ER2015 S3_1
Récupérer l'heure GPS

1. Simulateur de GPS

Le but est d'écrire un programme qui simule un GPS émettant toutes les 5 secondes des trames GPGGA depuis la position du bâtiment atelier GE. L'heure commencera à une valeur quelconque au reset du programme. La position ne changera pas. On prévoira la possibilité d'émettre aléatoirement des trames avec une cheksum fausse et des trames mal formées (ne commençant pas par $ et/ou ne se terminant pas correctement).

```
The most important NMEA sentences include the GGA which
provides the current fix data, the RMC which provides the
minimum gps sentences information, and the GSA which
provides the Satellite status data.
GGA - essential fix data which provide 3D location and
accuracy data.
 $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46
.9,M,,*47
Where:
    GGA          Global Positioning System Fix Data
    123519       Fix taken at 12:35:19 UTC
    4807.038,N   Latitude 48 deg 07.038' N
    01131.000,E  Longitude 11 deg 31.000' E
    1            Fix quality: 0 = invalid
                              1 = GPS fix (SPS)
                              2 = DGPS fix
                              3 = PPS fix
                     4 = Real Time Kinematic
                     5 = Float RTK
               6 = estimated (dead reckoning) (2.3 feature)
                7 = Manual input mode
                8 = Simulation mode
    08           Number of satellites being tracked
    0.9          Horizontal dilution of position
    545.4,M      Altitude, Meters, above mean sea level
    46.9,M   Height of geoid (mean sea level) above WGS84
                         ellipsoid
 (empty field) time in seconds since last DGPS update
```

```
 (empty field) DGPS station ID number
 47          the checksum data, always begins with *
```

If the height of geoid is missing then the altitude should be suspect. Some non-standard implementations report altitude with respect to the ellipsoid rather than geoid altitude. Some units do not report negative altitudes at all. This is the only sentence that reports altitude.

The checksum is the representation of two hexadecimal characters of an XOR of all characters in the sentence between – but not including – the $ and the * character.

Conception :

Pour la gestion du temps on utilisera trois variables globales :

```
uint8_t hour=23,min=58,sec=0;
```

et une fonction

```
void ComputeTime(void) ;
```

qui sera appelée chaque seconde à l'aide de la fonction millis() ;
La trame GPGGA sera stockée dans un tableau de char de taille suffisante. Les éléments fixes ($GPGGA, latitude, longitude,…) pourront être remplis une seule fois à initialisation.
Chaque fois que la trame doit être émise il faudra renseigner la date et la cheksum.
On écrira une fonction :

```
uint8_t computeCheksum(char * sentence) ;
```

qui calculera la cheksum (sous forme d'un octet).
Et une fonction :

```
void appendCheksum(char * sentence,\
                   size_t sentence_size ) ;
```

qui ajoutera les caractères de la cheksum à la trame.
Pour former la trame on pourra utiliser les fonctions

```
snprintf(..), strncpy(...), strncat(...)
```

de la librairie C standard.

Pour émettre des trames erronées, on pourra utilisateur la fonction random de la bibliothèque arduino à la fois pour obtenir une probabilité d'émettre une trames fausse et pour modifier la cheksum.

2. Récupération de l'heure dans une trame GPGGA

Le but est d'écrire un programme qui "récupère l'heure" dans la trame reçue : c'est à dire qu'il met à jour trois variables, `hour_pgs`, `min_gps` et `sec_gps` quand il reçoit une trame GPGGA valide (bonne cheksum) et complète : lors de la mise sous tension le module GPS émet quelques trames GPGGA incomplètes tant que le "fix" n'est pas fait. Il faudra détecter ce type de trame et ne pas les traiter.

Stockage de la trame :
La trame est reçue par un UART donc caractère par caractère. On va d'abord stocker une trame complète puis l'exploiter.

Le code ci dessous permet d'appeler la fonction retreiveSentence() à chaque fois que l'on reçoit un nouveau caractère sur l'UART de l'Ardiuno Uno. La méthode d'appel de la fonction retreiveSentence() pourra changer par la suite en fonction de l'UART utilisé (hardware 1 ou 2 sur les Mega ou UART soft).

```
char GPSsentence[128]; // pour stocker la trame
void loop() {

  int r=0;
  while (Serial.available()>0) {

       char c = Serial.read();
       r=retreiveSentence(c);

  }
  // r!=0 lorsque qu'une trame complète
  // est disponible dans  GPSsentence ;
  if ( r > 0) {
      exploiter_la_trame() ;
  }

}
```

Écrire la fonction int `retreiveSentence(char c)` qui met à jour la variable `GPSsentence` avec les caractères reçus et retourne 0 tant que la trame est incomplète et le nombre de caractère dans GPSsentence lorsque la trame est complète.
La fonction "attend" un $ avant de commencer à stocker les caractères. La fin de trame est obtenue par un '\n' ou '\r'. Attention à ne pas déborder du tableau si on ne reçoit jamais de caractère de fin !

Exploitation de la trame :
Après avoir vérifié qu'il s'agit bien d'une trame GPGGA valide et complète, mettre à jour les variables de temps.
On pourra utiliser le code ci-dessous (après l'avoir bien compris) :

```
uint32_t parsedecimal(char *str) {
  uint32_t d = 0; int i=0 ;
  while (str[i] != '\0') {
   if ((str[i] > '9') || (str[i] < '0'))
       return d;  // no more digits
   d = (d*10) + (str[i] - '0');
   i++;
  } return d; }
```