niveau à bulle électronique et disque stroboscopique actif pour **platines vinyle**

régler votre platine vinyle avec cet outil tout-en-un

Antonello Della Pia (Italie)

Pour un fonctionnement optimal, une platine vinyle doit être placé parfaitement à l'horizontale et tourner à la bonne vitesse. L'outil que nous présentons, basé sur le Raspberry Pi Pico permet de vérifier ces deux aspects. De plus, grâce à une approche novatrice, il intègre un disque stroboscopique qui se passe de la traditionnelle lumière stroboscopique pulsée de 50 ou 60 Hz.

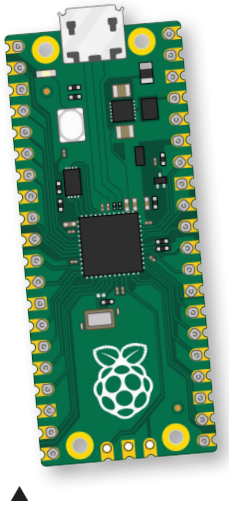


Figure 1. Le Raspberry Pi Pico utilisé dans le projet.

Malgré la prédominance du streaming multimédia en haute résolution, l'intérêt renouvelé pour la lecture de musique analogique, et en particulier pour les disques vinyles, est remarquable et peut sembler étonnant. Cette tendance est confirmée par les statistiques de vente dans l'industrie musicale. Par ailleurs, le marché des équipements dédiés (platines, cellule platine, préamplificateurs phono, accessoires) est plus vaste, plus diversifié et plus accessible qu'auparavant, attirant tant les amateurs que les audiophiles expérimentés dans le domaine de la reproduction musicale.

De l'idée à la réalisation

Sans entrer dans le débat sur la supériorité de la qualité sonore analogique ou numérique, il est clair que le fonctionnement optimal d'une platine vinyle repose sur un équilibre délicat entre divers facteurs électriques et mécaniques. À condition que le matériel soit en bon état et correctement réglé. Les facteurs critiques incluent l'alignement de la pointe de lecture, la mise à niveau de la platine et la vitesse de rotation précise du plateau.
Inspiré par cette idée, j'ai envisagé de créer une version numérique moderne du niveau à bulle classique et du disque stroboscopique généralement utilisés pour le réglage des platines vinyles. Cette idée est devenue réalisable grâce à la disponibilité d'un écran LCD de forme ronde de Waveshare [1] et aux capacités impressionnantes du Raspberry Pi Pico [2]. Ce projet intègre également le module GY-521, qui comprend le capteur de position MPU-6050 de TDK InvenSense, et deux boutons pour l'alimentation, le calibrage, la sélection du mode et la surveillance de la tension de la batterie.
Notre but est de concevoir un appareil auto-alimenté, compact et léger, qui puisse être déposé sur le plateau d'une platine vinyle sans perturber sa rotation. cet appareil offre une plage de mesure de niveau de +/-10° sur les axes x et y (avec une précision de 0,1°) et un mode "linéaire" pour des mesures de +/-90° sur l'axe x. Il ne se limite pas à l'affichage numérique de l'inclinaison, mais représente aussi graphiquement le mouvement de la "bulle" sur l'écran. En mode stroboscopique, il est capable de vérifier les vitesses de rotation standard de 33,33 et 45,00 RPM sans lumière externe.

Le Raspberry Pi Pico

Le cœur de ce projet est le Raspberry Pi Pico, une petite carte peu coûteuse (**figure 1**). Lancée en janvier 2021, elle a été récemment proposée avec une option de connectivité sans fil également. Contrairement aux modèles antérieurs de Raspberry Pi, qui sont des ordinateurs à carte unique (SBC) capables d'héberger et d'exécuter des SE basés sur le noyau Linux (typiquement Raspberry Pi OS), le Raspberry Pi Pico est doté d'un microcontrôleur et des

composants juste nécessaires à son fonctionnement. Son fonctionnement s'apparente à un Arduino UNO classique, mais le Pico offre des performances plus importantes, grâce au microcontrôleur RP2040, un Arm Cortex-M0+ à double cœur avec 264 KB de RAM interne et 2 MB de mémoire flash, et une fréquence d'horloge de 133 MHz qui peut être facilement surcadencée.
Grâce à ses interfaces USB, UART, SPI, I2C, ADC, PWM, et ses 26 broches GPIO multifonctions, sans oublier la version dotée de la connectivité sans fil 2,4 GHz 802.11n, on peut se procurer pour un prix assez bas une carte adaptée même à des projets assez exigeants en termes de puissance de calcul et de gestion de composants externes. Les caractéristiques complètes et toute la documentation relative au Raspberry Pi Pico sont disponibles sur le site du fabricant [3].

Le module GY-521 avec MPU-6050

Le module GY-521 est basé sur la puce MPU-6050, un capteur IMU (unité de mesure inertielle) à six axe en technologie MEMS (*micro electromechanical system*) qui intègre un gyroscope et un accéléromètre. Malgré son ancienneté, ce module est encore fréquemment utilisé dans les projets basés sur Arduino ou d'autres microcontrôleurs pour déterminer la position d'un objet dans l'espace environnant (par exemple, dans les robots, les drones, la reconnaissance de gestes, les interfaces de réalité virtuelle, les appareils portables). Il est disponible sur le marché, peu coûteux et bénéficie d'un support de bibliothèques dédiées.
Après quelques essais, il s'est avéré que le module est facile à interfacer et suffisamment précis et stable pour les exigences du projet. Le fonctionnement de ce type de capteur mérite une étude approfondie. Dans cette puce de quelques millimètres carrés, on trouve, en plus des composants électroniques, un MEMS que l'on peut décrire, pour simplifier, comme une minuscule masse en mouvement qui produit une variation de distance et donc de capacité entre un ensemble d'électrodes, en réponse aux contraintes dynamiques subies par la puce. Ces variations sont converties en un signal numérique de 16 bits, transmis via une interface I²C pour un traitement ultérieur. Pour le niveau à bulle électronique, les données des axes x et y seront converties en indications numériques et graphiques claires, affichées sur l'écran.
Un article informatif sur la technologie et l'utilisation de ce capteur MEMS et ainsi que d'autres capteurs MEMS est disponible sur le site de Last Minute Engineers [4]. La **figure 2** montre un schéma des axes de détection et de la polarité de rotation redessiné à partir d'un diagramme contenu dans la fiche technique [5].

Disque stroboscopique

Le fonctionnement du disque stroboscopique, couramment utilisé pour mesurer la vitesse de rotation des platines vinyle, repose sur un phénomène optique bien connu et utilisé depuis longtemps en mécanique (phasage des moteurs à combustion interne et vérification des dispositifs rotatifs à grande vitesse tels que les turbines). L'objet mobile observé, tournant à une vitesse constante, est éclairé par une source lumineuse pulsée, à une fréquence définie, pour qu'il ne devienne visible à l'œil humain que lorsqu'il est exposé à la lumière. Si la fréquence des impulsions lumineuses est égale ou multiple de la vitesse de rotation, l'objet apparaît immobile.
Sur le type de disque illustré dans la **figure 3**, on observe un motif de lignes équidistantes tracées sur la périphérie avec l'indication " 50 Hz ". En fait, il devait être utilisé avec avec une lampe à incandescence (aujourd'hui obsolète) alimentée par le courant alternatif du secteur, qui éclairait sur les demi-ondes positives et négatives, produisant ainsi cent impulsions lumineuses par seconde (50 × 2 = 100 Hz). Le nombre de " notches " nécessaires pour obtenir cet effet stroboscopique à cette fréquence est donné par la formule :

$$n = (f \times 2 \times 60) / \omega$$

où f est la fréquence du réseau en Hz et ω la vitesse angulaire à contrôler exprimée en tours par minute (33,33 et 45,00 RPM). À ce stade, il est clair que le même

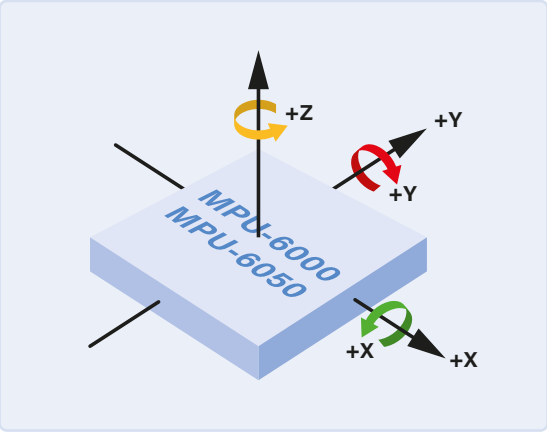
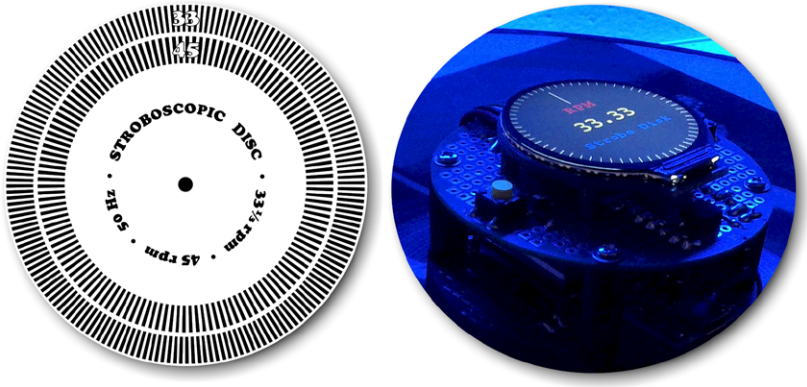


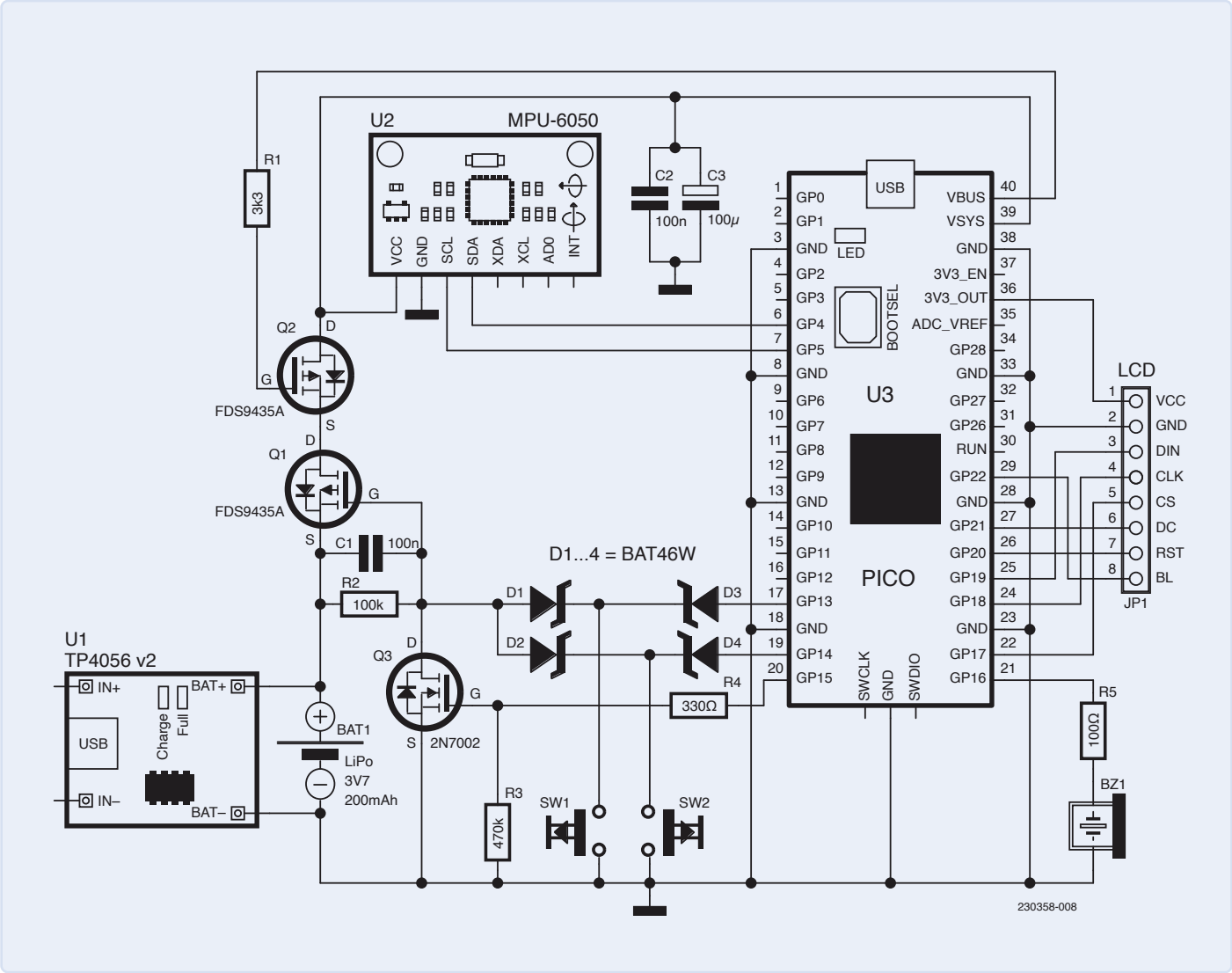
Figure 2. Axes de détection.

◀

Figure 3. Un disque stroboscopique ordinaire (à gauche) et un disque numérique actif (à droite). (Source de l'image du disque :Wikimedia Commons https://en.wikipedia.org/wiki/File:Stroboscopic_disc.svg)

▼





▲

Figure 4. Schéma du projet.

résultat peut être obtenu en conservant un nombre fixe de lignes de référence et en variant la fréquence de la source lumineuse. Plus spécifiquement, avec soixante points de référence, il s'avère que pour les voir immobiles, la fréquence des impulsions lumineuses devra être égale au nombre de tours par minute :

$$\omega = (f \times 60) / n = (f \times 60) / 60 \qquad \omega = f$$

Dans la version numérique du disque stroboscopique, soixante segments également espacés sont dessinés le long de la périphérie de l'écran LCD par un logiciel. Ensuite, en alimentant les LED responsables du rétroéclairage (borne BL de l'écran) avec un signal carré de fréquence appropriée (33,33 et 45,00 Hz), on verra les " notches " lorsque la vitesse de rotation de la platine vinyle (et donc de l'écran) coïncide avec la fréquence, sachant que les LED ne s'allument qu'à la demi-période (positive) du signal carré.

Ainsi, au lieu de dépendre d'une source lumineuse externe, ce sont les segments eux-mêmes qui s'illuminent et deviennent visibles à la fréquence souhaitée, d'où le nom de disque " actif ". Pour obtenir exactement les fréquences nécessaires, j'ai utilisé une méthode peu conventionnelle, mais qui s'est avérée très efficace, comme nous le verrons dans la description du micrologiciel. Il est important de souligner que les autres éléments graphiques de l'écran restent bien visibles, grâce à la persistance rétinienne.

Schéma de câblage

Comme le montre la **figure 4**, le schéma de câblage est simple. L'écran et le module de détection sont connectés, respectivement, aux broches standards des interfaces SPI et I²C sur le Raspberry Pi Pico, comme on peut le vérifier en consultant le brochage spécifié dans la fiche technique. Le connecteur JP1 est relié à l'écran LCD, et ses fils sont soudés directement sur la carte. L'alimentation de l'écran est assurée par le régulateur de la carte microcontrôleur, dont les broches GPIO ne supportent pas des tensions supérieures à 3,3 V.

Cela permet d'éviter tout problème d'interfaçage dès le départ. Le module du capteur intègre un LDO (régulateur à faible chute de tension), ce qui assure la compatibilité des niveaux de signal. Les boutons SW1 et SW2 sont contrôlés par les broches GP13 et GP14 (configurées en entrée avec une résistance *pull-up*) et permettent de sélectionner les fonctions disponibles et d'allumer et d'éteindre l'appareil. Le module U1, équipé de la puce TP4056 et de l'entrée Micro-USB, assure la charge d'une batterie au lithium polymère (LiPo) compacte de 3,7 V nominal et d'une capacité de 200 mAh. Cette batterie

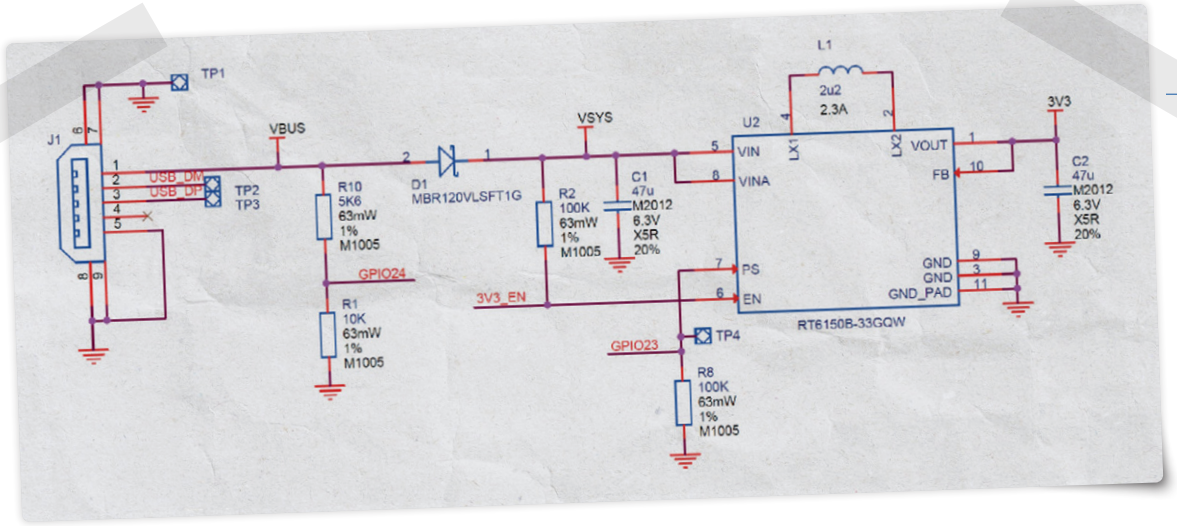


Figure 5. Cicuit d'alimentation du Raspberry Pi Pico. (Source : fiche technique du Raspberry Pi Pico [15])

est équipée d'un circuit de protection intégré, qui offre au moins une heure d'autonomie au projet.

Il est recommandé de charger la batterie (un chargeur de téléphone standard suffit) lorsque l'appareil est éteint, afin que la fin de la charge soit correctement indiquée. Le MOSFET Q1 [6] constitue l'interrupteur principal de l'appareil, normalement maintenu à l'état *OFF* par R2 (borne de la grille au même potentiel que la source). Lorsque l'un des interrupteurs est activé, la grille prend une valeur négative par rapport à la source à travers D1 ou D2, ce qui fait que Q1 conduit et que le signal atteint (via Q2) la borne VSYS de la carte Raspberry Pi Pico, activant ainsi le microcontrôleur.
Dès que l'exécution du micrologiciel commence, GP15 passe à l'état haut, activant Q3 (et par conséquent Q1), l'écran s'allume et on peut relâcher le bouton. Le MOSFET Q2 peut normalement être considéré comme " transparent " au flux de courant, jusqu'à ce que le câble USB soit connecté pour la programmation ; en effet, dans ce cas, la tension VBUS bloque Q2, empêchant le retour de la même tension vers la batterie à travers la diode interne Q1.
Il est possible d'éteindre le circuit en maintenant l'un des boutons enfoncé jusqu'à ce que l'écran s'éteigne. GP15 revient au niveau bas et Q3 passe à l'état OFF, puis le relâchement du bouton Q1 arrête le flux de courant. Comme mentionné précédemment, ce circuit marche/arrêt (couramment retrouvé sur le web, dans des versions similaires) peut être utilisé avec n'importe quel microcontrôleur, ne nécessite que deux broches libres (un seul bouton suffit) et quelques lignes de code. Le rôle de D3 et D4 est uniquement d'empêcher la tension de la batterie - normalement supérieure à 3,3 V - d'atteindre GP13 et GP14 via R2, D1 et D2, évitant ainsi tout risque de dommage.
Le schéma de la partie alimentation de la carte, extrait de la fiche technique du Raspberry Pi et présenté à la **figure 5**, permet de mieux comprendre le fonctionnement du circuit d'alimentation. On y remarque que la diode D1, qui empêche la tension appliquée à VSYS (comme c'est le cas ici) de circuler vers la prise Micro-USB et le diviseur résistif qui permet d'utiliser GPIO24 pour détecter la présence de la tension VBUS. Dans le schéma de câblage de ce projet, C1 empêche la mise sous tension immédiate lorsque la batterie est connectée, C2 et C3 sont sont utilisés comme dérivateurs d'alimentation habituels, et le buzzer émet un court bip pour signaler l'allumage et l'extinction de l'appareil.

Réaliser le prototype

J'ai construit le circuit sur une carte prototype, comme à mon habitude, mais cette fois, j'ai utilisé deux plaques de base circulaires de 60 mm de diamètre, répandues sur le marché. Elles sont empilées en insérant quatre entretoises, formant ainsi une structure en couches. En observant la **figure 6**, on peut voir que sur la face supérieure de la première plaque se trouvent les boutons, les diodes, le buzzer et le module de détection de mouvement. Au-dessus de ce dernier, l'écran circulaire est monté, avec son propre connecteur par lequel des fils descendent vers la face inférieure, qui abrite le Raspberry Pi Pico, comme le montre la **figure 7**. De ce côté, les connexions commencent - via un connecteur à cinq broches réalisé avec des contacts en bande - et passe à la seconde plaque de base, qui abrite les composants nécessaires à

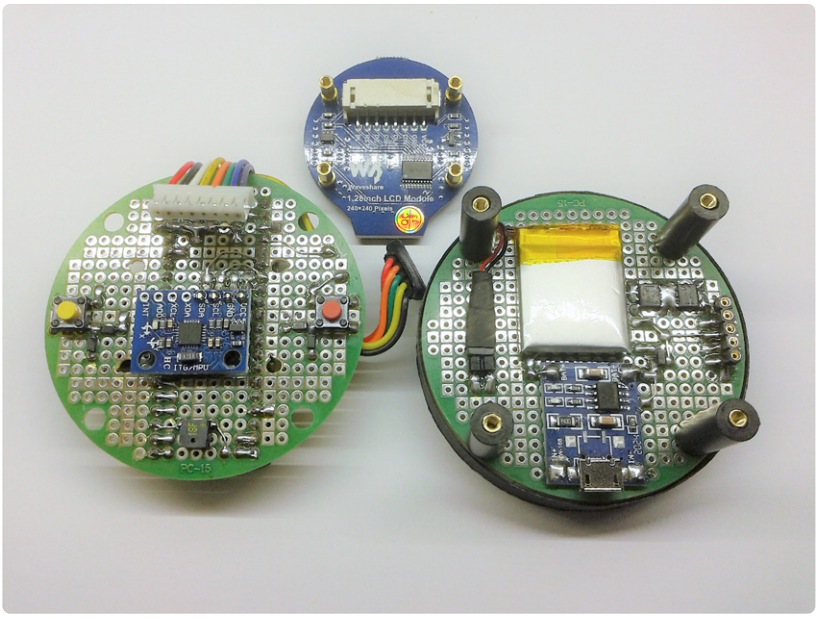
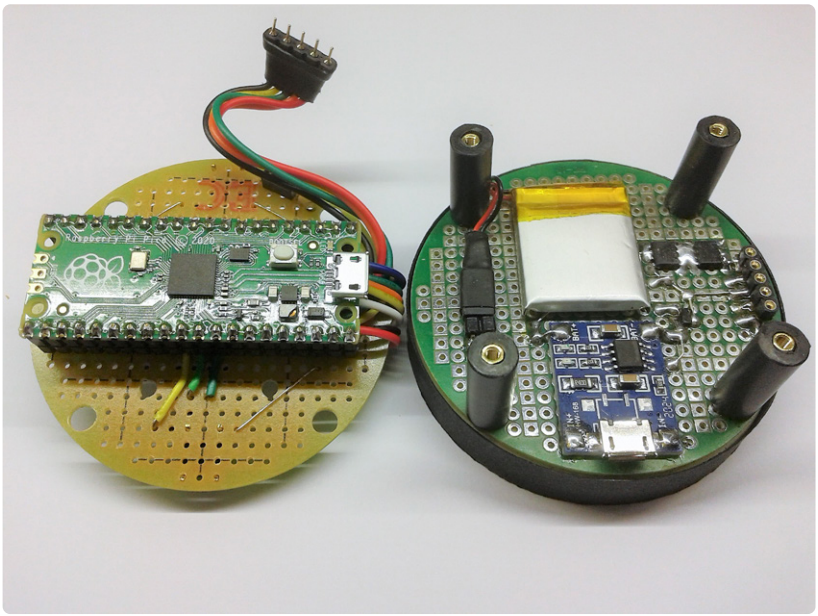


Figure 6. Le prototype - l'alimentation et l'afficheur



▲

Figure 7. Le prototype, présenté du côté Raspberry Pi Pico (à gauche).

l'alimentation (le module TP4056, la batterie, les MOSFET, etc.)

Cette plaque de base est fixée à un petit disque de même diamètre et d'un centimètre d'épaisseur (que j'ai fabriqué en bois, mais n'importe quel matériau rigide et léger convienne), percé en son centre pour laisser passer le pivot central du tourne-disque, pour un support stable aussi bien en mode niveau (tourne-disque immobile) qu'en mode stroboscopique (tourne-disque en rotation). J'ai utilisé des CMS et des traversants, selon la disponibilité, pour un assemblage qui a présenté des défis, compte tenu que tout devait tenir dans les dimensions des cartes, en veillant au centrage de l'écran et du capteur et en essayant d'obtenir un assemblage rigide et stable. Un circuit imprimé spécialement conçu aurait grandement simplifié la fabrication mais impliquant, à mon avis, trop d'efforts pour un seul prototype.

Le Raspberry Pi Pico et l'EDI Arduino

La comparaison peut paraître surprenante, mais en fait, le Pico embarque un microcontrôleur RP2040 et ressemble donc plus à un Arduino UNO (bien que la comparaison en termes de performances soit impitoyable) qu'aux nano-ordinateurs monocartes de la famille Raspberry Pi. De manière générale, l'EDI Arduino permet d'écrire du code pour programmer une vaste gamme de microcontrôleurs et de cartes, bien au-delà des modèles Arduino " officiels ". Le gestionnaire de cartes de l'EDI offre la possibilité d'installer ce que l'on appelle des " cores ", que l'on peut définir, en simplifiant, comme des modules logiciels supplémentaires capables d'assurer la compatibilité entre de nouvelles cartes et microcontrôleurs, même tiers, avec l'écosystème Arduino (EDI, bibliothèques existantes, structure des croquis).

La liste - assez impressionnante - des noyaux disponibles et donc des cartes et des microcontrôleurs compatibles est disponible sur [7]. Parmi les noyaux les plus populaires on retrouve ceux destinés aux microcontrôleurs AVR ATtiny et ATmega, ESP8266, ESP32, STM32, et à présent RP2040. Tous les dispositifs de la liste peuvent donc être programmés en utilisant l'EDI Arduino et le langage C/C++ - avec quelques astuces, il est possible d'inclure du code Assembleur dans les croquis - et en tirant parti de la plupart des bibliothèques déjà existantes pour Arduino. Ces noyaux, généralement dans le domaine public, sont écrits par des programmeurs passionnés et compétents. Le noyau développé par Earle F. Philhower, III pour le Raspberry Pi Pico surpasse même le noyau officiel d'Arduino en polyvalence et en performance. Toutes les informations et la documentation associées sont disponibles sur la page GitHub de l'auteur [8]. Lors de la première utilisation d'un Raspberry Pi Pico avec l'EDI Arduino, quelques étapes simples sont nécessaires. Après l'installation du noyau et la sélection de la bonne carte, (**figure 8**), branchez un câble micro USB à la carte et, avant de la connecter à l'ordinateur, appuyez sur le bouton BOOTSEL et maintenez-le enfoncé.

Sélectionnez le croquis classique *Blink* parmi les exemples et chargez-le. La LED intégrée du Raspberry Pi Pico commencera à clignoter, et le nom de la carte ainsi que son numéro de port seront affichés dans le menu *Outils* *Port*. À ce stade, vous pourrez éditer, compiler et charger des croquis directement, comme avec n'importe quelle carte Arduino.

Micrologiciel et fonctionnement

J'ai écrit le programme de ce projet en utilisant l'EDI Arduino 1.8.19 après avoir installé le noyau Raspberry Pi Pico Arduino, comme expliqué précédemment. L'écran est piloté par le contrôleur GC9A01, qui est peu répandu et n'est pas pris en charge par des bibliothèques librement disponibles, j'ai donc choisi d'utiliser la bibliothèque de démo fournie par le fabricant, en l'adaptant aux besoins du projet.

Tous les fichiers nécessaires (configuration, pilotes, polices, bibliothèques) et le fichier *main* sont contenus dans le dossier du croquis nommé *Raspberry_Pico_Livella_Digital_Strobo.ino* et sont tous accessibles depuis l'EDI Arduino et disponibles sur [9]. De plus, l'installation de deux autres bibliothèques est nécessaire, *MPU6050_light.2.1* [10], très bien documentée, pour contrôler le module capteur, et *RunningAverage 0.4.2* [11], utile pour calculer la moyenne mobile des données collectées en utilisant un buffer circulaire. Le listage est assez long - plusieurs centaines de lignes de code.

Je recommande donc à ceux qui sont intéressés de l'ouvrir dans l'éditeur et d'examiner les nombreux commentaires et détails que j'ai ajoutés. Cependant, je décrirai ici la logique du programme et les sections les plus intéressantes du code. En examinant le listage, vous devriez immédiatement remarquer la présence des fonctions `setup1()` et `loop1()` en plus de celles présentes par défaut dans chaque croquis Arduino. Cela est dû à la capacité double cœur du microcontrôleur RP2040. Chaque cœur peut exécuter indépendamment les

instructions contenues dans les fonctions `setup()` et `loop()` respectives. Au démarrage, après l'inclusion habituelle des bibliothèques et fichiers externes, la définition des variables et instances nécessaires, la routine principale de configuration, core0, assure la configuration des broches, l'initialisation de l'EEPROM, des protocoles de communication SPI et I²C, du capteur MPU-6050, et de l'afficheur.

Après vérification de la tension de la batterie (GP15 au niveau logique haut), l'alimentation est activée. Ensuite, une série d'instructions conditionnelles, basées sur l'état des boutons, gèrent les routines de calibrage, les premières informations affichées par l'écran et définissent le mode de fonctionnement. À ce stade, il est important de noter l'utilisation récurrente des instructions `rp2040.idleOtherCore()` et `rp2040.resumeOtherCore()`, qui sont nécessaires car les deux cœurs ne peuvent pas écrire simultanément dans la mémoire flash et d'autres périphériques. Pour assurer une exécution correcte et sans blocage du code, chaque cœur doit, si nécessaire, être en mesure de se mettre en pause et de réactiver l'autre par la suite.

Dans le cas où le mode niveau à bulle est sélectionné la boucle `main()` (toujours sur `core0`) est chargée de traiter les données du capteur MPU-6050 (coordonnées x et y), de traiter la moyenne mobile, de formater les chaînes avec la fonction `sprintf()` [12], d'afficher les données et de dessiner la "bulle" en mouvement sur l'écran avec les fonctions `Paint_DrawString_EN()` et `Paint_DrawCircle()` incluses dans la bibliothèque *GUI_paint.cpp* fournie par son fabricant. L'effet de mouvement de la bulle est obtenu simplement en dessinant un cercle noir avec les coordonnées actuelles et, immédiatement après, un cercle vert ou blanc avec les nouvelles coordonnées, dans un processus continu d'effacement et de réécriture.

Grâce à l'augmentation de la fréquence d'horloge du microcontrôleur à 250 MHz, paramétrable depuis le menu approprié de l'EDI - et au tampon de moyenne mobile, l'effet de mouvement est suffisamment réaliste. Sinon, si le mode *strobe* a été sélectionné, `loop()` exécute uniquement la routine qui pilote les LED de rétroéclairage de l'écran. Juste en cas d'appui simultané sur les deux boutons, la fonction qui affiche la tension de la batterie est appelée. `Setup1()` ne contient que l'instruction `delay(1000)` qui retarde le démarrage de `loop1()` de 1 s. Le second noyau exécute ensuite les fonctions `drawBackgroundElements()` et `buttonsOperation()`, qui sont chargées, respectivement, de redessiner en permanence les éléments fixes de l'arrière-plan de l'écran et de détecter et gérer les pressions sur les boutons.

La répartition des tâches entre les deux cœurs améliore considérablement la vitesse de réponse de l'interface et la fluidité des graphiques. Chaque fonction utilisée dans le code est décrite et commentée. Je vais ajouter ici quelques brefs commentaires explicatifs pour les fonctions qui me paraissent particulièrement intéressantes. La fonction `drawStrobeMarks()`, inspirée d'un exemple disponible en ligne [13], a été simplifiée et adaptée pour ce projet. Elle se charge de dessiner 60 segments équidistants, le long de la périphérie de l'écran. Partant du concept de division de la périphérie en 60 parties, chacune représentée par la valeur de 2π/60 radians. Une boucle `for` est utilisée pour calculer ces coordonnées, qui sont ensuite utilisées par la fonction `Paint_DrawLine()` pour dessiner chaque segment sur l'écran.

La fonction `flashBacklight()` gère le rétroéclairage en mode stroboscopique, en générant une onde carrée à une fréquence de 33,33 ou 45,00 Hz avec un cyclique de 50%, ce qui permet d'éclairer l'écran pendant la demi-période positive, rendant les segments sur la circonférence visibles aux intervalles de temps appropriés. La durée exacte, en µs, de chaque demi-période est obtenue en mesurant le temps écoulé avec la fonction `rp2040.getCycleCount()`, qui renvoie le nombre de cycles d'horloge exécutés depuis la mise sous tension du microcontrôleur. À une fréquence d'horloge de 250 MHz, un cycle dure 1/250 000 000 = 0,000000004 s. Cela donne une granularité de mesure de 4 ns. Une microseconde écoulée correspond à 250 cycles.

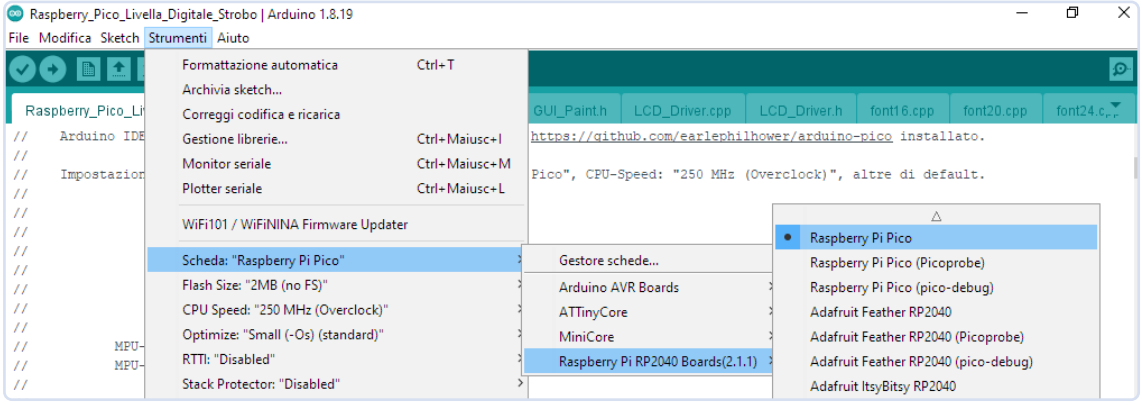
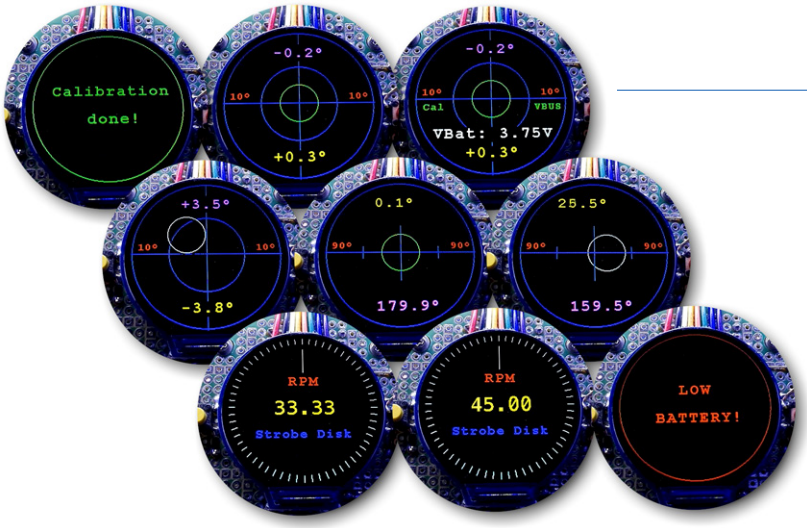


Figure 8. Sélection de la carte Raspberry Pi Pico dans l'EDI Arduino.



▲

Figure 9. Quelques photos du prototype.

L'onde carrée est générée sur la broche GP22 appelée *DEV_BL_PIN*, qui contrôle le rétroéclairage de l'écran. Sur le prototype, les fréquences obtenues, mesurées avec un compteur de fréquence/tachymètre, étaient de 33,333 Hz et 45,000 Hz, avec une précision extrême et une variation de seulement quelques millièmes de Hertz. De plus, la fonction `rp2040.fifo.push(pushCount)` est utilisée pour transmettre au deuxième cœur le nombre (divisé par 60) des demi-périodes écoulées, afin d'afficher un segment de référence plus long à chaque rotation effectuée. J'ai déjà mentionné l'utilisation d'une EEPROM pour stocker les paramètres de calibrage. Toutefois, le microcontrôleur RP2040 n'est pas doté d'une telle mémoire. Cependant, la bibliothèque *EEPROM.h* permet de la simuler en utilisant une partie de la mémoire flash. Avec l'instruction `EEPROM.begin(4096)` les sections de l'EEPROM émulée sont copiées en RAM pour que le programme puisse y accéder pour les opérations de lecture et d'écriture.

La fonction `storeOffsetsValues()` se charge de calculer les valeurs de décalage des mesures, de les stocker une structure définie par l'utilisateur (`struct`) et de les écrire, en bloc, dans l'EEPROM grâce à la fonction `EEPROM.put(eeAddress, offsetValues)`. À chaque mise sous tension, la fonction `setOffsetsValues()` lit les données en utilisant `EEPROM.get(eeAddress, offsetValues)` et fixe les valeurs d'offset correctes pour le gyroscope et l'accéléromètre. Le type de données `struct` (type composé défini par l'utilisateur) permet de traiter simultanément comme un seul bloc plusieurs valeurs appartenant à des types de données différents (byte, int, float, string, boolean, char). Il est important de préciser que la bibliothèque *EEPROM.h* et les fonctions `EEPROM.begin() et rp2040.xxxxx()` du programme sont spécifiques au cœur *Raspberry Pi Pico Arduino*.

Utilisation pratique

Pour conclure cet article, il convient de donner quelques recommandations pratiques sur l'utilisation de cet appareil. Avant tout, vous devez effectuer un calibrage, qui consiste à placer l'appareil au-dessus d'une surface de référence stable, plane et horizontale. Sans le déplacer, appuyez simultanément sur les deux boutons : le message *Calibration, wait...* apparaît sur l'écran et, après quelques secondes, le message *Calibration done!* s'affiche suivi de l'écran de fonctionnement du niveau à bulle circulaire, avec un indicateur vert *CAL* - visible uniquement dans ce cas - qui indique que le calibrage a été effectué avec succès. Cela signifie que les valeurs de référence pour la correction d'éventuelles erreurs d'offset sont enregistrées dans la mémoire de l'appareil et qu'elles seront automatiquement utilisées à chaque redémarrage.

Pour éteindre l'appareil, il suffit de maintenir enfoncé l'un des boutons pendant une période prolongée. En utilisation normale, une pression sur le bouton rouge allume l'instrument en mode niveau à bulle circulaire (des pressions courtes ultérieures sur le même bouton permettent de basculer entre le mode circulaire et le mode linéaire). Vous placez ensuite le niveau à bulle sur la platine vinyle, en utilisant la tige comme référence. Si la platine vinyle n'est pas parfaitement horizontale, vous devrez, selon votre configuration, ajuster les supports ou le système de suspension, jusqu'à ce que la bulle électronique sur l'écran se stabilise au centre et devienne verte, et que les indications numériques se rapprochent le plus possible de zéro.

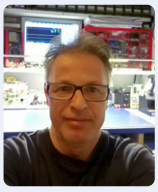
Utilisez le bouton jaune pour allumer et activer le mode stroboscopique (une nouvelle pression permet de basculer brièvement le mode de test entre 33,33 et 45,00 RPM). Ensuite, lorsque la platine vinyle est en rotation, de préférence dans une lumière ambiante faible, la vitesse sera vérifiée. Si elle est correcte, les segments sur le contour de l'écran apparaîtront immobiles. S'ils semblent tourner dans le sens horaire, cela indique que la vitesse est trop élevée ; s'ils semblent tourner dans le sens antihoraire, cela indique que la vitesse est trop basse. Le segment central le plus long, qui s'allume brièvement à chaque tour. Si la vitesse est correcte, il apparaîtra toujours dans la même position. Certaines platines vinyles permettent de régler le nombre de tours à l'aide d'une commande spéciale, d'autres à l'aide de trimmers plus ou moins accessibles. Sur les modèles à entraînement par courroie, une vitesse incorrecte peut aussi être causée par l'usure de la courroie. D'autres causes générales peuvent être une lubrification insuffisante du pivot ou des dysfonctionnements liés à l'alimentation ou au moteur. En cas de doute, il est toujours recommandé de consulter un technicien spécialisé pour une révision complète. Enfin, la **figure 9** montre quelques captures d'écran de l'appareil en fonctionnement.

Wrapping Up With a Video

Nous avons abordé divers sujets dans ce projet, comme l'utilisation du Raspberry Pi Pico et son potentiel, sa programmation sous l'EDI Arduino tout en exploitant sa fonction double cœur, les capteurs de mesure inertielle MEMS, avec un exemple pratique d'interfaçage et d'utilisation, l'utilisation de l'écran LCD circulaire,

le fonctionnement du disque stroboscopique, une méthode particulière pour générer une onde carrée avec une fréquence très précise, la gestion de l'alimentation avec un bouton, utilisable sur n'importe quel microcontrôleur avec seulement quelques lignes de code, et la mesure de la tension de la batterie. Bien que la prémisse puisse suggérer un projet destiné à un public ciblé, je pense que même tous les *maker* qui ne possèdent pas une platine vinyle et une collection de disques vinyles peuvent trouver des informations utiles et des idées intéressantes à développer et à utiliser pour de nouveaux projets différents. Enfin, une courte vidéo de démonstration est disponible sur YouTube [14].

230358-04



À propos de l'auteur

Depuis son enfance, Antonello Della Pia est passionné par l'électricité et les appareils électroniques. Il est diplômé en génie électrique. Antonello a toujours cultivé et développé sa passion pour l'électronique analogique et numérique. Actuellement, il s'amuse avec les microcontrôleurs et la programmation, cherchant à améliorer ses compétences. Antonello aime développer et proposer des projets aussi originaux que possible et, espère-t-il, intéressants.

Question or Comments?

Contactez Elektor (redaction@elektor.fr).

Liste des composants

Résistances

R1 = 3,3 kΩ (toutes 1/4 W 1%)
R2 = 100 kΩ
R3 = 470 kΩ
R4 = 330 Ω
R5 = 100 Ω

Condensateurs

C1, C2 = 100 nF, 50 V, multicouches en polyesther ou en céramique
C3 = 100 µF, 10 V, électrolytique

Semi-conducteurs

Q1, Q2 = FDS9435AMOSFET à canal P
Q3 = 2N7002 MOSFET à canal N
D1, D2, D3, D4 = BAT46W Diode Schottky, petits signaux
U1 = TP4056 Module de charge, sans protection
U2 = GY-521 MPU-6050 Module
U3 = Carte Raspberry Pi Pico

Divers

Écran = Écran LCD rond Waveshare de 1,28 pouce 240x240, 19192
BUZZER = Buzzer, CMS
SW1, SW2 = Bouton-poussoir N.O., pour circuit imprimé
BAT1 = 3.7 V Batterie LiPo, 200 mAh, avec protection
Connecteurs à bandes M/F
Spacers
carte de prototypage ronde, PC-15, 60 mm

Produits

- **Raspberry Pi Pico RP2040**
  www.elektor.fr/19562
- **Dogan Ibrahim, *Hardware Projects for Raspberry Pi,* Elektor 2014 (E-book)**
  www.elektor.fr/16969

LIENS

[1] Page web du module afficheur LCD de Waveshare : http://tinyurl.com/5n87a8pt
[2] Raspberry Pi : https://fr.wikipedia.org/wiki/Raspberry_Pi
[3] Page web du Raspberry Pi Pico : https://raspberrypi.com/products/raspberry-pi-pico/
[4] Tutoriel MPU6050 sur le site LastMinuteEngineers : http://tinyurl.com/rxr8av6k
[5] Fiche technique MPU6050 : http://tinyurl.com/mwr5fwb6
[6] Fiche technique FDS9435A : https://onsemi.com/pdf/datasheet/fds9435a-d.pdf
[7] Unofficial list of 3rd party boards support URLs — Arduino IDE : http://tinyurl.com/5edvv332
[8] Raspberry Pi Pico Arduino core by Earle F. Philhower, III : http://tinyurl.com/489kwb3k
[9] Logiciel pour ce projet : https://elektormagazine.fr/230358-04
[10] MPU6050_light — Bibliothèque Arduino : https://github.com/rfetick/MPU6050_light
[11] RunningAverage — Bibliothèque Arduino : https://github.com/RobTillaart/RunningAverage
[12] C library function — sprintf() — tutorialspoint : http://tinyurl.com/58ymn9wz
[13] ArduinoWatch — moononournation : https://github.com/moononournation/ArduinoWatch
[14] Vidéo de démo — YouTube: https://youtu.be/tun6wH5gKDA
[15] Fiche technique du Raspberry Pi Pico : https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf