voltmètre Raspberry Pi
à cadran et à bargraphe

Hermann Nieder
(Allemagne)

Quelques composants et un Raspberry Pi suffisent pour disposer d'un voltmètre capable de mesurer des tensions jusqu'à 5 V CC. Afficher les résultats à l'écran n'est ensuite plus qu'une affaire de codage. Python est ici utilisé pour modéliser deux voltmètres, l'un analogique, l'autre à bargraphe. En couleurs et en plein écran, s'il vous plaît !

J'ai écrit pour Elektor les trois volets de la série « de Basic à Python » [1]. Vous ne serez donc pas surpris que Python ait eu ma préférence pour l'écriture des deux applications présentées ici. Chaque programme se sert du Pi (révision 2) pour communiquer avec un convertisseur A/N et afficher le résultat des mesures à l'écran.

Circuit annexe

Le circuit à relier au Pi (**fig. 1**) est construit autour du convertisseur analogique-numérique sériel TLC549 de *Texas Instruments* [2]. Cette vénérable puce (plus de 30 ans d'âge !) a une résolution de 8 bits, est facile à trouver (en boîtier DIP bien sûr), et simple d'utilisation. Un diviseur de tension (R1, R2, R3) est relié à son entrée analogique. Durant mes essais, j'y ai connecté un potentiomètre de 10 kΩ et me suis servi d'un multimètre numérique pour comparer les mesures obtenues.

Les deux boutons HOLD1 et HOLD2 permettent chacun d'enregistrer le résultat d'une mesure. Les valeurs sauvegardées sont affichées en bas de l'écran et également indiquées au-dessus de l'échelle par un marqueur. Elles peuvent être effacées avec le bouton RES si l'utilisateur souhaite en afficher et marquer deux nouvelles.

Le circuit annexe tient aisément sur une de ces plaques sans soudure pour Raspberry Pi que l'on trouve en vente un peu partout et qui s'enfichent sur le connecteur d'extension du Raspberry. La tension d'alimentation de 3,3 V est fournie par le Pi. La plage de mesure peut être étendue par ajout de résistances en série avant R1, mais dans ce cas il vous faudra adapter en conséquence les deux programmes Python. Servez-vous des commentaires du code pour identifier les lignes à modifier.

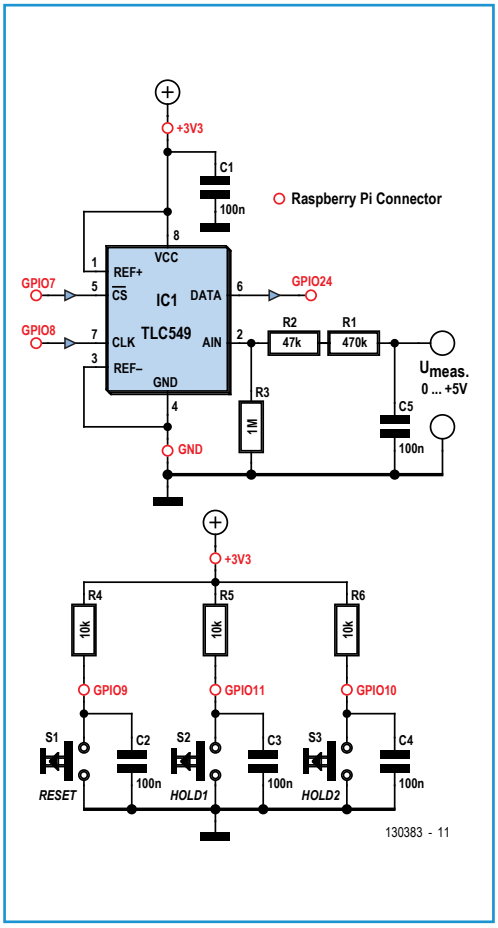

Figure 1. Le circuit annexe et son vénérable convertisseur A/N.

Les versions « bargraphe » et « cadran »

La **figure 2** montre le programme « bargraphe ». Trois valeurs sont affichées : celle du milieu a une police plus grande et représente la mesure actuelle ; les deux autres sont les valeurs de mesure enregistrées par l'utilisateur. Ces valeurs sont marquées au-dessus du bargraphe par un trait.

Si les deux valeurs que l'on souhaite enregistrer ne sont pas trop proches l'une de l'autre, une version légèrement modifiée du programme permet de représenter les marqueurs par des pointes de flèche (**fig. 3**). La version « cadran » reproduit un cadran de voltmètre et affiche la valeur mesurée. Comme précédemment les deux boutons HOLD1 et HOLD2 du circuit annexe permettent d'enregistrer deux valeurs, indiquées elles-aussi par deux traits situés au-dessus des graduations. Et là encore les valeurs numériques et les marqueurs peuvent être effacés au besoin. Une déclinaison graphique de cette version « cadran » dessine un cadre autour de l'aiguille et des graduations (**fig. 5**).

Quelle que soit la version utilisée, vous pouvez choisir entre plusieurs couleurs d'avant et d'arrière plan, et paramétrer le facteur d'échelle qui détermine la taille de la police. Vous pouvez également entrer un nom pour la fenêtre d'affichage.

Essai

Les programmes en Python peuvent être téléchargés depuis la page du site Elektor associée à cet article [3] ; préférez bien sûr un téléchargement depuis un Raspberry Pi connecté à Internet.
Créez à l'intérieur du répertoire */home/pi* un dossier dans lequel vous copierez puis extrairez les fichiers de l'archive.

Vous devez également installer la bibliothèque Python appelé *Pygame* puisque tous les exemples de programme l'utilisent. Si votre système d'exploitation est de type Debian, vous pouvez l'installer depuis un terminal avec :

```
sudo apt-get install python-game
```

Tout est prêt ; pour lancer le programme,

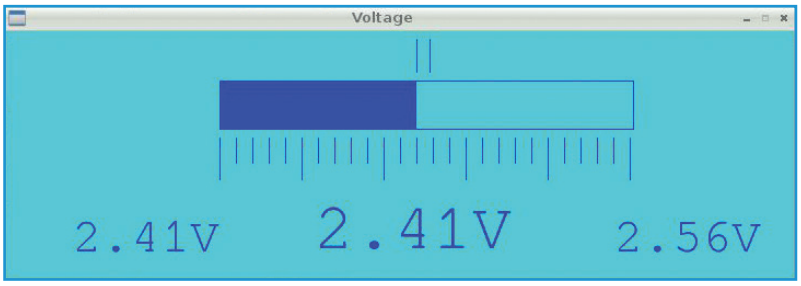

Figure 2. Le voltmètre numérique à bargraphe

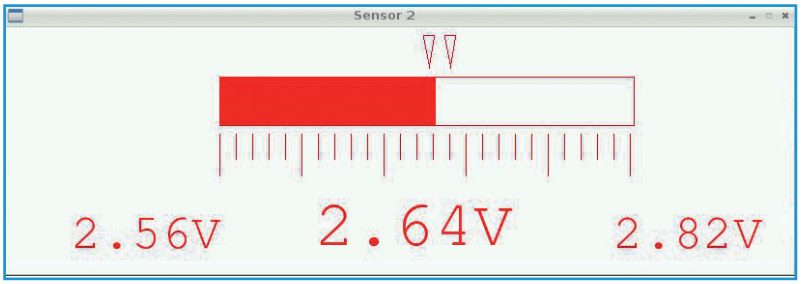

Figure 3. Dans cette variante, ce sont deux petites flèches qui marquent les valeurs enregistrées.

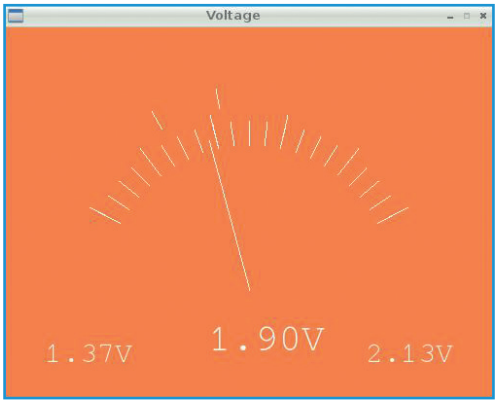

Figure 4.
Le voltmètre avec cadran et affichage numérique.

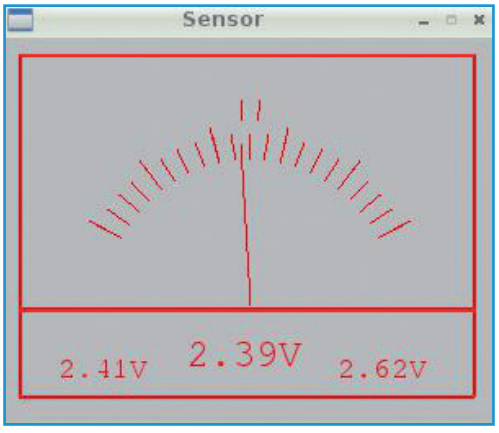

Figure 5.
La version de luxe avec un cadre !

placez-vous dans le dossier qui contient les fichiers Python et entrez :

```
sudo python ADW_PTR_D.py
```

(le fichier *ADW_PTR_D.py* est la version allemande du programme).
Choisissez une des couleurs d'arrière-plan disponibles et validez votre choix avec *Entrée* ; faites de même avec la couleur d'avant-plan, puis entrez un nom pour la fenêtre d'affichage, ainsi qu'un facteur d'échelle pour la taille de la police. La fenêtre s'ouvre dès que ce dernier paramètre est validé.
Si la taille de la police ne vous convient pas, fermez le programme depuis le terminal avec *Ctrl-C*, et essayez un nouveau paramètre. La procédure est la même pour les versions « cadran » et « bargraphe ».

Dans l'exemple reproduit sur la **figure 6**, j'ai choisi *white* pour l'arrière-plan, *red* pour les marqueurs, les graduations et l'affichage des valeurs de mesures. Comme facteur d'échelle de la taille de police, j'ai entré 0,7.
Avec les paramètres de la **figure 7**, les résultats d'une mesure de tension sur l'entrée du TLC549 s'affichent en bleu sur fond jaune.

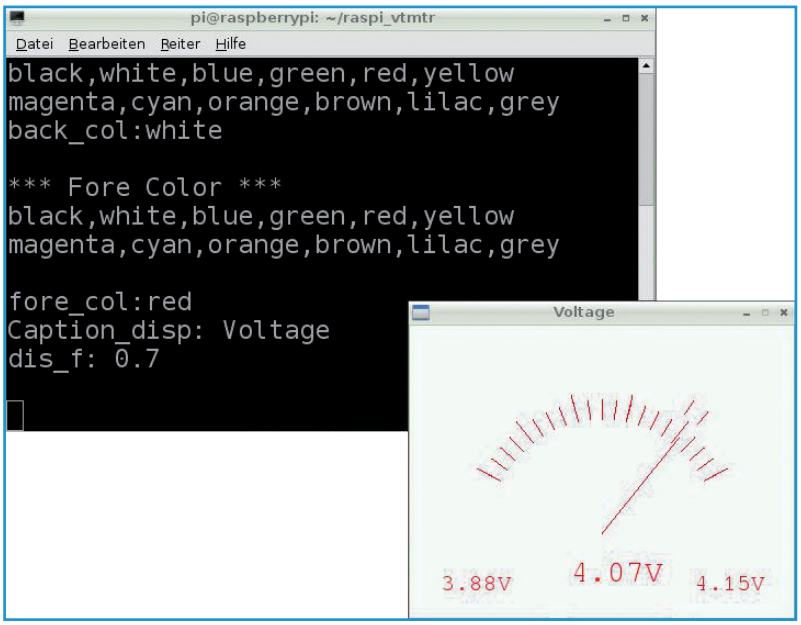

Figure 6. Lancement et paramétrage de l'application « cadran » depuis le terminal LXTerminal du Raspberry Pi.

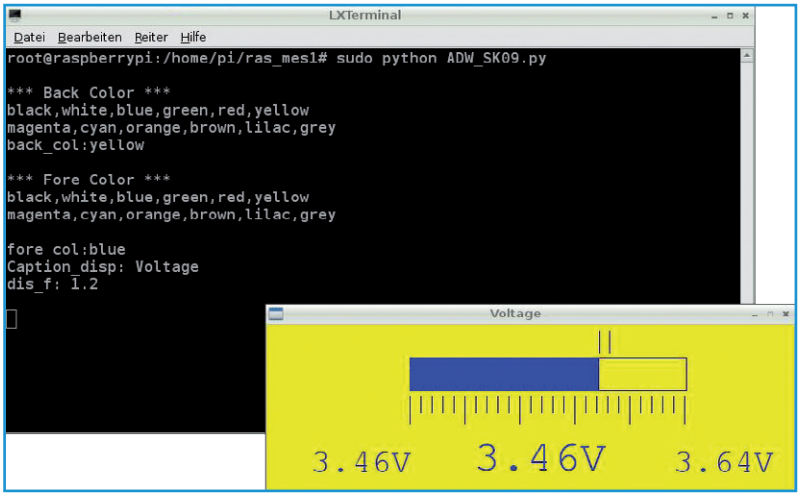

Figure 7. Paramétrage de l'application pour un modèle suédois.

Code Python

Le convertisseur A/N TLC549 se commande via les broches GPIO 7, 8 et 24. Le port GPIO7 est défini dans le programme comme `AD_CS`, le port GPIO8 avec le nom `AD_Clk`. Ces notations rendent le code plus lisible. L'utilisation des broches pour une lecture de mesures est décrite dans la fiche technique du convertisseur [3]. Les huit bits du résultat de la conversion numérique sont disponibles sur la sortie DATA pour un traitement ultérieur. Cette sortie sérielle DATA est reliée au port GPIO24 du Pi, appelé `AD_Dat` dans le programme.
Pour les deux versions du programme, le TLC549 est commandé par les lignes du premier listing reproduit à la fin de cet article.

Pour l'affichage des valeurs numériques et pour la représentation graphique, il faut les conversions du second listing reproduit à la fin de cet article.

Dans la version « bargraphe », pour représenter la valeur de mesure actuelle sur le bargraphe et afficher sa valeur numérique, le programme doit effacer la précédente surface de couleur ainsi que la précédente valeur affichée. Le programme commence par dessiner un rectangle « plein » invisible, c'est-à-dire rempli avec la couleur d'arrière-plan. Il dessine ensuite une nouvelle surface avec la couleur de la valeur actuelle, puis affiche la valeur numérique de la tension actuelle.
Le programme « cadran » a été conçu en

suivant la même démarche. Le codage a tout de même nécessité beaucoup plus de travail. Les valeurs nécessaires à la construction des points de départ et d'arrivée des lignes qui représentent les graduations sont déterminées à l'aide des fonctions trigonométriques sinus et cosinus. La déviation de l'aiguille correspond à la valeur mesurée actuelle, donc le programme utilise ici aussi les fonctions sinus et cosinus pour le calcul de sa position. L'extrémité de l'aiguille se positionne contre les graduations comme le fait l'aiguille d'un véritable instrument de mesure. Les deux marqueurs sont orientés dans la direction qu'avait l'aiguille au moment où l'utilisateur a appuyé sur un des boutons de sauvegarde.

À la différence du programme « bargraphe », c'est ici l'ensemble de la zone couverte par l'aiguille (un disque) qui est effacé avant que ne soit dessinée l'aiguille dans sa position actuelle. Je vous renvoie aux commentaires des programmes si vous souhaitez en savoir plus !

(130383 – version française : Hervé Moreau)

Liens

[1] de Basic à Python, Elektor, mai, juin, septembre ; www.elektor-magazine.fr

[2] www.ti.com/lit/ds/symlink/tlc549.pdf

[3] www.elektor-magazine.fr/130383

commande du TLC549

```
def ADin():

    GPIO.output(AD_Clk,GPIO.LOW)# AD_Clk to 0
    AD_res=0
    for n in range(10):
        time.sleep(0.05)
        GPIO.output(AD_CS,GPIO.HIGH)# AD_CS to 1
        MSB=128
        time.sleep(0.001)
        GPIO.output(AD_CS,GPIO.LOW)# AD_CS to 0
        time.sleep(0.0005)
        AD_value=0
        for z in range(8):
            if (GPIO.input(AD_Dat)):
                AD_value=AD_value+MSB
            GPIO.output(AD_Clk,GPIO.HIGH)# AD_Clk to 1
            time.sleep(0.0005)
            GPIO.output(AD_Clk,GPIO.LOW)# AD_Clk to 0
            MSB=MSB>>1
            time.sleep(0.0005)
            result=AD_value
        GPIO.output(AD_CS,GPIO.HIGH)# AD_CS to 1
        AD_res=AD_res+AD_value
    AD_res=AD_res/10
    result=AD_res
    return result
```

conversions nécessaires pour l'affichage des valeurs numériques et pour la représentation graphique

```
def conversion(value0):
    value=value0
    value=value*1960 #for 5V
    value=value/1000
    pic_value=value/2 #
    one=value/100
    rest_z=value % 100
    tenth=rest_z/10
    hundredth=rest_z%10
    return one, tenth, hundredth, pic_value,value
```