faites tourner votre moteur CC

exemples de projets du livre Motor Control Development (offre groupée)

Dogan Ibrahim (Royaume-Uni)

La meilleure façon de s'initier à la programmation et à l'électronique pratique est d'étudier comment on peut faire tourner des composants, les faire vibrer ou clignoter. Le mouvement implique souvent l'utilisation d'un moteur, et un petit moteur à courant continu à balais est idéal pour démarrer avant de se lancer dans la robotique et la mécatronique. Ici, nous proposons une introduction à la commande intelligente d'un moteur à courant continu, en nous appuyant sur un kit de développement exceptionnel d'Elektor qui combine le matériel, le logiciel et un guide pratique.

Note de l'éditeur : *cet article est un extrait du livre de 192 pages Motor Control Development Kit (Elektor, offre groupée) formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré* **« Questions ou commentaires ? »**.

Dans cet article, trois projets simples basés sur un petit moteur à courant continu, à balais et à aimant permanent, commandés par la carte de développement Cytron/Maker Pi RP2040, sont décrits. Les deux composants sont inclus dans l'offre groupée disponible sur l'e-choppe Elektor [1]. Les projets constituent des exemples exploratoires et éducatifs plutôt que des projets complets.

Commande marche/arrêt d'un moteur CC

Ce projet de base montre comment connecter un petit moteur CC à balais fonctionnant de 1...6 V CC à l'un des borniers DC MOTOR de la carte Maker Pi RP2040. Le moteur est commandé en marche avant pendant 5 secondes, puis arrêté pendant 5 secondes. Ensuite, il tourne en sens inverse pendant 5 secondes et s'arrête à nouveau pendant 5 secondes. Ce processus est répété indéfiniment jusqu'à ce qu'il soit arrêté manuellement.
Le but de ce projet est de se concentrer sur la connexion, le fonctionnement et la commande d'un moteur CC avec la carte Maker Pi RP2040. La **figure 1** montre le schéma fonctionnel du projet, et la **figure 2** présente le moteur utilisé. Il s'agit d'un petit moteur CC, à balais, conçu pour fonctionner de +1 V à +6 V avec une tension de fonctionnement recommandée de +3 V. Voyons comment le faire tourner.
Un double pilote de moteur à pont en H, commandant jusqu'à deux moteurs CC à balais, M1 et M2, ou un moteur pas à pas, est intégré sur la carte de développement. La disposition des broches d'E/S est la suivante

- M1A : GP8
- M1B : GP9
- M2A : GP10
- M2B : GP11

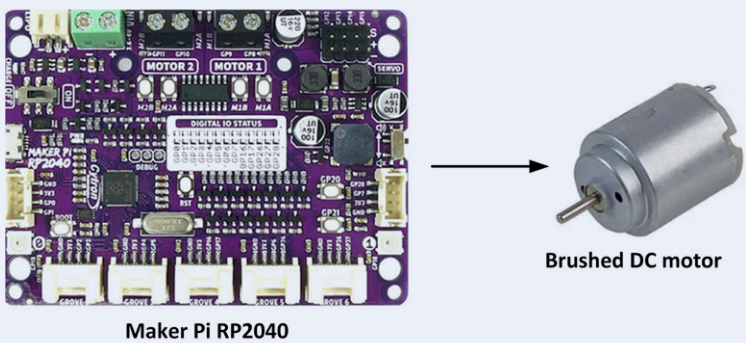


Figure 1. Les deux principaux composants du projet.

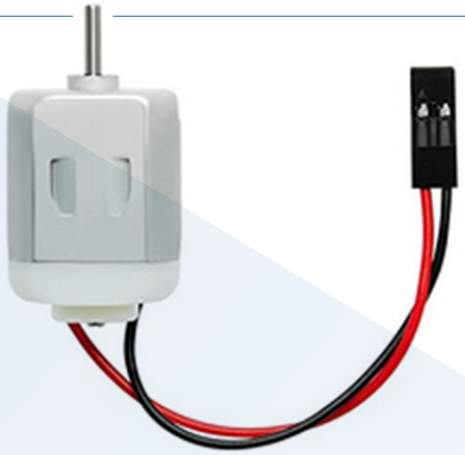


Figure 2. Le petit moteur à courant continu utilisé.

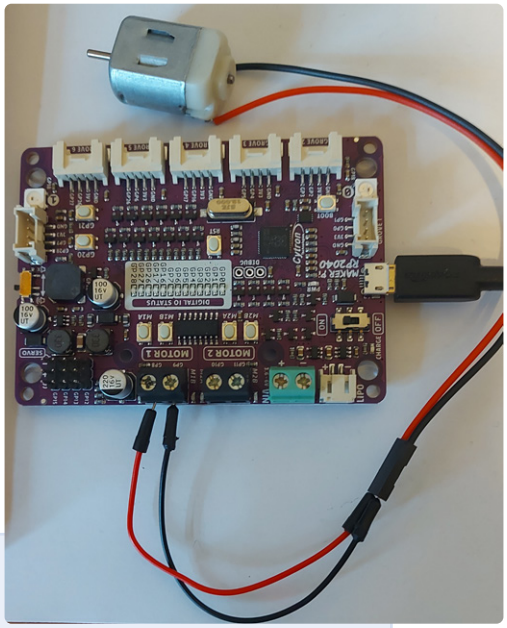


Figure 3. Connexion du moteur à la carte RP2040 Maker Pi.

La table de vérité du pilote de moteur est présentée dans le **tableau 1**. Dans ce projet, les entrées PWM1A (port GP8) et PWM1B (port GP9) du moteur sont utilisées avec les sorties du moteur sur M1A et M1B. Ces sorties (*Output A*) sont disponibles sur les bornes à vis MOTOR 1 situées au milieu en haut de la carte de développement. Comme le montre le tableau 1, le moteur est commandé par les ports GP8 et GP9. Par exemple, le fait de mettre GP8 au niveau haut (HIGH) et GP9 au niveau bas fait (LOW) tourner le moteur en avant. De même, le réglage de GP8 au niveau bas et GP9 au niveau haut fait tourner le moteur en arrière.

La **figure 3** montre le projet avec le moteur connecté aux sorties MOTOR 1.

Le programme de la démo de démarrage est donné dans le **listage 1**. Le programme appelé *DCMotor1.py* est contenu dans un fichier d'archive logiciel volumineux disponible gratuitement sur le site web d'Elektor [1]. Sur cette page, défilez vers *Téléchargements Software_Motor Control Development Kit.*

Le moteur est commandé par des formes d'ondes PWM comme décrit ailleurs dans le guide cité en référence. La bibliothèque intégrée de moteurs CC appelée *adafruit_motor* est utilisée dans ce projet. Au début du programme, les modules de bibliothèque *board*, *time*, *pwmio* et *motor* sont importés dans le programme. Les ports GP8 et GP9 de Motor 1 sont alors configurés pour fonctionner avec une forme d'onde PWM à une fréquence de 50 Hz. Le reste du programme s'exécute dans une boucle infinie.

La propriété de la classe `throttle` peut prendre une valeur comprise entre -1 et +1 et contrôle le moteur comme suit :

`throttle = 0` idle
`throttle = 1` le moteur tourne en avant à pleine vitesse
`throttle = 0.5` le moteur tourne en avant à 50 % de sa vitesse maximale
`throttle = -1` le moteur tourne en arrière à pleine vitesse
`throttle = -0.5m` le moteur tourne en arrière à 50 % de sa vitesse maximale

Dans ce programme, le moteur tourne à 25 % de sa vitesse maximale en réglant l'accélérateur à +0,25 dans un sens et à -0,25 dans l'autre.

Listage 1. DCMotor1.py

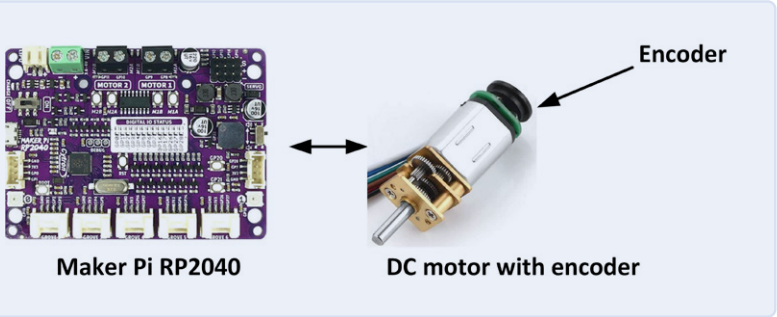
```
#-------------------------------------------------------------
#                BRUSHED DC MOTOR CONTROL
#
# In this program a brushed DC moor is controlled as follows:
# The motor is turned ON in forward direction for 5 seconds
# and then stopped for 5 seconds, then in reverse direction
# for 5 seconds and then stopped again for 5 seconds. This
# process is repeated forever
#
# Author: Dogan Ibrahim
# File  : DCMotor1.py
# Date  : February, 2023
#-------------------------------------------------------------
import board
import time
import pwmio
from adafruit_motor import motor
#
# Initialize DC Motor 1
#
m1a = pwmio.PWMOut(board.GP8, frequency=50)
m1b = pwmio.PWMOut(board.GP9, frequency=50)
motor1 = motor.DCMotor(m1a, m1b)
while True:
    motor1.throttle = 0.25              # 25% of full speed
    time.sleep(5)
    motor1.throttle = 0                 # Idle
    time.sleep(5)
    motor1.throttle = -0.25             # 25% of full speed
    time.sleep(5)
    motor1.throttle = 0                 # Idle
    time.sleep(5)
```

Tableau 1. Motor Driver Truth Table.

Input A	Input B	Output A	Output B	Motor Action
Low	Low	Low	Low	Brake
High	Low	High	High	Forward
Low	High	Low	High	Backward
High	High	Hi-Z (Open)	Hi-Z (Open)	Coast

Listage 2. DCMotor2.py.

```
#-------------------------------------------------------------
#        TWO SPEED BRUSHED DC MOTOR CONTROL
#
# In this program a brushed DC moor is controlled as follows:
# The motor normally rotates at 25% of its full speed. Pressing
# button at port GP20 increses the speed to 50% of its full
# speed. Releasing the botton returns the speed to 25%
#
# Author: Dogan Ibrahim
# File  : DCMotor2.py
# Date  : February, 2023
#-------------------------------------------------------------
import board
import time
import pwmio
import digitalio
from adafruit_motor import motor
#
# Initialize DC Motor 1
#
m1a = pwmio.PWMOut(board.GP8, frequency=50)
m1b = pwmio.PWMOut(board.GP9, frequency=50)
motor1 = motor.DCMotor(m1a, m1b)
#
# Configure button at port GP20. The button state is
# normally at logic 1
#
btn = digitalio.DigitalInOut(board.GP20)
btn.direction = digitalio.Direction.INPUT
btn.pull = digitalio.Pull.UP
while True:
    motor1.throttle = 0.25              # 25% of full speed
    while btn.value == 0:               # If button pressed
        motor1.throttle = 0.5           # Increase speed
```



▲ *Figure 4. Schéma fonctionnel du projet.*

Figure 5. Moteur/encodeur, GBMQ-GM12BY20.

▶

Commande de moteur CC à deux vitesses

Il s'agit d'un projet très simple où un petit moteur CC à balais est connecté à la carte Maker Pi RP2040. Ici, nous utilisons également le bouton poussoir embarqué sur le port GP20. Normalement, le moteur tourne à 25 % de sa vitesse maximale. En appuyant sur le bouton, il passe à 50 % de sa vitesse maximale.

Les composants et les connexions entre le moteur et la carte électronique pour ce projet sont les mêmes que pour le projet précédent.

Le **listage 2** montre le programme *DCMotor2.py*. Le programme principal s'exécute dans une boucle, où l'état du bouton sur GP20 est vérifié. Tant que le bouton est pressé, la vitesse du moteur est augmentée à 50% de sa vitesse maximale.

Mesure de la vitesse d'un moteur à courant continu avec un encodeur rotatif

Ce projet montre un exemple d'utilisation d'un encodeur rotatif pour mesurer la vitesse d'un moteur CC. La vitesse est ensuite affichée à l'écran. Dans une «représentation en diagramme», cela ressemble à la **figure 4**. Notez que le moteur, avec son encodeur rotatif intégré, n'est pas inclus dans le kit de développement et vous devez l'acheter séparément. (Disponible sur eBay, AliExpress et autres.) Nous utilisons un petit moteur CC à engrenages et à balais avec un encodeur rotatif intégré. La **figure 5** montre le moteur où l'encodeur rotatif est fixé à l'arbre arrière. Dans ce projet, le moteur suivant est utilisé : Moteur à engrenages CC 6 V, 2 W, GBMQ-GM12BY20 avec encodeur, 70 RPM. La vitesse à vide du moteur est spécifiée comme étant de 70 RPM. Dans ce projet, vous utiliserez une alimentation externe de +5 V pour le moteur et, par conséquent, vous vous attendrez à ce que la vitesse à vide soit inférieure à 70 RPM. Notez que la consommation de courant du moteur peut aller jusqu'à 200 mA.

Un encodeur rotatif (ou encodeur d'arbre) permet de convertir la position angulaire d'un composant en rotation, tel qu'un moteur, en un signal électrique. Les encodeurs rotatifs sont utilisés dans les applications de commande de moteur pour détecter la vitesse du moteur ou la position de l'arbre du moteur. Il existe essentiellement deux types d'encodeurs rotatifs : les encodeurs optiques et les encodeurs à effet Hall. Les encodeurs rotatifs optiques fonctionnent selon les principes de l'optique, où la lumière éclaire une photodiode à travers les fentes (ou trous) d'un disque métallique ou de toute autre forme. En comptant le nombre de trous passant devant la photodiode en un temps donné, vous (ou votre microcontroleur !) pouvez calculer la vitesse du moteur. Dans ce projet, nous utilisons un encodeur rotatif à effet Hall. Nous utilisons deux capteurs magnétiques statiques (appelés Phase A et Phase B) ainsi qu'un disque magnétique rotatif. Les capteurs fournissent des impulsions lorsque le moteur tourne. En comptant les impulsions

sur un temps donné, vous pouvez calculer la vitesse du moteur. La **figure 6** montre les formes d'ondes de sortie du moteur utilisé dans ce projet. La forme d'onde située en haut représente la phase A et celle située en bas la phase B. Ces types d'encodeurs sont également connus sous le nom d'encodeurs en quadrature car les capteurs magnétiques sont placés à 90° l'un par rapport à l'autre et il y a quatre états de sortie possibles. Le sens de rotation est facilement déterminé en trouvant l'ordre dans lequel les signaux de sortie passent de 0 à 1. Par exemple, si le signal Phase A remonte à partir de son état BAS, alors le moteur tourne dans un sens. Si, en revanche, le signal Phase B monte alors que le signal Phase A est au niveau BAS, alors le moteur tourne dans le sens opposé - voir **figure 7**.
Les caractéristiques du moteur utilisé dans ce projet sont les suivantes (en fonction de la tension appliquée) :

- Fonctionnement en courant continu (6 V)
- Vitesse de l'arbre après les engrenages : 70 RPM (à 6 V)
- Puissance : 2 W
- Courant 170...200 mA
- Deux capteurs à effet Hall intégrés
- Poids : 14 grammes

Un connecteur à 6 fils est fixé au panneau arrière du moteur. La configuration des broches du moteur est la suivante :

Fil noir	alimentation du moteur GND
Fil rouge	alimentation du moteur (+5 V dans ce projet)
Jaune	alimentation encodeur (+3.3 V dans ce projet)
Vert	alimentation encodeur GND
Bleu	sortie encodeur Phase A
Blanc	sortie encodeur Phase B

La **figure 8** montre le schéma de câblage du projet. Notez que le moteur fonctionne avec une alimentation externe de +5 V capable de fournir jusqu'à 500 mA. L'encodeur fonctionne sur une ligne de +3,3 V provenant de la carte de développement afin que la tension soit compatible avec les niveaux de tension d'entrée du processeur RP2040. Notez également que seule la sortie Phase A du moteur est utilisée dans ce projet.
Les fabricants et distributeurs de moteurs électriques n'indiquent pas le rapport d'engrenage et le nombre d'impulsions de l'encodeur émises par le moteur par seconde. Il s'agit pourtant de spécifications importantes. Dans cette section, nous écrivons un programme qui affiche ces paramètres importants du moteur utilisé. Vous devrez peut-être effectuer ces calculs pour trouver les spécifications importantes du moteur que vous utilisez. Le **listage 3** montre le programme MotorPulses.py écrit pour afficher le nombre d'impulsions du moteur en sortie de l'encodeur Phase A toutes les secondes. Dans ce programme, le port GP6 est assigné à l'objet `Encoder` et est configuré comme une entrée, et le nombre d'impulsions reçues de l'encodeur est calculé et affiché à la fin de chaque seconde. Notez que ce programme utilise la fonction `time.monotonic()` pour calculer le temps, et qu'il n'est pas très précis. L'idéal serait de recevoir les impulsions sous forme d'interruptions externes et d'utiliser des interruptions de temporisation d'une seconde pour calculer et afficher le nombre d'impulsions reçues chaque seconde. Malheureusement, CircuitPython ne prend pas en charge les interruptions externes ou temporisées.

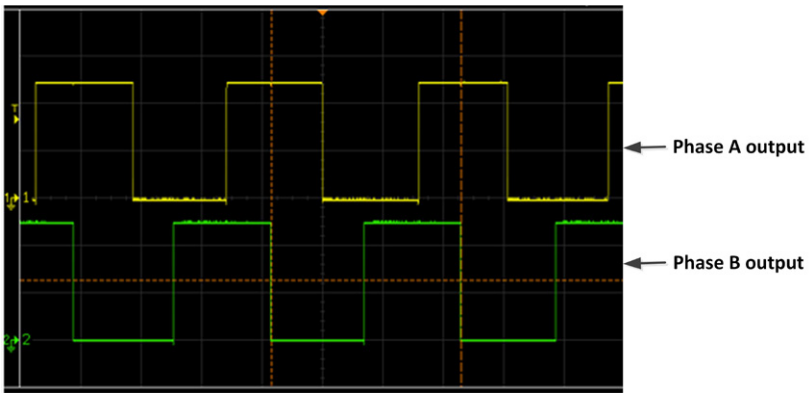


Figure 6. Formes d'onde de sortie de l'encodeur rotatif.

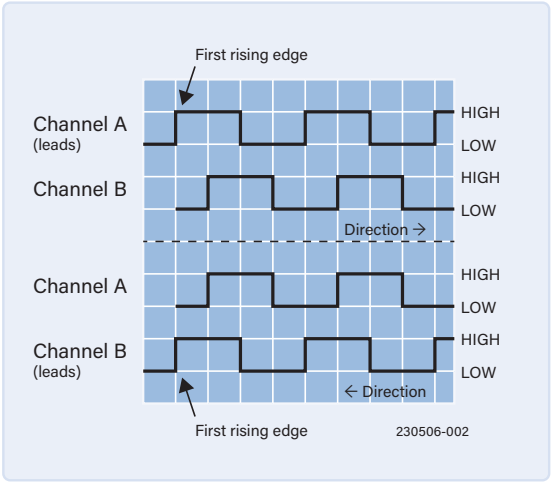


Figure 7. Détermination du sens de rotation (Source : robotoid.com).

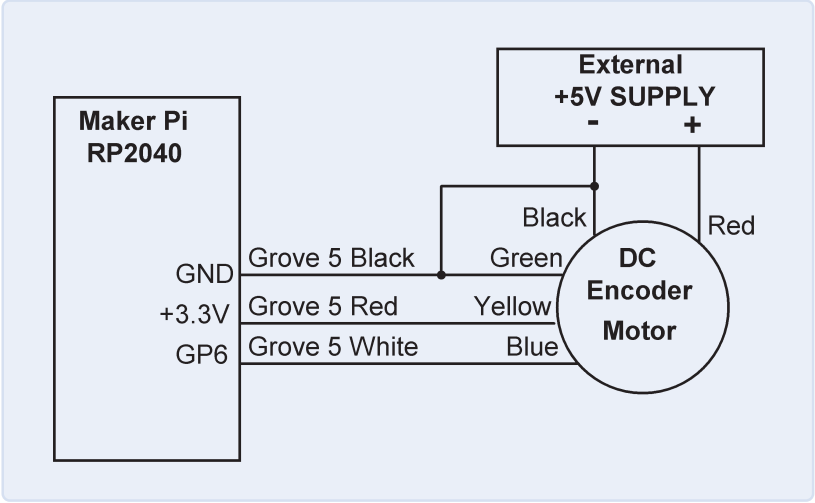


Figure 8. Le schéma de câblage du projet.

Listage 3. MotorPulses.py.

```
#-------------------------------------------------------
#       DISPLAYING THE NUMBER OF ENCODER PULSES
#
# In this program a geared DC motor with Hall Effect sensors
# is connected to the Maker Pi. This program calculates
# and displays the number of pulses received from the encoder
# every second. Only Phase A encoder output is used here
#
# Author: Dogan Ibrahim
# File  : MotorPulses.py
# Date  : March, 2023
#-------------------------------------------------------
import board
import digitalio
import time
Encoder = digitalio.DigitalInOut(board.GP6)
Encoder.direction = digitalio.Direction.INPUT
while Encoder.value == 0:          # Wait while 0
    pass
strtime = time.monotonic()         # Start time,rising edge
                                   # detected
count = 0                          # Intialize count
#
# Main program loop
#
while True:
    while Encoder.value == 1:      # Wait while 1
        pass
    while Encoder.value == 0:      # Wait while 0
        pass
    endtime = time.monotonic()     # End time
    if endtime-strtime > 1.0:      # Just over a second
        print(count)               # Display count
        strtime = time.monotonic()# Start time
        count = 0
    else:
        count = count + 1          # Increment count
    while(Encoder.value) == 1:     # Wait while 1
        pass
```

Listage 4: MotorSpeed.py.

```
#-------------------------------------------------------
#         DISPLAYING THE MOTOR SPEED
#
# This program displays the motor speed in RPM using the
# number of encoder pulses received every second and the
# formula given in the text
#
# Author: Dogan Ibrahim
# File  : MotorSpeed.py
# Date  : March, 2023
#-------------------------------------------------------
import board
import digitalio
import time
Encoder = digitalio.DigitalInOut(board.GP6)
Encoder.direction = digitalio.Direction.INPUT
while Encoder.value == 0:                Wait while 0
    pass
strtime=time.monotonic()                # Start time
count=0                                 # Intialize count
#
# Main program loop
#
while True:
    while Encoder.value == 1:           # Wait while 1
        pass
    while Encoder.value == 0:           # Wait while 0
        pass
    endtime=time.monotonic()            # End time
    if endtime-strtime > 1.0:           # Just over a second
        RPM = count * 60 / 880          # Motor speed
        print("Speed=%6.2f RPM" %RPM)   # Display speed
        strtime=time.monotonic()        # Start time
        count=0
    else:
        count=count+1                   # Increment count
    while(Encoder.value) == 1:          # Wait while 1
        pass
```

```
CircuitPython REPL
Speed= 63.95 RPM
Speed= 64.02 RPM
Speed= 64.02 RPM
Speed= 63.95 RPM
Speed= 63.89 RPM
Speed= 63.95 RPM
```

Figure 9. Sortie du programme.

Figure 10. Le tachymètre numérique DT-2234C.

Produits

- **Offre groupée Motor Control Development**
  www.elektor.fr/20534

Lorsque le moteur tourne, le programme *MotorPulses.py* affiche 938 pulses/seconde. Nous avons fixé un objet de forme ronde (un petit pneu) à l'arbre du moteur et nous avons fait une marque sur celui-ci afin de pouvoir compter les rotations, et nous avons observé que le nombre de tours de l'arbre du moteur à vide était de 64 RPM.
Or, le nombre d'impulsions reçues chaque minute est de 938 × 60 = 56,280 impulsions/minute, et cela correspond à une vitesse à vide de 64 RPM. Par conséquent, 56 280/64 = 879,38, ou 880 comme nombre entier le plus proche, puis la vitesse du moteur peut être calculée en utilisant la formule suivante :

vitesse (RPM) = (nombre d'impulsions mesurées par seconde × 60) / 880

Par exemple, si le nombre d'impulsions reçues est compté à 450 par seconde, la vitesse du moteur est la suivante :

vitesse = 450 × 60 / 880 = 30.68 RPM

La formule ci-dessus sera utilisée pour calculer la vitesse du moteur.
Le programme MotorSpeed.py présenté dans le **listage 4** est utilisé pour calculer puis afficher la vitesse du moteur en utilisant la formule ci-dessus. Ce programme est essentiellement le même que dans le listage 3, mais ici la vitesse du moteur est calculée et affichée à l'écran. La **figure 9** montre un exemple de sortie du programme.
Comme nous l'avons vu précédemment, nous aurions pu obtenir des résultats plus précis si des interruptions externes avaient été utilisées pour compter les impulsions de l'encodeur, et des interruptions de la minuterie pour mesurer le temps.

RP2040 ?

Vous pouvez facilement mesurer et **vérifier** la vitesse du moteur à l'aide d'un tachymètre numérique. Il existe de nombreux appareils de ce type à des prix différents. Celui utilisé par l'auteur était un modèle DT-2234C (**figure 10**) qui lui a coûté environ 8 £ plus le coût d'une seule pile de 9 V de type PP3. Avant d'utiliser cet appareil, il faut fixer un morceau de papier réfléchissant sur l'arbre du moteur, comme le montre la **figure 11**.
La vitesse du moteur est mesurée avec le tachymètre DT-2234C comme suit :

- Mettez le moteur en marche.
- Appuyez sur le bouton TEST du tachymètre et dirigez la lumière de l'appareil vers le papier réfléchissant.
- La vitesse du moteur s'affiche en continu sur l'écran LCD.
- Vous pouvez sauvegarder les relevés en appuyant sur le bouton MEM.

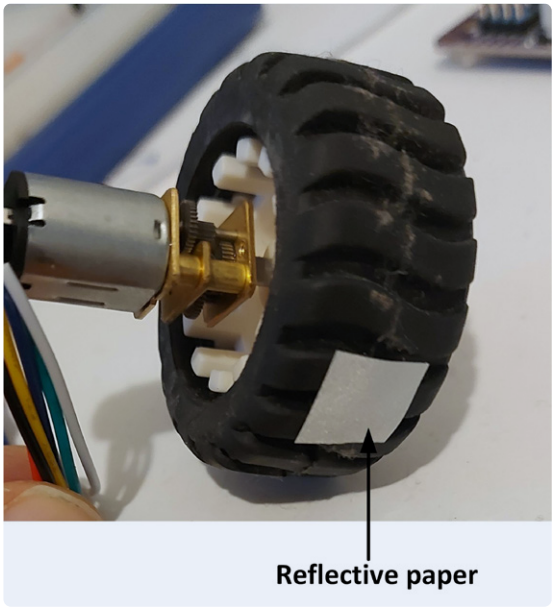


Figure 11. Fixation d'un morceau de papier réfléchissant sur l'arbre du moteur.

Note : Dans ce projet, nous n'utilisons qu'une seule phase de sortie de l'encodeur (Phase A). Il est possible d'obtenir des résultats plus précis sur la vitesse du moteur si les deux phases de l'encodeur (c'est-à-dire Phase A et Phase B) sont utilisées. Il est également possible de commander le moteur à partir des bornes à vis MOTOR 1 ou MOTOR 2 de la carte de développement Maker Pi RP2040. Cela limitera la tension maximale du moteur à +3,3 V et exigera également que la vitesse du moteur soit contrôlée en envoyant des formes d'ondes de tension PWM au moteur par les ports GP8/GP9 (MOTOR 1) ou les ports GP10/GP11 (MOTOR 2).

230506-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Dogan Ibrahim est titulaire d'une licence (avec mention) en ingénierie électronique, d'une Master en ingénierie du contrôle automatique et d'un doctorat en traitement des signaux numériques et microprocesseurs. Dogan a travaillé dans de nombreuses organisations et est membre de l'Institution of Engineering and Technology (IET) au Royaume-Uni et ingénieur électricien agréé. Dogan est l'auteur de plus de 100 livres techniques et de plus de 200 articles techniques sur l'électronique, les microprocesseurs, les microcontrôleurs et les domaines connexes. Dogan est un expert certifié d'Arduino et a de nombreuses années d'expérience avec presque tous les types de microprocesseurs et de microcontrôleurs.

LIEN

[1] Motor Control Development Bundle: https://elektor.fr/motor-control-development-bundle