de développement EUROP-44BOX (Figure 1). Si vous ne possédez pas un tel module il est toujours possible d'utiliser un simulateur sur PC, comme GPS Simulator (Figure 3).

Avec cet outil, vous pourrez sélectionner les trames transmises ainsi que leur fréquence d'émission.

En effet, un module GPS est capable de transmettre bien plus d'information que les coordonnées elles-mêmes. Vous pouvez ainsi obtenir la qualité du signal, le nombre de satellites détectés, l'heure fournie par les satellites. Pour cela, les informations sont rassemblées au sein de 6 trames :

- GPGLL,
- GPRMC,
- CPVTG,
- GPZDA,
- GPGGA,
- GPGSA.

Les trames qui nous intéressent sont les trames GPRMC et surtout GPGGA. Référez-vous aux spécifications NMEA 0183 et aux nombreux sites disponibles sur Internet pour connaître le détail de chacune des trames.

Module GPRS

Acquérir des informations depuis notre module GPS est une bonne chose pour ce qui est du stockage des informations, mais notre objectif final est de pouvoir transmettre ces informations à distance depuis n'importe quelle position géographique. Nous pouvons choisir de les transmettre au travers d'un support classique comme une transmission radio à 433MHz, mais les débits sont faibles et les portées réduites. La seule solution technique accessible est encore d'utiliser un téléphone mobile qui nous permettra d'établir une liaison simplement depuis n'importe quel point géographique.

Plutôt que de transmettre les informations au moyen de SMS, et vu qu'il nous faut transférer des données numériques de tout type, nous allons utiliser la transmission GSM en mode data, tel le GPRS. Des modules sont aisément disponibles en France à l'unité et nous choisissons donc de nous procurer un de ces modules : un SocketModem GSM/GPRS de la société Multitech (Figure 4).

Le SocketModem GSM/GPRS de Multitech est un modem sans fil prêt à être intégré. Il est compatible avec les recommandations multi bande GMS/GPRS classe 10. Il va nous permettre de mettre en place rapidement notre communication sans fil via une interface modem standard et l'envoie de commande AT.

Il s'alimente directement en 5V et nécessite une carte SIMet une antenne externe à connecteur MMCX. Nous trouvons l'antenne adéquate chez Farnell en France, et il ne nous reste plus alors qu'à nous procurer une carte SIM avec un compte permettant les transferrer en mode data. Nos premiers essais se feront donc avec une carte SIM standard telle que celles utilisées dans les téléphones GSM/GPRS grand public.

Par la suite, nous contactons les grands opérateurs téléphonique français afin d'obtenir une carte SIM destinée uniquement au transfert de données. Ces cartes, destinées au marché du *machine to machine*, permettent de souscrire à des abonnements spécifiques peu coûteux, de l'ordre de 5 euros par mois, avec un paiement au Ko transféré ou à la seconde de connexion.

Il suffit alors de communiquer avec le modem au travers d'une liaison série TTL. Des commandes AT (ASCII) devront être utilisées pour communiquer avec le modem et établir une connexion (Listing 1). Les deux premières commandes permettent de vérifier que la communications est bien établie avec le modem (il doit répondre *OK*), et les trois suivantes demandent au modem de passer en mode data (GPRS).

Enfin, on informe le modem de l'adresse du serveur auquel il va devoir se connecter. Pour SFR, il s'agit de *m2minternet,* alors que, pour Bouygtel, il faut spécifier *a2bouygtel.com*. Une fois la connexion établie avec le serveur de notre opérateur, nous sommes donc parfaitement reliés

Listing 4. Traitement des trames GPGGA

```
COMfd = InitCOMGPS(COMfd, "/dev/ttyUSB0");
// File descriptor to survey
FD_ZERO(&FD_Lect);
FD_SET(COMfd,&FD_Lect);
// infinite loop dedicated to the process
while(1) {
   // We setup the timeout of the GPS
   delay.tv_sec = 4;
   delay.tv_usec = 0;
   // we erase the old data treceived
   bzero(Buffer, sizeof(Buffer));
   // We suspend the process until a new frame is received (LF detected)
   retSelect = select(COMfd+1, &FD_Lect, NULL, NULL, &delay);
   if (retSelect) {
      ret = read(COMfd, Buffer, sizeof(Buffer));
      // If checksum is not correct we refuse the frame
      ret = ComputeChecksum(Buffer, ret);
      if (ret != 1) {
         printf("<ReadGPS> Bad checksum [ret=%d]..!\n", ret);
         continue;
      }
      // We only parse the $GPGGA frame
      if (strstr(Buffer, "$GPGGA") != NULL)
         ComputeGPGGA(Buffer, &ReceivedData);
      // We have lost the previously detected satellites...
      if (ReceivedData.QualiteIndicator[0] == '0') {
         printf("<ReadGPS> No sat found [%c]!\n", ReceivedData.
               QualiteIndicator[0]);
         continue;
      }
      printf("\n*************************************************\n");
      printf("Heure de l'enregitrement : %s\n", ReceivedData.Time);
      printf("Qualite du suivi : %c\n", ReceivedData.QualiteIndicator[0]);
      printf("Nombre de satellites en suivi: %s\n", ReceivedData.NbSat);
      printf("Longitude Est : %s\n", ReceivedData.Longitude);
      printf("Latitude Nord : %s\n", ReceivedData.Latitude);
      ret = write(GPSPipeFD, &ReceivedData, sizeof(GPSData));
      tcflush(COMfd, TCIFLUSH);
   }
}
```