elektor books

# Raspberry Pi Pico W en un tournemain

## Maîtrise de la puce **RP2040** avec plus de 60 projets à réaliser et à programmer


Dogan Ibrahim

elektor
créer > partager > vendre

# Raspberry Pi Pico W
## en un tournemain
### Maîtrise de la puce RP2040 avec plus de 60 projets à réaliser et à programmer

**Dogan Ibrahim**

elektor
design > share > earn

L'éditeur remercie d'avance le lecteur qui prendra la peine de lui signaler les erreurs éventuelles qui auront échappé à sa vigilance (écrire à redaction@elektor.fr).

● **ISBN 978-2-86661-214-6 Version papier**
**978-2-86661-215-3 Version numérique**

● 1e édition — 1er tirage - 02/2023

● Maquette : www.d-vision.nl | Julian van den Berg
Coordination : Mariline Thiebaut-Brodier
Traduction : Magda Lorne
Imprimé en Allemagne par Media-Print Informationstechnologie

# SOMMAIRE

# Préface

Un microcontrôleur est essentiellement un ordinateur monopuce comprenant une unité centrale, une mémoire, des circuits d'entrée-sortie, des temporisateurs, des circuits d'interruption, des circuits d'horloge et plusieurs autres circuits et modules, tous logés dans une seule puce de silicium. Les capacités et la vitesse des premiers microcontrôleurs étaient limitées, et leur consommation était considérable. La plupart des premiers microcontrôleurs étaient des processeurs à 8 bits avec des vitesses d'horloge de l'ordre du mégahertz, et ils ne disposaient que de centaines d'octets de mémoire de programme et de données. Ces microcontrôleurs étaient traditionnellement programmés avec le langage assembleur des processeurs cibles. Aujourd'hui, les microcontrôleurs à 8 bits sont encore couramment utilisés, notamment dans le cadre de petits projets où il n'est pas nécessaire de disposer d'une grande quantité de mémoire ou d'une vitesse élevée. Avec l'évolution de la technologie des puces, nous disposons désormais de microcontrôleurs à 32 et 64 bits, avec plusieurs gigaoctets de mémoire et des vitesses de l'ordre de plusieurs gigahertz. Les microcontrôleurs sont aujourd'hui programmés avec des langages de haut niveau tels que C, C#, BASIC, PASCAL, JAVA, etc.

Le Raspberry Pi Pico est une carte à microcontrôleur très puissante, conçue spécialement pour l'informatique interactive. Le lecteur doit savoir que ce type de carte est très différent des ordinateurs monocartes comme le Raspberry Pi 4 (et les autres membres de la famille Raspberry Pi). Il n'y a pas de système d'exploitation sur le Raspberry Pi Pico. Les cartes comme le Raspberry Pi Pico ne peuvent exécuter qu'une seule tâche. Elles sont utilisées dans des applications de commande et de surveillance en temps réel.

Le Raspberry Pi Pico est doté de la puce ARM Cortex-M0+ RP2040 à double cœur, rapide et très efficace, fonctionnant jusqu'à 133 MHz. La puce intègre 264 Ko de SRAM et 2 Mo de mémoire Flash. Ce qui rend le Raspberry Pi Pico très attrayant, c'est son grand nombre de broches GPIO, de fonctions de temporisation précises et de modules d'interfaçage avec des périphériques couramment utilisés, tels que SPI, $I^2C$, UART, PWM.

Sorti en 2022, le Raspberry Pi Pico **W** est le dernier-né de la famille des cartes à microcontrôleur Pico. Le « Pico W » est presque identique au « Pico » standard, car il présente une différence majeure : par rapport aux autres membres de la famille Pico, le « Pico W » dispose d'un module Wi-Fi embarqué, permettant ainsi à la carte d'être utilisée dans de nombreux projets de communication, de commande, et surtout dans les projets de type Internet des Objets. La carte Pico W peut également communiquer par Bluetooth, mais le micrologiciel Bluetooth n'était pas prêt au moment de la rédaction de ce livre.

Le Raspberry Pi Pico standard et le Raspberry Pi Pico W sont faciles à programmer avec certains des langages de haut niveau les plus courants, tels que MicroPython ou C/C++. L'internet regorge de notes d'application, de tutoriels et de fiches techniques sur l'utilisation du Pico ou du Pico W.

Ce livre est une introduction à l'utilisation de la carte de développement à microcontrôleur Raspberry Pi Pico W avec le langage de programmation MicroPython. Tous les projets du

livre sont décrits dans l'environnement de développement intégré (EDI) Thonny, nous recommandons aux lecteurs d'utiliser cet EDI. Le livre contient de nombreux projets testés et fonctionnels, couvrant presque tous les aspects du Raspberry Pi Pico W. À l'exception des montages qui requièrent le Wi-Fi, tous les projets du livre peuvent également être reproduits avec le Raspberry Pi Pico standard sans aucune modification.

En principe, la description des projets est structurée de la manière suivante afin de faciliter la lecture :

- Titre
- Brève description
- Objectif
- Schéma fonctionnel
- Câblage
- Listing du programme avec description complète

J'espère que vos prochains montages avec un microcontrôleur seront réalisés avec une carte Raspberry Pi Pico W, et que ce livre vous sera utile dans le développement de vos projets.

**Dogan Ibrahim**

# Chapitre 1 • Entrailles du Raspberry Pi Pico W

## 1.1 Préambule

Le Raspberry Pi Pico W est une carte dotée d'un microcontrôleur, conçue par la Fondation Raspberry Pi. Cette carte contient le microcontrôleur RP2040. Dans ce chapitre, nous examinons en détail le matériel de la carte à microcontrôleur Raspberry Pi Pico W. À partir de maintenant, cette carte avec microcontrôleur sera simplement appelée « Pico ».

## 1.2 Caractéristiques matérielles du Pico

Le Pico est une carte à microcontrôleur très bon marché (de 5 à 9 €), animée par la puce RP2040 avec un processeur Cortex-M0+ à double cœur. La **figure 1.1** est une vue du dessus de la petite carte Pico. Au milieu de la carte se trouve la minuscule puce RP2040 de 7×7 mm logée dans un boîtier QFN-56. Sur les deux bords longs de la carte, il y a au total 40 trous métalliques de couleur or pour les broches GPIO (*General Purpose Input/Output*). Vous devez souder des broches dans ces trous pour pouvoir facilement connecter des composants externes à la carte. Les trous sont repérés en commençant par le numéro 1 dans le coin inférieur gauche de la carte (sous le connecteur micro-USB) et les numéros augmentent tout autour de la carte jusqu'au numéro 40 qui se trouve dans le coin supérieur gauche (au-dessus du connecteur micro-USB). La carte est compatible avec une platine d'essai (c'est-à-dire avec un écartement des broches de 0,1 pouce). Après le soudage des broches, la carte peut être montée sur une platine d'essai pour faciliter la connexion aux broches GPIO avec des fils de liaison. À chaque extrémité de ces deux rangées de trous, il y a des découpes circulaires qui permettent d'enficher le Pico sur d'autres cartes sans avoir de broches physiques.

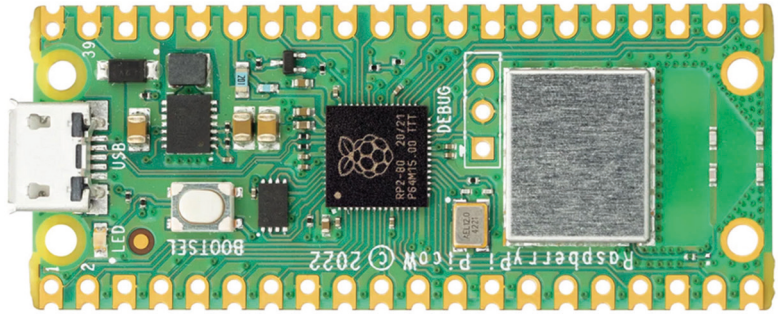


*Figure 1.1 : vue du dessus de la carte Pico.*

Sur l'un des bords de la carte se trouve le port micro-USB B qui permet d'alimenter la carte et de la programmer. À côté de ce port USB, il y a une LED « utilisateur » embarquée qui peut être utilisée pendant le développement des programmes. À côté de cette LED, il y a un bouton nommé BOOTSEL qui sert pendant la programmation du microcontrôleur comme vous le verrez dans les chapitres suivants. À côté de la puce RP2040, il y a trois trous où il est possible d'établir des connexions externes. Celles-ci sont utilisées pour déboguer vos programmes à l'aide du *Serial Wire Debug* (SWD). À l'autre extrémité de la carte se trouve le module Wi-Fi 2,4 GHz à bande unique (802.11n). L'antenne intégrée se trouve à côté du module Wi-Fi.

La **figure 1.2** montre le dessous de la carte Pico. Ici, toutes les broches GPIO sont identifiées par des lettres et des chiffres. Vous remarquerez les types de lettres et de chiffres suivants :

| | |
|---|---|
| GND | masse de l'alimentation (masse numérique) |
| AGND | masse de l'alimentation (masse analogique) |
| 3V3 | alimentation de +3,3 V (sortie) |
| GP0 à GP22 | GPIO numériques |
| GP26_A0 à GP28_A2 | entrées analogiques |
| ADC_VREF | tension de référence du convertisseur analogique-numérique |
| TP1 à TP6 | points de test |
| SWDIO, GND, SWCLK | interface de débogage |
| RUN | broche RUN par défaut. Appliquer un signal de niveau BAS pour réinitialiser le RP2040. |
| 3V3_EN | cette broche active par défaut l'alimentation de +3,3 V Il est possible de désactiver le +3,3 V en connectant cette broche à un signal de niveau BAS. |
| VSYS | tension d'entrée du système (1,8 V à 5,5 V) utilisée par l'alimentation monophasée (SMPS) intégrée pour produire une tension de +3,3 V pour la carte. |
| VBUS | tension d'entrée du port micro-USB (+5 V) |

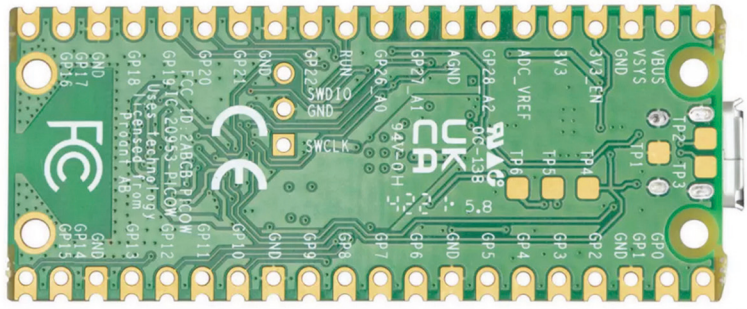


*Figure 1.2 : vue du dessous de la carte Pico.*

Certaines des broches GPIO sont utilisées pour les fonctions internes de la carte. Ce sont les suivantes :

| | |
|---|---|
| **GP29 (entrée)** | utilisée en mode CA/N (ADC3) pour mesurer VSYS/3 |
| **GP24 (entrée)** | détection VBUS ; niveau HAUT si VBUS présent, sinon niveau BAS |
| **GP23 (sortie)** | commande la broche d'économie d'énergie de l'alimentation SMPS embarquée |

Les spécifications de la carte Pico sont les suivantes :

- Processeur à 32 bits RP2040 Cortex-M0+ à double cœur, fonctionnant à 133 MHz
- Mémoire flash Q-SPI de 2 Mo
- Mémoire SRAM de 264 Ko
- 26 GPIO (broches d'entrée/sortie compatibles +3,3 V)
- 3 broches de convertisseur analogique-numérique à 12 bits
- Bibliothèques de virgule flottante accélérées sur puce
- Puce sans fil monobande Infineon CYW43439 intégrée, interface sans fil à 2,4 GHz (802.11b/g/n) et Bluetooth 5.2 (non pris en charge au moment de la rédaction)
- Port SWD (*Serial Wire Debug*)
- Port micro-USB (USB 1.1) pour l'alimentation (+5 V) et les données (programmation)
- Interface de bus : 2× UART, 2× $I^2C$, 2× SPI
- 16 canaux MLI (modulation de largeur d'impulsion)
- 1× *timer* (avec 4 alarmes), 1× compteur en temps réel
- Capteur de température embarqué
- LED intégrée, reliée à GPIO0, pilotée par le module Wi-Fi 43439
- Carte crenelée permettant une soudure directe sur des cartes porteuses
- 8× automate fini avec E/S programmable (PIO) pour mettre en œuvre des interfaces personnalisées
- Programmation en MicroPython, C, C++
- Programmation par glisser-déposer en utilisant une mémoire de masse connectée par USB

**Nota :** sur la carte Raspberry Pi Pico standard, la LED intégrée est reliée à la broche GP25 et est disponible pour l'utilisateur. Sur la carte Raspberry Pi Pico W, la LED intégrée est pilotée par le module Wi-Fi 43439.

Les broches GPIO du Pico supportent une tension de +3,3 V. Il est donc important de veiller à ne pas dépasser cette tension lors de la connexion aux broches GPIO de périphériques d'entrée externes. Pour relier aux broches GPIO du Pico des composants avec des sorties en +5 V, il faut se servir de circuits logiques de conversion du +5 V en +3,3 V ou de circuits diviseurs de tension avec des résistances.

La **figure 1.3** montre un circuit diviseur de tension avec des résistances qui permet d'abaisser le +5 V à +3,3 V.

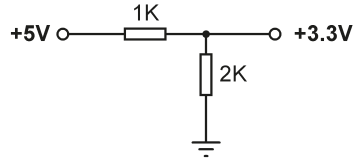


*Figure 1.3 : circuit diviseur de tension avec des résistances.*

## 1.3 Comparaison avec l'Arduino UNO

Arduino UNO est l'une des cartes de développement à microcontrôleur les plus populaires, utilisée par les étudiants, les ingénieurs en exercice et les amateurs. Le **tableau 1.1** présente une comparaison entre le Raspberry Pi Pico W et l'Arduino UNO. Il ressort clairement de ce tableau que le Pico W est beaucoup plus rapide que l'Arduino UNO, qu'il dispose d'une mémoire flash et d'une mémoire de données plus grosses, qu'il offre le Wi-Fi, qu'il dispose de plus de broches d'entrée/sortie numériques et d'un capteur de température intégré. L'Arduino UNO fonctionne en +5 V et ses broches GPIO sont compatibles avec le +5 V. Les avantages de l'Arduino UNO sont peut-être que cette carte possède une mémoire EEPROM intégrée et que son CA/N est à 6 canaux au lieu de 3.

| Fonction | Raspberry Pi Pico W | Arduino UNO |
|---|---|---|
| Microcontrôleur | RP2040 | Atmega328P |
| Noyau et bits | Double cœur, 32 bits, Cortex-M0+ | Simple cœur, 8 bits |
| RAM | 264 Ko | 2 Ko |
| Flash | 2 Mo | 32 Ko |
| Vitesse de la CPU | 48 MHz à 133 MHz | 16 MHz |
| EEPROM | Aucune | 1 Ko |
| Wi-Fi | Puce sans fil CYW43439 | Aucun |
| Alimentation | +5 V par le port USB | +5 V par le port USB |
| Autre alimentation | 2 à 5 V via la broche VSYS | 7 à 12 V |
| Tension de fonctionnement du MCU | +3,3 V | +5 V |
| Nombre de GPIO | 26 | 20 |
| Canaux de CA/N | 3 | 6 |
| Liaison UART matérielle | 2 | 1 |
| Bus $I^2C$ matériel | 2 | 1 |
| Bus SPI matériel | 2 | 1 |
| MLI matérielle | 16 | 6 |
| Langages de programmation | MicroPython, C, C++ | C (EDI Arduino) |
| LED embarquée | 1 | 1 |
| Coût | 9 € | 30 € |

*Tableau 1.1 : comparaison entre Raspberry Pi Pico W et Arduino UNO.*

## 1.4 Conditions de fonctionnement et mise sous tension du Pico

Les conditions de fonctionnement recommandées pour le Pico sont les suivantes :

- Température de fonctionnement : −20 ºC à +85 ºC
- Tension VBUS : +5 V ±10%
- Tension VSYS : +1,8 V à +5,5 V

Une alimentation monophasée (SMPS) intégrée produit les +3,3 V nécessaires à l'alimentation du RP2040 à partir d'une gamme de tensions d'entrée allant de 1,8 V à +5,5 V. Par exemple, trois piles alcalines AA permettent de fournir +4,5 V pour alimenter le Pico.

Le Pico peut être alimenté de plusieurs façons. La méthode la plus simple consiste à raccorder le port micro-USB sur une source d'alimentation de +5 V, comme le port USB d'un ordinateur ou un bloc d'alimentation avec une sortie de +5 V. L'entrée VSYS (voir **figure 1.4**) est alors alimentée par une diode Schottky. La tension sur l'entrée VSYS est donc la tension VBUS moins la chute de tension aux bornes de la diode Schottky (environ +0,7 V). Les broches VBUS et VSYS peuvent être court-circuitées si la carte est alimentée par un port USB externe de +5 V. Cela augmentera légèrement la tension d'entrée et réduira donc les ondulations sur VSYS. La tension VSYS est fournie à l'alimentation SMPS par le convertisseur RT6150 qui produit une tension fixe de +3,3 V pour l'unité centrale et les autres composants de la carte. La tension sur VSYS est divisée par trois et est disponible sur la broche d'entrée analogique GPIO29 (ADC3) qui peut facilement être surveillée. La broche GPIO24 vérifie l'existence de la tension VBUS et est au niveau logique HAUT si la tension VBUS est présente.

Une autre méthode pour alimenter le Pico consiste à appliquer une tension externe (+1,8 V à +5,5 V) directement sur l'entrée VSYS (par exemple avec des piles ou une alimentation externe). Vous pouvez également utiliser l'entrée USB et l'entrée VSYS ensemble pour alimenter le Pico, par exemple pour fonctionner à la fois avec des piles et le port USB. Dans ce cas, il faut alors placer une diode Schottky sur l'entrée VSYS pour éviter que les alimentations n'interfèrent entre elles. La tension la plus élevée alimentera VSYS.

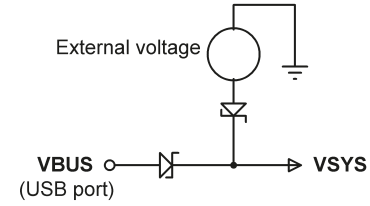


*Figure 1.4 : alimentation du Pico.*

## 1.5 Brochage du microcontrôleur RP2040 et de la carte Pico

La **figure 1.5** montre le brochage du microcontrôleur RP2040, logé dans un boîtier à 56 broches. Le brochage de la carte Pico est présenté en détail à la **figure 1.6**. Comme vous pouvez le constater, la plupart des broches ont plusieurs fonctions. Par exemple, la broche GPIO0 (broche 1) est à la fois la broche UART0 TX, la broche I2C0 SDA et la broche SPI0 RX.

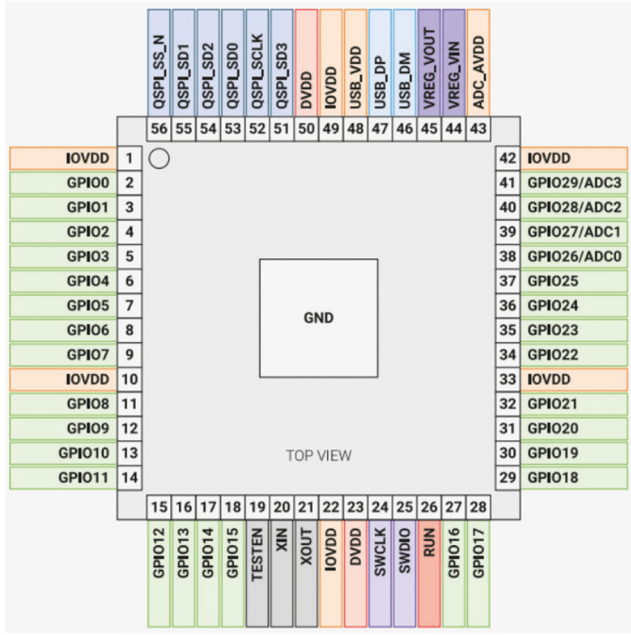


*Figure 1.5 : brochage du microcontrôleur RP2040.*

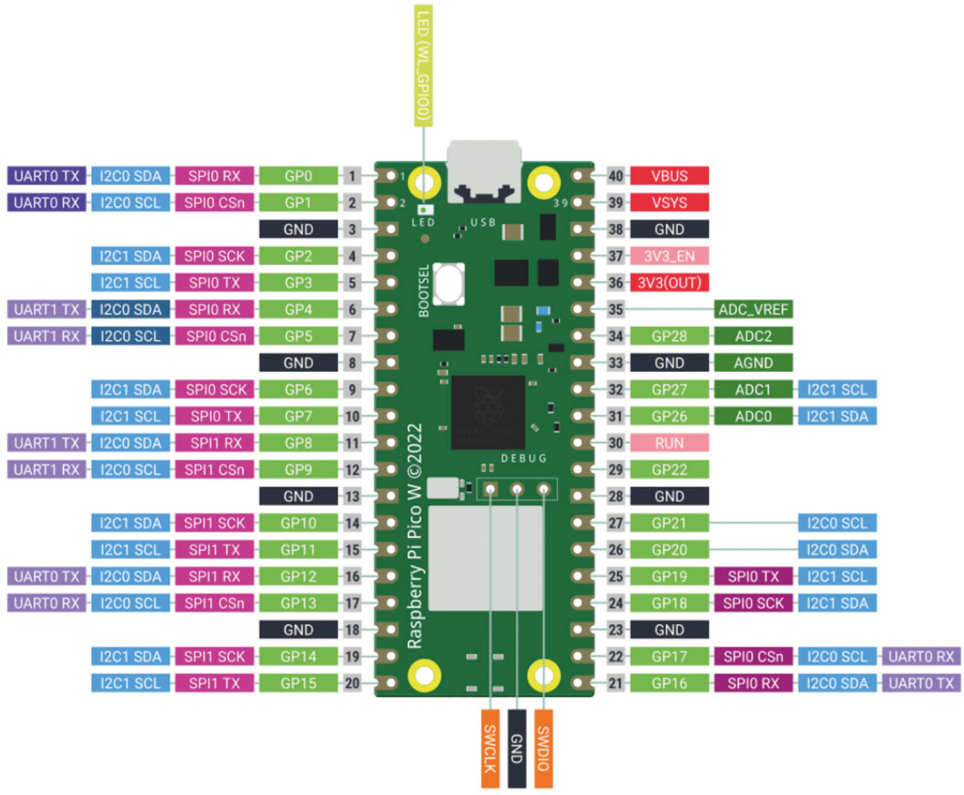


*Figure 1.6 : brochage du Pico.*

La **figure 1.7** présente un schéma fonctionnel simplifié de la carte Pico. Notez que les broches GPIO sont directement reliées de la puce au port GPIO. Les broches GPIO 26 à 28 peuvent être utilisées soit comme GPIO numériques, soit comme entrées de CA/N. Les entrées du CA/N GPIO26-29 ont des diodes en polarisation inverse jusqu'à 3 $V_{alim}$ et donc la tension d'entrée ne doit pas dépasser 3,3 V + 300 mV. Un autre point remarquable est que si le RP2040 n'est pas alimenté, les tensions appliquées sur les broches GPIO26-29 peuvent fuir à travers la diode vers l'alimentation (il n'y a pas de problème avec les autres broches GPIO et une tension peut être appliquée lorsque le RP2040 n'est pas alimenté).

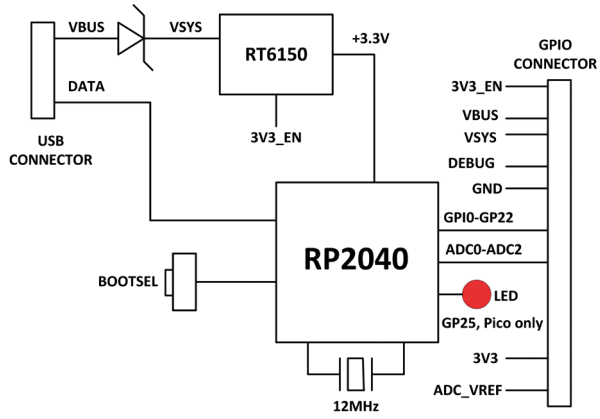


*Figure 1.7 : schéma fonctionnel simplifié.*

## 1.6 Autres cartes à base de microcontrôleur RP2040

Il existe de nombreuses autres cartes de développement basées sur le microcontrôleur RP2040. Au moment de la rédaction de ce livre, on peut citer entre autres :

- Adafruit - Feather RP2040
- Adafruit - ItsyBitsy RP2040
- Adafruit - QT Py RP2040
- Pimoroni - PicoSystem
- Pimoroni - Tufty 2040
- Arduino Nano RP2040 Connect
- SparkFun - Thing Plus RP2040
- Pimoroni - Pico Explorer Base
- Pimoroni - Pico Lipo
- SparkFun - MicroMod RP2040 Processor
- SparkFun - Pro Micro RP2040
- Pimoroni - Pico RGB Keypad Base
- Pimoroni - Pico Omnibus
- Pimoroni - Pico VGA Demo Base
- Cytron - Maker Pi RP2040 development board
- Technoblogy - Minimal RP2040 board
- Pimoroni - Tiny 2040
- Seeed Studio - Wio RP2040 Mini development board
- Seeed Studio - XIAO RP2040 development board
- Etc.

# Chapitre 2 • Programmation du Raspberry Pi Pico W

## 2.1 Vue d'ensemble

Au moment de la rédaction de ce livre, le Raspberry Pi Pico W peut être programmé avec les langages de programmation suivants :

- C/C++
- MicroPython
- Langage assembleur

Bien que le Pico soit configuré par défaut pour être programmé avec le puissant et populaire langage C/C++, de nombreux débutants trouvent plus facile d'utiliser MicroPython, une version du langage de programmation Python développée spécifiquement pour les microcontrôleurs.

Dans ce chapitre, vous apprendrez à installer et à utiliser le langage de programmation MicroPython. Vous utiliserez l'environnement de développement intégré (EDI) Thonny qui a été développé spécifiquement pour les programmes Python.

Les prochains chapitres présentent de nombreux projets avec le Pico programmés en MicroPython, fonctionnels et entièrement testés.

## 2.2 Installation de MicroPython sur le Pico W

Il faut installer MicroPython sur le Raspberry Pi Pico W avant que la carte puisse être utilisée. Une fois installé, MicroPython reste sur votre Pico à moins qu'il ne soit écrasé par quelque chose d'autre. L'installation de MicroPython nécessite une connexion à l'internet, qui n'est requise qu'une seule fois. Vous pouvez le faire en utilisant soit un Raspberry Pi (par ex. Raspberry Pi 4), soit un PC. Dans cette section, vous verrez l'installation avec un PC (par ex. Windows 10).

Les étapes sont les suivantes :

- Assurez-vous que votre PC est connecté à l'internet.
- Téléchargez le fichier Raspberry Pi Pico W MicroPython UF2 dans un dossier (par ex. Downloads) sur votre PC à partir du lien suivant. Au moment de la rédaction de ce livre, le fichier avait pour nom : **rp2-pico-w-20220909-unstable-v1.19.1-389-g4903e48e3.uf2**.

https://www.raspberrypi.com/documentation/microcontrollers/micropython.html#-drag-and-drop-micropython

- Appuyez sur le bouton **BOOTSEL** de votre Pico et maintenez-le enfoncé.
- Connectez votre Pico au port USB de votre PC à l'aide d'un câble micro-USB tout en maintenant le bouton enfoncé.
- Attendez quelques secondes et relâchez le bouton **BOOTSEL**.

- Le Pico doit apparaître comme un lecteur amovible baptisé **RPI-RP2**, comme illustré à la **figure 2.1** (lecteur **E** : dans ce cas).

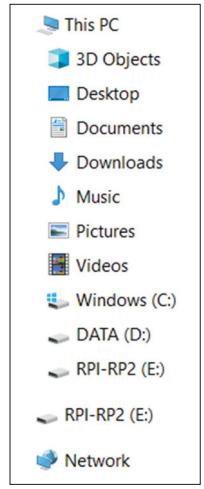


*Figure 2.1 : Pico comme lecteur amovible RPI-RP2.*

- Faites glisser et déposez le fichier MicroPython UF2 téléchargé sur le volume RPI-RP2. Votre Pico va redémarrer, et maintenant vous pouvez exécuter des programmes écrits en MicroPython sur votre Pico.
- La mise hors tension du Pico ne supprimera pas MicroPython de sa mémoire.

## 2.3 Utilisation de l'EDI Thonny sur PC

Dans cette section, vous allez apprendre à utiliser Thonny sur PC pour écrire et exécuter vos programmes.

Tout d'abord, vous devez installer Thonny sur votre PC (s'il n'est pas déjà installé). Les étapes sont les suivantes :

- Allez sur le site web Thonny.org :

  https://thonny.org/

- Téléchargez la dernière version de Thonny pour Windows et installez-la (voir **figure 2.2**). Vous pouvez sélectionner la langue de l'interface, le français est proposé.


*Figure 2.2 : cliquez pour installer Thonny.*

- Vous devriez voir une icône sur le bureau (**figure 2.3**) de votre PC. Double-cliquez dessus pour lancer Thonny.


*Figure 2.3 : icône Thonny sur le bureau.*

- L'écran de démarrage de Thonny sur PC est illustré à la **figure 2.4**.

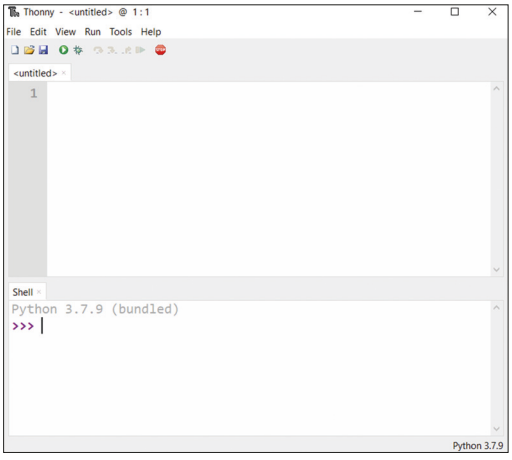


*Figure 2.4 : écran de démarrage de Thonny sur PC.*

- Cliquez sur l'étiquette **Python** dans le coin inférieur droit de l'écran puis sur « Configurer l'interpréteur », cherchez et sélectionnez **MicroPython (Raspberry Pi Pico)**.
- Vous êtes maintenant prêt à écrire vos programmes. Prenons une instruction Python très simple pour tester l'installation de MicroPython sur votre Pico.
- Saisissez l'instruction suivante dans la partie inférieure de l'écran (dans **Shell**) :

  **print("hello from Thonny on PC")**

- Si vous cliquez sur *Exécuter* > *Exécuter le script courant*, vous devriez voir le message **hello from Thonny on PC** s'afficher comme indiqué dans la **figure 2.5**.

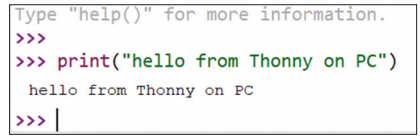


*Figure 2.5 : affichage du message.*

**Dans ce livre, vous utiliserez Thonny sur PC pour écrire des programmes et les exécuter sur le Raspberry Pi Pico W.**

Les sections restantes de ce chapitre présentent des programmes d'exemple simples, purement logiciels. L'objectif de ce chapitre est de passer en revue les notions de base de la programmation en Python. Ce livre n'a pas pour but d'enseigner le langage de programmation Python. Les lecteurs intéressés peuvent trouver de nombreux livres et tutoriels sur l'internet pour apprendre le langage de programmation Python.

## 2.4 Écrire un programme en utilisant Thonny

Dans la section précédente, vous avez appris à exécuter une instruction en ligne à l'aide du **Shell** de Thonny. Dans presque toutes les applications, vous devez écrire des programmes. À titre d'exemple, les étapes pour écrire et exécuter un programme très simple d'une ligne pour afficher le message **Bonjour de la part de...** sont données ci-dessous :

- Saisissez les instructions du programme dans la partie supérieure de l'écran, comme le montre la **figure 2.6**.

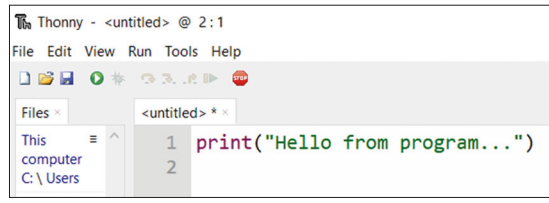


*Figure 2.6 : écrire le programme dans la partie supérieure de l'écran.*

- Cliquez sur *Fichier* puis sur *Enregistrer sous* et donnez un nom à votre programme, par exemple **FirstProg.** Vous avez la possibilité d'enregistrer le programme soit sur votre PC (*Cet ordinateur*), soit sur le Pico (*Raspberry Pi Pico*). Cliquez sur *Raspberry Pi Pico* pour l'enregistrer sur le Pico (**figure 2.7**). Entrez le nom de votre programme (**FirstProg**) et cliquez sur *OK* (remarquez que le fichier est enregistré avec l'extension **.py**).

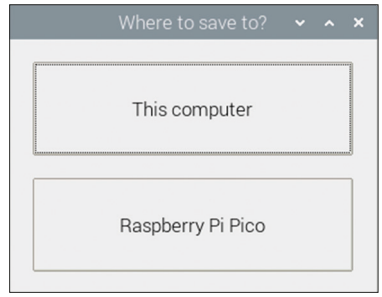


*Figure 2.7 : cliquez sur* **Raspberry Pi Pico** *pour enregistrer votre programme.*

- Cliquez sur l'icône de la flèche verte en haut de l'écran pour exécuter votre programme. Le résultat du programme s'affiche dans la partie inférieure de l'écran, comme le montre la **figure 2.8**.

```
>>> %Run -c $EDITOR_CONTENT
  Hello from program...
>>>
```

*Figure 2.8 : sortie du programme.*

## 2.5 Programmes MicroPython purement logiciels, exécutés sur le Raspberry Pi Pico W

**Exemple 1** : **moyenne de deux nombres saisis sur le clavier du PC**

Dans cet exemple, deux nombres saisis sur le clavier du PC sont lus et leur moyenne est calculée puis affichée. Le but de cet exemple est de montrer comment les données peuvent être lues à partir du clavier.

**Solution 1**

Le programme s'appelle **Average**. La **figure 2.9** présente le listing du programme et un exemple de son exécution. La fonction **input** est utilisée pour lire les nombres sous forme de chaînes à partir de la saisie au clavier. Ces chaînes sont ensuite converties en nombres à virgule flottante et stockées dans les variables **n1** et **n2**. La moyenne est calculée en additionnant les deux nombres puis en divisant par deux cette somme. Le résultat est affiché à l'écran.

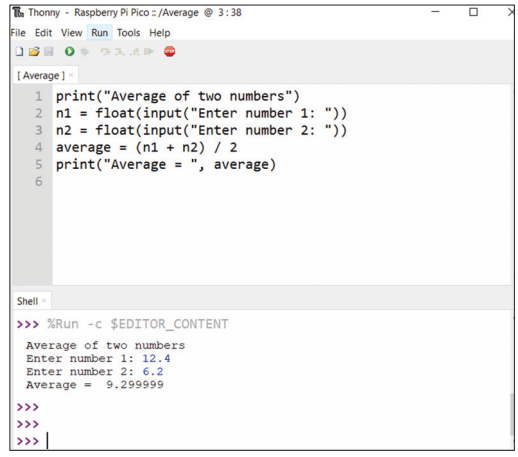


*Figure 2.9 : programme* **Average** *et son exécution.*

**Exemple 2** : **moyenne de dix nombres saisis sur le clavier du PC**
Dans cet exemple, dix nombres sont saisis sur le clavier et leur moyenne est affichée. Le but de cet exemple est de montrer comment construire une boucle en Python.

**Solution 2**
Le programme s'appelle **Average10**. La **figure 2.10** présente le listing du programme et un exemple de son exécution. Dans ce programme, une boucle est construite pour aller de 0 à 9 (c'est-à-dire dix itérations). À l'intérieur de cette boucle, les nombres saisis sur le clavier sont lus, additionnés les uns aux autres et stockés dans la variable **sum**. La moyenne est ensuite calculée et affichée en divisant **sum** par dix. Remarquez qu'il n'y a aucune nouvelle ligne après l'instruction de saisie, car cette dernière contient l'option **end = ' '**.

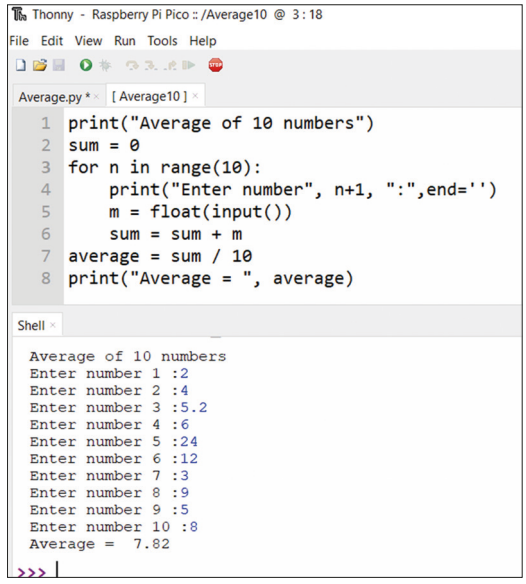


*Figure 2.10 : programme* **Average10** *et son exécution.*

**Exemple 3** : **surface d'un cylindre**
Dans cet exemple, le rayon et la hauteur d'un cylindre sont saisis sur le clavier et sa surface est affichée à l'écran.

**Solution 3**
Le programme s'appelle **CylArea**. La **figure 2.11** présente le listing du programme et un exemple de son exécution. La surface d'un cylindre est donnée par :

surface = 2πrh

Où **r** et **h** sont respectivement le rayon et la hauteur du cylindre. Dans ce programme, la bibliothèque **math** propre au langage Python est importée afin que la fonction **Pi** puisse être utilisée dans le programme. La surface du cylindre est affichée après lecture du rayon et de la hauteur.

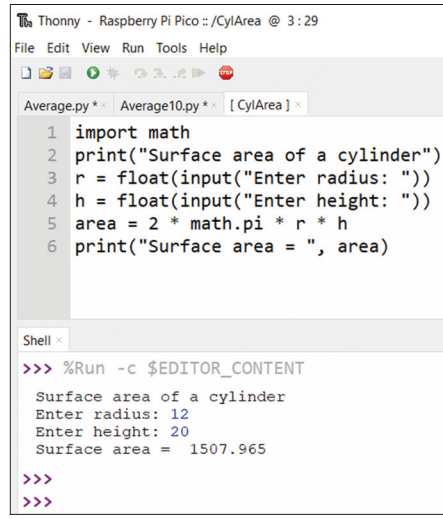


*Figure 2.11 : programme* **CylArea** *et son exécution.*

**Exemple 4** : **conversion de** ºC **en** ºF
Dans cet exemple, le programme lit une valeur en degrés Celsius saisie sur le clavier, la convertit en degrés Fahrenheit et affiche le résultat de la conversion.

**Solution 4**
Le programme s'appelle **CtoF**. La **figure 2.12** présente le listing du programme et un exemple de son exécution. La formule pour convertir les ºC en **ºF** est la suivante :

$$F = 1{,}8 \times C + 32$$

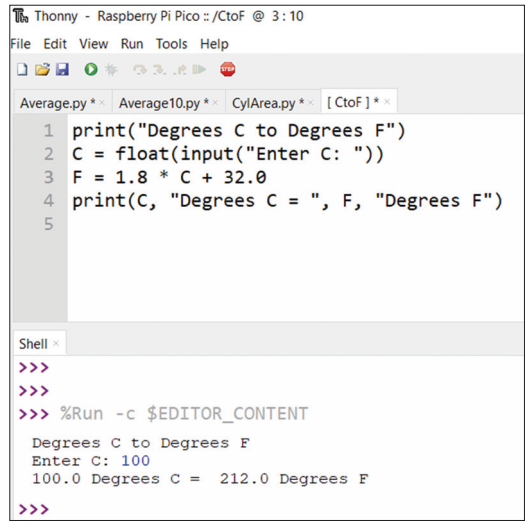


*Figure 2.12 : programme* **CtoF** *et son exécution.*

**Exemple 5** : **surface et volume d'un cylindre avec définition d'une fonction**
Dans cet exemple, on calcule la surface et le volume d'un cylindre pour un rayon et une hauteur donnés. La surface et le volume sont calculés et retournés par une fonction.

**Solution 5**
Le programme s'appelle **CylAreaSurf**. La **figure 2.13** présente le listing du programme et un exemple de son exécution. La surface et le volume d'un cylindre sont donnés par :

surface = 2πrh
volume = $\pi r^2 h$

Où **r** et **h** sont respectivement le rayon et la hauteur du cylindre. Le rayon et la hauteur du cylindre sont demandés dans le corps principal du programme. La fonction **Calc** calcule et renvoie la surface et le volume au programme principal, qui les affiche.

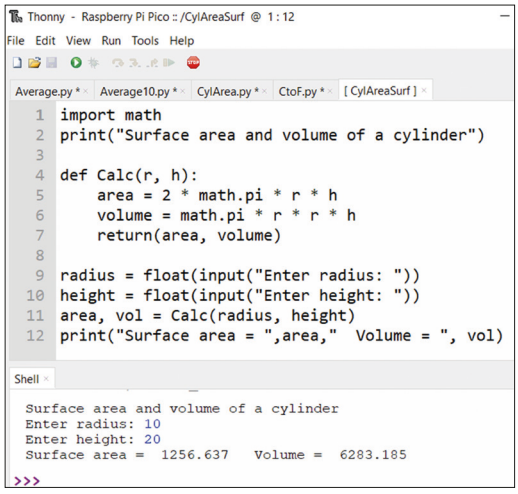


*Figure 2.13 : programme* **CylAreaSurf** *et exemple d'exécution.*

**Exemple 6** : **tableau des carrés de nombres**
Dans cet exemple, les carrés des nombres de 1 à 11 sont calculés et présentés sous forme de tableau.

**Solution 6**
Le programme s'appelle **Squares**. La **figure 2.14** présente le listing du programme et un exemple de son exécution. Remarquez que \t imprime une tabulation afin que les données se présentent sous forme de tableau.

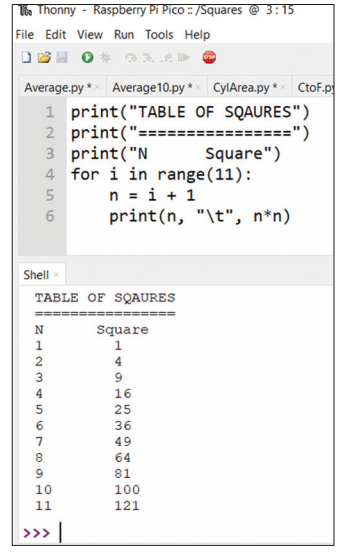


*Figure 2.14 : programme* **Squares** *et résultat de son exécution.*

**Exemple 7** : **tableau de la fonction trigonométrique sinus**
Dans cet exemple, le sinus trigonométrique est présenté sous forme de tableau, pour les angles de 0 à 45° par pas de 5°.

**Solution 7**
Le programme s'appelle **Sines**. La **figure 2.15** présente le listing du programme et un exemple de son exécution. Il est important de noter que les arguments des fonctions trigonométriques doivent être en radians et non en degrés (la fonction **math.radians()** se charge de la conversion des degrés en radians).

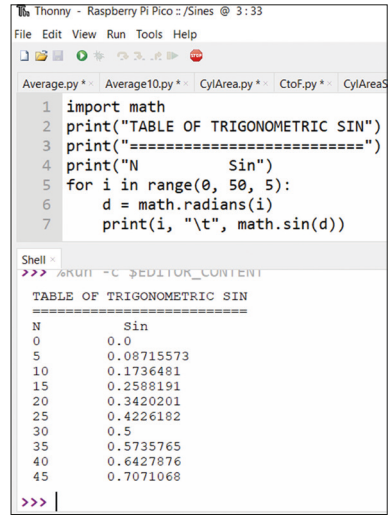


*Figure 2.15 : programme* **Sines** *et résultat de son exécution.*

**Exemple 8** : **tableau des fonctions trigonométriques sinus, cosinus et tangente**
Dans cet exemple, le sinus, le cosinus et la tangente trigonométriques sont présentés sous forme de tableau, pour les angles de 0 à 45° par pas de 5°.

**Solution 8**
Le programme s'appelle **Trig**. Le listing du programme ainsi qu'un exemple de son exécution sont présentés à la **figure 2.16**.

Trig.py

```
1  import math
2  print("          TABLE OF TRIGONOMETRIC FUNCTIONS")
3  print("          ================================")
4  print("N\t   Sin\t\t  Cos\t\t   Tan")
5  for i in range(0, 50, 5):
6      d = math.radians(i)
7      s = math.sin(d)
8      c = math.cos(d)
9      t = math.tan(d)
10     print(i, "\t%9.7f\t%9.7f\t%9.7f" %(s,c,t))
```

Shell

```
          TABLE OF TRIGONOMETRIC FUNCTIONS
          ================================
N          Sin             Cos             Tan
0       0.0000000       1.0000000       0.0000000
5       0.0871557       0.9961946       0.0874887
10      0.1736481       0.9848078       0.1763270
15      0.2588191       0.9659258       0.2679492
20      0.3420201       0.9396926       0.3639702
25      0.4226182       0.9063078       0.4663076
30      0.5000000       0.8660254       0.5773502
35      0.5735765       0.8191521       0.7002075
40      0.6427876       0.7660444       0.8390996
45      0.7071068       0.7071068       1.0000000
```

*Figure 2.16 : programme* **Trig** *et résultat de son exécution.*

**Exemple 9 : fonctions trigonométriques pour un angle donné**
Dans cet exemple, un angle est saisi au clavier. L'utilisateur précise également s'il veut le sinus (s), le cosinus (c) ou la tangente (t) de l'angle.

**Solution 9**
Le programme s'appelle **TrigUser**. Le listing du programme ainsi qu'un exemple de son exécution sont présentés à la **figure 2.17**.