# 1. Présentation

Dans ce TP vous allez vous familiariser avec le protocole AX25. Il s'agit d'un protocole de couche 2 (liaison) utilisé par les radioamateurs pour mettre en œuvre des transmissions numériques radio. Il est également utilisé dans les satellites radio-amateurs et universitaires.

L'objectif du TP est de découvrir le fonctionnement du protocole AX25 pour les transmissions satellites, de programmer le codage et le décodage du protocole et de tester avec des données reçues de divers nanosatellites.

Vous devrez mettre en œuvre un logiciel permettant de décoder les données.

Sur Moodle on donne des fichiers correspondant à des données AX25 issues de plusieurs satellites. On donne en annexe des éléments de programmation pour faciliter votre travail.

Dans une seconde partie vous devrez pouvoir décoder les données de charge utile émises par le satellite Robusta 1B de l'université. On donne en annexe le codage des données de la charge utile et de la télémétrie du satellite.

# 2. Protocole AX25

a) Écrire le programme en Python qui lit les données des trames AX25 issue d'un satellite depuis un fichier texte. Le fichier texte est compatible avec le format utilisé sur le site satnogs.org (voir question suivante) : chaque octet est représenté en hexadécimal, codé en ASCII, séparée par un espace. Votre programme doit décoder et afficher l'entête AX25.

On donne sur moodle quelques fichiers que vous pouvez utiliser.

# 3. Récupération de données

Le site https://satnogs.org est un site collaboratif de suivi de satellites. Il met en œuvre une base de données participatives des observations (onglet DB).
Après avoir choisi un satellite vous pouvez accéder aux relevés des observateurs, notamment les trames reçues. Parfois celles-ci sont inexploitables, partielles ou entachées d'erreurs. Tous les satellites n'utilisent pas AX25. Pour ce TP on se limitera donc aux trames sans erreurs de satellites utilisant AX25. Pour ceux-là le site affiche l'entête AX25 décodée.

Vous pouvez, par exemple, utiliser les trames des satellites suivants : UVSQ-SAT, TAUSAT-1,

CIRBE, RoseyCubesat-1, ROBUSTA-1B

a) Sur plusieurs exemples, vérifiez que votre programme fonctionne.

# 4. Décodage des données du satellite Robusta-1B

Vous trouverez en annexe un extrait du format des données encapsulées dans la trame AX25 pour le satellite Robusta-1B. Ce satellite embarque plusieurs expériences de mesures de l'effet des rayons cosmique sur des composants électroniques.
Pour visualiser ces données vous pouvez rejoindre l'onglet de la mission Robusta-1B du MCC de la machine virtuelle du segment sol (vgseg).

a) Selon vous, quelles sont les données de télémétrie du satellite ?
b) Que représentent les données de télémétrie ?
c) A quoi servent les données des événements passés ?
d) Compléter le programme précédent pour afficher les données de télémétrie du satellite.
e) Compléter le programme pour qu'il affiche quelques données de la charge utile.

La machine virtuelle du segment sol synchronise sa base de données des missions avec le site satnogs.org.

f) En récupérant les dernières données AX25 sur satnogs.org, vérifier le fonctionnement de votre programme.
g) Modifier votre programme pour qu'il récupère automatiquement les dernières données du satellite Robusta-1B sur le site satnogs.org et les affiche.

# 5. Uplink

L'annexe précise le fonctionnement des trames en uplink.

a) Produire un programme qui génère la trame montant permettant de réduire ou de baisser la puissance d'émission du satellite.
b) Produire un programme qui génère la trame montant qui permet de fixer les seuils de tension de la batterie.

# 6. Annexe

## AX25 et KISS

Le protocole AX.25 (Amateur X.25) est un protocole de communication utilisé principalement pour les communications de données numériques dans le domaine des radioamateurs et la télémétrie spatiale. Il est également utilisé dans les réseaux de messagerie radioamateur et les réseaux de données d'urgence.
Il est basé sur le protocole X.25, utilisé pour les réseaux de paquets publics, avec des adaptations spécifiques pour les transmissions radio.

Le protocole AX.25 fournit un service de niveau 2 (liaison). Il a été conçu pour fonctionner sur des

liens radio à faible débit et pour être résistant aux erreurs de transmission.

Les données à transmettre sont d'abord divisées en paquets de taille appropriée. Chaque paquet contient des informations sur la source, la destination, le type de paquet et les données elles-mêmes.

Le protocole AX.25 prend en charge le routage des paquets entre plusieurs stations. Les paquets peuvent être transmis à travers un réseau de stations intermédiaires avant d'atteindre leur destination finale.

Lorsque la communication se limite au lien satellite/TNC (Terminal Node Controler) le protocole KISS est utilisé (http://www.ax25.net/kiss.aspx) et, au final, on se limite à encapsuler les données de charge utile dans une seule trame AX25.

## Python

Ouvrir et lire dans un fichier :

```
with open("dump.ax25", "r") as f:
    # Lecture du contenu du fichier
    content = f.read()
```

Transformer la chaîne lue en tuple d'entiers (octet) :

```
trame=[]
    # Boucle pour traiter chaque paire de caractères hexadécimaux
for i in range(0, len(content), 3):
    # Conversion de la paire de caractères en un octet
    trame.append(int(content[i:i+2], 16))
```

Transformer un entier en caractère (code ASCII) : `c=chr(v)`

Transformer un tuple d'octets en entier non signé en considérant un codage little endian :

```
v=int.from_bytes(t,byteorder='little',signed=False)
```

Transformer un tuple d'octets en « timestamp », en considérant un codage little endian :

```
ts=datetime.fromtimestamp(int.from_bytes(t, byteorder='little'))
```

## En tête AX25

« Le champ adresse de toutes les trames doit être codé avec les indicatifs radioamateur source et destination (F4KLE pour l'IUT). Mis à part l'identificateur secondaire de station (SSID: Secondary Station IDentifier) , le champ adresse doit être composé seulement de caractères alphanumériques ASCII. Si des répéteurs de niveau 2 doivent être utilisés, leurs indicatifs devront également apparaître dans le champ adresse.

Le champ adresse HDLC est étendu au delà d'un octet en assignant au bit de poids faible de chaque octet le rôle de "bit d'extension". Le bit d'extension de chaque octet est positionné à zéro afin

d'indiquer que l'octet suivant contient une information supplémentaire d'adresse , ou bien à un pour indiquer qu'il s'agit du dernier octet du champ adresse. Pour libérer l'emplacement du bit d'extension, le codage de l'indicatif radioamateur est décalé d'un rang vers la gauche. »

La version 2.0 d'AX25 a mis en place l'information de commande/réponse dans le champ adresse. Afin de maintenir la compatibilité avec les versions précédentes d'AX.25, l'information commande/réponse est transportée en utilisant 2 bits.

Un DXE AX.25 ayant une compatibilité ascendante peut déterminer s'il communique avec un DXE qui utilise une ancienne version du protocole en testant le bit de commande/réponse situé dans le bit 7 du SSID des champs adresse source et destination. Si les deux bits C sont à zéro, le système utilise l'ancien protocole. La nouvelle version de protocole a toujours un de deux bits à la valeur un et l'autre à la valeur zéro, ceci dépendant du fait que la trame soit une réponse ou une commande.

```
# Partial AX25 packet (header+data), since the start/end
    # flags (0x7E), FCS (CRC) and low level AX25 processing (ie. bitstuffing)
    # are removed internally by the TNC.

    # +--AX25 Header-----------------------------------------------+----------------------------+
    # |                   ADDRESS                | CONTROL |  PID   |          AX25 INFO         |
    # |------------------------------------------------------------|----------------------------|
    # | Destination | D.SSID | Source  | S.SSID |         |        |                            |
    # | 6 Bytes     | 1 Byte | 6 Bytes | 1 Byte | 1 Byte  | 1 Byte |        N * Bytes           |
    # |             | 0x60   |         | 0x61   | 0x03    | 0xF0   |  ROB-1C Application Layer |
|
    # +------------------------------------------------------------+----------------------------+
    # Addresses are composed of CAPITAL LETTERS (ASCII) in HEX, left shifted one bit. (Ref. AX25
spec)
    # TERMINAL   VALUE (ASCII)   VALUE (HEX)           VALUE TRANSMITTED
    # SAT        "FX6FRA"        46 58 36 46 52 41     8C B0 6C 8C A4 82

#Le D-SSID est de la forme CRRSSID0 et identifie la sous-station d'émission soit 0x60 pour la #0
#Le S-SSID est de la forme cRRSSID1 et identifie la sous-station destinataire soit 0x61 pour la #0
# C et c = bit de commande/réponse AX25v2, RR = 2 bits réservés à 1, SSID le 4 bits pour le SSID
```

## Quelques données du satellite ROBUSTA-1B

| Nom | Type | 1er octet | taille | Conversion | Unité |
|---|---|---|---|---|---|
| type_de_trames | U int | 0 | 1 | LUT | |
| beacon_timestamp | timestamp | 1 | 4 | datetime | date |
| Distri_exp_1_OBDHS | U int | 5 | 1 | On=255, Off=0 | |
| Distri_init_1_OBDHS | U int | 6 | 1 | On=255, Off=0 | |
| Distri_distri_osl_OBDHS | U int | 9 | 1 | On=255, Off=0 | |
| Gain_osl_OBDHS | U int | 10 | 1 | H=255, M=240, L=0 | |
| Timer_puissance_OBDH | U int | 11 | 1 | | |

| | | | | | |
|---|---|---|---|---|---|
| Time_Tx_OBDH | U int | 12 | 1 | | |
| Time_Temp_OBDH | U int | 13 | 1 | | |
| Time_Dose_OBDH | U int | 14 | 2 | | |
| Time_Exp_OBDH | U int | 16 | 2 | | |
| Distri_exp_1_payloads | U int | 18 | 1 | On=255, Off=0 | |
| Distri_init_1_payloads | U int | 19 | 1 | On=255, Off=0 | |
| Distri_distri_osl_payloads | U int | 22 | 1 | On=255, Off=0 | |
| Gain_osl_payloads | U int | 23 | 1 | H=255, M=240, L=0 | |
| var_Iccp_LM124_exp1_payload | U int | 24 | 2 | 5/(1024*5,64) | mA ? |
| var_Iccm_LM124_exp1_payload | U int | 28 | 2 | 5/(1024*5,64) | mA ? |
| var_Iinp_LM124_exp1_payload | U int | 32 | 2 | $5.10^6$ /(8200 *4096) | mA ? |
| var_Iinm_LM124_exp1_payload | U int | 36 | 2 | $5.10^6$ /(8200 *4096) | mA ? |
| var_vsh1_LM124_exp1_payload | U int | 40 | 2 | 5/1024 | V |
| var_vsh2_LM124_exp1_payload | U int | 42 | 2 | 5/1024 | V |
| var_vsh3_LM124_exp1_payload | U int | 44 | 2 | 5/1024 | V |
| var_vsh4_LM124_exp1_payload | U int | 46 | 2 | 5/1024 | V |
| | | | | | |
| var_temp_1_payload | S int | 120 | 2 | 0,25 | °C |
| var_moy_temp_1_payload | S int | 122 | 2 | 0,25 | °C |
| var_ecart_type_temp_1_payload | S int | 124 | 2 | 0,02 | °K |
| var_cste_vbat_max | U int | 138 | 2 | 0,004 | V |
| var_cste_vbat_min | U int | 140 | 2 | 0,004 | V |
| var_cste_vbat_moy | U int | 142 | 2 | 0,004 | V |
| var_cste_pbat_max | U int | 144 | 2 | 2 | mW ? |
| var_cste_ibat_moy | U int | 146 | 2 | 2 | mA ? |
| var_cste_pbat_moy | U int | 148 | 2 | 2 | mW ? |
| | | | | | |
| event_code_1 | U int | 176 | 1 | | |
| event_timestamp1 | timestamp | 177 | 4 | datetime | date |
| event_data_1 | U int | 181 | 3 | | |

U int : unsigned int

S int : signed int
timestamp : nombre de seconde depuis le 1/1/1970 (U int sur 4 octets). Sert à horodater des événements. Fonctionne entre 1970 et 2038.

## Quelques commandes du satellite ROBUSTA-1B

En uplink le format des trames est donné ci-dessous. A l'émission il faut ajouter un octet de préambule (0x7E) et deux octets de CRC voir CRC-CCITT dans la norme ISO 3309 : http://practicingelectronics.com/articles/article-100003/article.php

```
# +---AX25 TC COMMAND-----------------------------------------------------+
# |  HEADER     | ROB1B command frame                                     |
# |-------------------------------------------------------------------------|
# | AX25 Header | Frame Length | Frame Type | Time stamp | TC code | params |
# | 16 bytes    | 1 byte       | 1 byte     | 4 bytes    | 1 byte  | N bytes|
# +-------------------------------------------------------------------------+
```

```
# +---AX25 PF COMMAND-----------------------------------------------------------------------------+
# |  HEADER     | ROB1B command frame                                                             |
# |-------------------------------------------------------------------------------------------------|
# | AX25 Header | Frame Length | Frame Type | PF code | Time Stamp | Nb of CAN | params  | CRC prm |
# | 16 bytes    | 1 byte       | 1 byte     | 1 byte  | 4 bytes    | 1 byte    | N bytes | 1 byte  |
# +-------------------------------------------------------------------------------------------------+
```

Les codes TC sont les suivants :

| **Commande** | **TC code** |
|---|---|
| TC : TTC Internal command | 0x01 |
| TM : Telemetry File Dump | 0x07 |
| ACK : Reply message | 0x0B |
| TM : Beacon | 0x10 |
| TM : Telemetry Dump Send Request | 0x15 |
| TC : Platform Telecommands | 0x1C |

La commande TC_TTC_CHANGE_POWER permet de demander au satellite d'augmenter ou de diminuer sa puissance d'émission. Cette commande étant un télécommande interne, le frame type est 0x01. Le « TC code » vaut 33h, il n'y a qu'un octet de donnée qui vaut 0x11=Decrease Power ou 0x66=Increase Power
Exemple : `8cb06c8ca484e08c689694b040e103f006`**`01`**`2206cb60`**`3311`**`10a2`

La commande TC_PF_EPS_VOLT_BAT_THRESHOLD permet de modifier les seuils de supervision des tensions de la batterie du satellite. Il y a 3 seuils bas (TH1L à TH3L) et 2 seuils hauts (TH1H et TH2H). Ils sont codés chacun sur un octet, le quantum est de 20mV. Cette commande est une commande pour la plateforme, le frame type est donc 0x1C. Le «PF code » vaut 0x60 et il y a Nb of CAN vaut 2. Le checksum est 2a il est calculé avec l'algorithme Dallas Maxim.
Exemple : `8cb06c8ca484e08c689694b040e103f010`**`1c53`**`d80acb60`**`02`**`01020304405`**`2a`**`230e`