blog.f0cks.net

DATA logger série avec une RaspberryPi

13-16 minutes

× **Info!** Ce projet est en cours! Il est mis à jour régulièrement!

Par F0cks le 13 Avril 2018. Màj: 14 Mai 2018

Déscription

J'ai eu la super bonne idée de faire un montage disposant d'*un GPS* et d'*un GSM*. Pour le debuguer, je dois prendre la voiture, mettre *un ordi sur le siège passager* et aller me balader. C'est déjà *contraignant* de mettre un ordi portable dans une voiture, mais *en moto, c'est carrément la galère!*

Ce projet a déjà quelques... années... de retard. Et je dois monter *très rapidement un DATA logger* pour enregistrer des données venant d'une UART. Les fonctions essentielles qu'il doit réaliser sont:

- *Enregistrer* les données venant d'une UART
- *Afficher* les données venant d'une UART
- Avoir une *autonomie* de quelques heures
- Pouvoir *récupérer les données facilement* sur un ordinateur

A quoi ressemble du monitoring sur moto...

Ma mère et ma copine me reprochent souvent de ne jamais rien jeter quand ça touche à l'électronique. Que j'accumule inutilement tout un tas de cartes et de composants qui ne serviront jamais à rien... C'est l'occas de *leurs donner tort* ♥ et de *faire du recyclage* d'anciens projets!

Composants

Carte principale: Raspberry Pi B+

Dans tout mon fourbi, j'ai presque tous les modèles de Raspberry Pi. Il me faut celle qui consome le moins parmis celles qui ont au moins 2 ports USB. J'ai donc récupéré une *Raspberry Pi B+*. Celle que je vais utiliser vient surement de chez Adafruit.

Cette carte a *4 ports USB*. Il m'en faut un pour *la communication série*, un pour le *stockage*. Elle consome à vide environ *1W*, la meilleure conso des Raspberries qui ont plus qu'un port.

En bonus, elle a un *port Ethernet* qui va rendre la programmation SSH

plus facile: pas besoin de configurer le WIFI. Elle est *surdimensionnée* pour faire un simple DATA logger: Processeur 700MHz, RAM 512Mo, OS Raspbian. La puissance, la gestion de fichiers et l'affichage seront *facilement et rapidement réalisés*. Les GPIO donnent accès à divers types de communication comme la *SPI*. On pourra *facilement utiliser un écran* tactile.

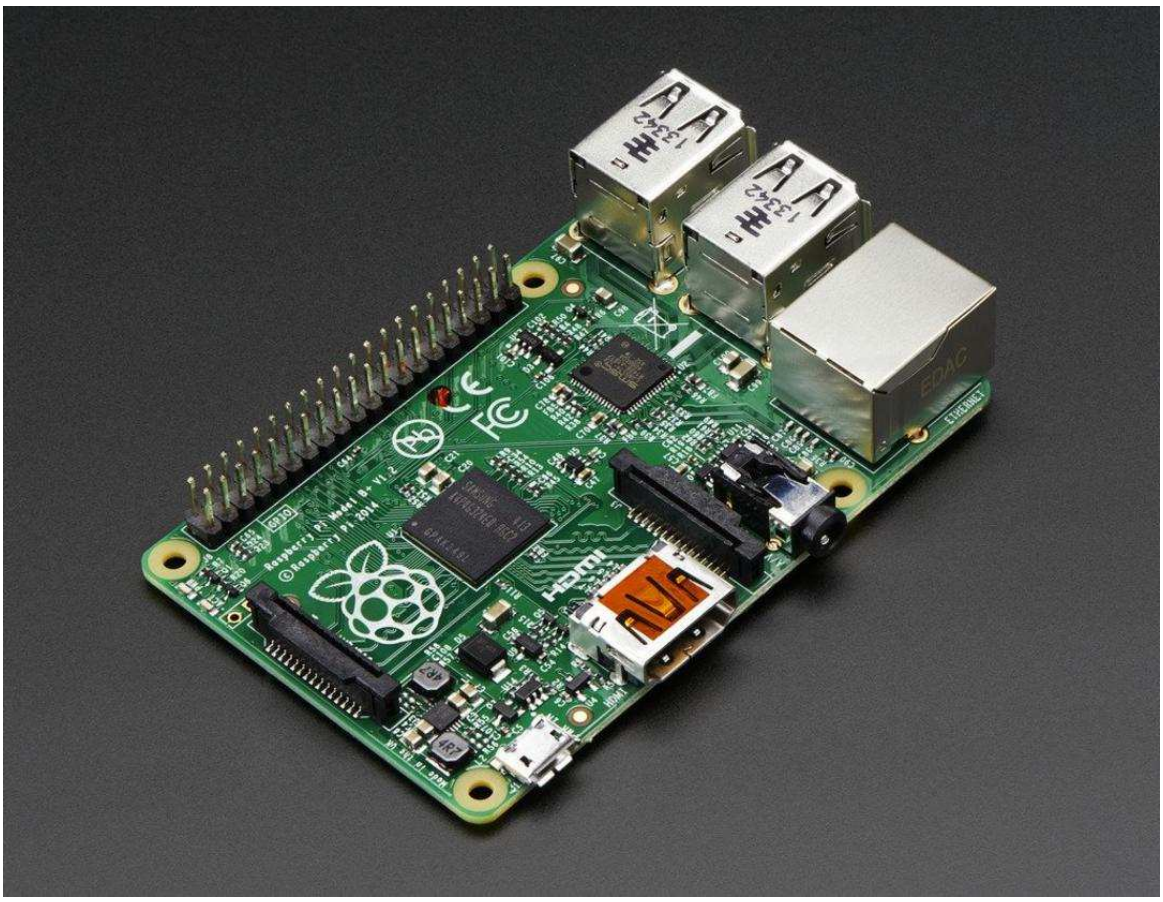


Photo Adafruit: Raspberry Pi B+

Affichage: Écran tactile PiTFT 3.5"

J'ai retrouvé un *écran tactile PiTFT 3,5"*, lui aussi acheté chez Adafruit, et qui n'a pas vraiment servi à grand chose...

Cette écran va me servir à *jeter un coup d'oeil aux trames* que je suis en train d'enregistrer. Je pourrai aussi *afficher quelques infos / erreurs*. L'écran tactile me servira de *bouton* pour sortir de veille.

Pour les caractéristiques, rien de fou. C'est un écran 16bits de 3,5" en 480x320. Il est associé à un écran tactile resistif. Le tout se plug directement sur les GPIO et utilise la SPI.

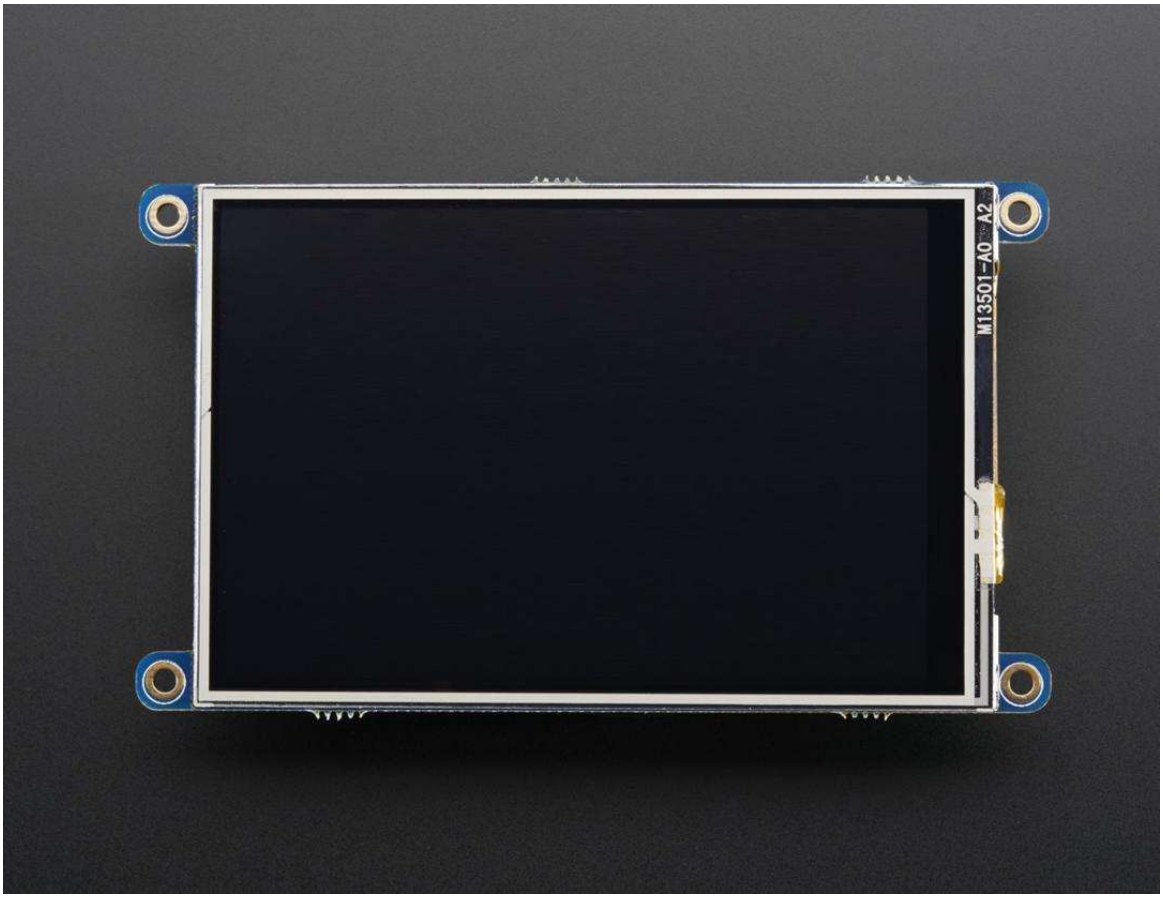


Photo Adafruit: PiTFT 3.5"

Communication série: Convertisseur série TTL vers USB

Pour varier un peu d'Adafruit, j'ai acheté ce cable sur Farnell.

Comme l'écran occupe tous les GPIO, et que je ne veux pas me casser la tête à souder sur les pads de la rasp des fils reliés à l'UART, j'utilse *un convertisseur TTL à USB*. Il va me permettre de convertir l'UART de la carte que je monitore en *port de communication* sur la raspberry.

Photo Adafruit: Convertisseur série TTL vers USB

Alimentation: Chargeur PowerBank Solaire SPBS 5000 B2

Le hasard fait bien les choses! J'ai besoin de faire un truc rapide et je vois une pub Lidl avec un chargeur solaire contenant *une batterie de 5000mAh*. Banco, je fonce à Lidl, je me trompe de jour, j'y retourne 3 jours après et voila! Une super batterie pour mon système.

Évidemment, le panneau solaire ne me sert à rien. Mais une option sympa de cette power bank est qu'*elle coupe l'alimentation lorsque le système qu'elle alimente consomme moins de 60mA pendant environ 10s*. Elle va donc couper toute seule l'énergie de la Raspberry quand elle s'éteind (Pour rappel, la raspberry continue de consommer quelques mA même éteinte. Elle ne peut pas couper seule son alimentation). Son *bouton ON/OFF* permettra de la rallumer! Plus besoin de brancher et débrancher le cable d'alimentation avec le risque de tuer la carte µSD.

Chargeur powerbank solaire SilverCrest

Stockage: Clé USB

Enfin, le dernier composant de ce système mais non le moindre: une clé USB! Alors cette fois ci, je suis bien incapable de dire d'où elle vient celle là...

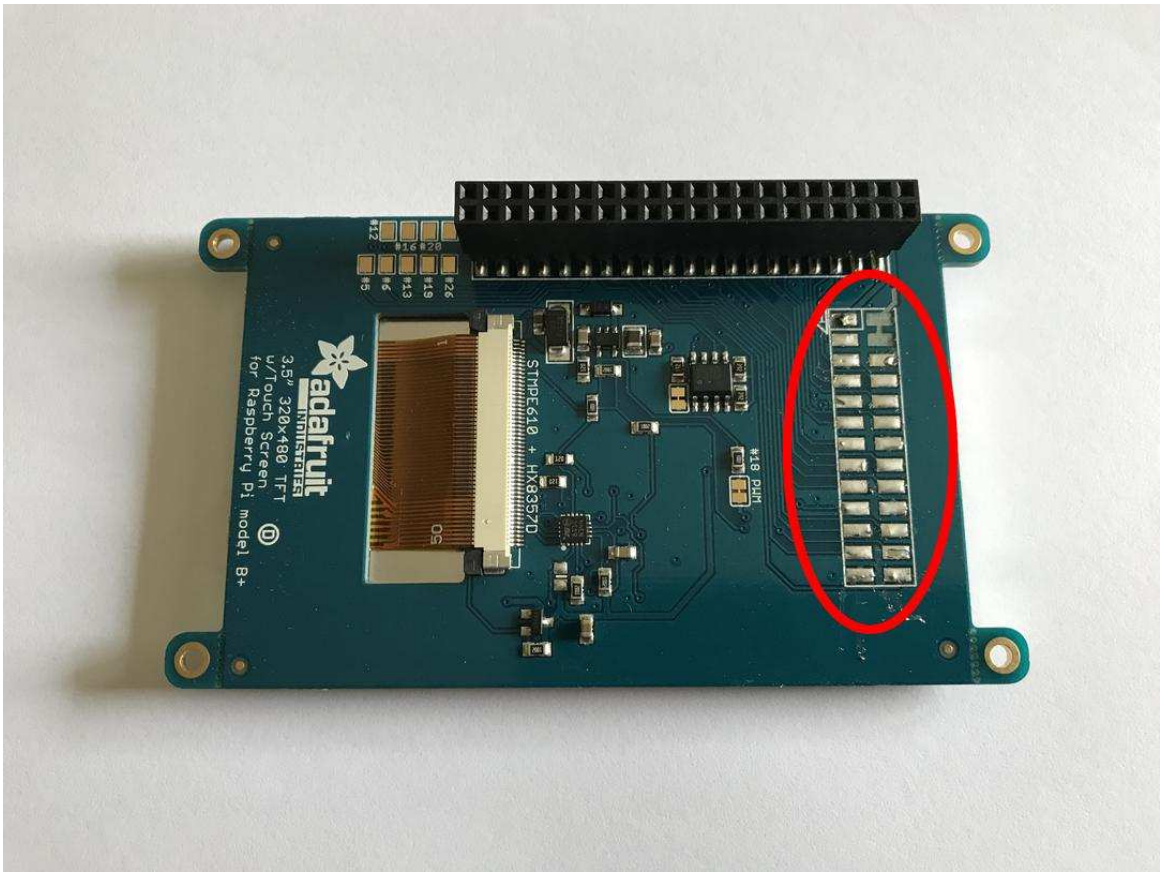
Que dire, c'est une clé USB Verbatim de 8Go, le modèle nano. *Les données seront enregistrées directement dessus*. Je n'aurai plus qu'à *la pluguer sur mon ordinateur* pour pouvoir traiter facilement des heures et des heures de data logging!

Clé USB Verbatim

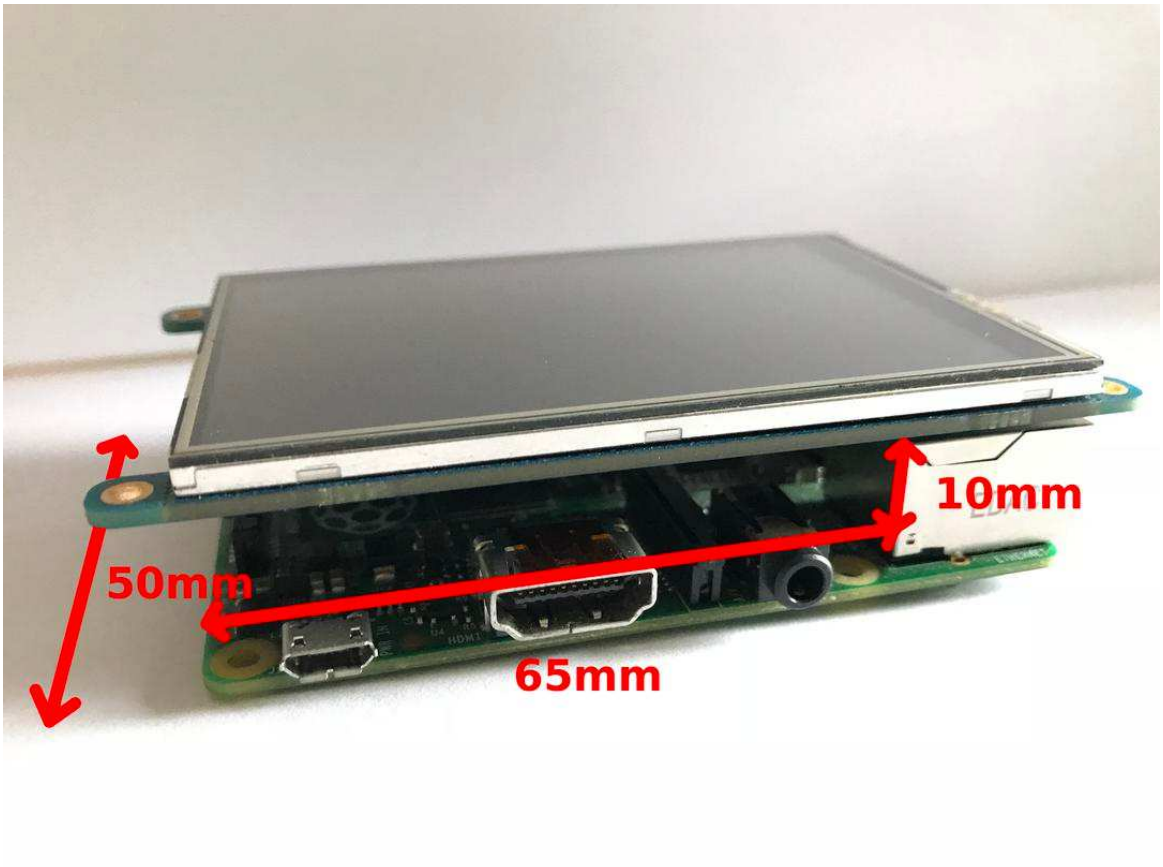
Réalisation

Assemblage

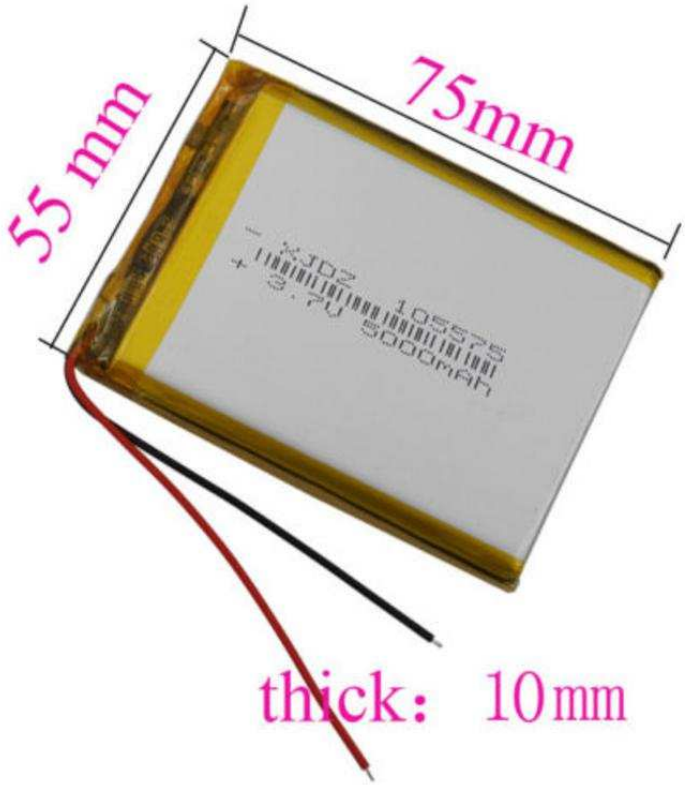
En assemblant l'écran avec la Raspberry, j'ai remarqué un espace conséquent entre les deux. Surtout en enlevant un connecteur de l'écran que je n'utilise pas. Je me retrouve avec *un volume libre de 50x65x10*. Pas beaucoup de jeu pour l'épaisseur, mais on peut déborder un peu sur la longueur et la largeur. Du coup, je me suis acheté une batterie lipo sur Ebay de *5000mAh de 55x75x10* pour rendre le système plus compact en enlevant la grosse power bank. Elle possède déjà une carte de protection contre les surtensions, les court-circuits etc... dont je vérifirai le fonctionnement vu que le tout vient de Chine... Je n'aurai plus qu'à ajouter un petit système de boost et de chargeur. Enfin je ferai tout ça plus tard... Quand j'aurai le temps!



Modification de l'écran



Espace libre après assemblage



Batterie lipo 5000mAh

Programmation

J'ai tout d'abord cloné la [version lite de Raspbian Stretch] sur une µSD de 8Go class 10. J'ai ensuite fait les opérations essentielles par SSH:

- [Intaller le PiTFT] (en sélectionnant bien le mode console et pas le mode bureau)
- [Eteindre le PiTFT lors du shutdown de la Rasp]
- [Mise en place un système de montage automatique des clés USB]
- [Désactiver l'économiseur écran en mode console] (je me charge de cette option moi même dans mon code)

J'ai ensuite créer un code python [dataLogger.py] dans le dossier */home/pi*. J'ai rendu son excution automatique lorsque l'utilisateur *Pi* est connecté:

```
echo "python /home/pi/datalogger.py" >> /home/pi
/.bashrc
```

Comment fonctionne mon code?

- Au démarrage, *il coupe le HDMI et les LEDs* pour gagner un peu en consommation
- Il attend qu'*un convertisseur TTL/USB soit connecté*, s'il ne l'est pas *dans les 60s, la raspberry s'éteint*
- Il attend ensuite qu'*une clé USB soit connectée*
- Il crée *un fichier data_X.log* où X est un nombre qui s'incrémente s'il existe déjà d'autre fichiers. Le data logging commence
- Les trames UART sont *affichées* sur l'écran et *enregistrées* dans le fichier
- *Le data logging se termine lorsque le cable de communication est déconnécté*. On peut ensuite enlever en toute sécurité la clé USB
- Il attend a nouveau qu'un convertisseur TTL/USB soit connecté, s'il ne l'est pas dans les 60s, la raspberry s'éteint
- Etc...

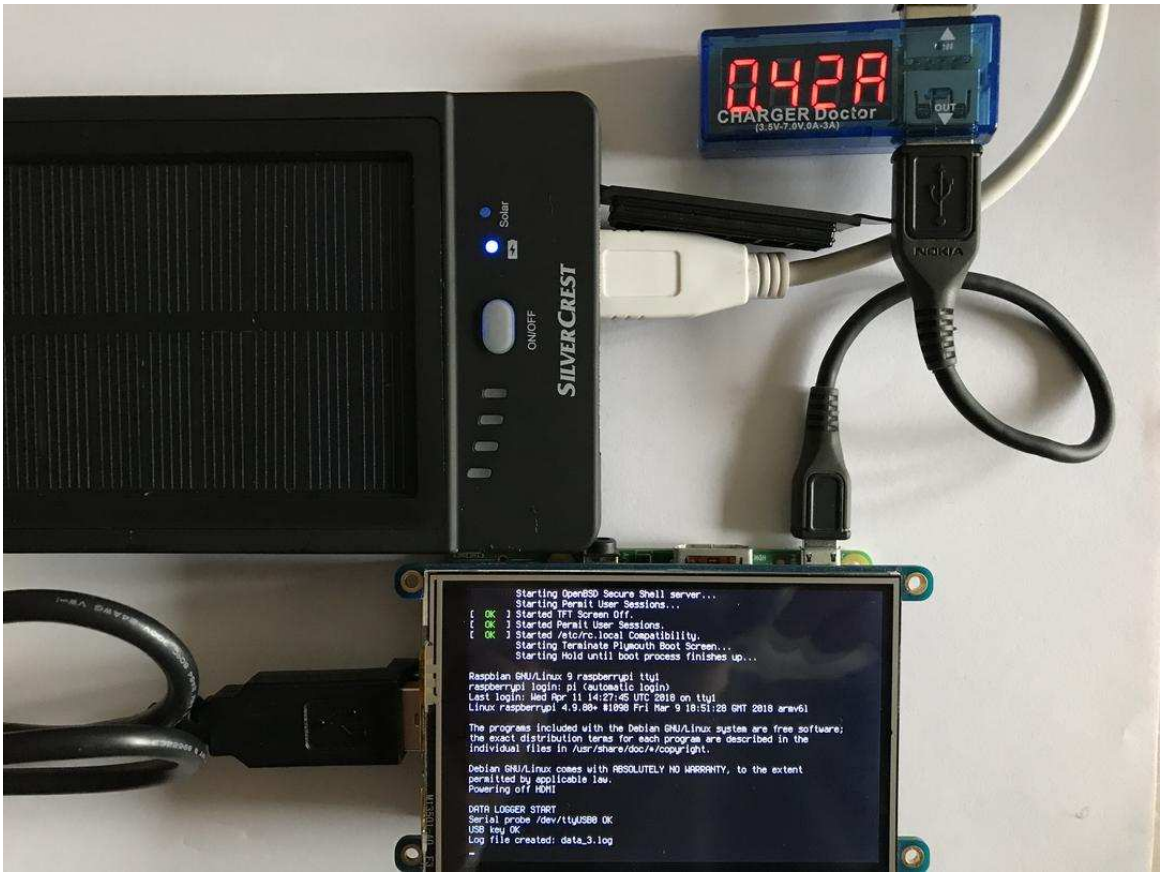
Dans un autre thread, *l'écran est éteint automatiquement au bout de 15s*. Pour le rallumer, il suffit d'*appuyer sur l'écran tactile*

Consommation

La power bank est de 5000mAh. Je suppose que cette valeur est celle de la batterie lipo à l'interieur. Cette valeur est donnée pour une tension de 3,7V dans la doc. Seulement, la sortie est de 5V! On va prendre une grosse marge et supposer que le *boost* qui va convertir la tension en 5V ait un *rendement de 80%*. Ça ne fait plus que *4000mAh*.

Lorsque l'*écran est en fonctionnement*, le courant consommé est de *420mA*. On aurait donc *une autonomie de 9,5h*.

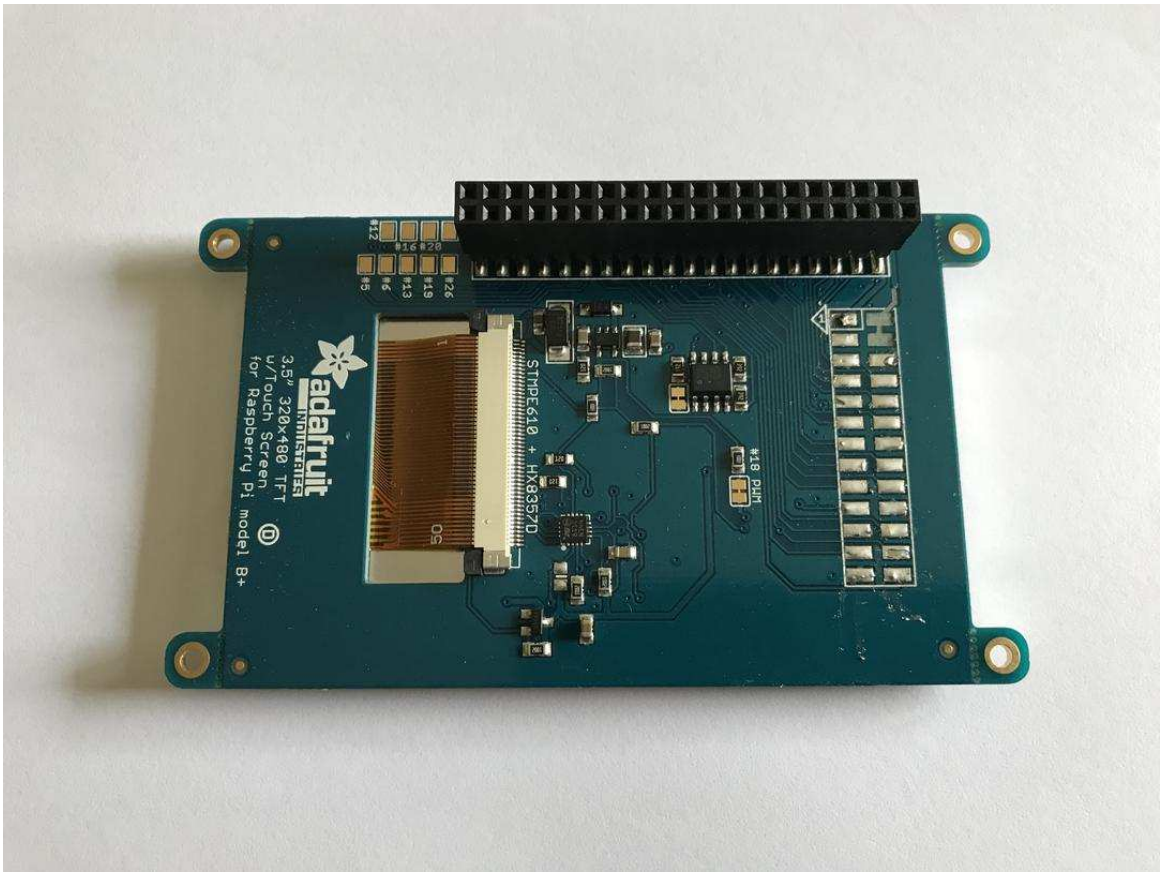
Grâce au soft, on met l'*économiseur d'écran* au bout de 15sec. On ne consomme plus que *260mA*. On a donc *une autonomie de 15,4h*. On a donc gratté presque 6h!



Consommation en fonctionnement normal

Consommation écran éteint

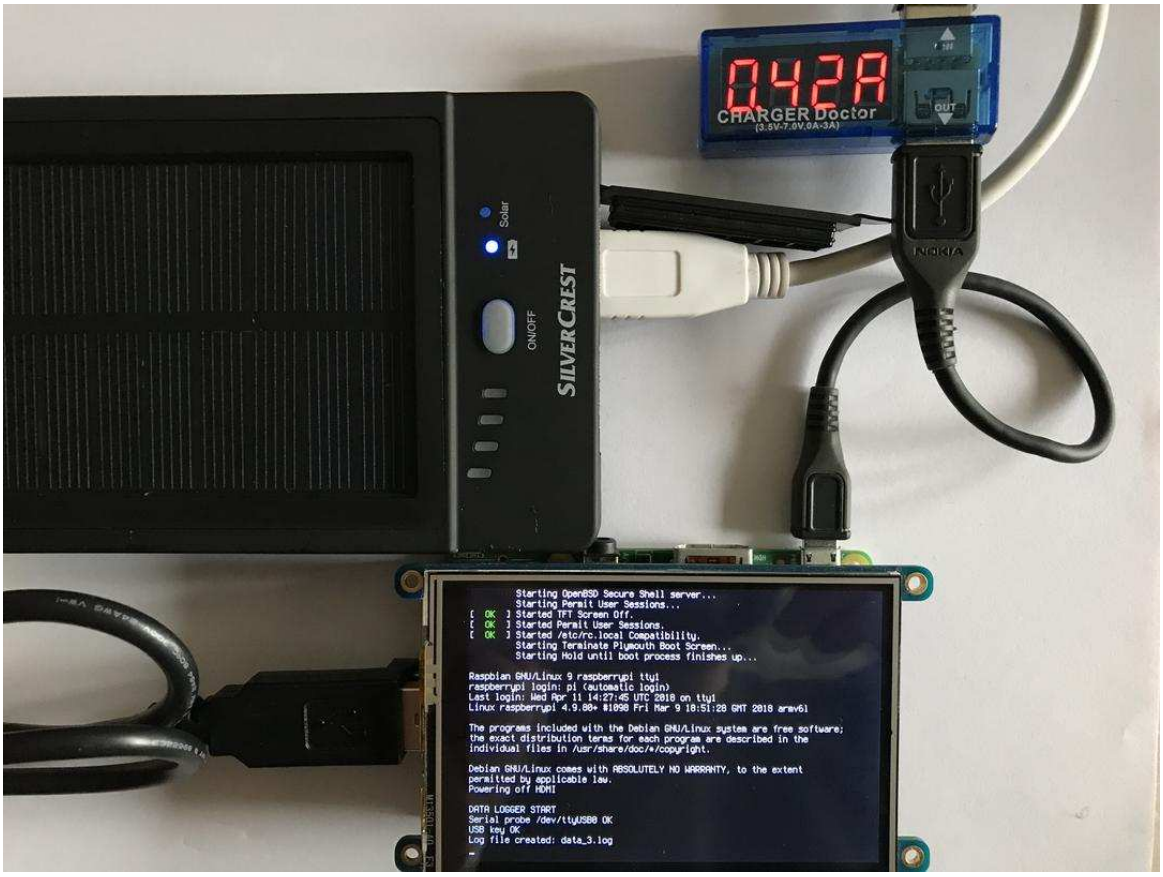
Galerie



Customisation de l'écran

Assemblage

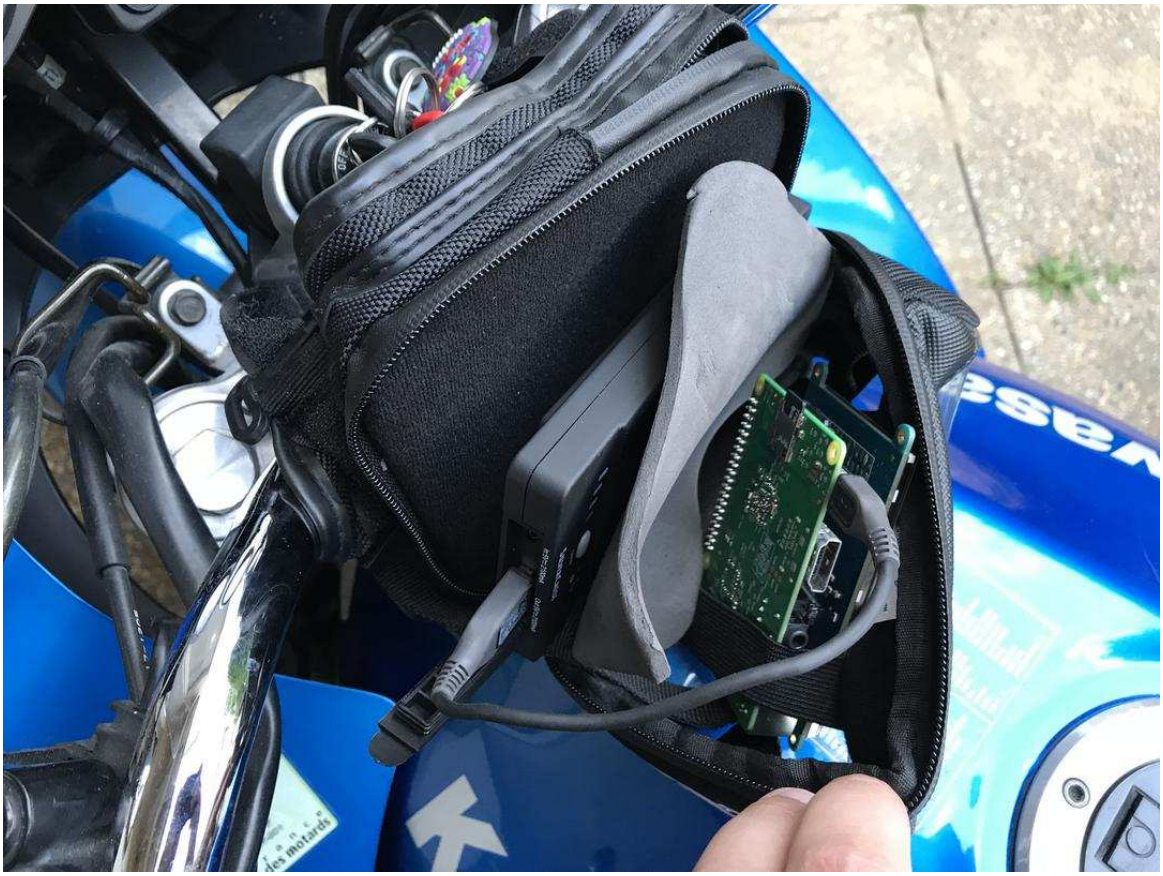
Système complet



Consommation en fonctionnement normal

Consommation écran éteint

La bête que je monte pour le data logging



Système sur la moto

Affichage sur la moto

Prochainement

Pour le moment, j'ai un système qui prend un peu de place (beaucoup moins qu'un ordinateur) mais qui est fonctionnel! *La prochaine étape sera de le fixer sur la moto* (un peu comme un GPS) et de commencer le data logging!

Je trouve ce système très pratique. Il a *une autonomie suffisante pour la plupart de mes projets* et *une puissance de calcul qui me pemettra même de faire du traitement en direct*! Dès que je peux, je dégage la Power bank pour y mettre la batterie commandée sur Ebay. Je n'ai qu'à *créer une carte qui fera boost, chargeur de lipo et qui possèdera aussi la coupure automatique*. Avec un boitier à l'imprimante 3D ou coupé en plexi, j'aurai un super data loggeur!