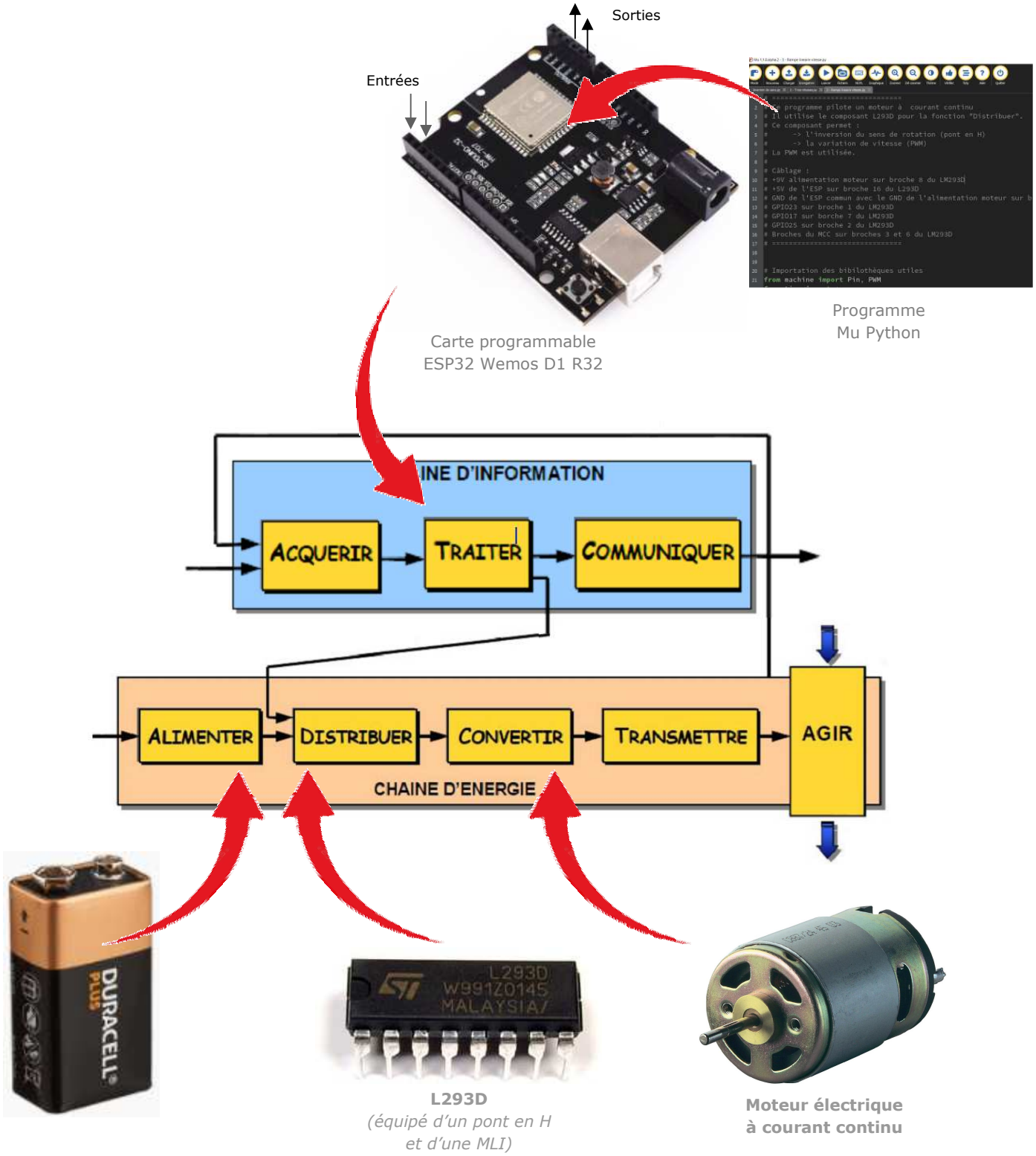
MISE EN ŒUVRE

→ **TRAITER : ESP32 WEMOS (EDI MU)**

→ **CONVERTIR : Moteur à courant continu**

→ **DISTRIBUER : L293D (PWM – MLI)**

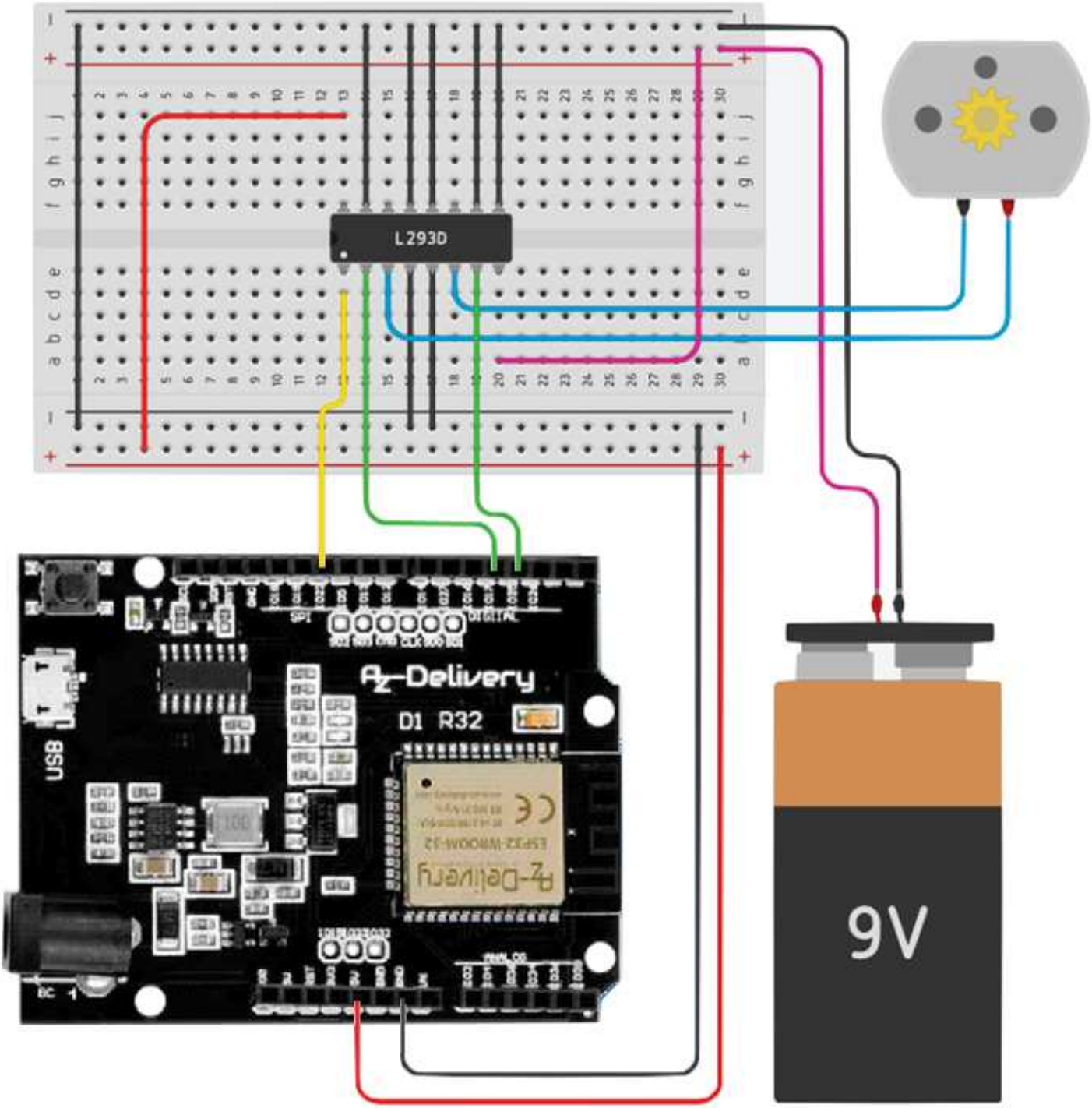
1 – Mise en situation



2 – Plan de câblage / Montage

Pour plus d'information sur le câblage du L293D, se reporter à la datasheet portant sur lui.

Attention : bien penser à mettre le GND en commun entre la carte ARDDUINO et la pile.



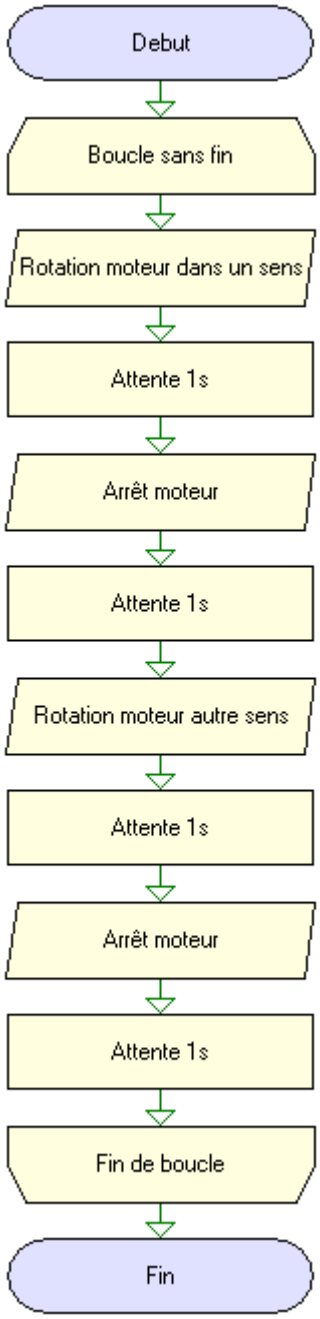
3 – Programmes

➘ PROGRAMME 1 : « Inversion sens rotation.py »

Ce programme fait tourner le moteur à sa vitesse maximale dans un sens puis dans l'autre avec un délai d'une seconde entre les deux.

➘ Ce programme n'utilise pas la MLI.

```python
# ==============================
# Ce programme pilote un moteur à  courant continu
# Il utilise le composant L293D pour la fonction "Distribuer".
# Ce composant permet :
#      -> l'inversion du sens de rotation (pont en H)
#      -> la variation de vitesse (PWM)
# La PWM n'est pas utilisée ici.
#
# Câblage :
# +9V alimentation moteur sur broche 8 du LM293D
# +5V de l'ESP sur broche 16 du L293D
# GND de l'ESP commun avec le GND de l'alimentation moteur sur broches 4 et 5 du LM293D
# GPIO23 sur broche 1 du LM293D
# GPIO17 sur borche 7 du LM293D
# GPIO25 sur broche 2 du LM293D
# Broches du MCC sur broches 3 et 6 du LM293D
# ==============================


# Importation des bibilothèques utiles
from machine import Pin
from time import *


# Définition des broches utilisées
PIN_vitesse = Pin(23, Pin.OUT)# Pas de PWM pour la variation (Tout Ou Rien sur l'éntrée Eneable (broche 1) du L293D)
PIN_sens1 = Pin(25, Pin.OUT)  # broche 25 (D3) sur ESP 32 cablée sur l'entrée 1 (broche 2) du L293D
PIN_sens2 = Pin(17, Pin.OUT)  # broche 17 (D4) sur ESP 32 cablée sur l'entrée 2 (broche 7) du L293D


while True :
    PIN_sens1.value(1)
    PIN_sens2.value(0)
    PIN_vitesse.value(1)
    sleep_ms(1000)

    PIN_vitesse.value(0)
    sleep_ms(1000)

    PIN_sens1.value(0)
    PIN_sens2.value(1)
    PIN_vitesse.value(1)
    sleep_ms(1000)

    PIN_vitesse.value(0)
    sleep_ms(1000)
```

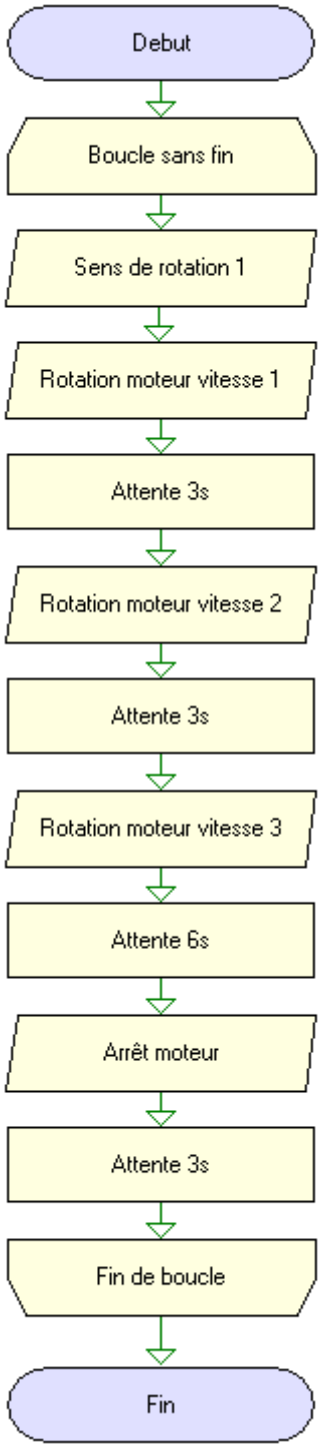


➘ PROGRAMME 2 : « Trois vitesses.py »

Ce programme fait tourner le moteur à trois vitesses successives différentes pendant des durées données puis s'arrête, là aussi pendant une durée donnée.

➘ Ce programme utilise la MLI.

```python
# ==============================
# Ce programme pilote un moteur à  courant continu
# Il utilise le composant L293D pour la fonction "Distribuer".
# Ce composant permet :
#       -> l'inversion du sens de rotation (pont en H)
#       -> la variation de vitesse (PWM)
# La PWM est utilisée.
#
# Câblage :
# +9V alimentation moteur sur broche 8 du LM293D
# +5V de l'ESP sur broche 16 du L293D
# GND de l'ESP commun avec le GND de l'alimentation moteur sur broches 4 et 5 du LM293D
# GPIO23 sur broche 1 du LM293D
# GPIO17 sur borche 7 du LM293D
# GPIO25 sur broche 2 du LM293D
# Broches du MCC sur broches 3 et 6 du LM293D
# ==============================


# Importation des bibilothèques utiles
from machine import Pin, PWM
from time import *


# Définition des broches utilisées
MCC = PWM(Pin(23), 5000)        # attache un objet nommé MCC de type MLI de fréquence 5000Hz sur broche 23 (D11)
PIN_sens1 = Pin(25, Pin.OUT)    # broche 25 (D3) sur ESP 32 cablée sur l'entrée 1 (broche 2) du L293D
PIN_sens2 = Pin(17, Pin.OUT)    # broche 17 (D4) sur ESP 32 cablée sur l'entrée 2 (broche 7) du L293D


while True:
# Rappel : l'envoi d'un nombre compris entre 0 et 1023 sur l'objet PWM provoque une signal impulsionnel sur la broche attaché
#                  à l'objet MLI dont le rapport cyclique vaut entre 0% et 100%
    PIN_sens1.value(1)
    PIN_sens2.value(0)

    MCC.duty(200)   # Applique la méthode duty à l'objet MCC (crée une impusion dont le rapport cyclique vaut 19,5% (200/1023)
    sleep_ms(3000)

    MCC.duty(511)   # Applique la méthode duty à l'objet MCC (crée une impusion dont le rapport cyclique vaut 50% (511/1023)
    sleep_ms(3000)

    MCC.duty(1023)  # Applique la méthode duty à l'objet MCC (crée une impusion dont le rapport cyclique vaut 100% (1023/1023)
    sleep_ms(6000)

    MCC.duty(0)     # Applique la méthode duty à l'objet MCC (crée une impusion dont le rapport cyclique vaut 0% 50/1023)
    sleep_ms(3000)
```



➘ PROGRAMME 3 : « Rampe linéaire vitesse.py »

Ce programme fait varier progressivement la vitesse du moteur.

Il l'amène d'une vitesse initiale (nulle ou non nulle) à une vitesse finale en suivant une rampe linéaire, sur une durée donnée.

➘ Les vitesses initiale, finale, et la durée de la rampe sont paramétrées.

➘ **Ce programme utilise la MLI.**

```python
# ================================
# Ce programme pilote un moteur à  courant continu
# Il utilise le composant L293D pour la fonction "Distribuer".
# Ce composant permet :
#       -> l'inversion du sens de rotation (pont en H)
#       -> la variation de vitesse (PWM)
# La PWM est utilisée.
#
# Câblage :
# +9V alimentation moteur sur broche 8 du LM293D
# +5V de l'ESP sur broche 16 du L293D
# GND de l'ESP commun avec le GND de l'alimentation moteur sur broches 4 et 5 du LM293D
# GPIO23 sur broche 1 du LM293D
# GPIO17 sur borche 7 du LM293D
# GPIO25 sur broche 2 du LM293D
# Broches du MCC sur broches 3 et 6 du LM293D
# ================================

# Importation des bibilothèques utiles
from machine import Pin, PWM
from time import *

# Définition des broches utilisées
MCC = PWM(Pin(23), 5000)  # attache un objet nommé MCC de type MLI de fréquence 5000Hz sur broche 23 (D11)
PIN_sens1 = Pin(25, Pin.OUT)  # broche 25 (D3) sur ESP 32 cablée sur l'entrée 1 (broche 2) du L293D
PIN_sens2 = Pin(17, Pin.OUT)  # broche 17 (D4) sur ESP 32 cablée sur l'entrée 2 (broche 7) du L293D

# Définition des paramètres de la rampe (régler ce que l'on veut)
duree = 10.0  # durée de la variation de vitesse (durée de la rampe)
N_initiale = 0  # nombre image de la vitesse initiale de la rampe (rapport cyclique minimal 0%)
N_finale = 1023  # nombre image de la vitesse finale de la rampe (1023 max) (rapport cyclique maximal 100%)
# !! attention !! : N_finale doit être différente de N_initiale.

while True:
    Ninst = N_initiale  # initialisation de la vitesse instantanée
    pause = int(1000 * duree / abs(N_finale - N_initiale))  # calcul de la durée entre 2 changements de rapport cyclique po

    while Ninst != N_finale:
        Ninst = Ninst + 1
        print("rapport cyclique = ", int(Ninst / 1023 * 100), "%")
        PIN_sens1.value(1)
        PIN_sens2.value(0)
        MCC.duty(Ninst)
        sleep_ms(pause)
    sleep_ms(5000)

    MCC.duty(0)
    print("rapport cyclique = 0%")
    sleep_ms(10000)
```