« Yes We CAN » avec PiCAN 3

Module de bus CAN pour le **Raspberry Pi 4**

Tam Hanna (Slovaquie)

Beaucoup de données ont circulé depuis la publication par Bosch de la spécification du bus CAN (*Controller Area Network*) en 1986 et sa première utilisation par Mercedes-Benz dans son modèle Classe S en 1991. Depuis, la notion de réseau CAN fait partie intégrante du développement et de la conception des automobiles. Le bus CAN est robuste, d'où son utilité dans de nombreux autres domaines d'application. Un module de bus CAN mobile est utile pour analyser les communications, diagnostiquer les problèmes et commander des dispositifs. Le HAT PiCAN examiné ici fait parfaitement l'affaire et se branche sur la toute dernière version du Raspberry Pi. Voyons cela de plus près.

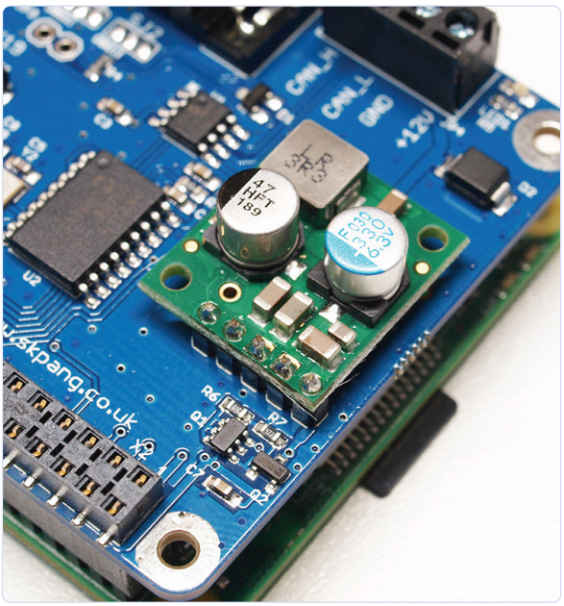


Figure 1. Cinq broches au pas de 0,1″ relient la carte d'alimentation (verte) à la carte principale.

Les systèmes électriques des véhicules sont des environnements hostiles pour les appareils électroniques délicats. J'ai le souvenir de nombreuses interventions de recherche de pannes consistant à poser des câbles de 40 m de long pour alimenter des oscilloscopes et des équipements de test fonctionnant dans une voiture. Si la simple évaluation des données du bus CAN vous intéresse, le PiCAN 3 est une autre ressource de diagnostic avantageuse et portable.

Pourquoi utiliser un Raspberry Pi ?

Lorsque vous expérimentez un système de communication par bus (I2C, SPI, etc.), les problèmes ne découlent pas de la mauvaise qualité du signal dans la couche physique, mais de l'organisation et du séquençage des données transmises. Si vous envoyez des paquets mal formés ou si vous vous trompez sur la façon dont les octets doivent être séquencés, par exemple, vous ne serez guère surpris si le moteur ou tout autre dispositif que vous espériez commander refuse de coopérer.

Des outils tels que PiCAN 3, associés à un Raspberry Pi, permettent d'effectuer des mesures mobiles. Dans l'idéal, vous n'avez besoin que d'un Raspberry Pi 4 et d'un petit écran pour construire un banc d'essai portable. Si vous intégrez l'environnement de développement du système CAN dans la distribution ARMbian du système, vous serez opérationnel.

Le matériel

L'environnement électrique d'une voiture peut être assez hostile pour les circuits électroniques. Le fabricant SK Pang en a tenu compte dans la conception de la carte en incorporant une alimentation à découpage (**fig. 1**) avec une large plage de tensions d'entrée (6 à 20 V continu). Elle se connecte à la carte via un connecteur à cinq broches. Le HAT PiCAN devrait être disponible sans l'alimentation à découpage – mais actuellement, ce n'est pas le cas et l'alimentation est fournie.

Ce tout dernier HAT CAN est baptisé « PiCAN 3 – carte à bus CAN pour Raspberry Pi 4 avec alim. à découpage 3 A et horloge en temps réel » pour le différencier de ses prédécesseurs. Cette dernière version inclut donc une alimentation à découpage de 3 A pour alimenter le Raspberry Pi 4, très gourmand en énergie. Il est possible de faire fonctionner le HAT directement à partir du Raspberry Pi lorsqu'il est alimenté par son port USB-C.

Le schéma de la carte peut être consulté en [1] et montre qu'elle fait appel à des circuits intégrés standard. Le manuel d'utilisation est disponible en [2].

L'interface du bus de signaux CAN est réalisée avec le MCP2515, qui communique avec le Raspberry Pi via SPI et une broche d'interruption GPIO. L'interface physique avec le bus est assurée par l'émetteur-récepteur de bus CAN MCP2562. Le schéma du circuit PiCAN 3 montre que l'alimentation de 3,3 V utilisée par certains composants de la carte provient du régulateur du Raspberry Pi, ce qui permet d'économiser un régulateur supplémentaire.

La carte possède deux connecteurs (**fig. 2**) pour relier la carte PiCAN 3 à un bus CAN externe. La première méthode fait appel à un bornier à quatre voies dont l'identification des signaux est imprimée sur la carte. La deuxième méthode utilise une prise sub-D

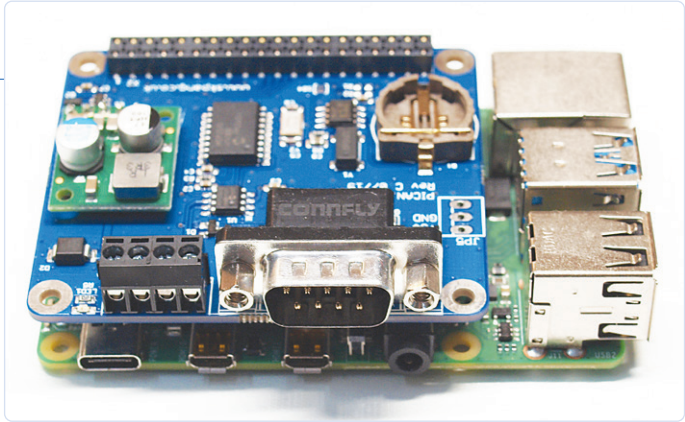


Figure 2. La carte PiCAN 3 permet de se connecter à un bus CAN de deux façons.

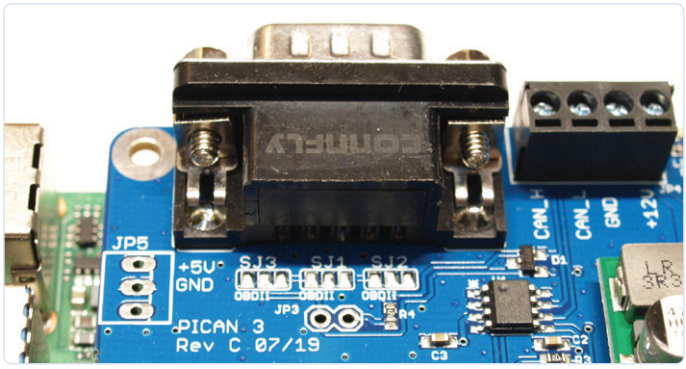
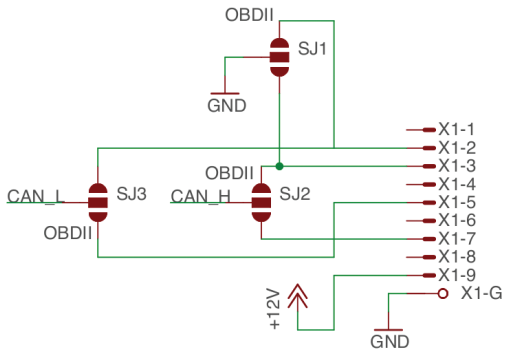


Figure 3. Grâce aux généreuses pastilles de soudure, le routage des signaux n'est pas trop compliqué.

à 9 broches, que les lecteurs d'un certain âge reconnaîtront comme celle des communications RS232 standard.
Le connecteur sub-D à 9 broches accepte un câble standard OBD-II vers DB9 pour se connecter à un système OBD. Malheureusement, il n'existe pas d'affectation standard des broches du connecteur et des signaux CAN. Pour parer à toutes les éventualités de câblage,



	CAN ANALYSER	OBDII	SOLDER-JOINT
CAN H	7	3	SJ2
CAN L	2	5	SJ3
GND	3	2	SJ1

Figure 4. Le tableau montre comment ponter les plages de soudure.

il est possible de ponter les pastilles de la carte (**fig. 3**) avec de la soudure. Un plan d'affectation (**fig. 4**) vous aide à décider lesquelles relier pour que le connecteur soit compatible avec le système auquel vous vous connectez. Ces pastilles sont complètement ouvertes sur la carte livrée. Ainsi, sans les ponts de soudure, les signaux ne seront pas acheminés vers les bonnes broches.
Les applications véhiculaires nécessitent souvent des informations précises sur l'heure du jour. Pour cela, le PiCAN 3 utilise une horloge en temps réel PCF8523. Celle-ci communique avec le Raspberry Pi via une interface I²C. La carte comporte un support pour pile bouton qui accepte une pile de type CR1220 pour alimenter l'horloge en temps réel.

L'écosystème CAN

Le système d'exploitation Linux a une certaine pertinence pour le secteur automobile. Il existe en fait tout un écosystème dans l'univers Linux permettant de prendre en charge les applications de bus CAN. Les utilitaires disponibles englobent aussi bien les pilotes de noyau et les logiciels de ligne de commande que de très nombreux autres puissants outils.
Pour commencer à communiquer avec le PiCAN 3, nous devons effectuer quelques adaptations dans le fichier */boot/config.txt* d'une installation Raspbian toute neuve. D'abord il faut activer le bus SPI avec le bloc suivant (notez l'appel supplémentaire *overlay*) :

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835-overlay
```

Les déclarations suivantes sont nécessaires pour utiliser l'horloge en temps réel :

```
dtparam=i2c_arm=on
dtoverlay=i2c-rtc,pcf8523
```

L'étape suivante consiste à télécharger les modules du noyau et quelques autres utilitaires. Fort à propos, il existe un package prêt à l'emploi disponible dans les référentiels :

```
sudo apt-get install can-utils
```

Après le déploiement des utilitaires CAN, il faut enregistrer l'interface sur le système d'exploitation. Pour cela, il suffit de créer une nouvelle interface selon le schéma suivant. La valeur 500 000 indique ici le débit maximal de données admis par le matériel :

```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Une fois l'interface prête, vous pouvez l'utiliser comme tout autre produit similaire pour communiquer à l'aide du bus CAN. De manière classique, vous pouvez utiliser l'outil `candump`, activable à l'aide de la ligne de commande, comme suit :

```
candump
```

Une fois en fonctionnement, elle affiche automatiquement et en continu tous les messages CAN du HAT. C'est particulièrement utile par ex. pour décortiquer les stratégies de commande de moteur inconnues dans les systèmes existants.

Il existe également une API CAN Python que vous pouvez installer :

```
git clone https://github.com/hardbyte/python-can
cd python-can
sudo python3 setup.py install
```

Ajout de l'horloge en temps réel

Le système d'exploitation Linux met en œuvre depuis longtemps des horloges en temps réel sur les ordinateurs portables et les PC. Les systèmes sans composant de type horloge en temps réel (pour des raisons de coût) utilisent le module émulateur `fake-hwclock` du système d'exploitation pour fournir des informations sur l'heure ; c'est également le cas du Raspberry Pi.
Pour pouvoir utiliser l'horloge en temps réel matérielle sur la carte CAN, il faut d'abord désactiver la « fausse » `hwclock` pour l'empêcher d'interférer avec l'heure fournie par la « vraie » :

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
sudo systemctl disable fake-hwclock
```

Ensuite, ouvrez le fichier */lib/udev/hwclock-set* (en mode super-utilisateur pour disposer des droits nécessaires) et mettez en commentaires les lignes de code dans ces deux blocs :

```
#if [ -e /run/systemd/system ] ; then
# exit 0
#fi

#/sbin/hwclock --rtc=$dev --systz --badyear
#/sbin/hwclock --rtc=$dev --systz
```

Les informations temporelles de l'horloge matérielle peuvent maintenant être lues à l'aide de `hwclock`. Les caractéristiques de cet outil d'administration de l'horloge matérielle sont présentées à la **figure 5**.

Fonctions de commande et de communication du bus CAN

Le HAT PiCAN 3 de SK Pang, avec son alimentation à découpage de 3 A embarquée, peut être utilisé avec un Raspberry Pi 4 pour offrir des capacités de communication et de commande par bus CAN. Avec le Raspberry Pi, il forme une plateforme mobile expérimentale compacte et relativement peu coûteuse, dotée de l'interface nécessaire pour se connecter directement à un bus CAN. Le matériel de la carte est *open source*, de sorte que le système peut être intégré à vos propres projets après évaluation, si nécessaire.

(210303-04)

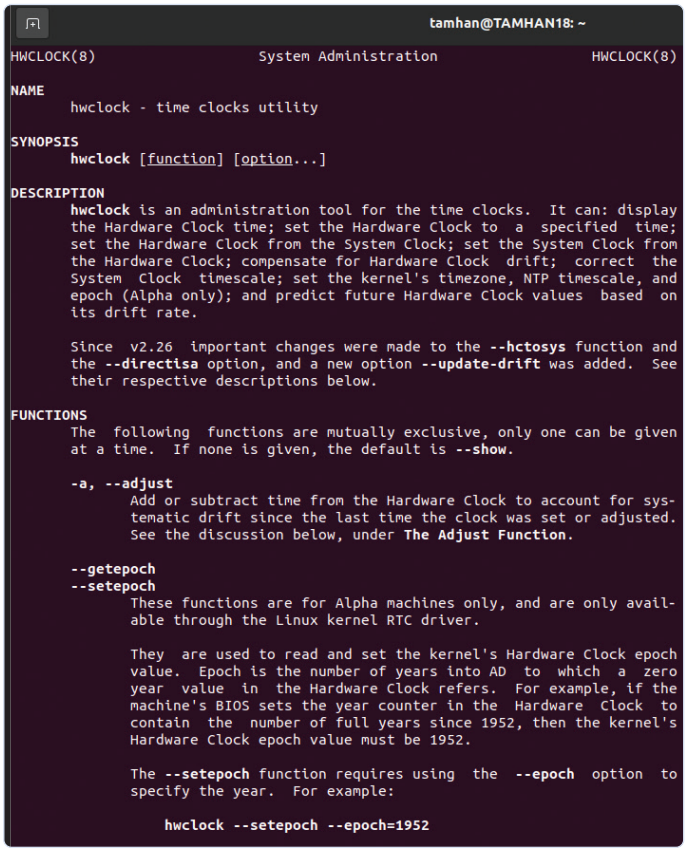


Figure 5. Pour en savoir plus sur l'accès par la ligne de commande à l'horloge en temps réel matérielle, utilisez `man` (abréviation de « manuel ») `<commande>`. Voici le résultat pour `hwclock`, avec des informations pour commander l'horloge en temps réel utilisée par la carte CAN.

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (tamhan@tamoggemon.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs
Texte et illustrations : **Tam Hanna**
Rédaction : **Thomas Scherer**
Traduction : **Pascal Godart**
Mise en page : **Giel Dols**

LIENS

[1] **Schéma de la carte PiCAN 3** : https://cdn.shopify.com/s/files/1/0563/2029/5107/files/pican3_rev_C.pdf?v=1619981690
[2] **Manuel de la carte PiCAN 3** : https://cdn.shopify.com/s/files/1/0563/2029/5107/files/PICAN3_UGA_10.pdf?v=1619981615