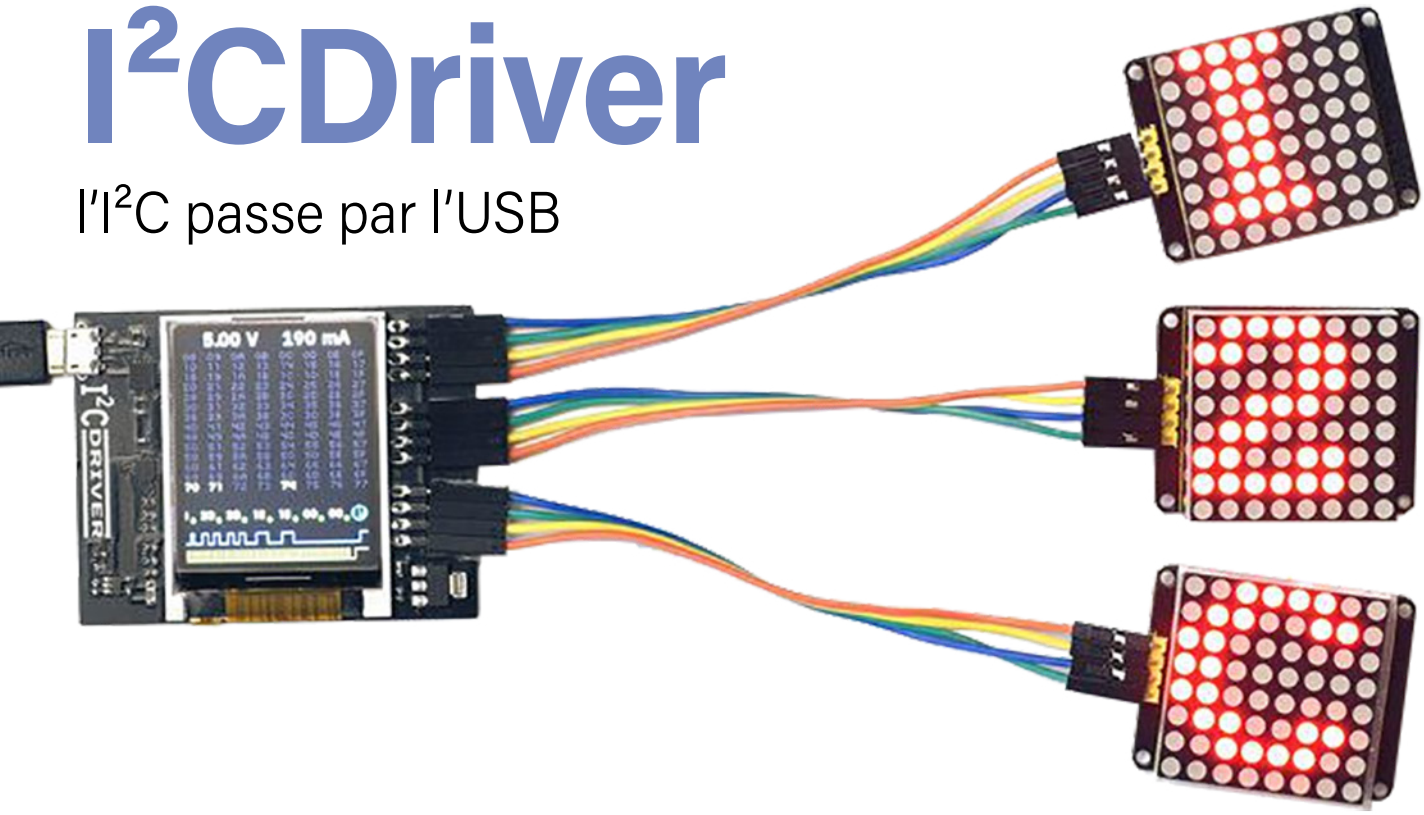
# I²CDriver

## l'I²C passe par l'USB




**Tam Hanna** (Slovénie)

Le bus I²C est utilisé dans de nombreuses applications embarquées. À des fins de test et de développement, *Excamera Labs* a développé l'*I²CDriver*, une carte qui fait le pont entre le logiciel PC et le matériel I²C (p. ex. les capteurs). La communication I²C est également enregistrée et affichée clairement en couleur.

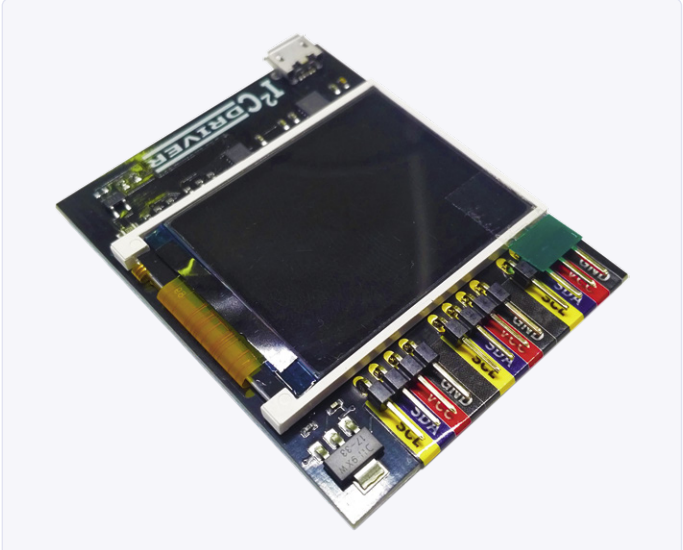



*Figure 1. L'I²CDriver est agréablement compact.*

Au déballage, vous trouverez comme le montre la **figure 1** une carte avec un connecteur microUSB et trois broches avec les signaux I²C. Des plots en caoutchouc sur la face inférieure l'empêchent de glisser. Pour le kit de base **I²CDriver**, *Excamera Labs* fournit trois jeux de câbles Dupont, ça facilite la connexion directe au matériel. Les broches d'alimentation délivrent jusqu'à 500 mA sous 3,3 V. Les **figures 2** à **4** montrent comment le produit se comporte en présence d'une charge. Comme les 3,3 V viennent de l'alimentation de l'ordinateur, il est tout à fait possible qu'une partie de l'ondulation visible dans les oscillogrammes soit due au bus USB.

Veuillez noter que l'I²CDriver n'offre pas l'isolation galvanique entre l'objet sous essai et l'ordinateur.

### Test avec du matériel réel

Comme je travaille actuellement sur un système de capteurs I²C avec l'*HygroSage*, c'est l'occasion rêvée pour moi de tester l'I²CDriver connecté entre le PC et mes capteurs (**fig. 5**). Comme la consommation est faible, je tente le coup avec l'alimentation directe à partir de l'I²CDriver. Ça passe.
*HygroSage* démarre bien (en dépit de mon câblage assez pathétique). L'affichage de la consommation en haut de l'écran se met à jour. Je signale que le contenu de l'afficheur de l'I²CDriver n'est apparu qu'après le démarrage du logiciel (sur lequel je reviens ci-dessous et qui s'est parfois planté au démarrage). Quand le démarrage réussit, vous voyez ce que montre la **figure 6 :** cet affichage vous informe sur la fréquence d'accès aux différents appareils (*heatmap*).

En travaillant avec cet afficheur, on constatera que son angle de vision est particulier, assez fermé pour que, si on le regarde de face, l'image est à peine visible.

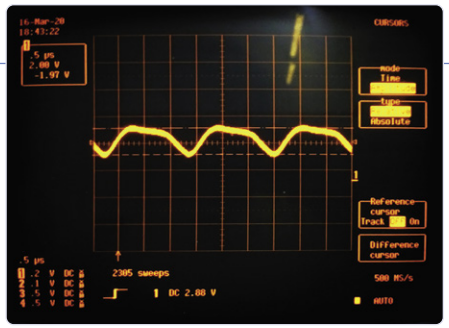



*Figure 2. L'I²CDriver à 500 mA.*

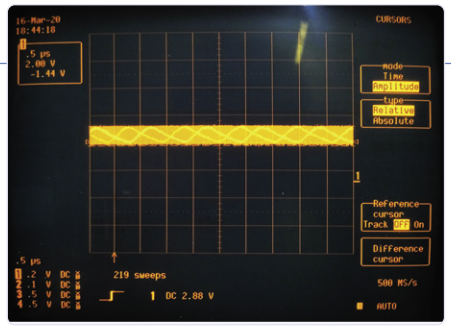



*Figure 3. … à 200 mA …*

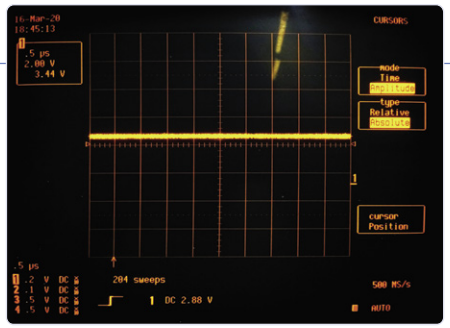



*Figure 4. … et au point mort.*

## Commande-moi !

Avec le lien [**1**], dans l'onglet *Ressources*, *vous* trouverez le fichier *i2cdriver-installer.exe* pour démarrer sous Windows. Après l'avoir téléchargé, afin que le système d'exploitation en autorise l'installation, il faudra d'abord un clic droit de souris pour le marquer le fichier dans la boîte de dialogue des paramètres comme provenant de l'ordinateur local. Pour qui travaille sous Linux ou MacOS, des instructions de configuration analogues se trouvent sur le même site.
Une fois le travail terminé, nous ouvrons le dossier *C:\Program Files (x86)\Excamera Labs\I2CDriver*, où nous trouvons à la fois un outil en ligne de commande et une version GUI.
Si vous démarrez le logiciel avec l'I²CDriver connecté et que vous cliquez sur le bouton *Monitor Mode*, *vous* obtiendrez des informations sur la dernière transaction effectuée (**fig. 7**).

En pratique, cependant, l'utilité de la fonction d'analyse (visuellement attrayante) est limitée ; dans presque tous les cas, sa capacité est débordée par la complexité du transfert. Dans ce cas, il vaut mieux cliquer sur *Capture Mode* (**fig. 8**).
Cette fonction s'est plantée de manière reproductible lors de mes tests, mais elle est quasi géniale en ce qu'elle permet d'enregistrer la communication les doigts dans le nez. Cela permet de débusquer les défauts furtifs, ceux qui sont si difficiles à repérer parce qu'ils ne se produisent que sporadiquement. Cela m'a fait penser à la procédure DPO de *Tektronix*.

## Interaction, programmatique

Il y a quelque temps, dans ma *forge à logiciel*, j'ai eu à mettre en œuvre un algorithme passablement complexe. L'approche la moins laborieuse a consisté à lancer la procédure sur le PC dans un premier temps, puis, une fois qu'elle marche, de la transposer sur le contrôleur.

Une procédure similaire convient pour la mise en service de capteurs complexes. Avec *i2ccl*, on dispose d'un programme dans l'invite de commande, par lequel vous pouvez envoyer des commandes à l'I²CDriver selon le schéma suivant :

```
C:\Program Files (x86)\Excamera Labs\I2CDriver>i2ccl.exe
Usage: i2ccl
```

Ici on apprécie la possibilité d'écrire ou de lire des informations dans les registres individuels des appareils connectés. Cela permet non seulement de prendre en main des capteurs inconnus, mais aussi de lire des informations lors de tests (automatisés).
Si vous ne voulez pas programmer dans le *shell*, vous pouvez passer par une API Python. Le fabricant fait la démonstration de l'utilisation d'un ensemble de modèles de pilotes tout prêts.

L'extrait suivant peut être utilisé pour exploiter un LM75B :

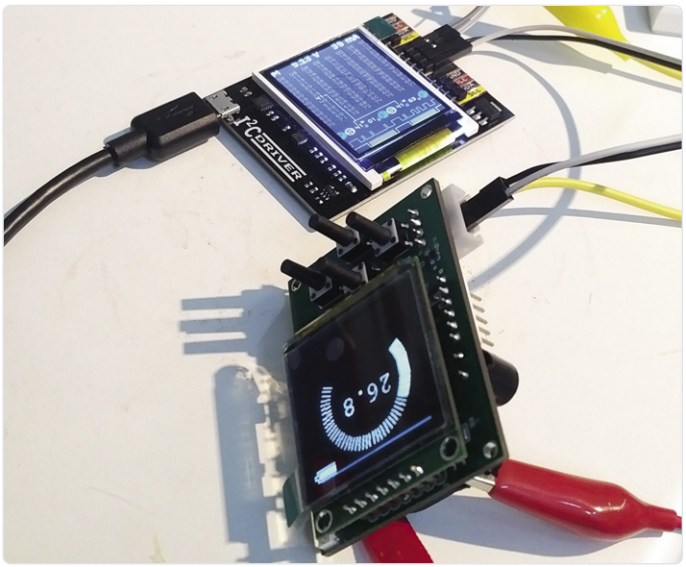



*Figure 5. L'I²CDriver connecté à une carte de capteurs de l'auteur.*

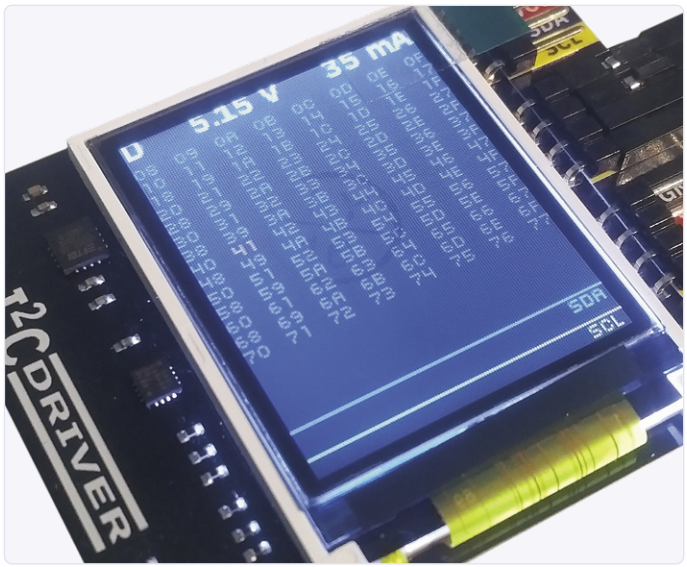



*Figure 6. Une seule couleur, cela signifie qu'il n'y a qu'un seul capteur.*

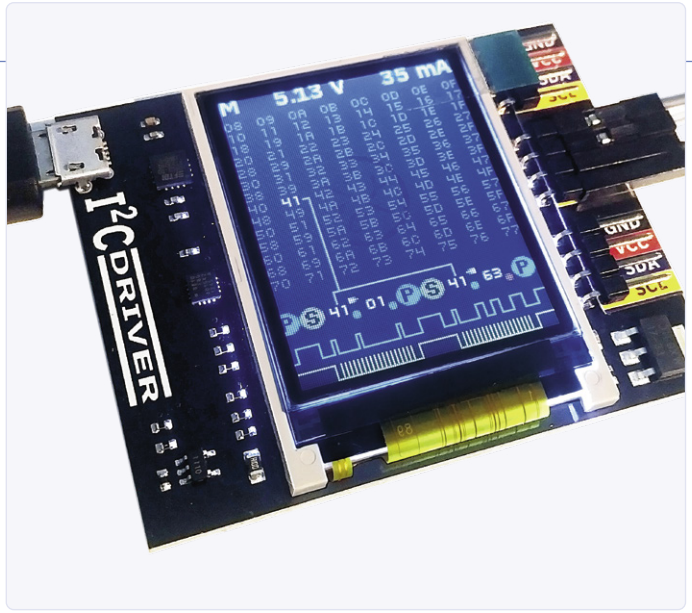



*Figure 7. Les opérations du registre apparaissent sur l'afficheur de l'I²CDriver.*

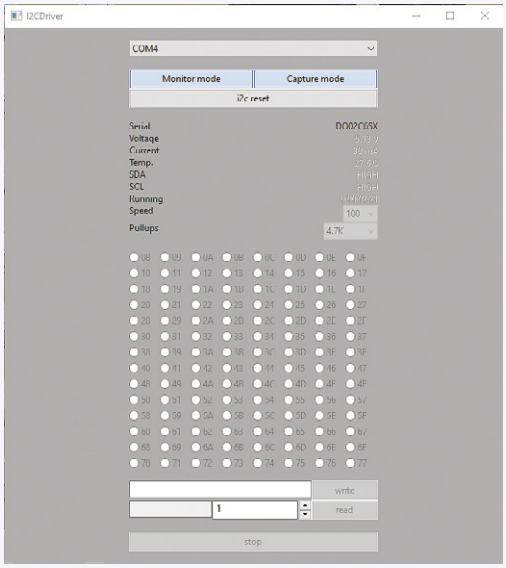



*Figure 8. Le logiciel de bureau I²CDriver a un charme rustique.*

```
import i2cdriver
i2c = i2cdriver.I2CDriver(«/dev/ttyUSB0»)
d=i2cdriver.EDS.Temp(i2c)
d.read()
17.875
d.read()
18.0
```

L'API de commande proprement dite est simple. Voir GitHub [**2**] :

```
class LM75B:
    def __init__(self, i2, a = 0x48):
        self.i2 = i2
        self.a = a
```

*Excamera Labs* implémente les pilotes matériels en utilisant l'API Python-OOP. `self` est un pilote requis par la spécification du langage, tandis que `i2` est un objet pilote I²C. Enfin, `a` est l'adresse du capteur. Les informations du registre sont ensuite lues selon le schéma suivant :

```
    def reg(self, r):
        return self.i2.regrd(self.a, r, «>h»)

    def read(self):
        return (self.reg(0) >> 5) * 0.125
```

Il existe une commande associée à *i2cdetect* qui, lorsqu'elle est appelée depuis la ligne de commande Python, exécute la fonction de *scan* connue d'OrangePi et compagnie.
J'attire l'attention du lecteur sur la documentation [**3**] qui explique l'I2C-API ainsi que le protocole de communication physique. Quelqu'un qui connaît l'API FTDI comme sa poche peut également accéder directement à l'I²CDriver.

## Conclusion

La révélation de l'I²CDriver ne vient pas tout de suite, c'est un de ces produits dont on ne capte pleinement l'essence qu'après une certaine période de contemplation. Après ça, impossible de s'en passer. Qu'il soit utilisé pour une analyse rapide de l'activité d'un réseau I²C ou pour la mise en service d'un capteur, cette carte fournit une aide précieuse. Le prix est raisonnable compte tenu du temps gagné, dommage qu'il n'y ait pas de mode autonome.

200148-03 VF



## LIENS

[1] **logiciel** : https://i2cdriver.com/
[2] **API de commande** : https://github.com/jamesbowman/i2cdriver/blob/master/python/lm75b.py
[3] **documentation** : https://i2cdriver.com/i2cdriver.pdf