HEH.be Sciences et technologies

PWM et moteurs

Nano-ordinateurs

François Roland

WALLONIE-BRUXELLES ENSEIGNEMENT

pôle hainuyer

Notes

1. Introduction
2. PWM
3. Moteur DC
4. Servomoteur
5. Conclusion

Notes

Objectif : contrôle d'une charge

- Contrôler la vitesse d'un moteur DC
- Ajuster l'intensité lumineuse d'une LED
- Créer des couleurs intermédiaires avec une LED RGB

⇒ nécessité de moduler la puissance délivrée à la charge.

Notes

Problématique liée aux sorties digitales

- Binaires (0ou1)
- Contrôlent uniquement la tension (0 Vou3,3 V)

Notes

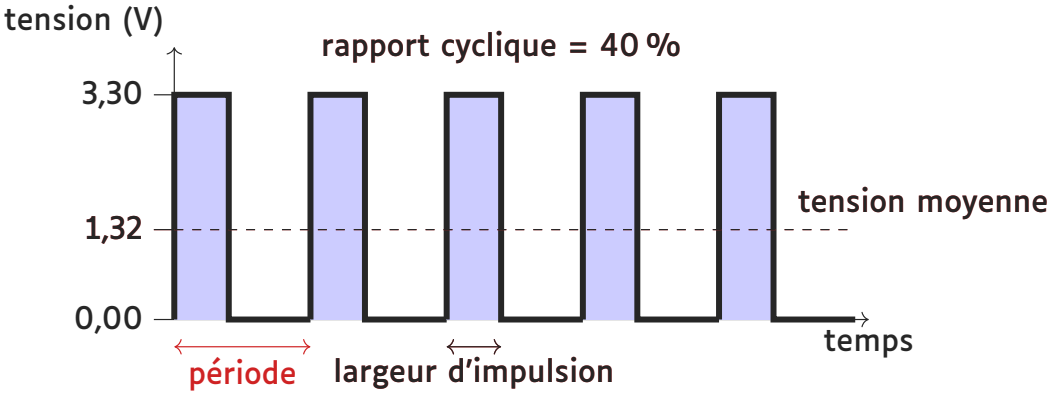
Notes

Principe du PWM

- PWM = Pulse Width Modulation (modulation de largeur d'impulsion)
- Signal numérique (0ou1) périodique (qui se répète)
- C'est le rapport cyclique qui détermine la puissance moyenne délivrée

Notes

Principe de la PWM



Le duty cycle (rapport cyclique) correspond au pourcentage du temps où la sortie est à 1. La tension moyenne est proportionnelle au rapport cyclique. La fréquence correspond au nombre de périodes par seconde.

Notes

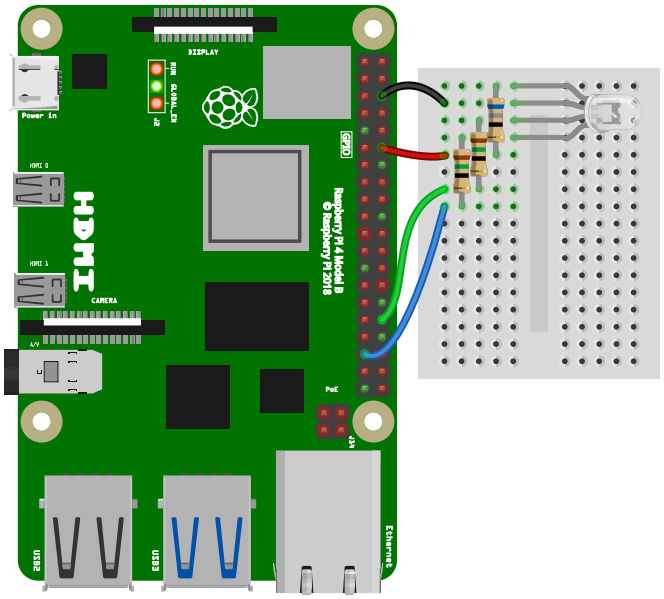
Choix de la fréquence

- Fréquence basse pour diminuer les coûts des composants.
- Fréquence basse pour éviter les interférences électromagnétiques.
- Fréquence élevée pour éviter les vibrations audibles ou visibles (LED qui scintille, moteur qui vibre).

⇒ le choix dépend de l'application et du type de charge.

Notes

LED RGB



Notes

LED RGB avec gpiozero

```
from time import sleep
from gpiozero import RGBLED
from colorzero import Color
from signal import signal, SIGINT

RED_PIN, GREEN_PIN, BLUE_PIN = 18, 12, 19
led = RGBLED(RED_PIN, GREEN_PIN, BLUE_PIN)

def pulse(color):
    led.pulse(
        on_color=color,
        n=3,
        fade_in_time=0.5,
        fade_out_time=0.5,
        background=False,
    )
```

```
signal(SIGINT, cleanup)
while True:
    pulse(Color("red"))
    pulse(Color("green"))
    pulse(Color("blue"))
    pulse(Color("yellow"))
    pulse(Color("magenta"))
    pulse(Color("cyan"))
    pulse(Color("white"))
    led.off()
    sleep(1)
```


Notes

Notes

Principes d'un moteur DC

- Demande un courant « élevé » pour démarrer et maintenir sa rotation
  ⇒ source d'alimentation dédiée
- Puissance proportionnelle à l'énergie électrique reçue
  ⇒ contrôle de la puissance par PWM
- Peut tourner dans les deux sens
  ⇒ contrôle de la direction par inversion de polarité

Notes

Limites des GPIO

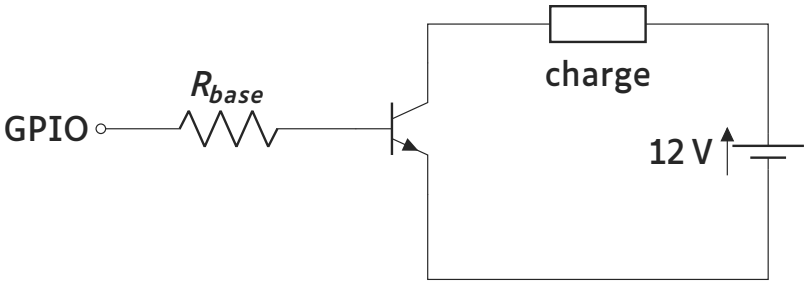
- Courant et tension limités → puissance limitée
- 16 mA pour 1 borne
- 2 mA en moyenne

Un moteur DC typique peut demander 100 mA ou plus au démarrage.

Notes

Transistor comme interrupteur

- Permet d'alimenter une charge plus gourmande
- Peut être commandé par une sortie GPIO
- Nécessite une résistance de base adaptée



Notes

Réseau Darlington ULN2003

- Intègre plusieurs transistors en boîtier unique
- Peut commander directement des relais ou moteurs
- Protégé contre les surtensions (diode de roue libre intégrée)

Notes

Commande bidirectionnelle d'un moteur DC

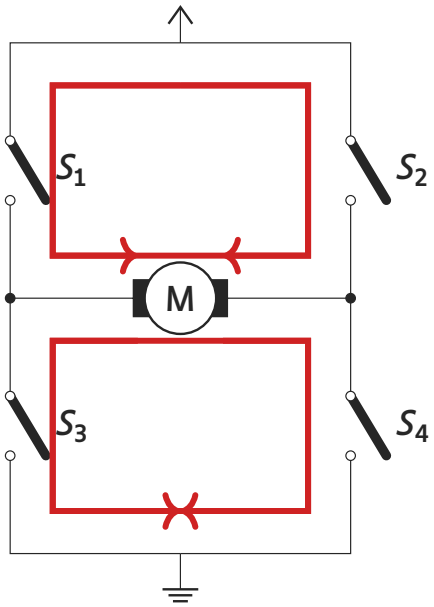
Pour changer le sens de rotation d'un moteur DC, il faut inverser la polarité de son alimentation.

Le pont en H permet d'inverser la polarité de l'alimentation du moteur, et ainsi de contrôler son sens de rotation.

Notes

Pont en H

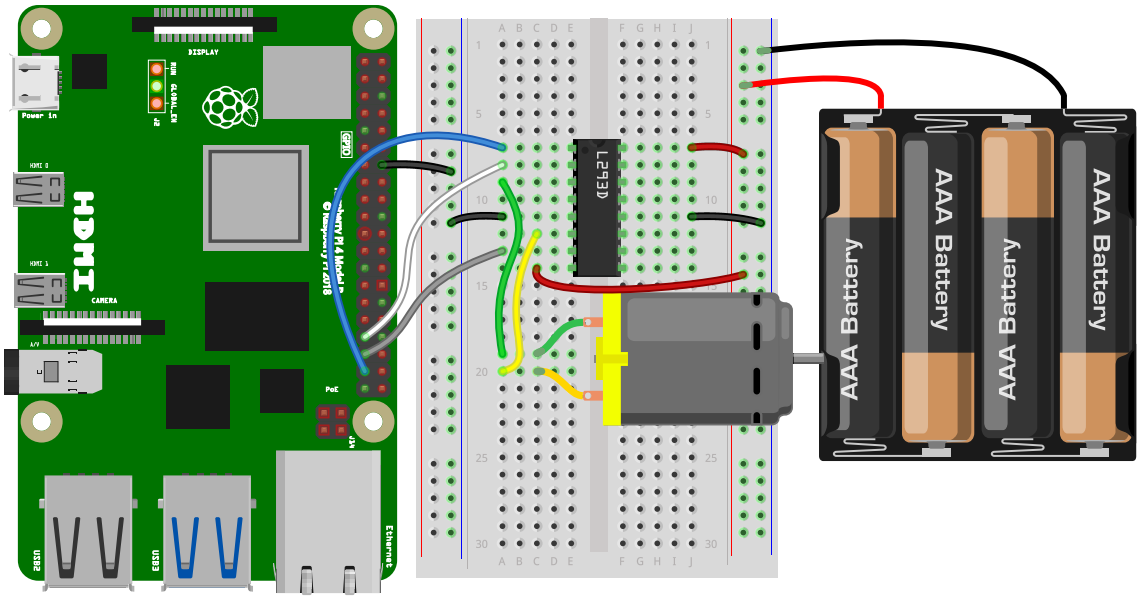
Schéma de principe



S_1	S_2	S_3	S_4	Action
ON	OFF	OFF	ON	sens direct
OFF	ON	ON	OFF	sens inverse
ON	ON	OFF	OFF	frein
OFF	OFF	ON	ON	frein
OFF	OFF	OFF	OFF	roue libre

Notes

Moteur DC avec pont en H L293D



Notes

Moteur DC avec pont en H

```python
from gpiozero import Motor
from signal import signal, SIGINT
from time import sleep

IN1_PIN, IN2_PIN, EN_PIN = 13, 19, 26
RUN_DURATION, STOP_DURATION = 10, 1
motor = Motor(IN1_PIN, IN2_PIN, enable=EN_PIN)
```

```python
while True:
    motor.forward(0.5)
    sleep(RUN_DURATION)
    motor.forward(1)
    sleep(RUN_DURATION)
    motor.stop()
    sleep(STOP_DURATION)
    motor.backward(0.8)
    sleep(RUN_DURATION)
    motor.stop()
    sleep(STOP_DURATION)
```


Notes

1 Introduction

2 PWM

3 Moteur DC

4 Servomoteur

5 Conclusion

Notes

Principe de fonctionnement

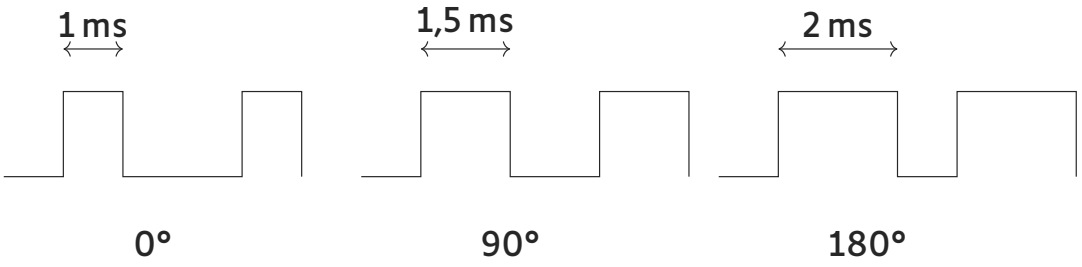
- Reçoit une consigne de position (angle)
- Conserve la position grâce à un système de feedback (rétroaction)

Notes

Commande d'un servomoteur

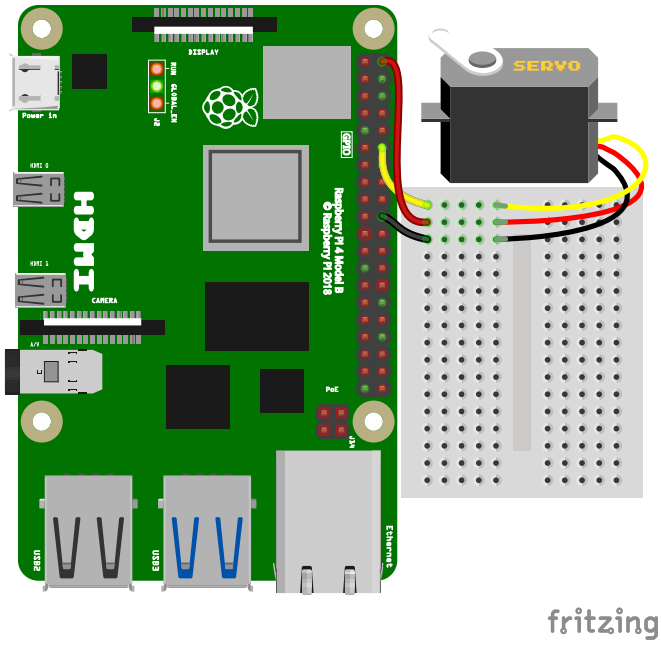
PWM avec propriétés spécifiques :

- Fréquence : 50 Hz
- Largeur d'impulsion → position



Notes

Servomoteur



Notes

Servomoteur

```
from gpiozero import AngularServo
from signal import signal, SIGINT
from time import sleep

servo = AngularServo(18, min_angle=0, max_angle=90)
while True:
    servo.angle = 0.0
    sleep(1)
    servo.angle = 45.0
    sleep(1)
    servo.angle = 90.0
    sleep(1)
```


Notes

1 Introduction

2 PWM

3 Moteur DC

4 Servomoteur

5 Conclusion

Notes

Résumé

- PWM pour moduler la puissance délivrée par une sortie digitale
- Contrôle d'une LED RGB par PWM
- Contrôle d'un moteur DC en puissance et en direction grâce à un pont en H
- Commande d'un servomoteur par PWM avec des propriétés spécifiques

Notes