Décodage d'Images satellites:

Proposition par: Vincent Perrier 24 septembre 2019 @ 00:52

Pourquoi ?

Des satellites passent au dessus de vos têtes tous les jours. Ces satellites ont le mandat de faire une diffusion d'images haute résolution de la surface et de l'atmosphère planétaire. Ces images sont utilisé par les météorologistes comme information pour leurs analyses du climat. Des sytèmes d'alertes précoces aux sinistres dépendent de ces photos. Environment Canada utilise des logiciels avancés et propriétaires pour obtenir ces données.

Une partie importante du système est le décodage. Certaines personnes le font avec Python et GNU-Radio. Le problème avec cette méthode est le temps réel. GNU radio ne supporte que l'analyse de fichiers de dompage. Une autre manière est l'utilisation de WXtolmg. C'est un logiciel propriétaire daté qui est utilisable seulement après avoir appliqué un key-gen. C'est parce que la compagnie n'existe plus...

Bref, c'est une infrastructure absolument fantastique en manque désespéré d'open-source. C'est aussi une merveilleuse introduction au monde du Software Defined Radio (SDR). Ce monde contient les radios amateurs et les télécommunications sans-fil. Si vous aimez le open-source et l'idée de descendre plus près du matériel, ce projet est pour vous !

Comment ?

Les satellites météos se nomment NOAA. Il y en a présentement 3 en opération, en ce moment même. Ils diffusent constamment un signal FM décrivant les images. Ces diffusions sont non encryptés et libre à tous d'intercepter. Selon moi le projet se déroulerai un peu comme ceci:

- Acquérir un module RTL-SDR. (disponible sur amazon pour ~20$).
- Acquérir une plateforme de traitement. E.g Raspberry Pi. Des ports USB sont essentiels.
- Avec l'API open-source rtl-sdr, créer un démodulateur FM. L'API est disponible en plusieurs languages comme C/C++, rust etc...
- Une fois que nous avons trouvé une manière fiable de convertir les ondes FM en séries de bits, il faut implémenter le protocole ATP. Ce protocol est la recette de convertion entre une séries de bits et un fichier d'image.
- Créer une architecture de site web pouvant télécharger les photos capturés de plusieures stations ou postes d'écoutes.
- Implémentation d'un service d'analyse de photo et de prédiction météorologiques
- Adapter l'architecture pour pouvoir recevoir toutes sortes de transmissions radio à partir de plusieurs stations différentes.

Buts

Même si nous regardons le cosmos avec envie, il faut garder les pieds au sol.

But initial

Le premier objectif est de pouvoir obtenir des images à l'aide de logiciel open-source et temps réel. Par temps-réel, on entend un modèle de streaming vers un serveur central. Ceci est différent de l'approche de GNU-Radio. GNU-radio fait des écritures répétitives dans un fichier donné. De manière évidente, cela peut apporter des problèmes de performances.

Par intérêt personnel envers le language, j'aimerais pouvoir implémenter le démodulateur FM et le décodeur ATP en language Rust. C'est bien supporté sur les Raspberry Pi de divers générations et assez performant pour nos besoins. De plus, cela donnerais une bonne raison aux participants de se mettre à apprendre le language plus sérieusement.

Améliorations futures

Dans un monde idéal, les divers étapes de traitement de donnée seraient indépendantes. On pourrait avoir des stations d'écoutes servant uniquement à streamer les données à une unité centrale. Cette unité aurait alors comme mandat de faire le décodage. Pour l'instant ce n'est rien de concret, plus à suivre et plus à discuter dans le futur.

À l'ordre du jour lors de la rencontre de mercedi 25 septembre 2019:

Voici quelques questions que j'aimerais vous poser par rapport au projet: * Est-ce une bonne idée ? * Est-ce que ce projet semble aligné avec le but du club ? * Le but initial vous semble-t-il réaliste ? * Aimez-vous la perspective d'analyse d'image après l'implémentation de l'infrastructure ? * Pensez-vous que le AI pourrait être introduit là-dedans ?

Plateformes d'intérêts potentiels

Par plateforme, on entend médium d'installation. La première plateforme me venant à l'esprit est le Raspberry Pi. Mais, il est absolument possible de faire un programme de décodage et de démodulation sur un PC. De plus, tant que nous avons accès à la librairie rtlsdr sous une forme ou l'autre il n'y a pas de limites. La plateforme choisie est donc très peu importante.

Toutefois, je ne recommande pas l'utilisation de microcontrolleurs (Arduino). Leurs basses vitesses de calculs est un facteur extrêmement limitant. De plus, la

majorité des micro-contrôleurs auront besoin d'un système extèrne de conversion USB->Port sériel. Finalement, ils n'ont pas de capacités de multi-threading (ne pas confondre avec des intérruptions). Ceci est un gros obstacle si on considère le besoin de futur de communiquation internet.

Information sur la famille rtlsdr

Le mot RTLSDR décrit une famille de dispositifs USB. Ces engins se basent tous sur le contrôleur realtek 2832u. Il existe plusieurs compagnies qui vendent des configurations différentes. Les variations sont principalement orientés autour du circuit de tuner. Ceci veut dire que la majorité des dispositifs fonctionneront avec l'API open-source (regardez le projet rtlsdr du groupe osmocom).

Cet API se base sur la librairie LibUsb-1.0. Ceci assure une compatibilité avec la majorité des systèmes Linux sur plusieurs architectures. Cet API est écrit en C. Il est toutefois possible de l'utiliser en C++ et en Rust.

Itérations

Le projet devra suivre quelques étapes importantes. On peut s'imaginer le projet en suivant le modèle des couches OSI.

- La version 0.1 représente la couche physique, liaison et réseau.
- La version 0.2 représente la couche transport.
- La version 0.3 représente la couche application.

Version 0.1

La première itération consiste à obtenir le matériel et à être capable de l'utiliser. Il faut donc obtenir un module RTLSDR et être capable de le contrôler. On peut obtenir plusieurs de ces modules pour une vingtaine de dollars sur Amazon. Tant qu'au contrôle, il suffit de produire un petit programme capable de contrôller des paramètres du module et obtenir des données de celui-ci.

Pour prouver que le programme fonctionne correctement, il graphera les samples obtenus auprès du module USB. De plus, il suffiera de faire rouler le programme de test sur un PC.

Selon ce site web, le signal reçu est un signal AM qui fût modulé dans un signal FM. Il faut donc tenir compte de cela dans la démodulation. De plus, le module USB capte des samples de tension en bits.

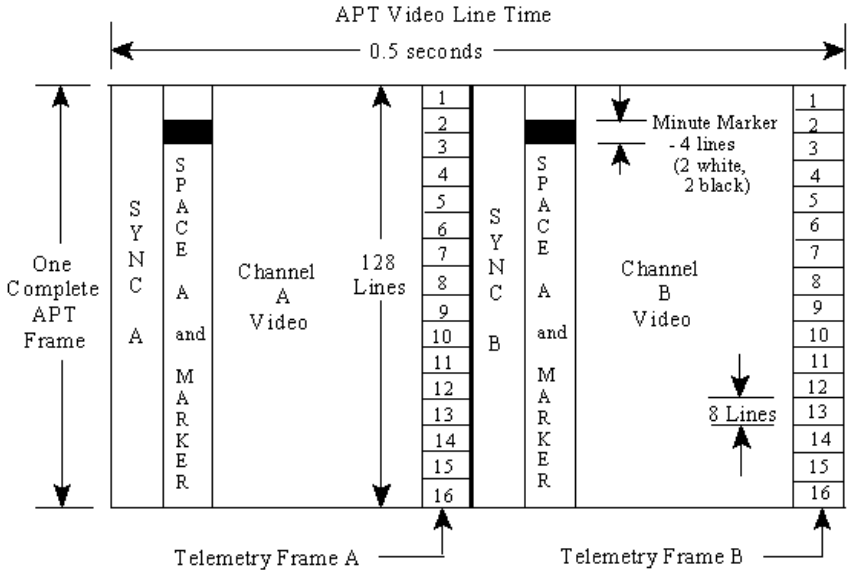
$$V_(signal) = \tag{1}$$

Version 0.2

La deuxième itération du projet se concentre sur la démodulation. En suivant la spécification APT qu'utilise les satellites de l'agence NOAA, on doit convertir les signaux radio en série d'octets. Pour se faire, il faut mettre la main sur le standard APT. Évidemment il y a plus de recherches à faire sur le sujet.

Concernant la plateforme, cette version peut aussi se contenter de fonctionner sur un PC.

Figure 4.2.2-1. APT Frame Format.



WEDGE #1	WEDGE #2	WEDGE #3	WEDGE #4	WEDGE #5	WEDGE #6	WEDGE #7	WEDGE #8
1	2	3	4	5	6	7	8
Zero Modulation Reference	Thermistor Temp. #1	Thermistor Temp. #2	Thermistor Temp. #3	Thermistor Temp. #4	Patch Temp.	Back Scan	Channel I.D. Wedge
9	10	11	12	13	14	15	16

Notes:
1) Each telemetry frames consists of 16 points
2) Telemetry frame rate is 1 frame per 84 seconds
3) Each telemetry point is repeated on 8 successive APT video lines

Figure 1: Image montrant le standard APT (https://en.wikipedia.org/wiki/Automatic_picture_transmission)

Version 0.3

Avec cette version, le programme doit être capable de convertir les octets reçus en format d'image.