

Calculs GNSS

Étudiant :

Ahmed Tsoroev

Enseignant :

Alexandre VERVISCH-PICOIS

Cours :

Projet de fin d'étude - PHY5035

Table des matières

Objectifs	3
Récupération des données	3
Traitement des données	6
Pseudodistances	6
Ephémérides.....	8
Calcul de la position	14
Résultats.....	17
Conclusion	19
Bibliographie.....	20

Objectifs

Première étape : Etude de la carte GPS.

Nous avons à notre disposition deux modules GNSS de u-blox : le NEO-M8N et le LEA-6T. Ce sont deux récepteurs GNSS professionnels miniaturisés pouvant communiquer par liaison série. La première étape consiste à étudier la liaison série et le protocole de communication UBX de u-blox pour comprendre comment établir une connexion avec le récepteur et extraire les données nécessaires.

Deuxième étape : Implémentation des algorithmes de position.

A partir des données brutes extraites du récepteur, le but est de calculer la position du récepteur.

Troisième étape : Comparaison avec le calcul de la carte.

Le module GNSS calcule lui-même sa position, mais les algorithmes utilisés sont propriétaires. Il s'agit donc de réaliser une étude comparative, afin de conclure sur la capacité à mener son propre calcul de position.

Pour réaliser ces objectifs, nous avons créé un outil en Python capable de se connecter en série avec le module u-blox, envoyer des messages avec le protocole UBX et parser les données reçues dans les réponses. Il s'agissait ensuite de traiter ces données pour appliquer les algorithmes de calcul de position.

L'outil [geoloc](#) réalisé est disponible sur GitHub.

Récupération des données

L'outil a été réalisé sous Python en utilisant la librairie open source [pyubx2](#). Pour se connecter en série au module de u-blox, il suffit de créer un objet [UBXStreamer](#) en spécifiant le port 'COMx' du module, le baud rate et le timeout souhaité.

La méthode [connect\(\)](#) de [UBXStreamer](#) permet d'ouvrir la connexion, la méthode [start_read_thread\(\)](#) permet de démarrer le thread qui lira toutes les données venant du module en utilisant lui-même la méthode [_read_thread\(\)](#). Il s'agit ensuite de définir les messages ou requêtes voulues comme des objets [UBXMessage](#), de leur appliquer la méthode [serialize\(\)](#) et de les envoyer vers le module grâce à la méthode [send\(\)](#). Comme le thread de lecture de la communication série est toujours activé, les réponses venant du module seront traitées par la méthode [_read_thread\(\)](#), en particulier les réponses seront stockées dans les variables [raw_data](#) et [parsed_data](#) sous forme brute et parsé respectivement.

Lorsque les requêtes souhaitées ont été envoyées, il est nécessaire d'arrêter le thread et de couper la connexion série avec le module, en utilisant successivement les méthodes `stop_read_thread()` et `disconnect()`.

Voici un exemple du code Python pour établir la connexion série et envoyer les requêtes **UBX-NAV-CLOCK** et **UBX-AID-EPH**.

```
if __name__ == "__main__":

    print("Instantiating UBXStreamer class...")
    ubxs = UBXStreamer(PORT, BAUDRATE, TIMEOUT)
    print(f"Connecting to serial port {PORT} at {BAUDRATE} baud...")
    ubxs.connect()
    print("Starting reader thread...")
    ubxs.start_read_thread()

    print("\nPolling receiver...\n\n")
    msg1 = UBXMessage('NAV', 'NAV-CLOCK', POLL)
    msg2 = UBXMessage('AID', 'AID-EPH', POLL)
    ubxs.send(msg1.serialize(), msg2.serialize())

    print("\n\nStopping reader thread...")
    ubxs.stop_read_thread()
    print("Disconnecting from serial port...")
    ubxs.disconnect()
    print("Done")
```

Figure 1 - Exemple de code pour la communication avec le module u-blox

Le port série du module, le baud rate et le timeout ont préalablement été définis dans le fichier de configuration `config.py`.

Pour connaître la nature des requêtes et les données reçues en réponse, il est nécessaire de consulter la documentation officielle de u-blox : [u-blox 6](#), [u-blox 8](#).

La librairie `pyubx2` recense la majorité des requêtes UBX, à l'exception de certaines requêtes des anciennes générations du module u-blox qui sont désormais obsolète pour les nouvelles générations. Ceci permet de générer le message en suivant le protocole UBX, en spécifiant simplement le nom de la requête, comme on peut le voir avec `msg1` et `msg2` dans l'exemple au-dessus. De plus, la réponse aux requêtes suit une structure particulière et très précise détaillée dans la documentation officielle de u-blox. La librairie `pyubx2` se charge de parser correctement les réponses en fonction de cette structure. Exemple avec le message **UBX-NAV-CLOCK** :

35.2 NAV-CLOCK (0x01 0x22)

35.2.1 Clock Solution

Message	NAV-CLOCK				
Description	Clock Solution				
Firmware	Supported on u-blox 6 from firmware version 6.00 up to version 7.03.				
Type	Periodic/Polled				
Comment	-				
Message Structure	Header	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x01 0x22	20	see below	CK_A CK_B
Payload Contents:					
Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS Millisecond Time of week

GPS.G6-SW-10018-F

Public Release

Page 167 of 210



locate, communicate, accelerate

NAV-CLOCK continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
4	I4	-	clkB	ns	Clock bias in nanoseconds
8	I4	-	clkD	ns/s	Clock drift in nanoseconds per second
12	U4	-	tAcc	ns	Time Accuracy Estimate
16	U4	-	fAcc	ps/s	Frequency Accuracy Estimate

Figure 2 - Extrait de la documentation officielle u-blox 6

Le document officiel décrit la structure de la réponse à la requête avec l'emplacement de chaque champ dans la série d'octet envoyé par le module ainsi que le sens de ce même champ. La réponse est parsée grâce à la librairie et le résultat, stocké dans [*parsed_data*](#), est affiché de la sorte :

```
Instantiating UBXStreamer class...
Connecting to serial port COM5 at 9600 baud...
Starting reader thread...

Polling receiver...

<UBX(NAV-CLOCK, iTOW=19:37:37, clkB=-37831, clkD=149, tAcc=6, fAcc=223)>
```

Figure 3 - Extrait de la console Python correspondante au code de la Figure 1

On reconnaît bien les champs spécifiés dans la documentation de u-blox.

Traitement des données

Deux types de données étaient nécessaires pour calculer la position du récepteur : les **éphémérides** et les **pseudodistances**.

Les éphémérides sont un ensemble de données transmises par chaque satellite et sont un regroupement de données caractérisant l'orbite du satellite et les correctifs à appliquer au calcul de sa position. La pseudodistance est la distance entre le récepteur et le satellite, calculée en faisant la différence entre le temps à la réception du message et le temps à l'émission, multiplié par la vitesse de la lumière. Comme l'horloge embarquée du satellite et celle du récepteur ne sont pas synchronisées, cette différence de temps ne sera pas exactement le temps que le signal a mis pour parcourir la distance satellite-récepteur, d'où le « pseudo » dans pseudodistance. En se référant à la documentation de u-blox, on identifie les requêtes UBX qui fournissent ces données :

- **UBX-AID-EPH** pour les éphémérides.
- **UBX-RXM-RAW** pour les pseudodistances.

Pseudodistances

Le plus simple des deux types de données à obtenir étaient les pseudodistances :


36.4 RXM-RAW (0x02 0x10)					
36.4.1 Raw Measurement Data					
Message	RXM-RAW				
Description	Raw Measurement Data				
Firmware	Supported on u-blox 6 from firmware version 6.00 up to version 7.03 (only available with raw data product variant).				
Type	Periodic/Polled				
Comment	This message contains all information needed to be able to generate a RINEX file.				
Message Structure	Header	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x02 0x10	8 + 24*numSV	see below	CK_A CK_B
Payload Contents:					
Byte Offset	Number Format	Scaling	Name	Unit	Description
0	I4	-	iTOW	ms	Measurement integer millisecond GPS time of week (Receiver Time)
4	I2	-	week	weeks	Measurement GPS week number (Receiver Time).
6	U1	-	numSV	-	# of satellites following.
GPS.G6-SW-10018-F					
Public Release					
Page 185 of 210					
					
locate, communicate, accelerate					
RXM-RAW continued					
Byte Offset	Number Format	Scaling	Name	Unit	Description
7	U1	-	reserved1	-	Reserved
Start of repeated block (numSV times)					
8 + 24*N	R8	-	cpMes	cycles	Carrier phase measurement [L1 cycles]
16 + 24*N	R8	-	prMes	m	Pseudorange measurement [m]
24 + 24*N	R4	-	doMes	Hz	Doppler measurement [Hz]
28 + 24*N	U1	-	sv	-	Space Vehicle Number
29 + 24*N	I1	-	mesQI	-	Nav Measurements Quality Indicator: >=4 : PR+DO OK >=5 : PR+DO+CP OK <6 : likely loss of carrier lock in previous interval
30 + 24*N	I1	-	cno	dbHz	Signal strength C/No. (dbHz)
31 + 24*N	U1	-	lli	-	Loss of lock indicator (RINEX definition)
End of repeated block					

Figure 4 - Extrait de la documentation officielle u-blox 6, message UBX-RXM-RAW

On reconnaît la donnée souhaitée dans le champ « prMes », qui se trouve dans un bloque de champs qui se répètent autant de fois qu'il y a de satellites pour lesquels le module a reçu les pseudodistances (seules les données des satellites actuellement visibles par le récepteur sont transmises).

```
<UBX(RXM-RAW, iTOW=20:04:57, week=2142, numSV=9, reserved1=0,
cpMes_01=8072104.550874984, prMes_01=24304352.619048595, doMes_01=-3747.046875, sv_01=31, mesQI_01=7, cno_01=38, lli_01=1,
cpMes_02=-4491873.638587657, prMes_02=21037426.700071678, doMes_02=1527.3387451171875, sv_02=16, mesQI_02=7, cno_02=47, lli_02=1,
cpMes_03=-2346627.826599138, prMes_03=25487325.250800572, doMes_03=3266.13720703125, sv_03=10, mesQI_03=7, cno_03=37, lli_03=1,
cpMes_04=4129959.674833714, prMes_04=25933908.151956115, doMes_04=-3986.23974609375, sv_04=25, mesQI_04=4, cno_04=17, lli_04=3,
cpMes_05=66789.42364150938, prMes_05=20396015.887099653, doMes_05=-531.582275390625, sv_05=18, mesQI_05=5, cno_05=28, lli_05=3,
cpMes_06=-4060538.973396397, prMes_06=23589603.6538305, doMes_06=2999.635009765625, sv_06=23, mesQI_06=7, cno_06=32, lli_06=1,
cpMes_07=-6551190.06604685, prMes_07=22940295.727370165, doMes_07=2703.6845703125, sv_07=27, mesQI_07=7, cno_07=42, lli_07=1,
cpMes_08=-29247.906714080833, prMes_08=20417175.951329917, doMes_08=-599.12060546875, sv_08=26, mesQI_08=7, cno_08=46, lli_08=1,
cpMes_09=3423087.9850068674, prMes_09=23033876.620800365, doMes_09=-3107.180908203125, sv_09=29, mesQI_09=4, cno_09=27, lli_09=3)>
```

Figure 5 - Extrait de la console Python correspondante à la réponse de la requête UBX-RXM-RAW

Attention : la façon dont ce message est affiché a été modifiée de manière forcée dans le code de la librairie pour une meilleure lisibilité à l'affichage. Il se peut donc que le message ne soit pas affiché de la même façon chez un autre utilisateur. Le contenu n'a cependant pas été modifié et correspond à ce que retourne le récepteur.

On reconnaît ici également la structure décrite dans la documentation, avec par exemple `prMes_01` correspondant à la pseudodistance du satellite dont l'identifiant est 31.

Lorsque le thread de lecture de la communication série est activé, la méthode `_read_thread()` tourne en permanence, elle permet non seulement d'afficher les données reçues (résultant dans les données visibles dans la console Python dans les exemples au-dessus) mais aussi de les manipuler. En effet, la variable `parsed_data` prends la structure de la réponse à la requête. Par exemple lorsque le module envoie la réponse à la requête `UBX-RXM-RAW`, `parsed_data.numSV` donnera le nombre de satellites pour lesquels le module a reçu la pseudodistance. C'est donc essentiellement dans cette méthode que se fera le traitement des données.

Ephémérides

On se réfère à la documentation officielle pour trouver la structure de la réponse à la requête permettant d'obtenir les éphémérides :

30.6.3 GPS Aiding Ephemeris Input/Output Message					
Message	AID-EPH				
Description	GPS Aiding Ephemeris Input/Output Message				
Firmware	Supported on u-blox 6 from firmware version 6.00 up to version 7.03.				
Type	Input/Output Message				
Comment	<ul style="list-style-type: none"> SF1D0 to SF3D7 is only sent if ephemeris is available for this SV. If not, the payload may be reduced to 8 Bytes, or all bytes are set to zero, indicating that this SV Number does not have valid ephemeris for the moment. This may happen even if NAV-SVINFORM and RXM-SVSI are indicating ephemeris availability as the internal data may not represent the content of an original broadcast ephemeris (or only parts thereof). SF1D0 to SF3D7 contain the 24 words following the Hand-Over Word (HOW) from the GPS navigation message, subframes 1 to 3. The Truncated TOW Count is not valid and cannot be used. See IS-GPS-200 for a full description of the contents of the Subframes. In SF1D0 to SF3D7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored. When polled, the data contained in this message does not represent the full original ephemeris broadcast. Some fields that are irrelevant to u-blox receivers may be missing. The week number in Subframe 1 has already been modified to match the Time Of Ephemeris (TOE). 				
Message Structure	Header	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x0B 0x31	(8) or (104)	see below	CK_A CK_B
Payload Contents:					
Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	svid	-	SV ID for which this ephemeris data is (Valid Range: 1 .. 32).
4	U4	-	how	-	Hand-Over Word of first Subframe. This is required if data is sent to the receiver. 0 indicates that no Ephemeris Data is following.
Start of optional block					
8	U4[8]	-	sf1d	-	Subframe 1 Words 3..10 (SF1D0..SF1D7)
40	U4[8]	-	sf2d	-	Subframe 2 Words 3..10 (SF2D0..SF2D7)
72	U4[8]	-	sf3d	-	Subframe 3 Words 3..10 (SF3D0..SF3D7)
End of optional block					

Figure 6 - Extrait de la documentation officielle u-blox 6, message UBX-AID-EPH

Ici, on trouve une particularité des éphémérides. Les champs du message sont appelés des « subframe » contenant des « words ». En effet, les éphémérides sont sous-divisées en 5 « subframes » contenant toutes 10 « words ». Il est *important de noter* que pour ce message, le module u-blox envoie que les subframe 1, 2 et 3 et les words 3 à 10 pour chacune. Ceci ne pose pas de problèmes car ce sont exactement les informations nécessaires pour calculer la position des satellites dans le ciel, mais il est important de le savoir pour traiter les données correctement.

Pour connaître le contenu de chaque subframe et de chaque word, il faut se référer à la documentation officielle du système GPS : [IS-GPS-200H](#) :

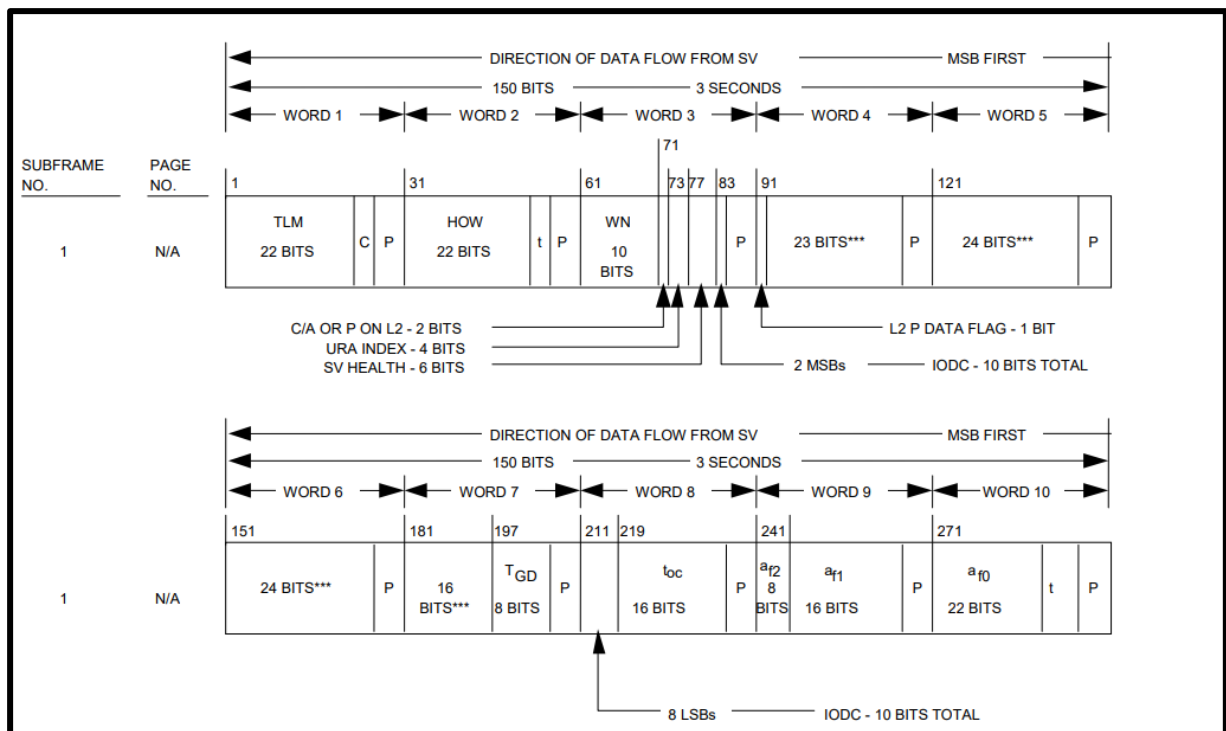


Figure 7 - Extrait de la documentation officielle du système GPS explicitant le contenu de la Subframe 1

Grâce à cette description du contenu de chaque word de la subframe 1, il s'agit de tirer les variables comme IODC, T_{GD} , t_{oc} , a_{f2} , a_{f1} , a_{f0} en comptant les bits pour extraire seulement les bits correspondants à ces variables dans chaque word.

Il est *important de noter* que u-blox ne transmet pas exactement les bits tel qu'ils sont décrits dans la documentation officielle du système GPS. Par exemple, du point de vue du système GPS, un word fait 30 bits, alors que u-blox transmet les word en 32 bits. Il faut donc croiser les deux documents pour comprendre où récupérer quelles données dans ce qui est transmis par u-blox.

Une fois que les bons emplacements des tous les paramètres ont été identifiés, il faut se référer encore une fois à la documentation officielle du système GPS pour connaître les facteurs d'échelles à appliquer, savoir s'il faut prendre le complément à deux et connaître l'unité des paramètres :

Table 20-III. Ephemeris Parameters				
Parameter	No. of Bits**	Scale Factor (LSB)	Effective Range***	Units
IODE	8			(see text)
C_{rs}	16*	2^{-5}		meters
Δn	16*	2^{-43}		semi-circles/sec
M_0	32*	2^{-31}		semi-circles
C_{uc}	16*	2^{-29}		radians
e	32	2^{-33}	0.03	dimensionless
C_{us}	16*	2^{-29}		radians
\sqrt{A}	32	2^{-19}		$\sqrt{\text{meters}}$
t_{oe}	16	2^4	604,784	seconds
C_{ic}	16*	2^{-29}		radians
Ω_0	32*	2^{-31}		semi-circles
C_{is}	16*	2^{-29}		radians
i_0	32*	2^{-31}		semi-circles
C_{rc}	16*	2^{-5}		meters
ω	32*	2^{-31}		semi-circles
$\dot{\Omega}$	24*	2^{-43}		semi-circles/sec
IDOT	14*	2^{-43}		semi-circles/sec
<p>* Parameters so indicated shall be two's complement, with the sign bit (+ or -) occupying the MSB;</p> <p>** See Figure 20-1 for complete bit allocation in subframe;</p> <p>*** Unless otherwise indicated in this column, effective range is the maximum range attainable with indicated bit allocation and scale factor.</p>				

Figure 8 - Extrait de la documentation officielle du système GPS explicitant les paramètres des éphémérides

Il est *important de noter* que les paramètres dont les unités sont des demi-cercles, doivent être multipliés par π pour être correctement utilisés dans le calcul de la position des satellites.

Les paramètres éphémérides sont extrait et traités par l'outil :

```
self.svid = eph_raw.svid
self.tgd = twos_comp(int(eph_raw.sf1.word7.bin[16:], 2), 8) * (2 ** -31)
self.iode = int(eph_raw.sf1.word3.bin[22:] + eph_raw.sf1.word8.bin[:8], 2)
self.toc = int(eph_raw.sf1.word8.bin[8:], 2) * (2 ** 4)
self.af2 = twos_comp(int(eph_raw.sf1.word9.bin[:8], 2), 8) * (2 ** -55)
self.af1 = twos_comp(int(eph_raw.sf1.word9.bin[8:], 2), 16) * (2 ** -43)
self.af0 = twos_comp(int(eph_raw.sf1.word10.bin[:22], 2), 22) * (2 ** -31)
self.iode = int(eph_raw.sf2.word3.bin[:8], 2) # Found in two different subframes for some reason ...
self.crs = twos_comp(int(eph_raw.sf2.word3.bin[8:], 2), 16) * (2 ** -5)
self.delta_n = twos_comp(int(eph_raw.sf2.word4.bin[:16], 2), 16) * (2 ** -43) * math.pi
self.m0 = twos_comp(int(eph_raw.sf2.word4.bin[16:] + eph_raw.sf2.word5.bin, 2), 32) * (2 ** -31) * math.pi
self.cuc = twos_comp(int(eph_raw.sf2.word6.bin[:16], 2), 16) * (2 ** -29)
self.e = int(eph_raw.sf2.word6.bin[16:] + eph_raw.sf2.word7.bin, 2) * (2 ** -33)
self.cus = twos_comp(int(eph_raw.sf2.word8.bin[:16], 2), 16) * (2 ** -29)
self.sqrt_a = int(eph_raw.sf2.word8.bin[16:] + eph_raw.sf2.word9.bin, 2) * (2 ** -19)
self.toe = int(eph_raw.sf2.word10.bin[:16], 2) * (2 ** 4)
self.cic = twos_comp(int(eph_raw.sf3.word3.bin[:16], 2), 16) * (2 ** -29)
self.omega0 = twos_comp(int(eph_raw.sf3.word3.bin[16:] + eph_raw.sf3.word4.bin, 2), 32) * (2 ** -31) * math.pi
self.cis = twos_comp(int(eph_raw.sf3.word5.bin[:16], 2), 16) * (2 ** -29)
self.i0 = twos_comp(int(eph_raw.sf3.word5.bin[16:] + eph_raw.sf3.word6.bin, 2), 32) * (2 ** -31) * math.pi
self.crc = twos_comp(int(eph_raw.sf3.word7.bin[:16], 2), 16) * (2 ** -5)
self.omega = twos_comp(int(eph_raw.sf3.word7.bin[16:] + eph_raw.sf3.word8.bin, 2), 32) * (2 ** -31) * math.pi
self.omega_dot = twos_comp(int(eph_raw.sf3.word9.bin, 2), 24) * (2 ** -43) * math.pi
self.idot = twos_comp(int(eph_raw.sf3.word10.bin[8:-2], 2), 14) * (2 ** -43) * math.pi
```

Figure 9 - Extrait du code de l'outil explicitant l'extraction des paramètres éphémérides

Avec ces paramètres éphémérides nous pouvons calculer la position des satellites dans le ciel en suivant la procédure décrite dans la documentation officielle du système GPS :

Table 20-IV. Elements of Coordinate Systems (sheet 1 of 2)	
$\mu = 3.986005 \times 10^{14} \text{ meters}^3/\text{sec}^2$	WGS 84 value of the earth's gravitational constant for GPS user
$\dot{\Omega}_e = 7.2921151467 \times 10^{-5} \text{ rad/sec}$	WGS 84 value of the earth's rotation rate
$A = (\sqrt{A})^2$	Semi-major axis
$n_0 = \sqrt{\frac{\mu}{A^3}}$	Computed mean motion (rad/sec)
$t_k = t - t_{oc}^*$	Time from ephemeris reference epoch
$n = n_0 + \Delta n$	Corrected mean motion
$M_k = M_0 + nt_k$	Mean anomaly
$M_k = E_k - e \sin E_k$	Kepler's Equation for Eccentric Anomaly (may be solved by iteration) (radians)
$v_k = \tan^{-1} \left\{ \frac{\sin v_k}{\cos v_k} \right\}$	True Anomaly
$= \tan^{-1} \left\{ \frac{\sqrt{1-e^2} \sin E_k / (1 - e \cos E_k)}{(\cos E_k - e) / (1 - e \cos E_k)} \right\}$	
$E_k = \cos^{-1} \left\{ \frac{e + \cos v_k}{1 + e \cos v_k} \right\}$	Eccentric Anomaly
$\Phi_k = v_k + \omega$	Argument of Latitude
$\delta u_k = c_{us} \sin 2\Phi_k + c_{uc} \cos 2\Phi_k$ $\delta r_k = c_{rs} \sin 2\Phi_k + c_{rc} \cos 2\Phi_k$ $\delta i_k = c_{is} \sin 2\Phi_k + c_{ic} \cos 2\Phi_k$	Argument of Latitude Correction Radius Correction Inclination Correction
	Second Harmonic Perturbations
$u_k = \Phi_k + \delta u_k$	Corrected Argument of Latitude
$r_k = A(1 - e \cos E_k) + \delta r_k$	Corrected Radius
$i_k = i_0 + \delta i_k + (IDOT) t_k$	Corrected Inclination
$x_k' = r_k \cos u_k$ $y_k' = r_k \sin u_k$	Positions in orbital plane.
$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e) t_k - \dot{\Omega}_e t_{oc}$	Corrected longitude of ascending node.
$x_k = x_k' \cos \Omega_k - y_k' \cos i_k \sin \Omega_k$ $y_k = x_k' \sin \Omega_k + y_k' \cos i_k \cos \Omega_k$ $z_k = y_k' \sin i_k$	Earth-fixed coordinates.

Figure 10 - Extrait de la documentation officielle du système GPS explicitant les calculs à réaliser pour calculer la position des satellites

En utilisant la procédure décrite au-dessus on peut, en partant des éphémérides, calculer la position d'un satellite dans le ciel, dans le référentiel ECEF. On peut ensuite transformer cette position pour obtenir la latitude, la longitude et l'altitude. Il existe des ressources en ligne qui suivent les satellites en temps réel et qui permettent de vérifier si les données que nous obtenons sont bonnes.

Calcul de la position

Nous avons obtenu les éphémérides et les pseudodistances des satellites captés par le récepteur. Grâce à ces données, nous avons pu calculer la position des satellites dans le ciel. Il s'agit donc maintenant de choisir un minimum de quatre satellites pour calculer la position du récepteur.

Pour commencer, nous pouvons établir une relation entre la pseudodistance, la position du satellite, la position du récepteur et le biais de l'horloge du récepteur :

$$\begin{aligned}\rho_j &= \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + c\delta t_u \\ &= f(x_u, y_u, z_u, \delta t_u)\end{aligned}$$

ρ_j = pseudorange measurement from satellite j (m)

x_j, y_j, z_j = ECEF position of satellite j (m)

x_u, y_u, z_u = ECEF position of user (m)

δt_u = receiver clock error (sec)

On reconnaît la formule pour calculer la distance entre deux points dans un plan en trois dimensions. Comme nous ne connaissons pas la position du récepteur ni le biais de son horloge, nous avons une équation à quatre inconnus. On retrouve ici la nécessité d'avoir au minimum quatre satellite.

On peut alors établir un système d'équations avec les n satellites dont nous avons les données :

$$\begin{aligned}\rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + c\delta t_u \\ \rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + c\delta t_u \\ \rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + c\delta t_u \\ &\vdots \\ \rho_n &= \sqrt{(x_n - x_u)^2 + (y_n - y_u)^2 + (z_n - z_u)^2} + c\delta t_u\end{aligned}$$

Dans le but de calculer le vecteur position du récepteur :

$$\mathbf{X} = \begin{bmatrix} x_u \\ y_u \\ z_u \\ c\delta t_u \end{bmatrix}$$

Il existe plusieurs méthodes pour résoudre ce système d'équations. Ici, nous allons le résoudre par linéarisation et itération grâce à la méthode des moindres carrés.

Pour cela, on part d'un point fictif, une sorte d'estimation de la position du récepteur (qui en réalité peut juste être le centre de la Terre) et on peut réécrire le vecteur position de la sorte :

$$\mathbf{X}_u = \hat{\mathbf{X}}_u + \Delta\mathbf{X}_u$$

Où la position est égale à la position du point fictif additionné à la différence entre la position réelle et la position du point fictif. Dans l'outil, nous avons définis de manière arbitraire la position de la Tour Eiffel comme point de départ.

On peut alors réécrire les relations de la sorte :

$$\begin{aligned}\hat{\rho}_j &= \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2} + c\delta\hat{t}_u \\ &= f(\hat{x}_u, \hat{y}_u, \hat{z}_u, c\delta\hat{t}_u)\end{aligned}$$

$$f(x_u, y_u, z_u, c\delta t_u) = f(\hat{x}_u + \Delta x_u, \hat{y}_u + \Delta y_u, \hat{z}_u + \Delta z_u, c\delta\hat{t}_u + \Delta c\delta t_u)$$

On peut ensuite applique le développement en série de Taylor pour obtenir le système suivant :

$$\begin{aligned}\Delta\rho_1 &= a_{x1}\Delta x_u + a_{y1}\Delta y_u + a_{z1}\Delta z_u - \Delta c\delta t_u \\ \Delta\rho_2 &= a_{x2}\Delta x_u + a_{y2}\Delta y_u + a_{z2}\Delta z_u - \Delta c\delta t_u \\ \Delta\rho_3 &= a_{x3}\Delta x_u + a_{y3}\Delta y_u + a_{z3}\Delta z_u - \Delta c\delta t_u \\ &\vdots \\ \Delta\rho_n &= a_{xn}\Delta x_u + a_{yn}\Delta y_u + a_{zn}\Delta z_u - \Delta c\delta t_u\end{aligned}$$

Que l'on peut résoudre par calcul matriciel :

$$\Delta\mathbf{p} = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \vdots \\ \Delta\rho_n \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & -1 \\ a_{x2} & a_{y2} & a_{z2} & -1 \\ a_{x3} & a_{y3} & a_{z3} & -1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{xn} & a_{yn} & a_{zn} & -1 \end{bmatrix} \quad \Delta\mathbf{x} = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ \Delta c\delta t_u \end{bmatrix}$$

$$\Delta \rho = H \Delta x$$

Il suffit désormais de calculer Δx et l'additionner à la position du point fictif pour obtenir une estimation de la position du point réel. Il peut être nécessaire de réaliser plusieurs itérations de ce calcul en remplaçant le point fictif par le nouveau point obtenu, jusqu'à ce que le calcul converge. Cette méthode est relativement bien décrite dans un cours du International Centre for Theoretical Physics : [Calculation of GPS PNT Solution](#).

Résultats

Nous avons utilisé les cartes de Google Maps pour placer les points de position obtenus et faire des comparaisons :



Figure 11 - Extrait de carte avec différents points de position

Légende de la carte :

Point violet --> position donnée par le module u-blox en utilisant ses algorithmes propriétaires.

Point orange --> position obtenue par calcul sur les données brutes.

Point marron --> position correcte estimée.

On remarque aisément que la position obtenue par calcul se trouve à une distance d'environ 14 mètres et est moins précise que la position donnée par le module. Ceci est dû à différents facteurs qui influent sur la précision de la position, comme :

- La dilution de précision
- Les trajets multiples du signal satellite
- L'algorithme de calcul de précision utilisé est primitif
- Le traitement des données, en particulier les corrections appliquées, est insuffisant

On peut également observer les variations dans la position au cours du temps :



Figure 12 - Extrait de carte avec différents points de position prise au cours du temps

On remarque une grande variation dans les positions obtenues, certaines sont très précises, d'autres sont très imprécises. L'emplacement de l'antenne, sur l'appui d'une fenêtre peut influencer les mesures car une grande partie des satellites n'est pas directement visible, or une bonne répartition géographique des satellites au-dessus du récepteur est cruciale pour atténuer la dilution de précision.

Nous avons donc fait des mesures en étant éloigné des bâtiments :

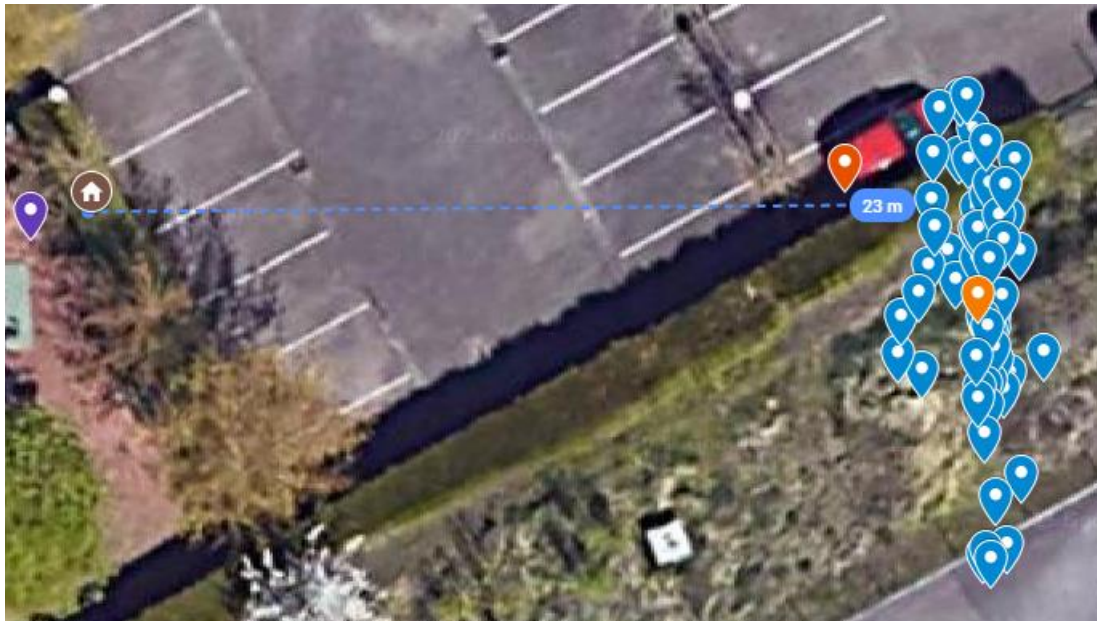


Figure 13 - Extrait de carte avec différents points de position prise au cours du temps

On peut remarquer ici que même si la distance entre la position estimée et la position calculée est plus grande, il y a plus de régularité dans les positions calculées au cours du temps.

Conclusion

Dans le cadre de ce travail, nous avons créé un outil capable d'établir une connexion série avec un module GNSS de u-blox, envoyer des requêtes en suivant le protocole UBX et parser les données reçues. Nous avons utilisé cet outil pour obtenir les éphémérides et les pseudodistances des satellites visibles par le module, pour ensuite calculer leur position dans le ciel. Avec les positions des satellites et les pseudodistances nous avons pu calculer la position du module.

La position que nous obtenons par calcul est moins précise que celle donnée par le récepteur en utilisant ses algorithmes propriétaires.

Pour la suite de ce travail, il serait intéressant de tenter d'améliorer le résultat obtenu avec un meilleur traitement des données reçues, notamment en réfléchissant à comment choisir les meilleurs satellites à utiliser dans le calcul de position. Il peut également être intéressant de tenter d'appliquer un autre algorithme de calcul de position et comparer les résultats.

Bibliographie

[Librairie pyubx2, GitHub](#)

[u-blox 6 Receiver Description, u-blox](#)

[u-blox 8 Receiver Description, u-blox](#)

[IS-GPS-200H, GPS Directorate](#)

[Calculation of GPS PNT Solution, International Centre for Theoretical Physics](#)

[GPS Accuracy: HDOP, PDOP, GDOP, Multipath & the Atmosphere, GISGeography](#)