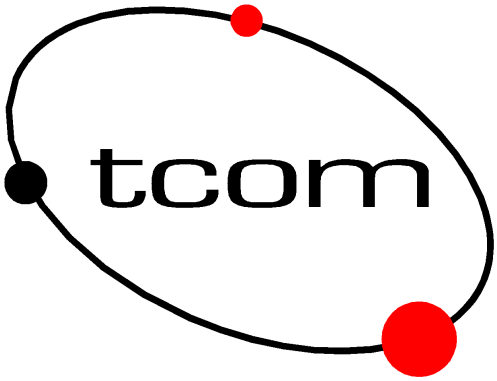


TRAVAIL DE DIPLÔME 2001

Transmission de la localisation GPS par GPRS

Candidat	Pierre Daccord
Professeur	Stephan Robert
Expert	Gianpaolo Cecchin
	Swisscom

1 Avant propos

Initialement, le but de ce travail de diplôme était de fournir un outil de mesure de la qualité de service du réseau GPRS. Ce projet a été abandonné pour des raisons techniques. Nous ne savions pas comment calculer la qualité de service.

1.1 But

Le but de ce projet est de fournir un système pouvant être embarqué de manière autonome dans un véhicule. Ce module fournira, au besoin, sa position géographique en utilisant le mode toujours connecté de GPRS. L'appareil sera activé par un SMS. Un roaming devra être prévu, en cas d'absence de GPRS.

Ce projet se compose de quatre parties :

- Une application chargée d'attendre un SMS d'activation-configuration, de collecter des données en provenance d'un GPS et de les envoyer à un serveur contenant une base de données par une connexion TCP à l'aide d'un modem GPRS.
- Un serveur capable de réceptionner les données en tout temps, de les traiter et de les insérer dans une base de données. Le serveur doit pouvoir traiter plusieurs clients simultanément.
- Une base de données pouvant contenir tous les points envoyés par les véhicules, ainsi que des informations concernant les propriétaires et leur identification.
- Un site Internet graphique pour que les utilisateurs du système puissent consulter en toute simplicité la position de leur véhicule sur une carte. Le système doit être transparent par rapport à la méthode de localisation du véhicule. Le site doit aussi pouvoir permettre l'identification de l'utilisateur et de son véhicule avant de transmettre les information concernant la localisation. La sécurité doit être garantie, un utilisateur doit pouvoir consulter la position que de sa voiture.

Une cinquième partie pourrait être la miniaturisation du système de manière à pouvoir le faire tenir dans un petit boîtier électronique, probablement à microcontrôleur. Ceci pourrait représenter le produit fini mais n'est pas le but de ce travail de diplôme.

1.2 Remerciements

Les remerciements vont à MM Gianpaolo Cecchin et Stephan Robert, respectivement initiateur du projet et professeur responsable, à M.Bron, responsable du parc informatique pour sa disponibilité ainsi qu'à toute la classe ETR7 pour son soutien et sa collaboration.

1.3 Planning

Dans un premier temps, il faut créer une application minimale pouvant être embarquée dans une automobile ainsi qu'un petit serveur pouvant collecter quelques données et les mettre dans une base de données. Une fois que cette "application minimale" sera en fonction, il faudra la perfectionner et créer un site Internet fonctionnel et accueillant.

La mise en fonction de "l'application minimale", devrait vraisemblablement prendre environ un mois et demi, le reste du temps sera consacré au perfectionnement, au site WEB ainsi qu'au rapport.

Nous disposons déjà d'un lecteur de données envoyées par le GPS (NMEA 183) trouvé sur Internet lors du travail de semestre. Il s'agit donc en premier lieu de comprendre son fonctionnement, de supprimer de ce programme tout ce qui est superflu et d'intégrer les classes restantes dans l'application que nous voulons mettre dans le véhicule.

Dans un second temps, il faut pouvoir ouvrir et fermer des connexion GPRS, transmettre des données. Ensuite, il faudra s'occuper de recevoir, de lire et de comprendre des SMS pour pouvoir tenir compte de leur contenu pour configurer l'envoi des positions.

Enfin, il restera à définir et créer la base de données. Une fois la base de données fonctionnelle, il faudra que le serveur puisse la remplir. Les points géographiques entrant correctement dans la base de données, "l'application minimale" pourra donc être considérée comme réalisée.

1.4 Schéma de principe

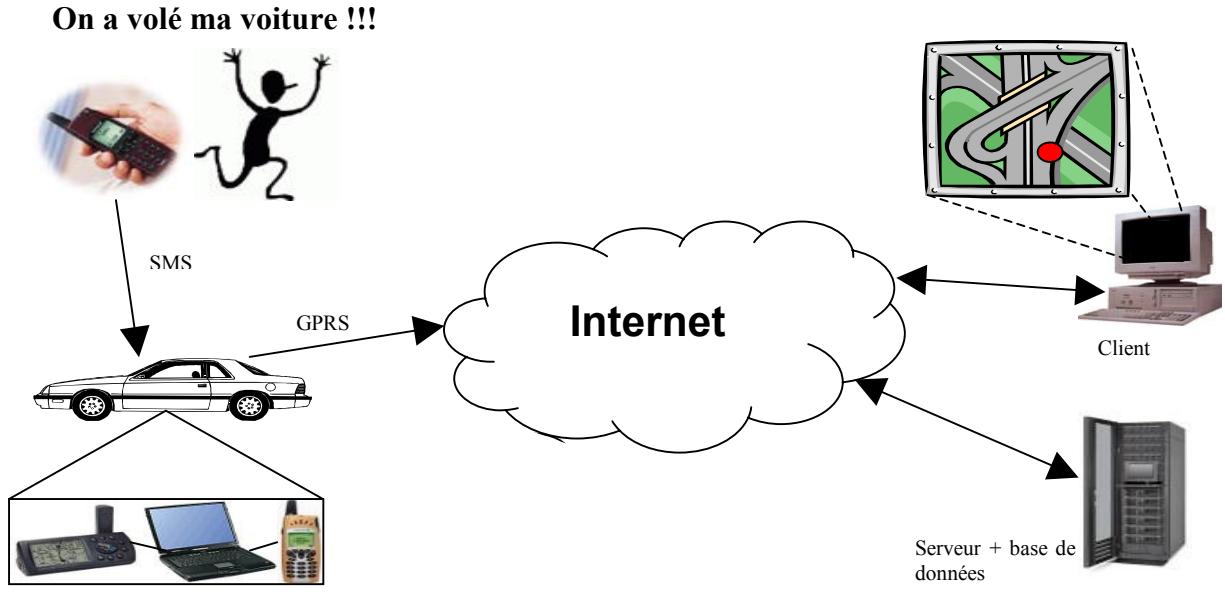


Figure 01: schéma de principe

1.5 Le matériel

Pour que notre système puisse fonctionner, il faut disposer du matériel suivant :

- Client
  - Un GPS avec câble série
  - Un téléphone GPRS avec câble série
  - Un adaptateur USB-série
  - Un PC portable avec machine virtuelle Java (jdk1.3)

- Serveur
  - Un PC ne se trouvant pas derrière un firewall ou avec les droits sur le firewall avec
    - Mysql
    - Apache
    - Tomcat
    - Machine virtuelle Java (jdk1.3)

Chacun de ces points est plus détaillé dans la suite de ce rapport.

Nous disposons du matériel suivant:

- Client
  - Un PC portable Dell 1GHz ram 256
  - Un GPS garmin II avec câble série
  - Un téléphone Ericsson R520m ou T39m avec câble série
  - Un adaptateur USB-série Distrelec

- Serveur
  - Un PC P166 ram100

2 Partie théorique

2.1 Le système GPS

Figure 02 : satellites GPS

GPS est l'abréviation pour "Global Positioning System", un système de positionnement et de navigation par satellites à l'échelle mondiale. Mis en oeuvre par le département de la défense des Etats Unis d'Amérique, qui reste le seul responsable de sa précision et de sa maintenance. Actuellement 24 satellites tournent autour de la terre à une altitude de 20'000 km environ et envoient des signaux que le récepteur GPS utilise pour calculer sa position. La plupart des récepteurs GPS actuels peuvent recevoir simultanément les signaux de 8 à 12 satellites et choisir les meilleurs pour construire le point représentant la position actuelle de l'utilisateur. Pour calculer une position en longitude et latitude (2D) 3 satellites sont nécessaires et pour une position en 3D comprenant l'altitude il en faut 4.

Auparavant la précision était volontairement d'environs 100 mètres, mais par une décision du président Clinton, depuis le début du mois de mai 2000, la précision mise à disposition de chacun est de 5-20 mètres. Ce paramètre peut être réglé par le département de la défense américain en introduisant une dérive de l'horloge. En période de guerre, l'Amérique diminue la qualité de la précision, de manière à ce que les adversaires des USA ne puissent pas utiliser correctement leur GPS. Ceci n'affecte que les GPS civils.

L'altitude est environ 1.5 fois moins précise que la longitude et la latitude, et change vers le haut et le bas. Mais elle n'a aucune influence sur la précision horizontale.

La précision peut être augmentée en utilisant le système DGPS (Differential GPS) ce système nécessite un élément supplémentaire, une balise connaissant sa position et qui simule un satellite supplémentaire. Comme cet émetteur se trouve généralement à une distance faible comparée aux 20'000 km de l'orbite des satellites, il est possible de calculer plus précisément la position puisqu'il faut moins de temps pour transmettre le signal. Dans les voitures équipées de GPS avec cartographie, le système DGPS est aussi utilisé, il n'y a cette fois pas de d'émetteur placé par l'utilisateur le temps d'une mesure. Les informations DGPS sont émises dans la bande RDS des chaînes de radio locales, de même que l'heure, le nom de la station ou d'autres publicités. Ceci permet au système de positionner la voiture sur la bonne route pour pouvoir donner des informations cohérentes au conducteur.

2.2 Le GPRS

GPRS (General Packet Radio Service) est employé comme un service de transmission de données, il s'agit d'une mise à niveau de n'importe quel réseau GSM. Il permet aux réseaux GSM d'être vraiment compatible avec l'Internet. GPRS emploie une technique de transfert en mode paquet pour transférer le trafic de données de façon efficace. Il permet des taux de transmission de 9.6 kbps à plus de 150 kbps par utilisateur.
Les deux clés bénéfiques de GPRS sont une meilleure utilisation des ressources radio et du réseau et le support complètement transparent de IP. Il emploie des ressources radio seulement quand il y a des données à envoyer ou à recevoir. Comme une vraie technologie en paquets il permet aux applications des utilisateurs finaux d'occuper le réseau seulement lorsque des données utiles (payload) sont transférées. Une autre particularité importante de GPRS est qu'il fournit la connectivité immédiate.

Le taux de transfert de données le plus rapide avec GPRS est, en théorie, 171.2 kbit/s. GPRS emploie un maximum de 8 intervalles de temps (timeslot), chacun avec une vitesse maximale de 21.4 kbit/s. Cette vitesse pourrait ne jamais être atteinte et certainement pas au début.

C'est en raison des facteurs suivants :

- les limitations des téléphones. Les premiers téléphones GPRS seront équipés d'un maximum de 4 intervalles de temps en downlink
- partage de la largeur de bande avec GSM
- partage de la largeur de bande avec d'autres utilisateurs GPRS
- dépendance de la force du signal

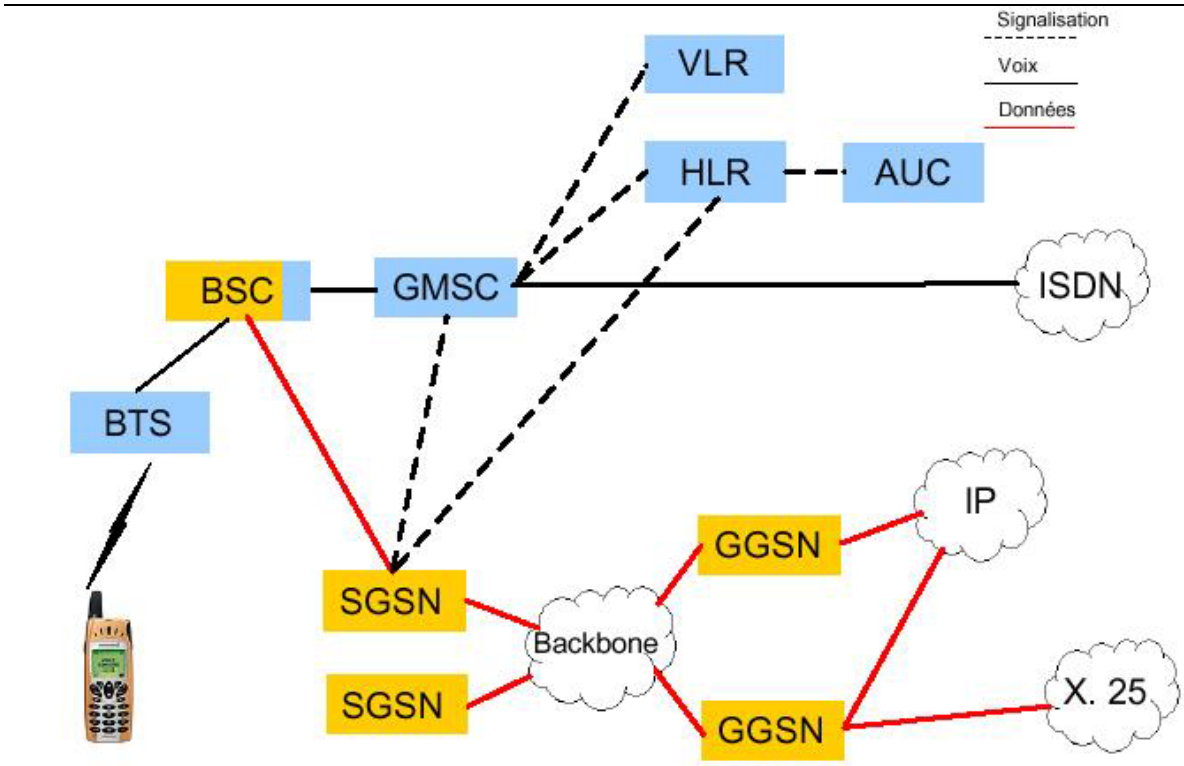
En pratique, les vitesses entre 18 et 50 kbit/s sont réalistes (avec 4 DL et CS1 ou CS2). C'est entre 2 et 6 fois plus rapide que le 9.6 kbit/s de GSM, selon le nombre d'utilisateurs connectés simultanément sur la même antenne.

CS signifie Coding Scheme. C'est le moyen par lequel les renseignements entrants seront décodés (il y a beaucoup d'overhead (redondance) en ayant une mauvaise connexion et peu quand la connexion est bonne.)

CS1 : égale la qualité de la norme GSM, 9.05 kbit/s
CS2 : 13.4 kbit/s
CS3 : 15.6 kbit/s
CS4 : la qualité la plus haute, 21.4 kbit/s (si par exemple, l'utilisateur est directement à côté d'une antenne et qu'il y a peu d'autres utilisateurs à ce moment là)

Une raison de ne pas utiliser des vitesses plus grandes que celles de CS1 et CS2, est que les taux de transfert plus rapides exigent trop des batteries actuelles.

L'architecture d'un réseau GPRS est la même que celle d'un réseau GSM, seuls quelques éléments doivent être modifiés ou ajoutés.



MS : Mobile Station
BTS : Base Transceiver Station
BSC : Base Station Controller
MSC : Mobile Switch Center
GMSC : Gateway Mobile Switch Center
GGSN : Gateway GPRS Serving Node

HLR : Home Location Register
VLR : Visited Location Register
EIR : Equipment Identity Register
AUC : Authentification Location Register
SGSN : Serving GPRS Service Node

Figure 03: architecture d'un réseau GPRS

Pour intégrer GPRS dans une architecture GSM existante, il faut ajouter de nouveaux noeuds réseaux, appelés nœuds de support GPRS *(GPRS support node*s, GSN). Les GSNs sont responsables du routage et de la remise des paquets de données entre les stations mobiles (MS) et les réseaux externes de paquets de données *(packet data network*s, PDN). La séparation de la voix et des données se fait au niveau du BSC.

Un nœud de support GPRS de service *(serving GPRS support nod*e, SGSN) est responsable de la remise des paquets de données depuis et vers les stations mobiles à l'intérieur de sa zone de service. Dans ses tâches sont inclus le routage et le transfert des paquets ainsi que le management de toutes les données relatives aux utilisateurs GPRS se trouvant dans sa zone de responsabilité. Il assure donc la fonction VLR pour les services data. Ceci implique que ce dernier n'a pas besoin d'être modifié.

Un nœud de support GPRS passerelle *(gateway GPRS support nod*e, GGSN) se comporte comme une interface entre le réseau backbone GPRS et les autres réseaux à commutation de paquets. Il convertit les paquets GPRS provenant du SGSN dans le format du protocole approprié (par exemple, IP ou X.25) et les envoie vers le réseau correspondant. En général, il existe une relation du type plusieurs à plusieurs entre les SGSNs et les GGSNs. Un SGSN peut router ses paquets vers différents GGSNs pour atteindre différents réseaux à commutation de paquets.

2.3 Le Langage de programmation

Pour ce projet, il y a deux parties de programmation, le client et le serveur. Il faut donc choisir un langage pour réaliser ces deux applications.

En résumé, ce que le client doit faire, c'est gérer les ports série et communiquer avec, ainsi qu'établir une connexion TCP pour transmettre les données. Ce que le serveur doit faire, c'est accepter une connexion d'un client, se connecter sur la base de données et insérer les données reçue par la liaison TCP dans la base.

Ce que nous recherchons aussi c'est la portabilité. Dans ce cas, Java est un bon langage, même si il est un peu lent et lourd pour la machine sur laquelle s'exécute le programme.

Java offre un paquetage appelé "javax.comm" pour tout ce qui est des interactions avec les ports série. Les protocoles TCP et UDP sont déjà implémentés et les accès aux bases de données sont simples.

Une autre raison qui pousse à l'utilisation du langage Java est la possession de codes sources d'un programme qui interagit avec un GPS. Il serait bien de réutiliser ces classes.

2.4 Le serveur

Pour la partie serveur de l'application, il faut une machine qui puisse être atteinte depuis l'extérieur de l'école. Ceci pose quelques problèmes en raison du firewall externe qui assure la sécurité de l'école. Nous devons donc installer une machine dédiée avec une adresse IP spécifique à la DMZ (zone démilitarisée) de l'école. Les machines se trouvant dans cette zone virtuelle située entre deux firewall bénéficient d'une sécurité plus faible, ainsi il est possible de les atteindre depuis le monde entier. C'est dans cette zone que se trouvent des serveurs tels que www.eivd.ch et www.tcom.ch. Pour disposer d'une adresse DMZ, il faut en faire la demande et avoir un peu de patience. De plus, nous devons annoncer ce que nous comptons mettre sur le serveur pour que les ports correspondants soient laissés ouverts sur les firewalls. Notre serveur s'appelle "gprs.eivd.ch" et a l'adresse IP fixe "193.134.216.180".

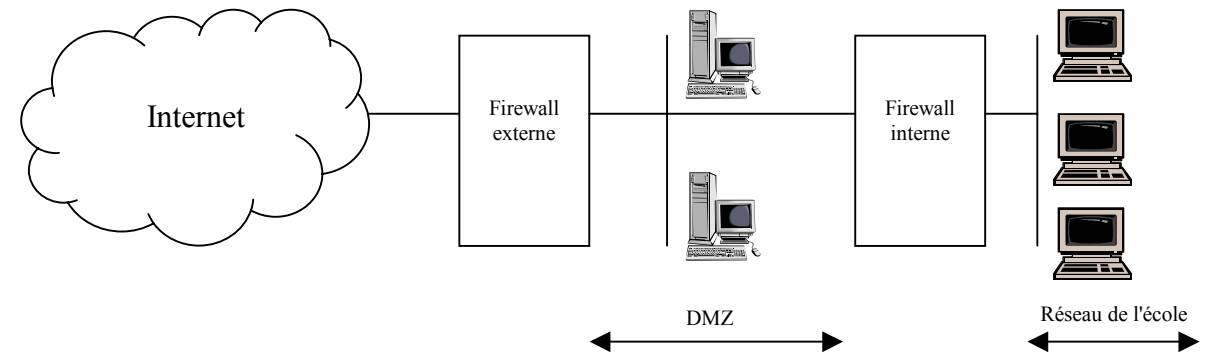


Figure 04: position de la DMZ

Pour l'installation d'un serveur, il y a encore des choix à faire, tels que celui du système d'exploitation, du serveur Web, du système de base de données, du serveur FTP etc…

2.5 Détail des éléments

Ce projet peut être découpé en 4 parties :

- Le client
- Le serveur
- La base de données
- Le site Internet de consultation

2.5.1 Le client

Ce que nous appelons client, c'est l'élément qui pourra être multiplié dans le cas d'une utilisation réelle. Cette partie est destinée à fonctionner de manière autonome. Nous voulons ensuite pouvoir la miniaturiser pour, par exemple, la mettre en option dans des véhicules neuf, au même titre que d'autres équipements tels qu'une alarme.

Le client doit :

- Attendre un SMS de configuration, le lire et interpréter le contenu
- Se connecter au serveur par une liaison mobile GPRS
- Transmettre ses données pendant un temps défini dans le SMS de configuration
- Se déconnecter du réseau GPRS
- Se remettre en état d'attente d'un SMS pour une utilisation ultérieure
- Pouvoir envoyer la position du véhicule au propriétaire par SMS si le réseau GPRS n'est pas accessible ou pas utilisable. C'est une sorte de roaming.

2.5.2 Le serveur

Ce que nous appelons le serveur est la partie qui sert uniquement à collecter les données et à remplir la table de la base de données qui contient les positions des véhicules.

Le serveur doit:

- Attendre la connexion d'un ou plusieurs clients
- Se connecter à la base de données
- Recevoir les données, les traiter et les insérer dans la base de données
- Se déconnecter de la base de données
- Attendre la connexion de nouveaux clients

2.5.3 La base de données

La base de données doit pouvoir être remplie et consultée simultanément, les clients qui envoient leur positions ne doivent pas être bloqués par quelqu'un qui serait entrain de consulter la position de son véhicule sur le site Internet.

La base de données doit contenir:

- Les données concernant les propriétaires de véhicules
- Les données concernant les véhicules
- Les positions des véhicules

2.5.4 Le site Internet de consultation

Le site Internet de consultation est la partie visible du projet, tout le reste est caché. L'utilisateur final ne doit qu'envoyer un SMS à sa voiture et aller consulter la position de son véhicule sur Internet. Ce site doit aussi permettre d'administrer le système, par là nous entendons qu'il faut bien pouvoir inscrire de nouveaux utilisateurs du système et leur voiture. C'est également ici qu'est gérée la sécurité du système, un utilisateur ne doit pouvoir consulter que la position de son ou ses véhicules.

Le site doit permettre :

- L'identification de manière sûre du propriétaire
- La restitution de la position du véhicule
- Contenir une partie administrateur avec un formulaire pour inscrire les nouveaux utilisateurs

2.6 Technologie

2.6.1 Les données NMEA

Sur la ligne série, le GPS émet des trames NMEA. L'exemple montre la structure d'une trame, envoyée toujours au même format toutes les deux secondes. Il faut remarquer que la trame est très longue et à 4800 Bauds elle occupe la moitié du temps disponible sur le bus. Cela est très pénalisant pour les autres appareils se trouvant sur le même bus NMEA 183. S'il y a de nombreux appareils les collisions peuvent bloquer le bus. Mais nous n'avons pas ce problème puisque le GPS est seul sur le bus.

```
$GPRMC,130330,A,4643.866,N,00634.317,E,000.0,360.0,281101,000.3,W*62
$GPRMB,A,,,,,,,,,,,V*71
$GPGGA,130330,4643.866,N,00634.317,E,1,08,2.0,450.0,M,48.2,M,,*4C
$GPGSA,A,3,05,,07,09,14,,21,,26,28,,30,4.8,2.0,3.0*3F
$GPGSV,3,1,12,05,73,267,49,06,00,201,00,07,24,047,40,09,76,102,49*72
$GPGSV,3,2,12,14,25,313,41,18,04,237,00,21,11,275,34,24,02,120,00*75
$GPGSV,3,3,12,26,13,169,36,28,08,064,32,29,02,328,00,30,33,251,43*74
$PGRME,15.0,M,22.5,M,15.0,M*1B
$GPGLL,4643.866,N,00634.317,E,130330,A*21
$PGRMZ,1476,f,3*2F
$PGRMM,CH-1903*44
$GPBOD,,T,,M,,*47
$GPRTE,1,1,c,0*07
```

Figure XX : exemple de trame NMEA

De toutes ces trames, ce dont nous avons besoin c'est : la latitude, la longitude, les orientations de la latitude et de la longitude, l'heure la date. Dans cet objectif, une seule de ces trames nous suffit : $GPRMC (Recommended minimum specific GNSS data). En plus des données que nous voulons collecter, cette trame offre un champ qui indique si les données reçues sont valides.

Name	Example	Units	Description
Message ID	$GPRMC		RMC protocol header
UTC Time	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Speed Over Ground	0.13	knots	
Course Over Ground	309.62	degrees	True
Date	120598		ddmmyy
Magnetic Variation[a]		degrees	E=east or W=west
Checksum	*10		
CR LF			End of message termination

Figure 05 : descriptif de $GPRMC

2.6.2 La conversion des coordonnées

Dans le tableau ci-dessus il est possible de voir que les valeurs envoyées par le GPS pour la latitude et la longitude sont au format "dd mm.mmm" (WGS84). Pour notre application, il sera beaucoup plus facile de travailler avec des coordonnés suisse.

Les méthodes modernes de mesure par satellites (GPS) se réfèrent toujours à un des ellipsoïdes globaux (WGS 84), il est nécessaire de définir les relations entre les différents systèmes (transformation de coordonnées). Il s'agit de transformer les coordonnées géographiques sphériques (longitudes et latitudes) en coordonnées nationales planes. Le système suisse est une projection cylindrique conforme à l'axe oblique dont l'origine est l'ancien observatoire de Berne avec les coordonnées nationales Y = 600'000 m et X = 200'000 m.

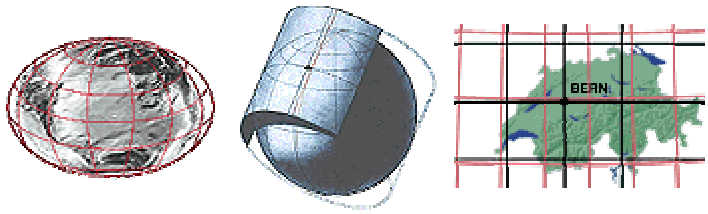


Figure 06: coordonnées suisses

La conversion des latitudes et longitudes en coordonnées suisses peut être faite par un simple calcul. Les coefficients présentés ci dessous dérivent de coordonnées elliptiques et de fonctions de bessel.

Conversion de données géographiques WGS84 -> CH 1903

N	46	46.879	φ	
E	6	39.624	λ	

φ'	(φ -169028.66)/10000	=	-0.061592
λ'	(λ -26782.5)/10000	=	-0.280506

					Résultat intermédiaire	
Y =	600072.37					600072.37
+	211455.93	* λ'			+	-59314.6571
-	10938.51	* λ'	* φ'		-	188.983823
-	0.36	* λ'	* φ' ^{2}		-	-0.00038308
-	44.54	* λ' ^{3}			-	-0.98305242
					Y =	540569.713
X =	200147.07					200147.07
+	308807.95		* φ'		+	-19020.0993
+	3745.25	* λ' ^{2}			+	294.689813
+	76.63		* φ' ^{2}		+	0.29070161
-	194.56	* λ' ^{2}	* φ'		-	-0.94289249
+	119.79		* φ' ^{3}		+	-0.02798939
					X =	181422.866

Figure 07 : calcul de conversion des coordonnées

Ref : www.swisstopo.ch

2.6.3 Les commandes AT

La plupart des modems modernes disposent d'un jeu de commandes textuelles qui peuvent être appelées en mode de commande (lorsque le modem n'est pas connecté à un modem distant ou qu'il a été remis en mode de commandes locales). Le jeu le plus courant de commandes textuelles est appelé le jeu *AT* ou encore *Hayes* , du nom de l'entreprise ayant commercialisé les premiers modems dits intelligents. Le jeu se compose de commandes simples préfixées par `AT`. En général, ces deux codes sont utilisés pour synchroniser le modem à la bonne vitesse. Il est donc recommandé de les taper avec la même casse. Les commandes agissent sur des registres internes qui sont manipulables directement. Il est souvent possible de sauvegarder la configuration dans une mémoire interne non volatile. Il faut savoir que la plupart du temps beaucoup d'extensions incompatibles ont été implantées: il

n'existe pas à proprement parler de norme. Nous parlerons cependant des commandes les plus courantes. En général, les fabricants fournissent avec le modem de la documentation suffisante pour la paramétrisation.

Pour ce projet, nous disposons de téléphones Ericsson, nous devons donc utiliser les commandes AT spécifiques aux appareils Ericsson. Voici quelques exemples.

```
AT                  Attention Command
AT*                 List All Supported AT Commands
ATA                 Answer Incoming Call Command
ATD                 Dial Command
AT+CSQ              Signal Quality

AT+CPMS             Preferred Message Storage
AT+CMGL             List Message
AT+CMGR             Read Message
AT+CMGS             Send Message
AT+CMGD             Delete Message
```

Figure 08: commandes AT GSM+SMS

```
AT+CGDCONT          Define PDP Context
AT+CGQREQ           Quality of Service Profile (Requested)
AT+CGQMIN           Quality of Service Profile (Minimum Acceptable)
AT+CGATT            GPRS Attach or Detach
AT+CGACT            PDP Context Activate or Deactivate
AT+CGDATA           Enter Data State
AT+CGEREP           GPRS Event Reporting
AT+CGREG            GPRS Network Registration
AT+CGPADDR          Show PDP Address
ATD*                Extension of ATD - Request GPRS Service
ATD*                Extension of ATD - Request GPRS IP Service
```

Figure 09 : commandes AT GPRS

2.7 Les SMS

2.7.1 Leur format

Les messages SMS sont spécifiés par l'ETSI (http://www.etsi.org) et peuvent contenir jusqu'à 160 caractères. Chaque caractère est codé sur 7 bits selon le "7-bits default alphabet". Les messages contenant des caractères codés sur 8 bits ne sont habituellement pas lisibles par les téléphones comme des messages de texte.

Il y a deux façons d'envoyer et de recevoir des messages SMS : le mode texte et le mode PDU (protocol description unit). Le mode texte (non disponible sur tous les téléphones) est juste un codage du train binaire représenté par le mode PDU.

La chaîne de caractères de PDU contient non seulement le message, mais également des informations au sujet de l'expéditeur, son centre de messagerie, le timbre horaire etc... Tout est sous forme d'octets hexadécimaux ou de semi-octets décimaux .

```
at+cmgl=0

+CMGL: 8,0,,29
07911497949900F0040B911497969823F70000101192013230000BC8329BFD06A1CB6CF61B
```

Figure 10: exemple de SMS reçu par la ligne série

Octet(s)	Description
07	Longueur de l'information SMSC
91	Type de l'adresse du SMSC
1497949900F0	Numéro du centre de service (en semi-octet décimal). La longueur du numéro de téléphone est 11 alors un F a été ajouté pour former des octets propres. Le numéro de téléphone est "+41794999000"
04	Premier octet du message à délivrer
0B	Longueur de l'adresse de l'expéditeur (hex : 0B = dec : 11)
91	Type de l'adresse de l'expéditeur
1497969823F7	Numéro de téléphone de l'expéditeur, en semi-octets
00	Identificateur de protocole
00	Schéma de codage des données
10119201323000	Timbre horaire
0B	Longueur des données de l'utilisateur (message) (nombre de septets)
C8329BFD06A1CB6CF61B	Message, octets de 8 bits représentant les données 7 bits

Figure 11: Description des champs, SMS reçu

```
at+cmgs=23
> 0011000A817029090381000AA0BC8329BFD06A1CB6CF61B→
+CMGS: 101
```

Figure 12: exemple de SMS envoyé par la ligne série

Octet(s)	Description
00	Longueur de l'informations SMSC
11	Premier octet du message à soumettre
00	Référence du message (0 indique que le téléphone met les références tout seul)
0A	Longueur de l'adresse du destinataire (numéro de téléphone)
81	Type de l'adresse du destinataire
7029090381	Numéro de téléphone du destinataire, en semi-octet
00	Identificateur de protocole
00	Schéma de codage des données
AA	Période de validité (AA = 4 jours)
0B	Longueur des données de l'utilisateur
C8329BFD06A1CB6CF61B	Données de l'utilisateur, codées sur 7 bits

Figure 13: Description des champs, SMS envoyé

2.7.2 Le codage / décodage

Le message codé reçu dans l'exemple est `C8329BFD06A1CB6CF61B` un exemple de décodage s'impose.

C8	32	9B	FD	06	A1	CB	6C	F6	1B
1100 1000	0011 0010	1001 1011	1111 1101	0000 0110	1010 0001	1100 1011	0110 1100	1111 0110	0001 1011

La valeur hexadécimale est simplement convertie en une chaîne de 8 bits.

1001000	1100101	1101100	1101100	1101111	0100000	1101000	1100101	1101100	1101100	1101111
0100 1000	0110 0101	0110 1100	0110 1100	0110 1111	0010 0000	0110 1000	0110 0101	0110 1100	0110 1100	0110 1111
48	65	6C	6C	6F	20	68	65	6C	6C	6F
H	e	l	l	o		h	e	l	l	o

De la chaîne de bits du tableau précédent, il faut prendre les bits sept par sept en incluant les bits restants de l'octet précédent (en gris) et rajouter un zéro comme bit de poids fort. Il ne reste ensuite qu'à recalculer les valeurs hexadécimales qui correspondent à des caractères de la table ASCII. Tous les 8 octets, un octet vient se rajouter puisque le reste de l'octet précédent à une longueur de 7. Le codage des SMS inclut donc aussi une certaine forme de compression.

Pour le codage, les opérations sont inversées mais le principe reste identique.

Ref: http://www.dreamfabric.com/sms/

3 Partie pratique

3.1 Le client

3.1.1 Une application semblable

Lors du projet de semestre et des recherches préalables à la réalisation du projet, nous avions trouvé sur Internet, le travail de MM Rikard & Henrick Bjorkman (http://www.stacken.kth.se/~bjorkman/ChartPlotter/ChartPlotter.html). Ces deux personnes mettent librement à disposition leur travail ainsi que les fichier sources de leurs classes Java. Nous décidons de commencer le projet sur cette base. Leur programme, appelé "ChartPlotter" (traceur de diagramme) réceptionne les données envoyées par un GPS sur la ligne série, en extrait les coordonnées de la position courante et dessine sur une carte du monde l'endroit où se trouve l'utilisateur.

Sur cette base, il est décidé que le client serait programmé en Java. Un argument de poids et le fait que si nous voulons miniaturiser système pour l'embarquer de manière fixe dans un véhicule, il faut que le programme soit portable.

3.1.2 Javax.comm

Nous disposons aussi des classes Java permettant de communiquer et d'interagir avec les ports série d'un PC. Ce paquetage s'appelle "javax.comm" et est disponible, libre de droits sur le site http://java.sun.com.

Ce paquetage de Java sert pour la communication sur les ports série, il n'est pas inclus d'office avec le JDK-1.3. Il faut télécharger "javacomm20-win32.zip" depuis le site de Sun et l'installer, en particulier le fichier "comm.jar". Attention, il faut modifier le classpath en conséquence. Le plus simple est de tout décompresser dans un répertoire et de rajouter un classpath au système pour qu'il puisse chercher les classes dans le répertoire contenant la librairie.

3.1.3 La communication réseau (TCP/UDP)

Pour transférer des données sur un réseau IP, deux protocoles sont utilisables TCP et UDP.

TCP et UDP emploient le même schéma d'adressage. Une adresse IP (nombre de 32 bits, toujours écrit en quatre nombres de 8 bits exprimé en nombres décimaux à 3 chiffres non signés, séparés par des points comme 193.174.25.26) et un numéro de port (un nombre de 16 bits exprimé en nombre décimal non signé).

L'adresse IP est utilisé dans le protocole de bas niveau (IP) pour acheminer le datagramme à l'hôte correct sur le réseau indiqué. Le numéro de port est alors utilisé pour délivrer le datagramme au processus hôte correct (programme sur l'hôte).

Pour un protocole donné (TCP ou UDP), un processus hôte unique peut exister à un moment donné pour recevoir des données envoyées au port défini. D'habitude un port est dédié à un processus.

UDP signifie *Protocole* de *Datagramme Utilisateur*. Il est décrit dans STD-6/RFC-768 et fournit une voie de communication d'hôte à hôte sans connexion. UDP a des faibles entêtes; chaque paquet sur le réseau est composé d'une petite en-tête et de données. Il est appelé datagramme UDP. Ce mode est sans connexion. Cela signifie qu'un datagramme peut être envoyé à tout moment sans avis préalable, négociation ou préparation. Il envoie juste le datagramme en espérant que le récepteur soit capable de l'interpréter. Sinon, le datagramme est perdu.UDP est un protocole incertain. Il n'y a absolument aucune garantie que le datagramme soit livré à l'hôte de destination. Mais à vrai dire, le taux d'échec est très bas sur Internet et quasi nul sur un réseau local, sauf si le réseau est saturé.

Non seulement le datagramme peut ne pas être livré, mais il peut aussi être livré dans un ordre incorrect. Cela signifie que vous pouvez recevoir un paquet avant un autre, même si le deuxième a été envoyé avant le premier que vous avez reçu. Vous pouvez également recevoir deux fois le même paquet. Les principaux avantages d'UDP sont que les limites des datagrammes sont respectées, et que la transmission est rapide.L'inconvénient principal est le manque de fiabilité, ce qui complique le développement du programme.

TCP signifie *Protocole* de *Contrôle* de *Transmission*. Il est décrit dans STD-7/RFC-793. TCP est un protocole orienté connexion qui est garant de la fiabilité de la communication entre deux processus. L'unité de données transférées est appelée un flot, qui est simplement une séquence d'octets. Etre orienté connexion signifie qu'avant de transmettre des données, vous devez ouvrir la connexion entre les deux hôtes. Les données peuvent être transmises en full duplex (envoyer et recevoir sur une seule connexion. Quand le transfert est terminé, vous devez clore la connexion pour libérer les ressources système. De chaque côté on sait quand la session débute et quand elle se termine. La transmission de données ne peut pas avoir lieu tant que les 2 hôtes n'ont pas approuvé la connexion. La connexion peut être fermée par chacun des hôtes; l'autre en est notifié. Il est décidé préalablement si la connexion est terminée correctement ou si elle est seulement interrompue.

Etre orienté flot signifie que les données formes une séquence inconnue d'octets. Il n'y a rien à faire pour avoir des données apparentes. Le récepteur n'a aucun moyen de savoir comment les données ont été réellement transmises. L'expéditeur peut envoyer beaucoup de petits paquets de données et le récepteur seulement recevoir un gros paquet, ou l'expéditeur peut envoyer un gros paquet, le récepteur le recevant en un certain nombre de paquets plus petits. La seule chose que l'on garantit est que toutes les données envoyées seront reçues sans une erreur et dans l'ordre correct. Si une erreur se produit, elle sera automatiquement corrigée (retransmise selon le besoin) ou l'erreur sera notifiée si elle ne peut pas être corrigée.

A la vue de ces arguments, nous choisissons la fiabilité de transmission. Le média que nous allons utiliser est très lent et de plus il n'est pas sûr. Notre application utilisera une connexion TCP.

Ref: http://users.swing.be/francois.piette/fren/support/tcpudp_primer.html

3.2 Methode de travail

Pour réaliser un client comme celui dont nous avons besoin, il faut créer les fonctionnalités une par une et les ajouter aux autres une fois qu'elles sont satisfaisantes.

Pour éditer et écrire les programmes Java, nous allons utiliser l'environnment de dévelopement *Jbuilder 4*.

3.2.1 Un lecteur de NMEA

Pour cette partie, le but était de reprendre le code Java de MM. Bjorkman, mais ce qu'ils avaient fait était beaucoup trop compliqué et surtout gourmand en ressources pour notre application. Il a donc été décidé, après une phase d'analyse, de réécrire un lecteur simple de NMEA depuis le début.

Le lecteur de NMEA est basé sur les événements survenant sur le port série. A chaque fois qu'un événement survient, il est analysé et si il s'agit d'un `DATA_AVAILABLE` les données sont lues et mises dans un tampon. Ensuite, il faut rechercher les positions des chaînes de caractères "GPRMC" et "GPRMB", les données que nous voulons prendre se trouvent entre ces deux index. La chaîne de caractères restante après la découpe est envoyée au serveur sans autre traitement. L'extraction finale pour avoir les positions, la date et l'heure se fait au niveau du serveur.

3.2.2 L'émetteur TCP/LAN

Cette classe comporte trois méthodes principales. Il a été vu plus haut que TCP nécessite l'établissement d'une connexion. Les méthodes principales sont donc `connect`, `send` et `deconnect`. `Connect` est appelé une fois pour la connexion au serveur, `send` est utilisé à chaque fois qu'une trame doit être envoyée et `deconnect` est appelé à la fin de la période de travail pour clore la liaison avec le serveur. Le client restera connecté au serveur tout le temps de travail. Ceci est l'utilisation d'un des avantages de GPRS qui ne compte que les bytes envoyé et non le temps de connexion.

Au début du développement nous ne disposions pas encore du serveur atteignable depuis l'extérieur de l'école. La solution provisoire était donc mettre le serveur sur une machine du réseau local et d'utiliser le réseau local pour connecter le client. Il faut toutefois faire attention au fait que les réactions du réseau ne seront pas les mêmes, surtout en termes de vitesse. Les connexions, déconnexions et transmissions par GPRS seront forcément plus lentes.

A partir de ce point, le développement de notre application implique une interaction entre deux machines, le serveur et le client. Il va de soi qu'ils sont développés de manière parallèle, même si ce rapport semble indiquer qu'ils ont été faits l'un après l'autre.

3.2.3 L'émetteur TCP/GPRS

Nous disposons maintenant du serveur DMZ, il faut modifier l'application déjà existante pour qu'elle envoie les données par GPRS. Pour le moment nous ne maîtrisons pas les connexions et déconnexions automatiques sur le réseau GPRS. Nous pouvons toutefois quand même tester un transfert de données par GPRS. Pour cela, il suffit d'établir la liaison manuellement avant de lancer le programme. Visiblement le changement du média utilisé pour la transmission ne pose pas de grands problèmes. Si ce n'est un ralentissement visible des réactions du serveur.

3.2.4 L'activation par SMS

Ce que nous voulons, c'est pouvoir activer à distance le système de localisation. Pour cela, rien de mieux qu'un SMS. Il aurait aussi été intéressant de pouvoir baser cette fonction sur une gestion des événements. Les téléphones Ericsson dont nous disposons ne transmettent des alarmes sur la ligne série que lors d'un appel vocal. Il est donc impossible de gérer l'arrivée des nouveaux messages seulement en attendant qu'un événement du type `new_message` survienne sur la ligne série. Notre système client doit donc pouvoir aller périodiquement contrôler l'arrivée d'un nouveau message dans le téléphone mobile. Lire ce message et en interpréter le contenu. La lecture des messages par la ligne série et à l'aide de commandes AT décrites plus haut, ne pose pas de problèmes. Le point le plus ennuyeux est que les messages arrivent codés. Il faut donc les décoder pour disposer du contenu de manière lisible. Une fois le message décodé, si il commence par le mot clé "Start" ce message est intéressant. Si il commence par autre chose le message est ignoré et le contrôle périodique de l'arrivée de message reprend.

3.2.5 La connexion automatique sur GPRS

En principe, pour garantir la portabilité, nous voulions établir la connexion sur GPRS avec une suite de commandes AT spécifiques.

```
AT+CGDCONT=1,"IP","www.gprs.ch","",0,0

AT+CGQREQ=1,0,0,0,0,0

AT+CGQMIN=1,0,0,0,0,0

AT+CGATT=1
```

Figure 14: essai de connexion à GPRS

Après des recherches, nous sommes toujours incapables d'établir une connexion au réseau GPRS à l'aide des commandes AT ci-dessus. Une solution de rechange se présente à nous, un appel au système d'exploitation de la machine supportant notre client. Ceci revient a remplacer l'utilisateur qui va cliquer dans les menus pour établir une connexion distante. Cette solution fonctionne correctement mais réduit à néant tous les espoirs de portabilité. Après quelques tests, il s'avère que cette solution de secours supporte tous les Windows, excepté Windows CE. Cette fonctionnalité a été mise dans une classe Java séparée, en cas de portage du système sur une machine Linux, seule cette classe doit être modifiée. La commande du Windows utilisée est `rasdial`.

```
C:\>rasdial /?
Utilisation :
   Rasdial nom_d'entrée [nom_d'utilisateur [mot_de_passe|*]] [/DOMAIN:domaine]
           [/PHONE:numéro_de_tél] [/CALLBACK:numéro_de_rappel]
           [/PHONEBOOK:fichier_annuaire_téléphonique] [/PREFIXSUFFIX]

   rasdial [nom_d'entrée] /DISCONNECT

   rasdial

La commande a été exécutée.

3.2.5.1.1.1  C:\>rasdial gprs_usb gprs gprs /phone:*98*1#
Connexion à GPRS_USB...
Vérification du nom d'utilisateur et du mot de passe...
Enregistrement de votre ordinateur sur le réseau...
Connexion à GPRS_USB réussie.
La commande a été exécutée.

3.2.5.1.1.2  C:\>rasdial gprs_usb /disconnect
La commande a été exécutée.
```

Figure 15: la commande RasDial

En décomposant la commande "`rasdial gprs_usb gprs gprs /phone:*98*1#`" il est possible de voir qu'il faut indiquer le nom de la connexion à utiliser, le nom d'utilisateur, le mot de passe et le numéro à composer pour se connecter.

3.2.6 Le roaming par SMS

Le cahier des charges indique qu'il faut que l'utilisateur de notre système puisse connaître la position de son véhicule même si GPRS est inutilisable ou absent. Cette situation nous fait penser au cas d'un véhicule se trouvant à l'étranger. Le roaming des SMS est en principe garanti alors que celui de GPRS est plus incertain.

Le programme doit donc détecter s'il a pu établir une connexion sur un réseau GPRS afin d'envoyer les données par TCP. Si cette connexion n'a pas pu être établie, il ne faut pas tenter d'utiliser TCP mais envoyer un SMS indiquant la position sur le numéro du propriétaire du véhicule. Dans ce cas, il faut créer un message contenant la position actuelle, le coder de manière à ce que le téléphone recevant notre texte puisse le lire, et envoyer la chaîne d'octets à l'aide de commandes AT décrites plus haut. Attention, le message codé doit être suivi d'un caractère `ctrl-Z` pour valider l'envoi du message.

3.2.7 La ligne de commande

En lançant le programme, il faut pouvoir donner quelques paramètres tels que le numéro de téléphone du propriétaire, le numéro de plaques du véhicule l'adresse du serveur, les ports à utiliser etc… Tous ces paramètres doivent être entrés dans la console, à la suite du nom du programme.

En fait, tant que le système ne supporte pas beaucoup de clients, nous pouvons penser qu'un seul serveur suffit à gérer toutes les connexions. Dans ce cas, seuls les paramètres indiquant le numéro de téléphone du propriétaire et le numéro de plaques du véhicule ne peuvent pas être prévus à l'avance. Dans ce cas, la ligne de commande devient :

```
C:\gps> java gps.Controle

Use : Controle -IV NoPlates -OPN OwnerPhoneNumber [-GP GpsPort] [-MP ModemPort]
        [-SA ServerAdress] [-SP ServerPort] [-TTS TimeToSleep]
        [-TTW TimeToWork] [-TBC TimeBetweenCheck]

-IV NoPlates                    : car plates number (sample: VD111111)
-OPN OwnerPhoneNumber           : owner phone number (sample: 0798889977)
[-GP GpsPort]                   : local GPS port (sample : COM1, COM2 or COM3)
[-MP ModemPort]                 : local modem port (sample : COM1, COM2 or COM3)
[-SA ServerAdress]              : server IP adress (sample : 111.111.222.222)
[-SP ServerPort]                : server remote port (sample : 1024 - 65000)
[-TTS TimeToSleep]              : time to sleep in miliseconds
[-TTW TimeToWork]               : time to work in miliseconds
[-TBC TimeBetweenCheck]         : time between SMS check in miliseconds
```

Figure 16 : ligne de commande de démarrage du client

Tous les autres paramètres ont des valeurs par défaut, ceci évite de devoir taper une ligne de commande trop grande et pouvant contenir des erreurs.

Nom	Identificateur	Valeur par défaut
Port du GPS	-GP	COM1
Temps de sommeil	-TTS	5000 (millisecondes)
Temps de travail	-TTW	20000 (millisecondes)
Adresse IP du serveur	-SA	193.134.216.180
No de port du serveur	-SP	4051
No de plaque du véhicule	-IV	-
Temps entre deux check de SMS	-TBC	20000 (millisecondes)
Port du téléphone	-MP	COM3
No de téléphone du propriétaire	-OPN	-

Figure 17 : paramètres de démarrage de l'application cliente

3.2.8 Liste des classes

Le client est composé d'un paquetage Java appelé `GPS` et comportant 8 classes.

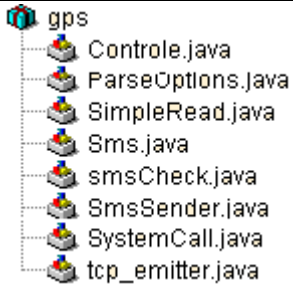


Figure 18: représentation du paquetage GPS

La classe `Controle` est responsable de l'ordonnancement des opérations, c'est elle qui gère le lancement d'une action en fonction du résultat rendu par une activité s'étant exécutée précédemment.

La classe `ParseOption` reçoit la ligne de commande, la découpe pour en extraire les paramètres donnés par l'utilisateur de notre système.

La classe `SimpleRead` est utilisée pour recevoir les données NMEA envoyées par le GPS sur la ligne série.

La classe `Sms` contient les méthodes pour coder et décoder les messages en format PDU, elle est utilisée lors de la réception d'un message pour connaître le contenu en texte clair et lors de l'envoi d'un message pour encoder les informations de manière à ce que le récepteur puisse avoir un texte lisible sur son téléphone portable.

La classe `smsCheck` contrôle périodiquement si un nouveau message est arrivé au téléphone portable, elle est tout le temps active lorsque le système attend l'arrivée d'un message lui indiquant d'envoyer sa position géographique.

La classe `SmsSender` peut envoyer des messages. Elle est utilisée pour envoyer la position au propriétaire de la voiture lorsque le réseau GPRS n'est pas disponible.

La classe `SystemCall` s'occupe des appels au système utilisé pour l'établissement de la connexion au réseau GPRS.

La classe `tcp_emitter` gère les communications TCP avec le serveur, elle établit une connexion TCP avec le serveur au début de la phase de travail, transmet périodiquement les données et déconnecte la partie mobile à la fin de sa période d'activité.

Si les paramètres de configuration contenus dans le SMS diffèrent de ceux donnés par l'utilisateur à l'aide de la ligne de commande. Le système travaille sur la base des derniers arrivés, c'est à dire ceux du SMS.

3.3 Le serveur

Le client étant écrit en Java, il semble naturel que le serveur réceptionnant les trames TCP soit aussi écrit dans le même langage. Les fonctions de notre serveur sont de recevoir les trames TCP, d'en extraire les coordonnées, ainsi que de créer une requête SQL pour insérer les informations dans la base de données.

3.3.1 Choix d'une base de données

Pour choisir une base de données, il faut prendre en compte quelques paramètres tels que la rapidité et la fiabilité du système de gestion de la base, le nombre de connexions admissibles simultanément, le prix, les performances nécessaires de la machine sur laquelle va être installé le système de gestion de bases de données. La portabilité du serveur ne doit pas non plus être étrangère à ce choix.

Dans notre cas, nous voulons une base de données légère, fiable, gratuite et supportant un nombre pas très élevé de connexions simultanées. Le meilleur choix est `MySql`. Les autres bases de données prises en compte sont Access et Oracle. Chacun de ces systèmes de base de données a un inconvénient tel que le prix et les performances demandées à la machine pour Oracle. Une base comme Access de Microsoft est très simple à utiliser mais ne peut pas être portée sur un machine ayant un autre système d'exploitation que Windows. De plus il reste à signaler un problème de liscences.

Dans tout les cas, le programme du serveur doit permettre de changer facilement de base de données. La connexion sur la base de données est réalisée par ces deux lignes :

```
…
// driver loading
Class.forName("org.gjt.mm.mysql.Driver");
…
// database connection establishment
con = DriverManager.getConnection("jdbc:mysql://localhost/positions",
                                                  "javaserver","java");
…
```

Figure 19 : connexion sur une base de données en Java

Un changement de base de données nécessite juste l'indication d'un nouveau driver spécifique à la base de données remplaçante et l'écriture d'une ligne pour la connexion contenant le nom de l'utilisateur, son mot de passe et le nom de la base de données à utiliser.

3.3.2 La base de données MySql

Avant de pourvoir être utilisée, une base de données, doit être crée dans le système de gestion de MySql. Ceci permet d'utiliser simultanément plusieurs bases contenues dans un seul système de gestion de base de données. Ainsi la machine serveur peut servir pour plusieurs projets.

Après une analyse, la base de données nécessaire pour notre projet peut être schématisée de cette manière :

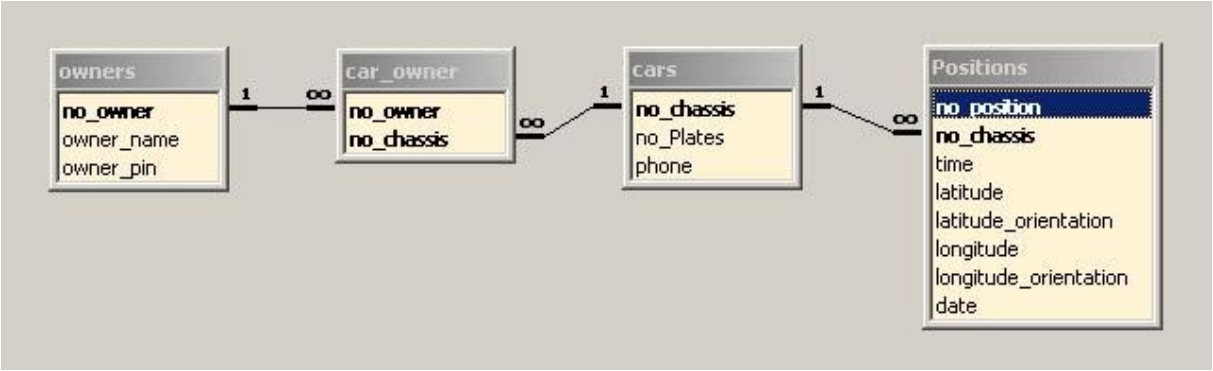


Figure 20 : schéma de la base de données

L'utilité de la table `car_owner` n'est pas forcément évidente. Elle est utile pour gérer des problèmes tels que la copropriété ou la multipropriété. Ceci évite d'avoir plusieurs fois dans la base de données, les informations concernant une seule voiture.

MySql n'offre pas d'interface graphique performant, la plupart des actions se font à l'aide d'une ligne de commande. Toutefois, sur certaines versions, il est possible d'utiliser une interface html. Pour des raisons de reproductibilité des actions, nous allons utiliser la ligne de commande.

Une fois la base de données crée, il faut accorder les droits aux utilisateurs, dans notre cas, il n'y aura qu'un utilisateur : 'javaserver' et il accédera la base de données seulement depuis la machine locale. Ainsi, il sera impossible de consulter les données depuis une autre machine que le serveur, même si nous connaissons le mot de passe de l'utilisateur 'javaserver'. Il s'agit là d'un niveau de sécurité non négligeable pour nos données. Il faut aussi indiquer ce que l'utilisateur a le droit de faire en donnant une liste de commandes qu'il peut exécuter.

```
c:\>cd mysql

C:\mysql>cd bin

C:\mysql\bin>mysql -h localhost -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20 to server version: 3.23.44-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database positions;
Query OK, 1 row affected (0.00 sec)

mysql> grant select, insert, update, delete, create, drop
    -> on positions.*
    -> to javaserver@localhost
    -> identified by 'java';              <-- ça c'est le password
Query OK, 0 rows affected (0.08 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.12 sec)

mysql> quit
Bye

C:\mysql\bin>mysql -h localhost -u javaserver -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22 to server version: 3.23.44-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use positions;
Database changed
mysql> select * from positions
    -> ;
Empty set (0.00 sec)

mysql> quit
Bye
```

Figure 21 : création de la base de données et accord des droits à un utilisateur

Pour la création des tables, il est possible d'utiliser le MySqlManager, c'est une interface graphique pour écrire des requêtes SQL et les envoyer à la base de données.

```
create table positions (
no_position integer not null,
no_chassis char(20) not null,
time integer,
latitude float(10, 4),
latitude_orientation char(1),
longitude float(10, 4),
longitude_orientation char(1),
date integer,
primary key (no_position, no_chassis))

create table cars (
no_chassis char(20) not null,
no_plates char(10),
phone integer,
primary key (no_chassis))

create table owners(
no_owner integer not null,
owner_name char(20),
owner_pin integer,
primary key (no_owner))

create table car_owner (
no_owner integer not null,
no_chassis char(20) not null,
primary key (no_owner, no_chassis))
```

Figure 22 : création des tables

La base de donnée existe maintenant et contient les tables dont nous avons besoin. Le serveur TCP peut donc insérer les données à l'aide de requêtes SQL simple.

Pour des raisons de facilité, nous avons choisi d'identifier les utilisateurs par un PIN au lieu d'un mot de passe. Ceci peut être changé en modifiant le type de la colonne `owner_pin` de la table `owners`.

3.3.3 Choix d'un système d'exploitation

Pour un serveur, les deux systèmes d'exploitation possible sont Windows et Linux. Dans notre cas, les deux sont valables. Il est possible de trouver des distributions gratuites de MySql pour les deux plateformes. Il existe des serveurs Web et FTP gratuits pour les deux. La seule restriction avec un serveur Linux est que nous ne pouvons pas faire le site de consultation des données en ASP (propre à Microsoft).

Ce qui nous fait pencher pour l'utilisation d'un serveur Windows est qu'un projet parallèle utilise une base de données Access et que cet élément n'est pas disponible pour Linux. Notre serveur sera donc basé sur une machine Windows NT4.

3.4 Plan de travail

Pour cette partie du projet, nous disposions déjà d'un émetteur-récepteur TCP testé dans lors d'un laboratoire de dernière année. Nous décidons donc de reprendre cette partie et d'y ajouter des fonctionnalités supplémentaires nécessaires à notre application. Au lieu de faire un simple affichage de la trame reçue, nous devons la découper pour extraire les valeurs de la latitude, de la longitude, de la date, de l'heure etc… avant l'insertion dans la base de données.

Initialement, le serveur était prévu pour limiter le nombre de points stockés par véhicule, ceci pour éviter que la base de données ne prenne trop d'ampleur. Les points étaient stockée de manière circulaire, le point le plus récent écrasant le plus ancien. Cette contrainte d'insertion à été éliminée à la demande de l'initiateur du projet. La base de données garde maintenant tout l'historique du trajet du véhicule pendant qu'il émet sa position.

Nous disposons du port 4051 ouvert sur le firewall d'entrée de l'école. Nos applications clientes et serveur peuvent communiquer par cette ouverture dans la sécurité de l'école.

3.4.1 La ligne de commande

Pour démarrer, le serveur n'a besoin de savoir que le nom de la base de données, le nom de l'utilisateur de la base de données ainsi que son mot de passe. Tous les autres paramètres tels que le port sur lequel les clients viennent se connecter sont fixes. Une amélioration pourrait être de les rendre configurables par la ligne de commande.

3.4.2 Liste des classes

Le serveur est composé d'un paquetage Java appelé `Server` et comportant 4 classes.

Figure 23 : représentation du paquetage Server

La classe `ConnectMe` est responsable de la connexion, de la déconnexion et de tous les accès à la base de donnée. Il s'agit en particulier de calculer le numéro du point à insérer et de le stocker dans la base de donnée.

La classe `DonneeGPS` contient les variables concernant le point en cours de traitement ainsi que des méthodes permettant de les lire et de leur assigner une valeur.

La classe `tcp_receiver` a pour utilité d'ouvrir le port utilisé pour notre application, d'accepter la connexion d'un client ainsi que de lancer un thread de traitement de ce client.

La classe `Traitement` s'occupe de lire les données qui arrivent par le port TCP, de décomposer la trame ainsi reçue et d'assigner les valeurs aux variables de la classe donneeGPS.

3.5 Le serveur Web

Cette partie du projet est la seule visible directement par l'utilisateur de notre application. Il s'agit donc de la soigner particulièrement.

3.5.1 Choix d'un serveur http

La machine dont nous disposons déjà pour réceptionner les trames TCP envoyées au travers d'Internet par notre client peut être réutilisée pour y disposer un site Internet consultable depuis le monde entiers au même titre que http://www.eivd.ch ou encore http://www.tcom.ch. Notre site s'appelle http://gprs.eivd.ch. C'est cette adresse qu'il faut taper dans un browser pour accéder aux informations concernant votre véhicule.

Pour qu'une machine puisse servir de serveur Web, il faut qu'elle dispose d'un logiciel tel qu'IIS (Internet Information Server) de Microsoft ou Apache. Apache est écrit en Java, il est donc possible de l'installer sur une machine n'ayant pas Windows comme système d'exploitation. Ce qui nous fait choisir Apache pour ce projet est l'argument du prix. Apache est gratuit alors que Microsoft IIS est payant.

Maintenant qu'Apache a été choisi, il faut savoir qu'il ne sait que distribuer des pages HTML statiques. Si nous voulons faire un site Web dynamique il nous faut donc ajouter un autre programme écrit en Java, appelé Tomcat. Ce 'module' est utilisé lorsque nous voulons faire des sites dynamiques basés sur des servlets (mini-applications exécutées sur le serveur) ou des JSP (Java Server Pages). Ces deux méthodes servent à créer une page en HTML pur qui sera envoyée au browser du client. Le code Java exécuté sur le serveur ne sert qu'a écrire des lignes de code HTML. Ces deux technologies très semblables sont très utilisées pour des sites ayant des interactions avec des bases de données. C'est ce que nous voulons faire.

Maintenant que nous avons le matériel et les logiciels, il ne reste plus qu'a créer les pages qui seront vues par l'utilisateur depuis chez lui sur l'écran de son ordinateur connecté à l'Internet.

Le langage utilisé pour ces pages est Java, il reste maintenant à choisir entre les servlets ou les JSP. Les servslets sont simplement du code java qui écrit des lignes de HTML les unes après les autres. Les JSP sont composées de code HTML avec quelques parties de Java. Les JSP sont plus accessible à des personnes ne connaissant pas le Java, il est possible de trouver des programmes servant à des "designers" pour faire les graphismes des sites Web. Il suffit ensuite de rajouter les parties de code Java dans les pages ainsi générés.

Le site Web que nous devons faire est simple, il sera donc écrit totalement à la main. Mais dans un souci d'évolution de notre projet vers un produit commercial nous choisissons les JSP. Ainsi il sera plus facile de rendre le site Web existant plus en accord avec la ligne graphique du site principal de Swisscom.

La connexion d'un utilisateur désirant connaître la position de son véhicule se déroule comme suit :

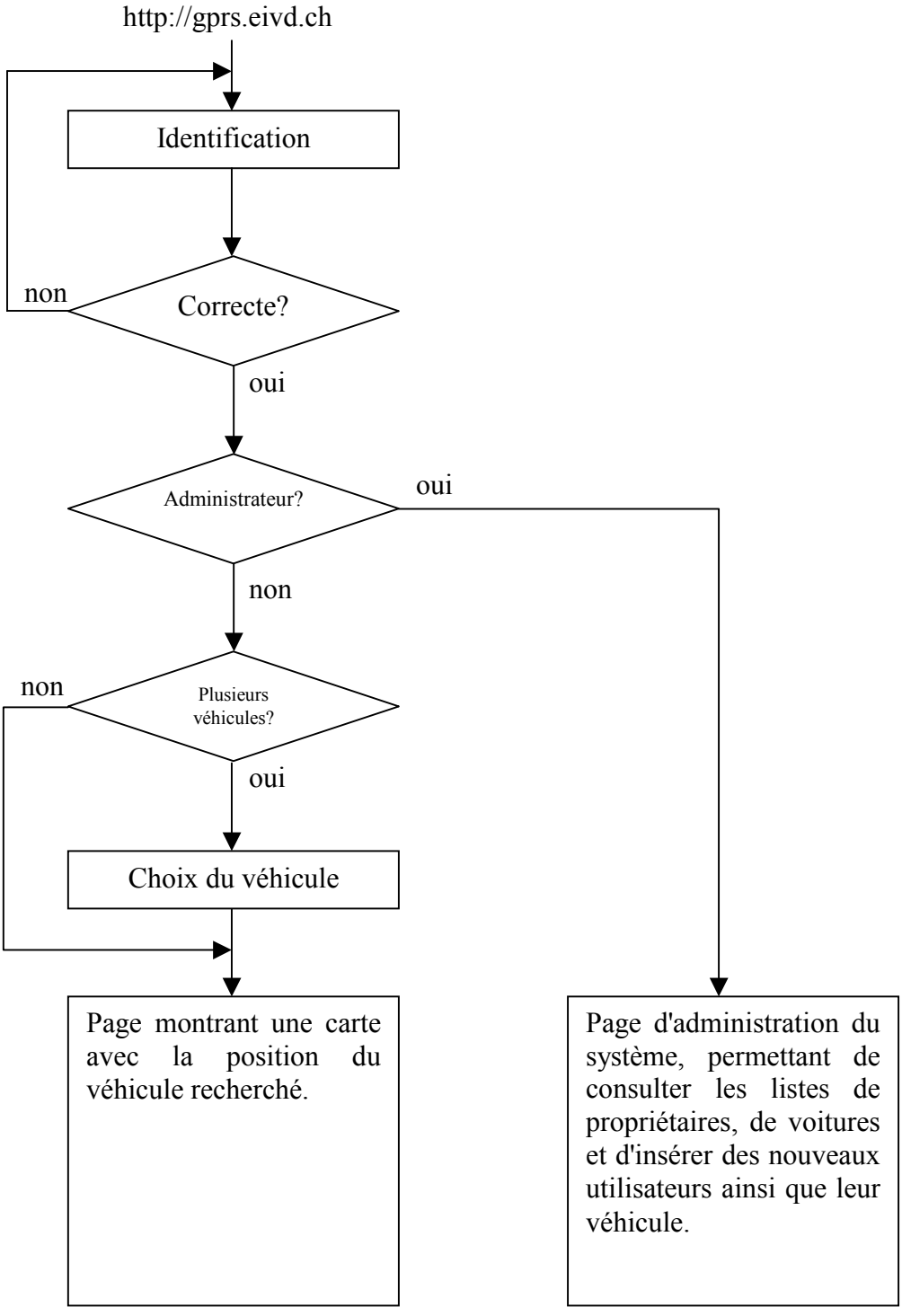


Figure 24: organigramme de fonctionnement du site Web

3.6 Méthode de travail

Les pages s'écrivent dans l'ordre ou elles vont être utilisées, la méthode utilisée pour transmettre les paramètres (champs remplis) à la page suivante est 'post'. Tout le site est conçu pour que son utilisation soit intuitive. La version présentée dans ce rapport n'est pas définitive, il faudrait encore rajouter des fonctionnalités d'administration telles que la suppression d'utilisateurs et de voitures. Ainsi que quelques fonctionnalités graphiques telles que le déplacement d'une carte à l'autre par curseur ou alors une sorte de zoom arrière permettant de voir quelle partie de la Suisse est affichée de manière détaillée.

3.6.1 Liste des pages

Le serveur Web est composé de cinq pages.

Figure 25: structure du serveur Web

La page `identification` demande le nom et le code d'identification de la personne désirant consulter la position de son véhicule. Elle utilise la méthode 'post' pour appeler la page 'choose'.

La page `choose` détecte si l'utilisateur est autorisé à utiliser le système ou si c'est l'administrateur. Si l'administrateur est détecté, la page 'admin' est appelée. Si c'est un autre utilisateur, le système compte le nombre de voiture dont l'utilisateur courant est propriétaire. S'il possède seulement un véhicule, la page 'print est appelée en postant le numéro de châssis. S'il possède plusieurs voitures, une liste est affichées et l'utilisateur est invité à choisir celle pour laquelle il veut afficher des informations.

La page `print` affiche la carte sur laquelle il est possible de voir la dernière position transmise par le véhicule. Sur cette carte, il est possible de voir, indiquée par un point clignotant, la dernière position et, par un point normal les autres positions pouvant être affichée sur cette même carte.

La page `admin`, visible seulement par l'administrateur, lui propose une liste d'opérations qu'il peut effectuer, telles que lister des utilisateurs ou des véhicules, insérer des utilisateurs ou encore des voitures.

La page `Bdaccess` met en forme des requêtes en fonction de ce que l'administrateur veut faire, ainsi il est possible d'administrer tout le système sans connaître le langage d'accès aux bases de données qu'est SQL.

3.7 Les cartes géographiques

Les cartes géographiques détaillées fournies avec le site Internet proviennent du programme *Swiss Map100* et sont sous licence de l'office fédéral de topographie. Elles ne sont donc là qu'a titre de démonstration.

3.8 Un répéteur GSM

Le niveau de signal GSM étant très faible, voir nul dans notre laboratoire, il nous est presque impossible de faire des tests en utilisant GPRS. Swisscom Mobile nous a donc prêté un répéteur GSM. Il s'agit d'une antenne extérieure reliée par un câble coaxial à un boîtier ré-émetteur qu'il faut placer à l'intérieur. Cet élément augmente significativement le niveau de réception autour de lui. L'antenne extérieure doit être pointée avec précision sur un émetteur de Swisscom.

3.9 Résultat

Après quelques tests en grandeur nature, il s'avère que le système fonctionne bien, nous avons juste à déplorer un petit manque de précision quand à l'affichage des points, ainsi un véhicule circulant sur l'autoroute risque bien d'être indiqué comme visitant le champ voisin.

Figure 26: exemple d'imprécision

Ce manque de précision, provient en partie de nos calculs lors de la conversion entre les systèmes de coordonnées. Sur cette opération, nous introduisons une erreur de ± 10 mètres. Le reste de l'imprécision vient du système GPS en lui même. En temps normal, pour l'utilisateur, cette précision est d'une dizaine de mètres. Nous sommes en décembre 2001, les USA sont en guerre, la précision des satellites GPS peut être estimée à 50-100m. Au regard de ces paramètres nous ne pouvons pas fournir un système plus précis. Ceci peut poser un problème lors de la recherche d'un véhicule en ville, les rues étant souvent distante de moins de 100m, un véhicule risque d'être indiqué dans une rue voisine de celle ou il se trouve réellement.

4 Conclusions

4.1 Améliorations possibles

Le produit présenté dans ce rapport n'est pas fini, il est encore possible d'y apporter quelques améliorations telles que:

- Pour le client, ajouter un contrôle de la connexion TCP pendant la phase de travail. Actuellement, si la liaison est interrompue après que la connexion ait été établie, le système continue son travail même si les données n'arrivent plus au serveur.
- Pour le serveur, modifier l'acceptation des clients de manière à ce que des connexions simultanées puissent être acceptées. Actuellement, le serveur n'accepte qu'un client à la fois, c'est une limitation imposée par le système d'exploitation du serveur à savoir: Windows NT4.
- La base de données peut aussi être améliorée, elle devrait contenir plus d'information sur les utilisateurs. Le stockage des points peut aussi être amélioré. Actuellement c'est la latitude et la longitude qui sont mémorisées. Il serait positif de travailler avec les coordonnées suisses, ainsi il serait plus facile de retrouver un véhicule, même sans voir la carte.
- Le site Web, n'est pas parfait non plus, la partie utilisateur mérite une amélioration de la gestion des cartes. Il serait bien de pouvoir aller consulter les cartes voisines de celle sur laquelle se trouve actuellement le véhicule. Un zoom arrière devrait aussi être prévu pour pouvoir visualiser dans quelle partie de la suisse se trouve la partie de carte affichée. Il serait bien aussi de 'rafraichir' périodiquement l'affichage de la carte ainsi il serait possible de suivre un véhicule encore en mouvement. Une dernière fonction consisterait à afficher les informations d'un point lorsque l'utilisateur clique dessus sur la carte.

Une extension du système pourrait consister à afficher plusieurs véhicules sur la carte, il serait ainsi possible d'imaginer qu'une entreprise qui veut suivre sa flotte de véhicules d'entreprise puisse le faire avec notre système.

4.2 Produits semblables

Il est possible de trouver sur Internet des systèmes semblables au notre. Cependant, la principale nouveauté de notre projet est l'utilisation d'un réseau GPRS. Les autres systèmes transmettent leurs informations par GSM.

http://www.saphelec.fr/radio/gps/suivregsm.htm
Il s'agit là d'un système qui mémorise une série de points avant d'ouvrir une liaison GSM pour les envoyer. Il n'y a pas d'activation à distance et ce système nécessite l'emploi d'un logiciel pour consulter les données récoltées.

http://www.europesatellite.com/gps/suivi/suivi.htm
Ici aussi, le boîtier embarqué mémorise les positions et les envoie lorsqu'il reçoit un appel d'une machine distante. Cela revient à télécharger par GSM les données lorsqu'un opérateur les demande.

4.3 Conclusion

Je pense que le résultat obtenu pour ce projet lors de ce travail de diplôme est satisfaisant et même encourageant. Certes il reste quelques modifications à apporter mais il est déjà possible de tester et d'utiliser l'ensemble du système de localisation.

Sur le plan personnel, la réalisation de ce projet m'a apporté beaucoup d'enthousiasme et de motivation. J'ai appris à gérer les problèmes imprévus et à les résoudre. A étudier plusieurs solutions avant d'en choisir une et de la réaliser. J'ai aussi découvert le langage Java, auquel je n'avais jamais vraiment porté attention auparavant.

Yverdon-les-bains, le 20 décembre 2001

Pierre Daccord

Bibliographie

[1] Markus Jaton
Java : le langage
Java : les extension en télécommunications
Tcom 2000/2001

[2] Michel Bonjour, Gilles Falquet, Jaques Guillot et André Le Grand
Java: de l'esprit à la méthode
Thomson Paris 1996

[3] Eliotte Rusty Harold
Programmation réseau avec Java
O'Relly Paris 1997

[4] Internet
http://java.sun.com
http://www.javavorld.com
http://www.garmin.ch
http://www.ericsson.com
http://www.gpsy.com
http://www.3gpp.org
http://www.etsi.org