– d'afficher certains renseignements concernant le modem
– de manipuler les registres internes du modem

Les commandes AT (ou Commandes Hayes) sont des commandes que l'on peut directement envoyer au modem, lorsque celui-ci est en mode Command.

Chaque commande est envoyée sous la forme d'une ligne de texte encodée en ASCII, terminée par le caractère '\r' seul (code ASCII 0x13). Le modem retourne une réponse sous la forme d'une ou plusieurs lignes selon la commande envoyée, chaque ligne se terminant par les caractères '\r' suivi de '\n' (codes ASCII 0x13 et 0x10).

Terminologie

GSM GSM (*Global System for Mobile Communications*) est une norme numérique de seconde génération pour la téléphonie mobile. Elle a été spécifiée et mise au point par l'ETSI (*European Telecommunications Standard Institut*) pour la gamme de fréquences des 900 MHz. Cette norme est particulièrement utilisée en Europe, en Afrique, au Moyen-Orient et en Asie. Tel qu'il a été conçu, le réseau GSM est idéal pour les communications de type « voix » (téléphonie). Le réseau étant commuté, les ressources ne sont allouées que pour la durée de la conversation, comme lors de l'utilisation de lignes téléphoniques fixes. Les clients peuvent soit acheter une carte prépayée, soit souscrire un abonnement.

GPRS GPRS (*General Packet Radio Service*) est une norme (protocole réseau) pour la téléphonie mobile dérivée du GSM et complémentaire de celui-ci, permettant un débit de données plus élevé. On le qualifie souvent de 2,5G ou 2G+. Le G est l'abréviation de génération et le 2,5 indique que c'est une technologie à mi-chemin entre le GSM (deuxième génération) et l'UMTS (troisième génération).

EDGE EDGE (*Enhanced Data Rates for GSM Evolution*) est une norme de téléphonie mobile, une évolution du GPRS qui est elle-même une extension du GSM avec rétrocompatibilité. Le débit maximal descendant a été fixé à 384 kbit/s par l'UIT (Union Internationale des Télécommunication) dans le but de respecter la norme IMT-2000. EDGE est quatre fois plus efficace que le GPRS.

UMTS UMTS (*Universal Mobile Telecommunications System*) est l'une des technologies de téléphonie mobile de troisième génération (3G). Elle est basée sur la technologie W-CDMA, standardisée par le 3GPP et constitue l'implémentation dominante, d'origine européenne, des spécifications IMT-2000 de l'UIT pour les systèmes radio cellulaires 3G. L'UMTS est parfois appelé 3GSM, soulignant la filiation qui a été assurée entre l'UMTS et le standard GSM auquel il succède. Elle est également appelée 3G, pour troisième génération.

W-CDMA Le W-CDMA (*Wideband Code Division Multiple Access*, « multiplexage par code à large bande ») est une technique de codage utilisée dans la partie radio (UTRAN) des réseaux de téléphonie mobile UMTS, de troisième génération.

HSPA HSPA (*High Speed Packet Access*), aussi appelé 3G+ dans sa dénomination commerciale, est la combinaison de deux protocoles utilisés en téléphonie mobile pour améliorer les performances obtenues avec la 3G UMTS : le *High Speed Downlink Packet Access* (HSDPA) et le *High-Speed Uplink Packet Access* (HSUPA). Ils permettent d'atteindre des débits théoriques maximum de 14,4 Mbit/s en descente et 5,8 Mbit/s en montée selon la mise en service de ces deux normes par les opérateurs et la compatibilité du terminal utilisé.

Carte SIM Les téléphones mobiles contiennent une carte SIM qui permet d'identifier l'utilisateur et parfois de stocker un certain nombre de numéros de téléphone. Chaque appareil est identifié, quelle que soit sa marque, par un numéro IMEI que l'on obtient, en entrant sur le clavier, la séquence : *#06#. Il convient de noter ce numéro et de le signaler à son opérateur, en cas de vol, de façon à procéder à son blocage. Cet identifiant ne doit pas être confondu avec l'IMSI contenu dans la carte SIM.

Code PIN Le code PIN est le mot de passe de la carte SIM ; le code PUK permet de débloquer une carte SIM, bloquée après l'introduction de 3 codes PIN erronés. Le code PIN2, s'il existe, est un mot de passe pour un sous-ensemble de fonctions précises de la carte SIM ; le code PUK2 lui est associé, de la même manière.

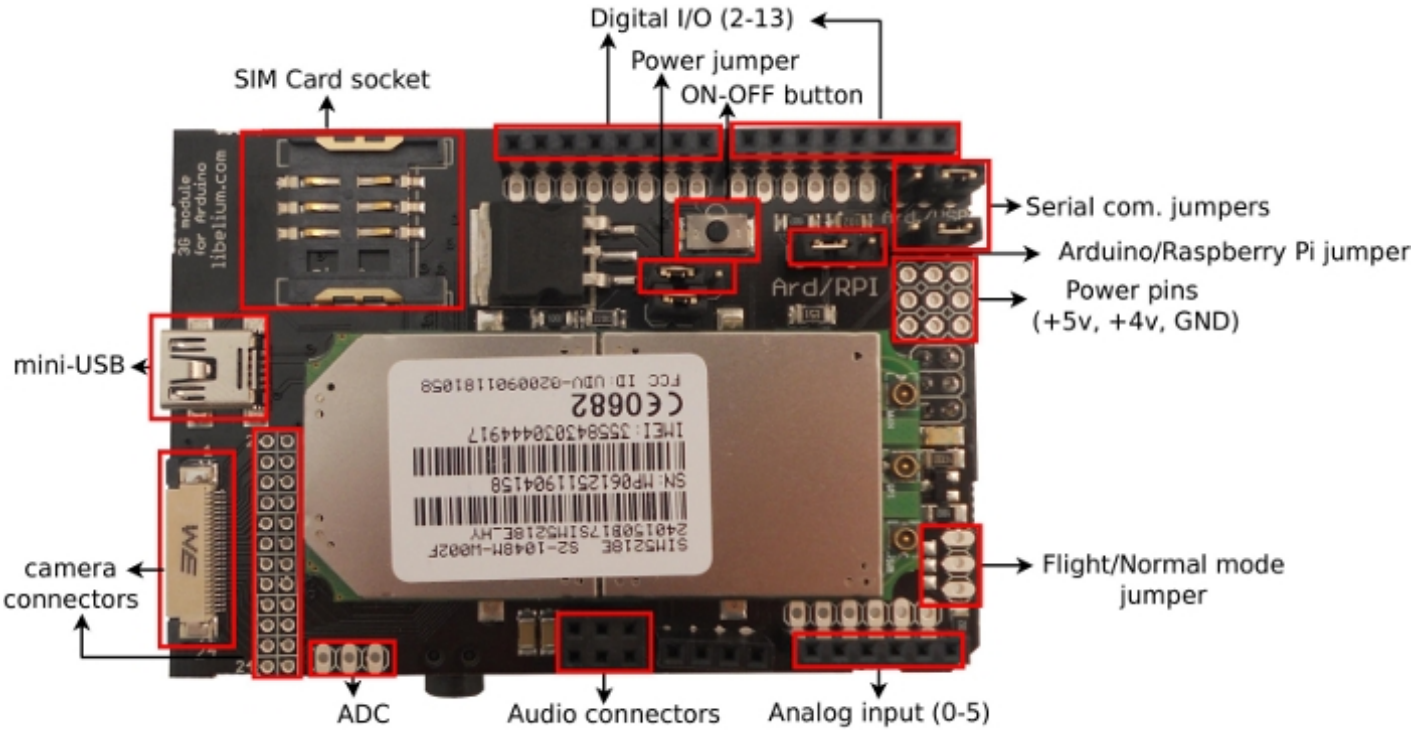
Code TMSI Sur un réseau cellulaire, un appareil est identifié via un code TMSI (*Temporary Mobile Station Identifier*) dérivé du code IMSI. Grâce à ce système de IMSI/TMSI, un téléphone portable ne voit pas son numéro d'appel divulgué sur le réseau, ce qui permet la confidentialité des appels : comme les TMSI changent souvent et sont alternativement attribués à plusieurs appareils, une personne interceptant le trafic a très peu de chance d'associer un numéro de téléphone à un TMSI.

Modem À lorigine, Le modem (mot-valise, pour modulateur-démodulateur), est un périphérique servant à communiquer avec des utilisateurs distants par l'intermédiaire d'un réseau analogique (comme une ligne téléphonique). Il permet par exemple de se connecter à Internet. Depuis la fin des années 1990, de nombreuses normes de télécommunications sont apparues et, donc autant de nouveaux types de modems : RNIS (ou ISDN), ADSL, GSM, GPRS, Wi-Fi, Wimax ...

SIM5218

La série SIM5218 est un module GSM / GPRS / EDGE quadri-bande supportant les protocoles HSPA et la technologie WCDMA. Il autorise un débit descendant (*download*) de 7.2Mbps et un débit montant (*upload*) de 5,76Mbps.

Lien : www.cooking-hacks.com



Le module SIM5218 sera détecté comme un port série virtuel [PDF] lorsqu'il est relié par une liaison USB [PDF]. On le programmera à partir de commandes AT.

Lire : SIM5218_AT_command_manual.pdf

Objectifs

Être capable de communiquer avec un modem GPRS/GSM.

Séquence 1 : commandes AT

On peut tester les commandes AT avec la commande `screen` (`Ctrl-a k` pour sortir), `picocom` (`Ctrl-a Ctrl-x` pour sortir) ou `cutecom`.

Séquence 2 : mise en oeuvre de la classe ModemGPRS

La déclaration de la classe `ModemGPRS` :

```
#ifndef MODEMGPRS_H
#define MODEMGPRS_H

#include "qextserialenumerator.h"
#include "qextserialport.h"
#include <QStringList>
#include <QMap>
#include <QTime>
#include <unistd.h>

#define PORT_DEFAUT "/dev/ttyUSB2"

//#define DEBUG_MODEMGPRS

class ModemGPRS
{
public:
    ModemGPRS();
    ModemGPRS(QString peripherique);
    ~ModemGPRS();

    QString     authentifier(const QString &pin);
    bool        demarrerGPS();
    bool        arreterGPS();

    QStringList getGPS();
    QString     getInformations();
    QString     getNomFabricant();
    QString     getNomModel();
    QString     getNumeroSerie();
    QString     getJeuDeCaracteres();
    QString     getIMSI();
    QString     getCapacitesGlobales();
    QString     getIMEI();
    QString     getRevision();
    QString     getOperateur();
    QString     getListeOperateurs();
    QString     getNumero();
    QString     getNumero(int index);
    QString     getListeNumeros();
    QString     getQualiteSignal();
    QString     getNomFournisseur();
    QStringList getListeAppelsEmis();
    QStringList getListeAppelsManques();
    QStringList getListeAppelsRecus();
    QStringList getListeRepertoire();
    bool        setModeTexteSMS();
    bool        setStockageSMS(const QString &mem1);
```



```
    bool        setStockageSMS(const QString &mem1, const QString &mem2, const QString &mem3);
    int         getNbSMS();
    int         getNbSMS(const QString &mem1);
    QString     getSMS(int index);
    QString     getSMS(int index, const QString &mem1);
    QStringList getListeSMS();
    QStringList getListeSMS(const QString &mem1);
    QStringList getListeSMSNonLus();
    QStringList getListeSMSNonLus(const QString &mem1);
    bool        envoyerSMS(const QString &numero, const QString &message);
    bool        envoyerEmail(const QString &adresse, const QString &nom, const QString &sujet, const QString &
    message);
    QString     getURL(const QString &url, const QString &requete, unsigned int timeout=20000);
    void        testerSocketTCP(const QString &serveur, const int port, const QString &message);

    bool        envoyerCommandeAT(const QString &message, const QString &reponseAttendue, unsigned int timeout
    =100);
    int         transmettre(const QString &message);
    QString     recevoir(unsigned int timeout=100);

private:
    QextSerialPort     *port;
    QString             peripherique;
    bool                gps;
    bool                modeTexte;
    QMap<QString, int>  stockageSMS;
    QMap<QString, int>  totalSMS;

    bool    ouvrir();
    void    fermer();
};

#endif // MODEMGPRS_H
```

La définition partielle de la classe `ModemGPRS` :

```
#include "modemgprs.h"

#include <QtCore/QDebug>

ModemGPRS::ModemGPRS() : port(NULL), peripherique(""), gps(false), modeTexte(false)
{
    // mode asynchrone : QextSerialPort::Polling
    port = new QextSerialPort(QLatin1String(PORT_DEFAUT), QextSerialPort::Polling);
    ouvrir();
}

ModemGPRS::ModemGPRS(QString peripherique) : port(NULL), peripherique(peripherique), gps(false), modeTexte(false
    )
{
    port = new QextSerialPort(peripherique, QextSerialPort::Polling);
    ouvrir();
}

ModemGPRS::~ModemGPRS()
{
    fermer();
}

bool ModemGPRS::ouvrir()
{
    port->setBaudRate(BAUD115200);
    port->setParity(PAR_NONE);
    port->setDataBits(DATA_8);
    port->setStopBits(STOP_1);
    port->setTimeout(100);
```



```
    port->open(QIODevice::ReadWrite | QIODevice::Unbuffered);
    #ifdef DEBUG_MODEMGPRS
    qDebug("%s ouvert : %d", peripherique.toAscii().data(), port->isOpen());
    #endif

    if (!port->isOpen()) return false;
    return true;
}

void ModemGPRS::fermer()
{
    if (port->isOpen())
        port->close();
    #ifdef DEBUG_MODEMGPRS
    qDebug("%s ferme : %d", peripherique.toAscii().data(), !port->isOpen());
    #endif
}

QString ModemGPRS::authentifier(const QString &pin)
{
    QString reponse;
    QString statut;

    // déjà autjentifié ?
    if(transmettre("AT+CPIN?"))
    {
        reponse = recevoir();
        if(reponse.length() == 0)
            return "";
        if(!reponse.contains("OK"))
            return "";
        if(!reponse.startsWith("\r\n+CPIN:"))
            return "";
        reponse.remove(0, 9);
        while(reponse[0] != '\r')
        {
            statut.append(reponse[0]);
            reponse.remove(0, 1);
        }
        #ifdef DEBUG_MODEMGPRS
        if(statut.contains("READY"))
            qDebug() << "READY : ME is not pending for any password";
        else if(statut.contains("SIM PIN"))
            qDebug() << "SIM PIN : ME is waiting SIM PIN to be given";
        else
            qDebug() << "Statut : " << statut;
        /*
        SIM PUK : ME is waiting SIM PUK to be given
        PH-SIM PIN : ME is waiting phone-to-SIM card password to be given
        SIM PIN2 : ME is waiting SIM PIN2 to be given
        SIM PUK2 : ME is waiting SIM PUK2 to be given
        PH-NET PIN : ME is waiting network personalization password to be give
        */
        #endif
    }
    if(!statut.contains("READY"))
    {
        if(transmettre("AT+CPIN=\""+pin+"\""))
        {
            reponse = recevoir();
            if(reponse.length() == 0)
                return "";
            if(!reponse.contains("OK"))
                return "";
            usleep(5000*1000);
```



```
            return reponse;
        }
    }
    return "";
}

QString ModemGPRS::getURL(const QString &url, const QString &requete, unsigned int timeout/*=20000*/)
{
    QString commandeAT;
    bool res;

    commandeAT = "AT+CREG?";
    res = envoyerCommandeAT(commandeAT, "+CREG: 0,5", 500);
    if(res == false)
        return "";

    // sets APN, user name and password
    /*
     * APN (Access Point Name) est un identifiant qui permet à un utilisateur de téléphonie mobile d'un réseau 2
    G ou 3G
     * de se connecter à Internet en identifiant le Gateway GPRS Support Node (GGSN) qu'il veut utiliser.
     * L'APN est généralement constitué d'un code identifiant le GGSN (et le réseau IP derrière lui) et
     * des codes MCC et MNC identifiant l'opérateur de réseau mobile.
     */
    res = envoyerCommandeAT("AT+CGSOCKCONT?", "+CGSOCKCONT: 1,\"IP\",\"Free\"", 200);
    if(res == false)
    {
        // sets APN, user name and password
        res = envoyerCommandeAT("AT+CGSOCKCONT=1,\"IP\",\"Free\"", "OK", 2000);
        if(res == false)
            return "";
        res = envoyerCommandeAT("AT+CSOCKAUTH=1,1,\"\",\"\"", "OK", 2000);
        if(res == false)
            return "";
    }

    res = envoyerCommandeAT("AT+CHTTPACT=\""+url+"\",80", "+CHTTPACT: REQUEST", 10000);
    if(res == true)
    {
        qDebug() << "envoie requete : " << "GET /"+requete+" HTTP/1.1\r\nHost: "+url+"\r\nContent-Length: 0\r\n"
    ;
        // Ctrl-Z -> 0x1A
        if(transmettre("GET /"+requete+" HTTP/1.1\r\nHost: "+url+"\r\nContent-Length: 0\r\n\r\n\x1a"))
        {
            QString reponse = recevoir(timeout);
            qDebug() << "reponse : " << reponse;
            if(reponse.length() == 0)
                return "";
            if(!reponse.contains("+CHTTPACT: DATA"))
                return "";
            reponse.remove(0, 2); // enleve \r\n au début
            return reponse;
        }
    }

    return "";
}

/*
 * client TCP
 *
 * pour les tests, il faut lancer un serveur avec netcat :
 * $ nc.traditional -l -p 64465
 *
 */
void ModemGPRS::testerSocketTCP(const QString &serveur, const int port, const QString &message)
```



```
{
    QString commandeAT;
    bool res;

    commandeAT = "AT+CREG?";
    res = envoyerCommandeAT(commandeAT, "+CREG: 0,5", 500);
    if(res == false)
        return;

    // sets APN, user name and password
    /*
     * APN (Access Point Name) est un identifiant qui permet à un utilisateur de téléphonie mobile d'un réseau 2
    G ou 3G
     * de se connecter à Internet en identifiant le Gateway GPRS Support Node (GGSN) qu'il veut utiliser.
     * L'APN est généralement constitué d'un code identifiant le GGSN (et le réseau IP derrière lui) et
     * des codes MCC et MNC identifiant l'opérateur de réseau mobile.
     */
    res = envoyerCommandeAT("AT+CGSOCKCONT?", "+CGSOCKCONT: 1,\"IP\",\"Free\"", 200);
    if(res == false)
    {
        // sets APN, user name and password
        res = envoyerCommandeAT("AT+CGSOCKCONT=1,\"IP\",\"Free\"", "OK", 2000);
        if(res == false)
            return;
        res = envoyerCommandeAT("AT+CSOCKAUTH=1,1,\"\",\"\"", "OK", 2000);
        if(res == false)
            return;
    }

    commandeAT = "AT+NETOPEN=\"TCP\","+QString::number(port);
    res = envoyerCommandeAT(commandeAT, "Network opened", 5000);
    if(res == false)
    {
        res = envoyerCommandeAT("AT+NETCLOSE", "OK", 5000);
        return;
    }

    commandeAT = "AT+TCPCONNECT=\""+serveur+"\","+QString::number(port);
    res = envoyerCommandeAT(commandeAT, "Connect ok", 5000);
    if(res == false)
    {
        res = envoyerCommandeAT("AT+NETCLOSE", "OK", 5000);
        return;
    }

    commandeAT = "AT+TCPWRITE="+QString::number(message.length());
    res = envoyerCommandeAT(commandeAT, ">", 100);
    if(res == false)
    {
        res = envoyerCommandeAT("AT+NETCLOSE", "OK", 5000);
        return;
    }

    res = envoyerCommandeAT(message, "Send ok", 10000);
    if(res == false)
    {
        res = envoyerCommandeAT("AT+NETCLOSE", "OK", 5000);
        return;
    }

    res = envoyerCommandeAT("AT+NETCLOSE", "OK", 5000);
}

bool ModemGPRS::envoyerCommandeAT(const QString &message, const QString &reponseAttendue, unsigned int timeout/*
    =100*/)
{
```



```
    if (!port->isOpen()) return false;

    if(transmettre(message))
    {
        //usleep(timeout*1000);
        QString reponse = recevoir(timeout);
        if(reponse.length() == 0)
            return false;
        if(reponse.indexOf(reponseAttendue) == -1)
            return false;
    }
    else return false;

    return true;
}

int ModemGPRS::transmettre(const QString &message)
{
    QString delimiteur = "\r\n";

    if (!port->isOpen()) return 0;

    int i = port->write(message.toLatin1() + delimiteur.toLatin1());
    #ifdef DEBUG_MODEMGPRS
    qDebug("-> send (%d) : %s", i, message.toAscii().data());
    #endif

    return i;
}

QString ModemGPRS::recevoir(unsigned int timeout/*=100*/)
{
    QString message = "";
    char buffer[1024];
    int  nbOctets = 0;
    QTime t;
    int debut = 0;

    t.start();

    if (!port->isOpen()) return "";

    if(timeout != 0)
    {
        debut = t.elapsed();
        while((nbOctets == 0) && (t.elapsed() < (int)(debut + timeout)))
        {
            nbOctets = port->bytesAvailable();
            if(nbOctets > 1024)
                nbOctets = 1024;
            if(nbOctets > 0)
            {
                int i = port->read(buffer, nbOctets);
                if (i != -1)
                {
                    if (buffer[i-1] == '\n' && buffer[i-2] == '\r')
                        buffer[i-2] = '\0';
                    else
                        buffer[i] = '\0';
                }
                else
                    buffer[0] = '\0';
                message += QLatin1String(buffer);

                #ifdef DEBUG_MODEMGPRS
                qDebug("-> receive (%d/%d) : %d ms (%d ms)", i, nbOctets, t.elapsed(), (debut + timeout));
```



```
                #endif

                nbOctets = 0;
            }
        }
        #ifdef DEBUG_MODEMGPRS
        qDebug("-> receive (%d) : %s", message.length(), message.toAscii().data());
        //qDebug("en %d ms", t.elapsed());
        #endif

        return message;
    }
    else
    {
        while(nbOctets == 0)
        {
            nbOctets = port->bytesAvailable();
            if(nbOctets > 1024)
                nbOctets = 1024;
        }
    }

    int i = port->read(buffer, nbOctets);
    if (i != -1)
    {
        if (buffer[i-1] == '\n' && buffer[i-2] == '\r')
            buffer[i-2] = '\0';
        else
            buffer[i] = '\0';
    }
    else
        buffer[0] = '\0';
    message = QLatin1String(buffer);

    #ifdef DEBUG_MODEMGPRS
    qDebug("-> receive (%d/%d) : %s", i, nbOctets, message.toAscii().data());
    qDebug("en %d ms", t.elapsed());
    #endif

    return message;
}
```

Un programme de test de la classe `ModemGPRS` :

```
#include "modemgprs.h"

#include <QStringList>
#include <QtCore/QDebug>

void afficherDetails(ModemGPRS *modemGPRS);

int main()
{
    ModemGPRS *modemGPRS = new ModemGPRS("/dev/ttyUSB2");
    QString commandeAT;
    QString resultat;
    bool res;

    // Test
    commandeAT = "AT";
    res = modemGPRS->envoyerCommandeAT(commandeAT, "OK");
    if(res == false)
    {
        qDebug() << "Pas de reponse du modem !";
        delete modemGPRS;
        return 0;
```

```
    }

    modemGPRS->authentifier(QString("1234"));

    afficherDetails(modemGPRS);

    // ...

    qDebug() << "Test client socket TCP";
    /*
        -> send (24) : AT+NETOPEN="TCP",64465
        -> receive (20) :
        Network opened
        OK
        -> send (37) : AT+TCPCONNECT="92.153.12.130",64465
        -> receive (16) :
        Connect ok
        OK
        -> send (16) : AT+TCPWRITE=13
        -> receive (3) :
        >
        -> send (15) : hello world !
        -> receive (32) :
        OK
        +TCPWRITE: 13, 13
        Send ok
        -> send (13) : AT+NETCLOSE
        Network closed
        OK
    */
    modemGPRS->testerSocketTCP("92.153.12.130", 64465, "hello world !");

    delete modemGPRS;

    return 0;
}

void afficherDetails(ModemGPRS *modemGPRS)
{
    QString resultat;

    //resultat = modemGPRS->getInformations();
    //qDebug() << resultat;

    resultat = modemGPRS->getNomFabricant();
    qDebug() << "Fabricant : " << resultat;
    resultat = modemGPRS->getNomModel();
    qDebug() << "Model : " << resultat;
    resultat = modemGPRS->getNumeroSerie();
    qDebug() << "Numero de serie : " << resultat;
    resultat = modemGPRS->getNomFournisseur();
    qDebug() << "Fournisseur : " << resultat;
    resultat = modemGPRS->getJeuDeCaracteres();
    if(resultat.contains("IRA"))
        qDebug() << "Jeu de caracteres : IRA (International Reference Alphabet)";
    else if(resultat.contains("GSM"))
        qDebug() << "Jeu de caracteres : GSM  (GSM default alphabet); this setting causes easily software flow
    control (XON/XOFF) problems";
    else if(resultat.contains("UCS2"))
        qDebug() << "Jeu de caracteres : UCS2 (16-bit Universal multiple-octet coded Character Set); UCS2
    character strings are converted to hexadecimal numbers from 0000 to FFFF";
    else
        qDebug() << "Jeu de caracteres : " << resultat;
    resultat = modemGPRS->getIMSI();
    qDebug() << "IMSI : " << resultat;
    resultat = modemGPRS->getCapacitesGlobales();
```



```
    qDebug() << "Capacites : " << resultat;
    QStringList liste = resultat.split(",");
    for (int i = 0; i < liste.size(); ++i)
    {
         if(liste.at(i).contains("CGSM"))
             qDebug() << "CGSM : GSM function is supported";
         else if(liste.at(i).contains("FCLASS"))
             qDebug() << "FCLASS : FAX function is supported";
         else if(liste.at(i).contains("DS"))
             qDebug() << "DS : Data compression is supported";
         else if(liste.at(i).contains("ES"))
             qDebug() << "ES : Synchronous data mode is supported";
    }
    resultat = modemGPRS->getIMEI();
    qDebug() << "IMEI : " << resultat;
    resultat = modemGPRS->getRevision();
    qDebug() << "Revision : " << resultat;
}
```

Code source : test-mo-sim5218.zip

Retour au sommaire

