Série TP n° 6 : commande prototype

N. Mesnier
Lycée Jean Perrin, Lyon

2023–2024

Objectifs

- démystifier les cartes de prototypage et leur programmation ;
- réaliser des câblages élémentaires de prototypage en robotique ;
- acquérir et traiter des informations analogiques ou numériques ;
- commander un moteur à courant continu par l'intermédiaire d'un hacheur ;
- mettre en place un asservissement !

Vous présenter des
outils
pouvant vous être utiles pour votre
TIPE !

Objectifs

- démystifier les cartes de prototypage et leur programmation ;
- réaliser des câblages élémentaires de prototypage en robotique ;
- acquérir et traiter des informations analogiques ou numériques ;
- commander un moteur à courant continu par l'intermédiaire d'un hacheur ;
- mettre en place un asservissement !

Vous présenter des

outils

pouvant vous être utiles pour votre

TIPE !

Plan de la synthèse

1. Robot Arexx
2. Axe seul & vérin électrique

Robot Arexx

Robot Arexx

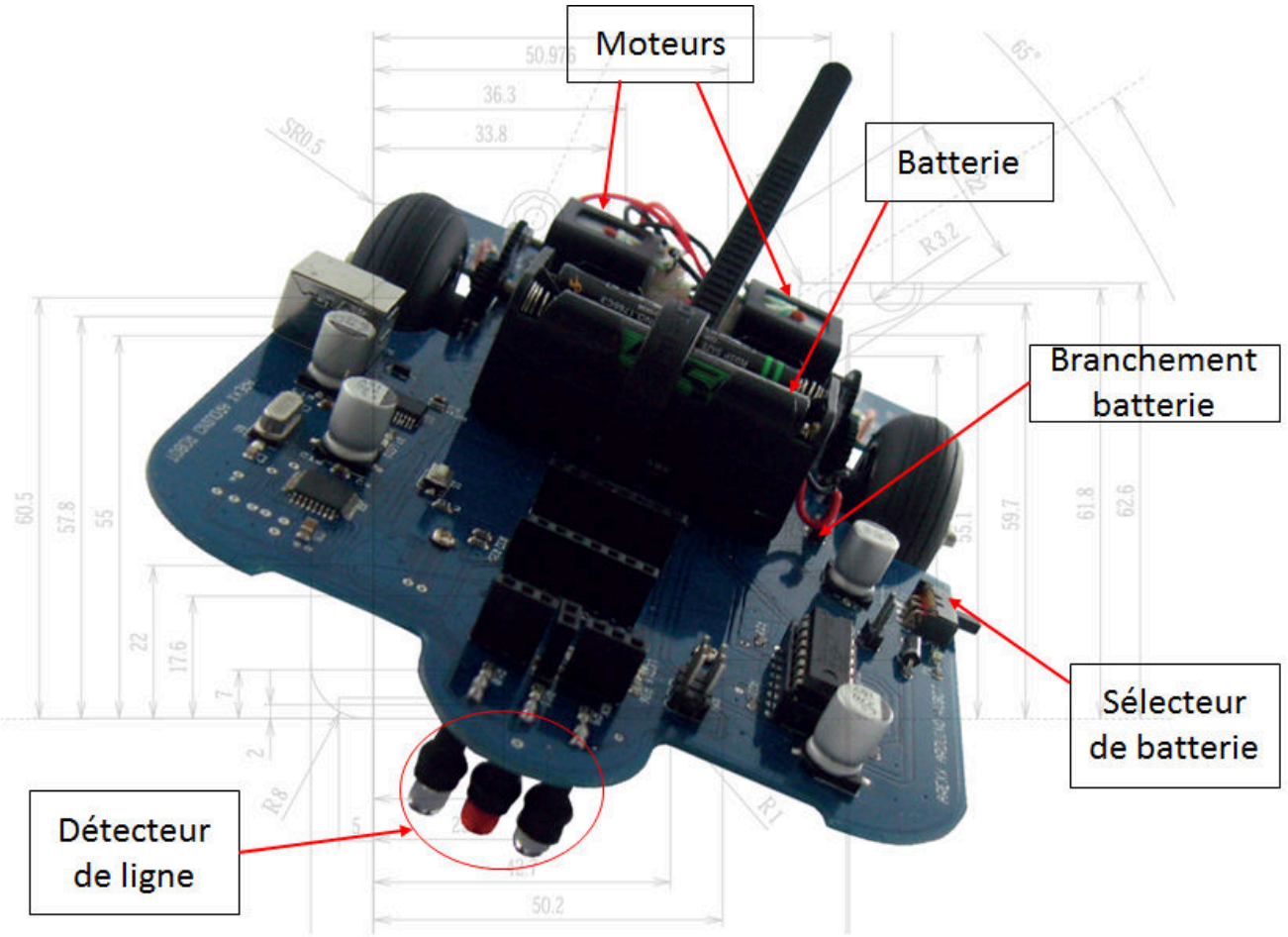

Moteurs
Batterie
Branchement batterie
Sélecteur de batterie
Détecteur de ligne

Schéma électrique

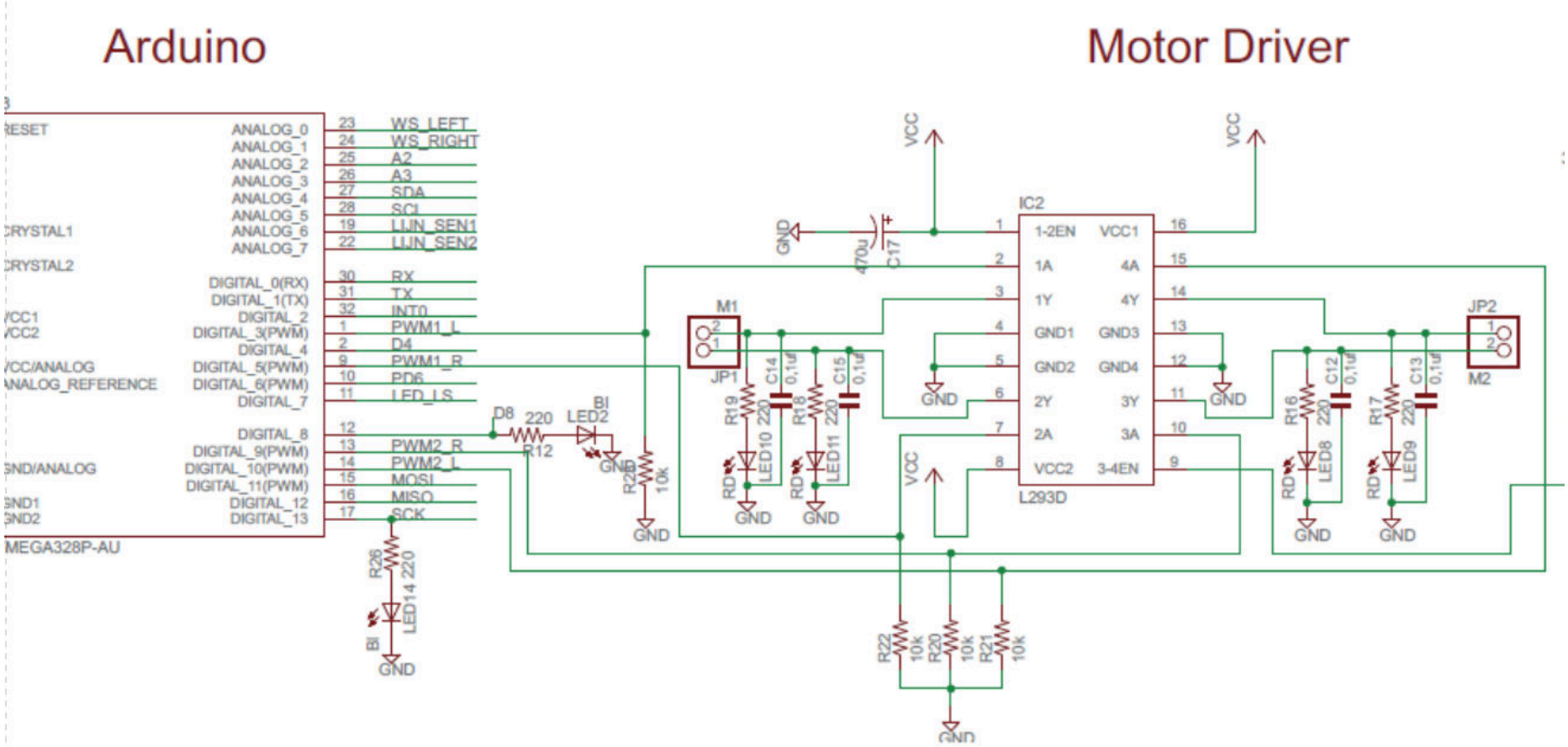
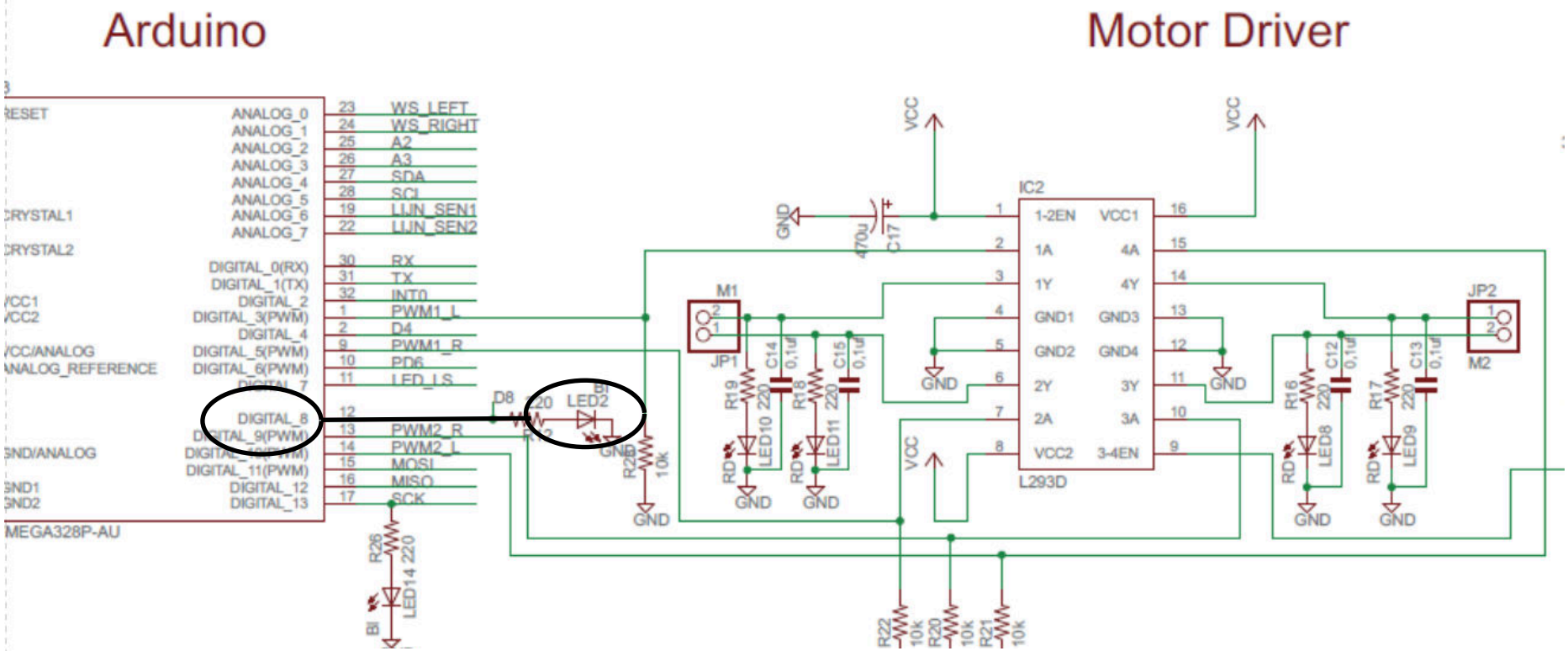
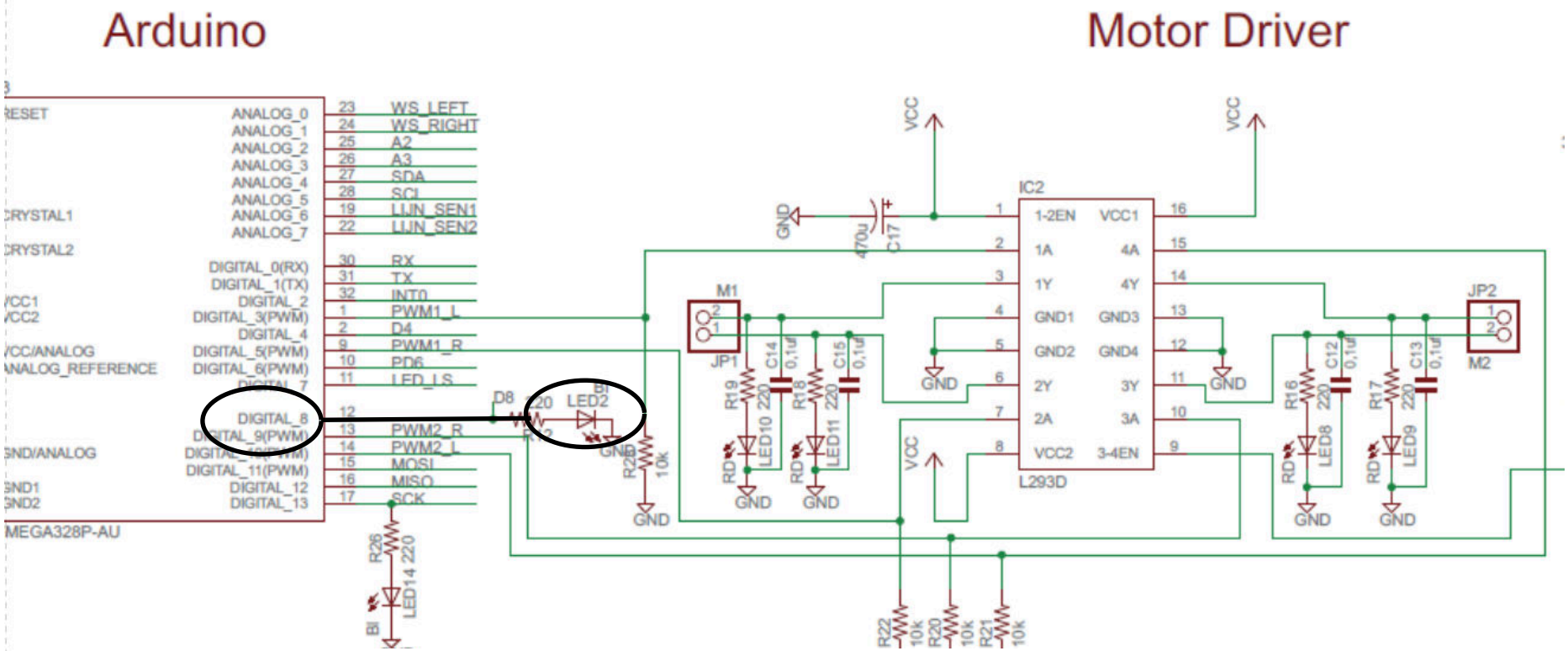


Schéma électrique : LED2



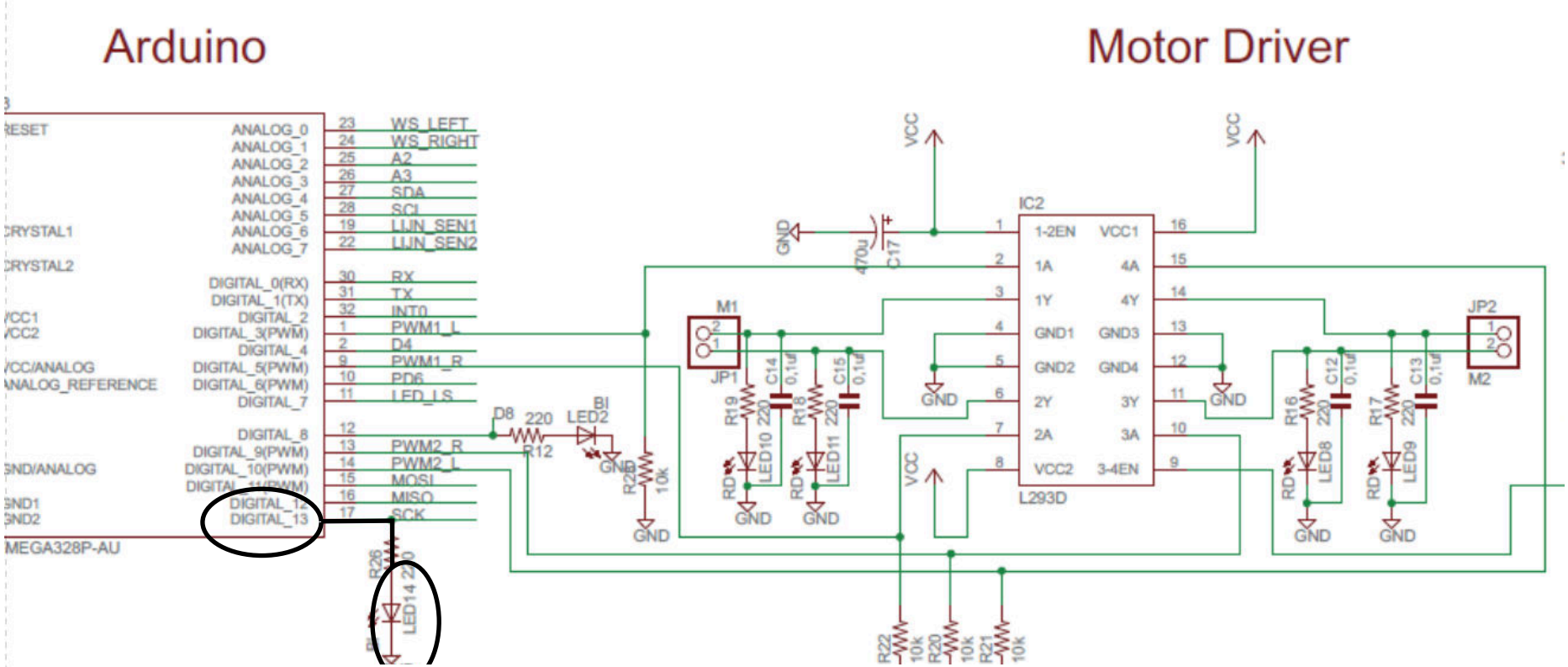
LED2 ⟺ OUTPUT PIN 8

Schéma électrique : LED2



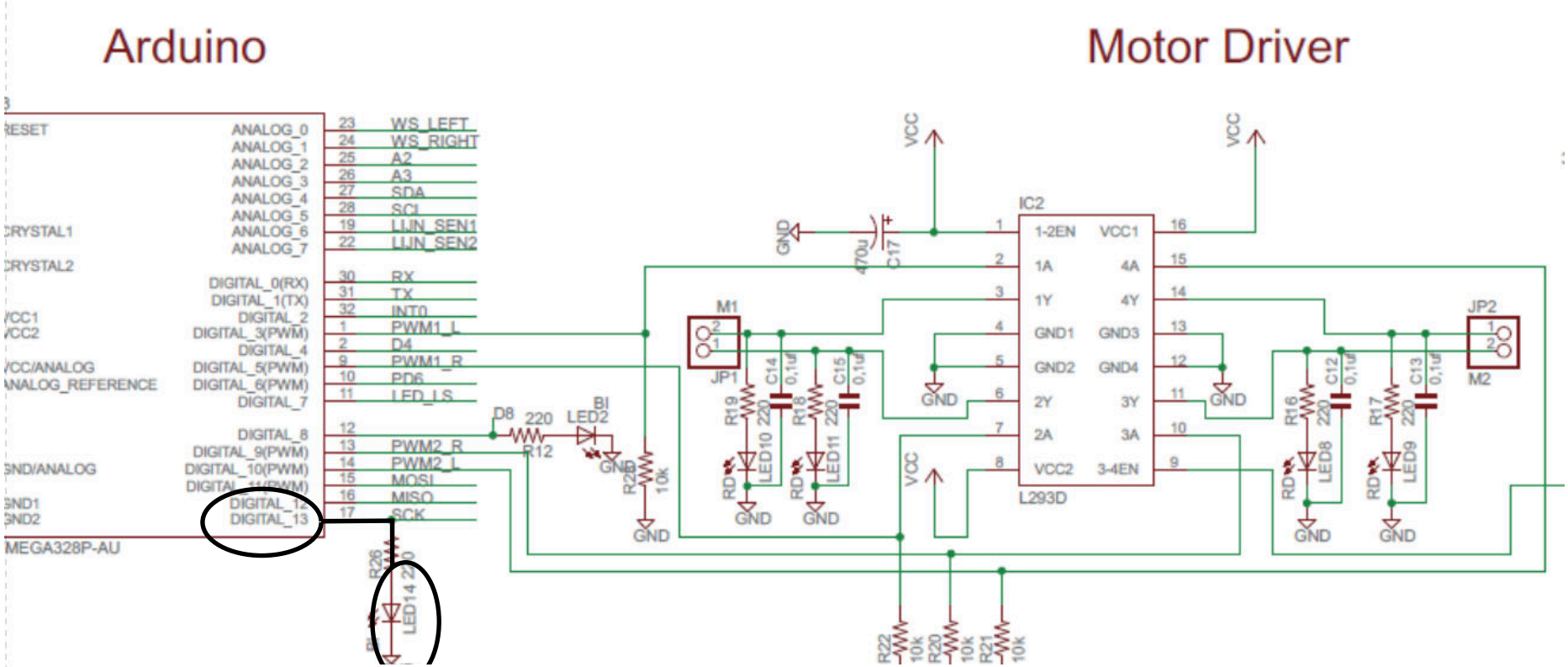
LED2 $\Longleftrightarrow$ OUTPUT PIN 8

Schéma électrique : LED14



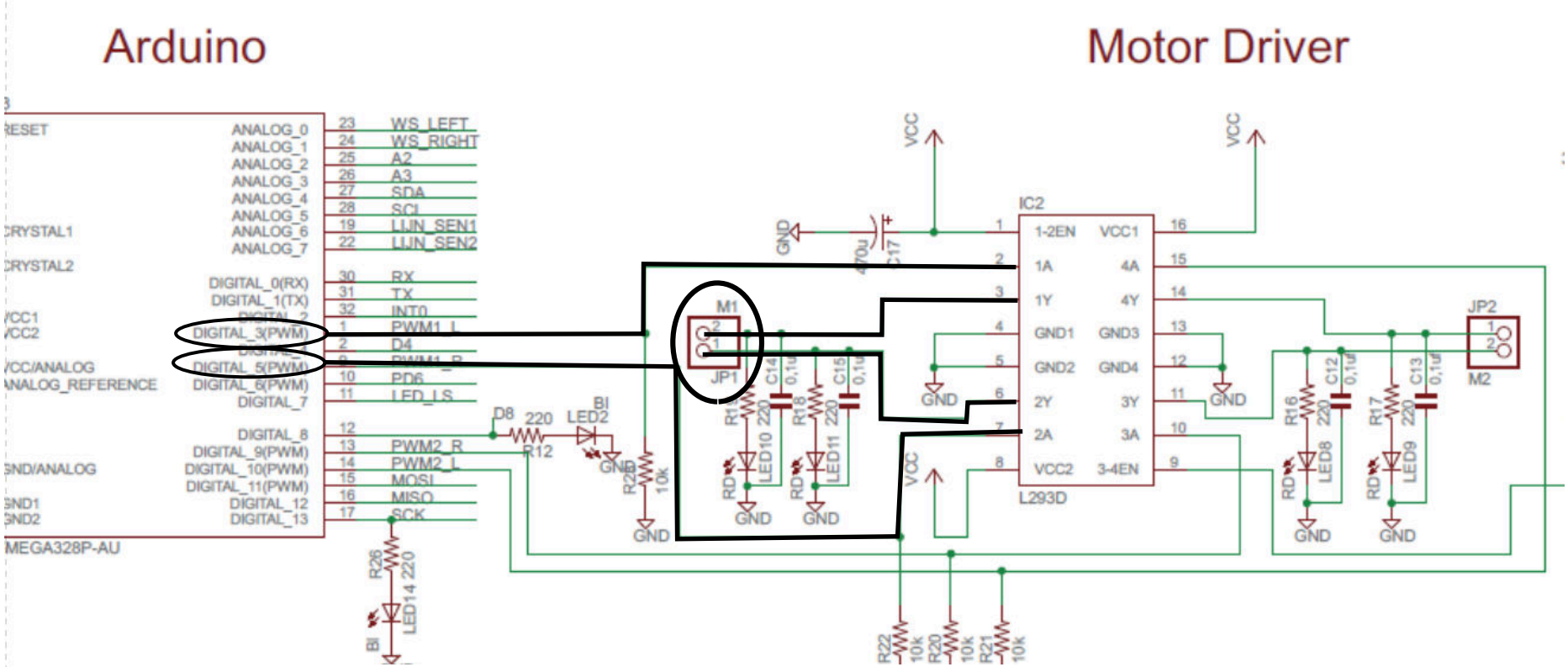
LED14 ⟺ OUTPUT PIN 13

Schéma électrique : LED14



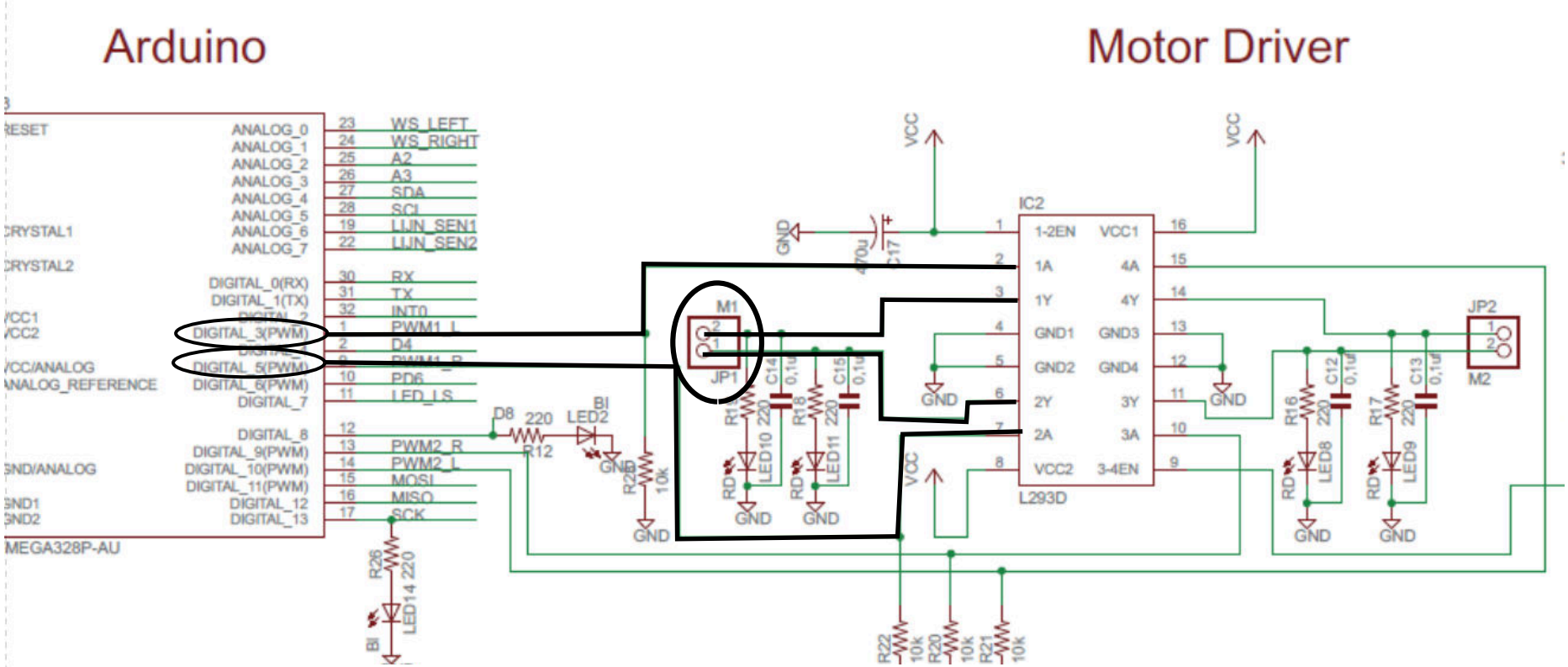
LED14 $\Longleftrightarrow$ OUTPUT PIN 13

Schéma électrique : Moteur M1



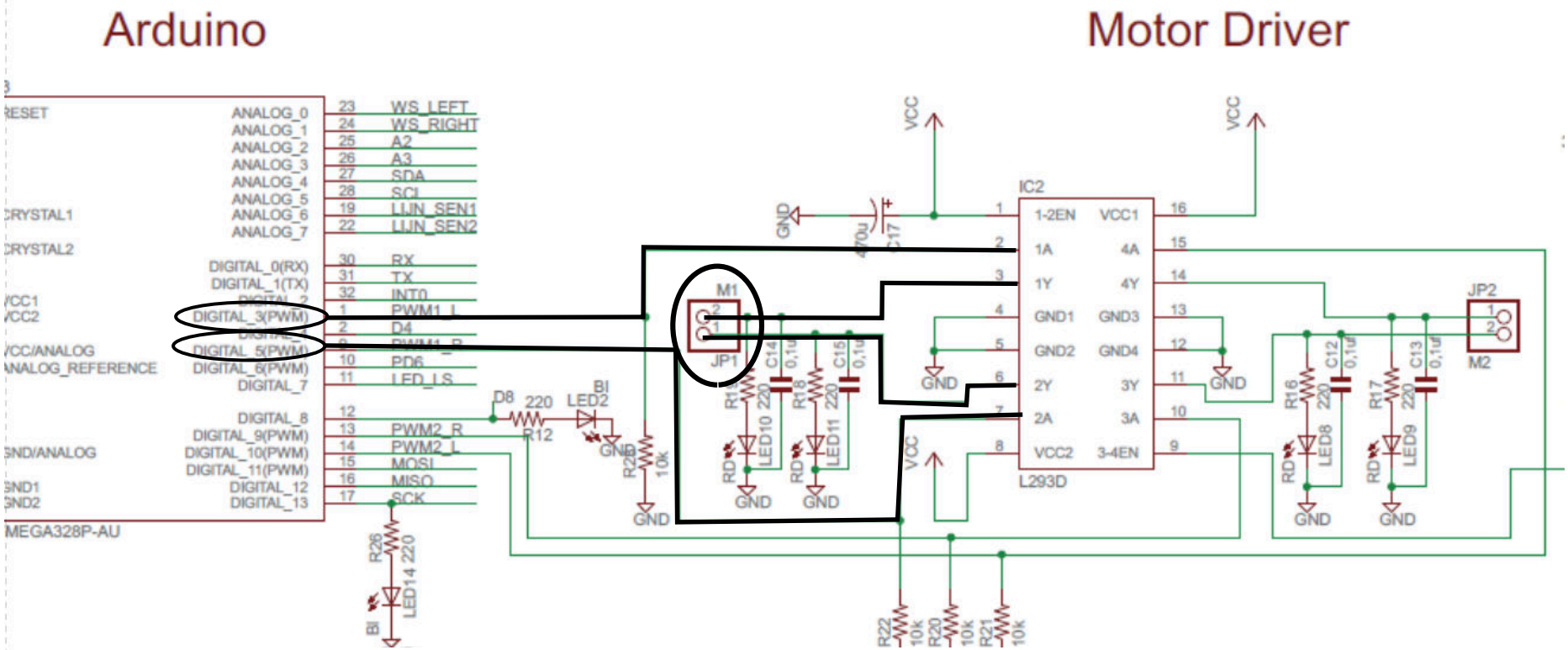
M1 ⟺ OUTPUT PIN 3 & 5

Schéma électrique : Moteur M1



M1 $\Longleftrightarrow$ OUTPUT PIN 3 & 5

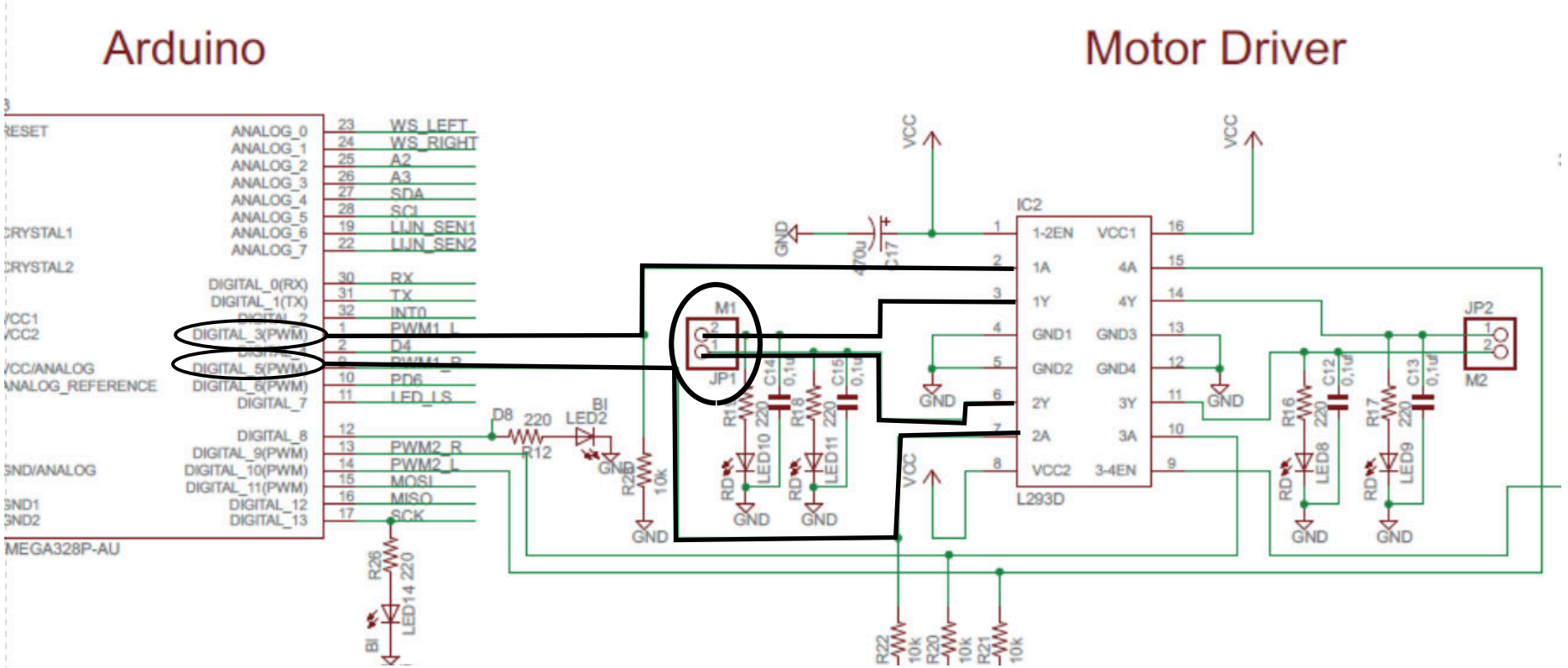
Schéma électrique : Moteur M1



M1 $\Longleftrightarrow$ OUTPUT PIN 3 & 5

Faire un essai pour avoir le sens !

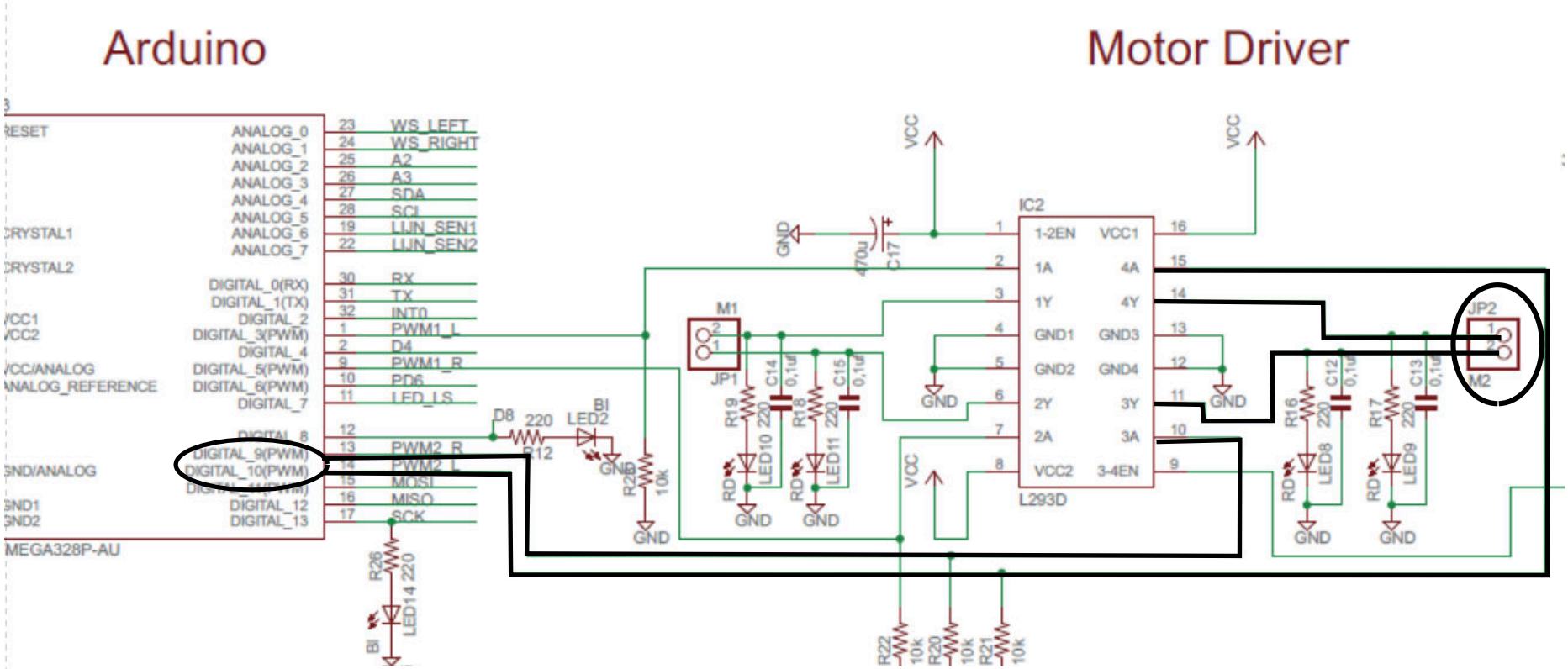
Schéma électrique : Moteur M1



M1 $\Longleftrightarrow$ OUTPUT PIN 3 & 5

Après essai : moteur gauche 5 (avant) & 3 (arrière)

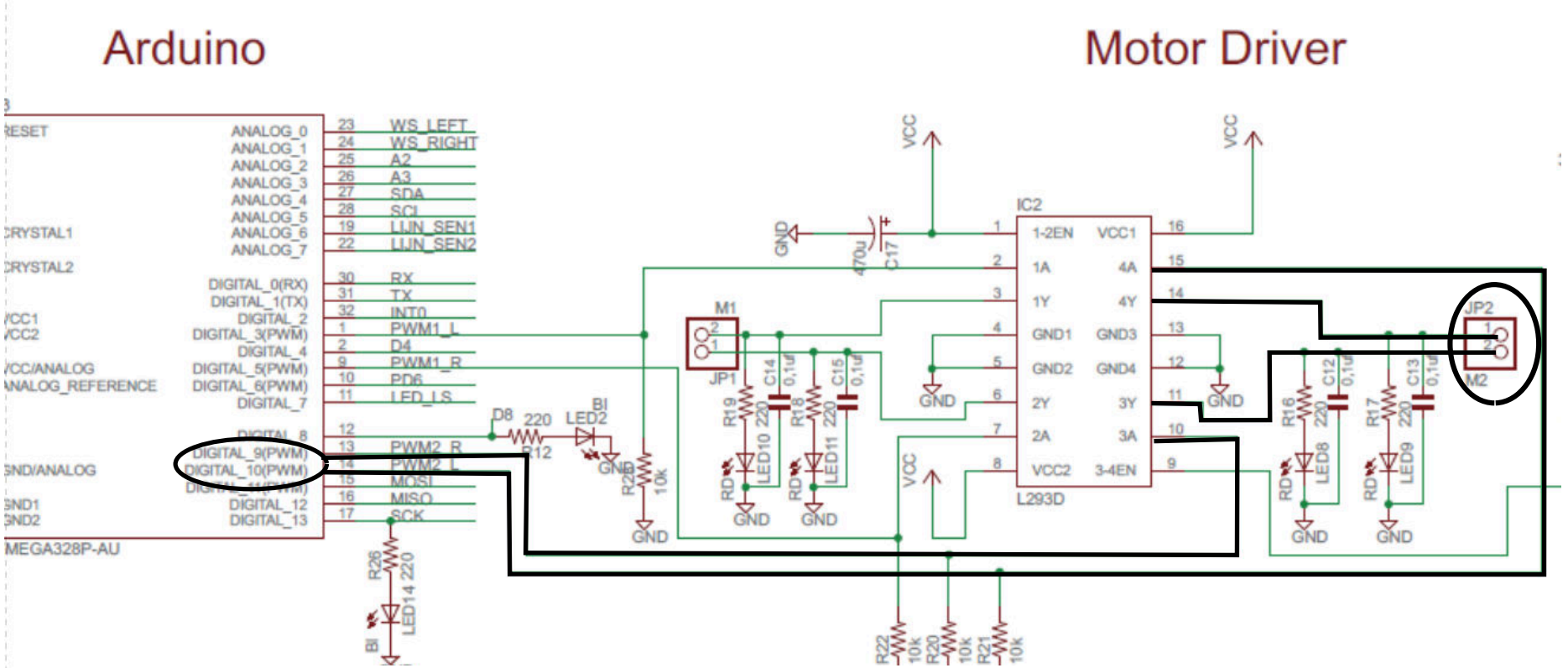
Schéma électrique : Moteur M2



M2 ⟺ OUTPUT PIN 9 & 10

Après essai : moteur droite 9 (avant) & 10 (arrière)

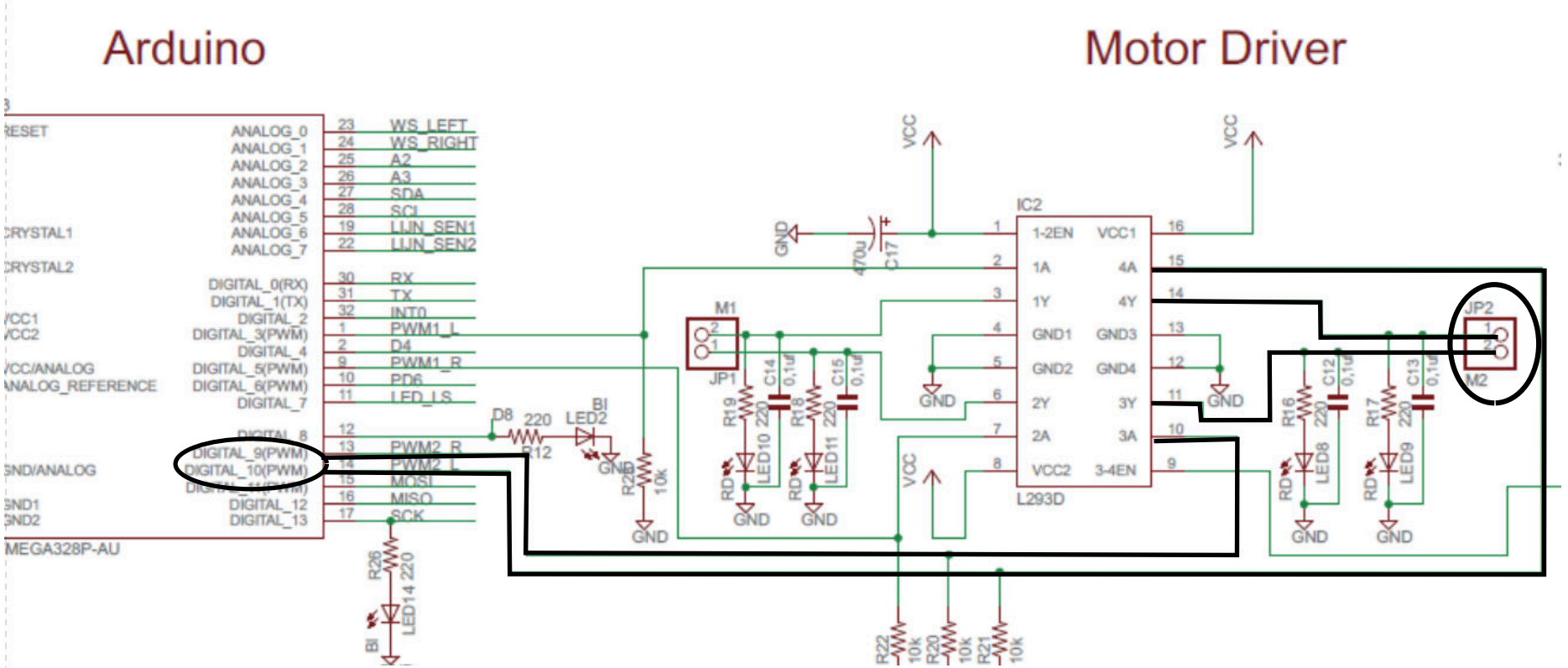
Schéma électrique : Moteur M2



M2 $\Longleftrightarrow$ OUTPUT PIN 9 & 10

Après essai : moteur droite 9 (avant) & 10 (arrière)

Schéma électrique : Moteur M2



M2 $\Longleftrightarrow$ OUTPUT PIN 9 & 10

Après essai : moteur droite 9 (avant) & 10 (arrière)

Robot Arexx

Ligne droite

- pins 5 et 9 pour la marche avant ;
- commande du hacheur sur 8 bits ($[\![0, 255]\!]$) ;

Problème observé

Le robot ne se déplace pas en ligne droite !

Solutions possibles

- trouver une consigne pour chaque moteur
  de sorte que le* robot aille en ligne droite
  $\star$ *à chaque robot son jeu de valeurs (...)* $\Longrightarrow$ bidouille $\neq$ science !
- mettre en place un asservissement.

Robot Arexx

Ligne droite

- pins 5 et 9 pour la marche avant ;
- commande du hacheur sur 8 bits ($[\![0, 255]\!]$) ;

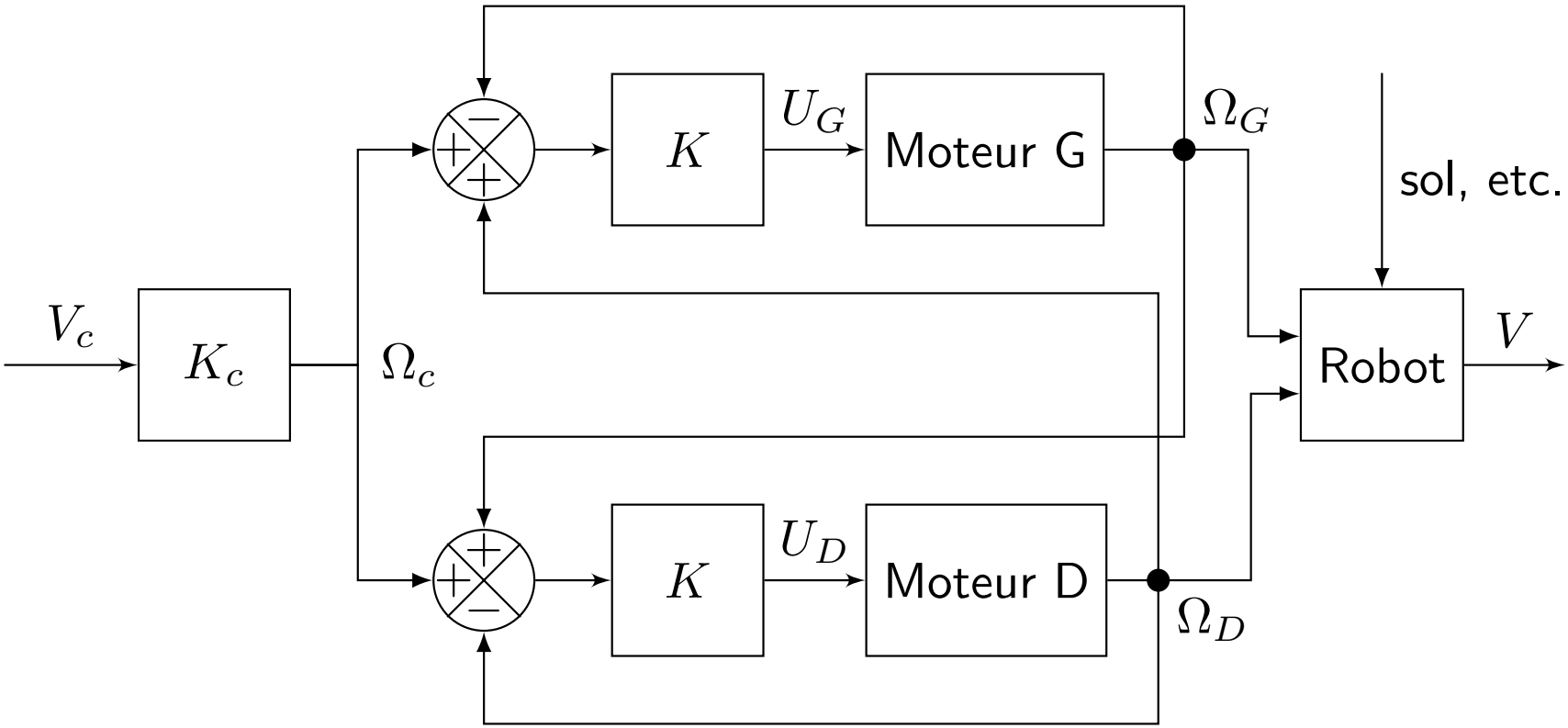
Problème observé

Le robot ne se déplace pas en ligne droite !

Solutions possibles

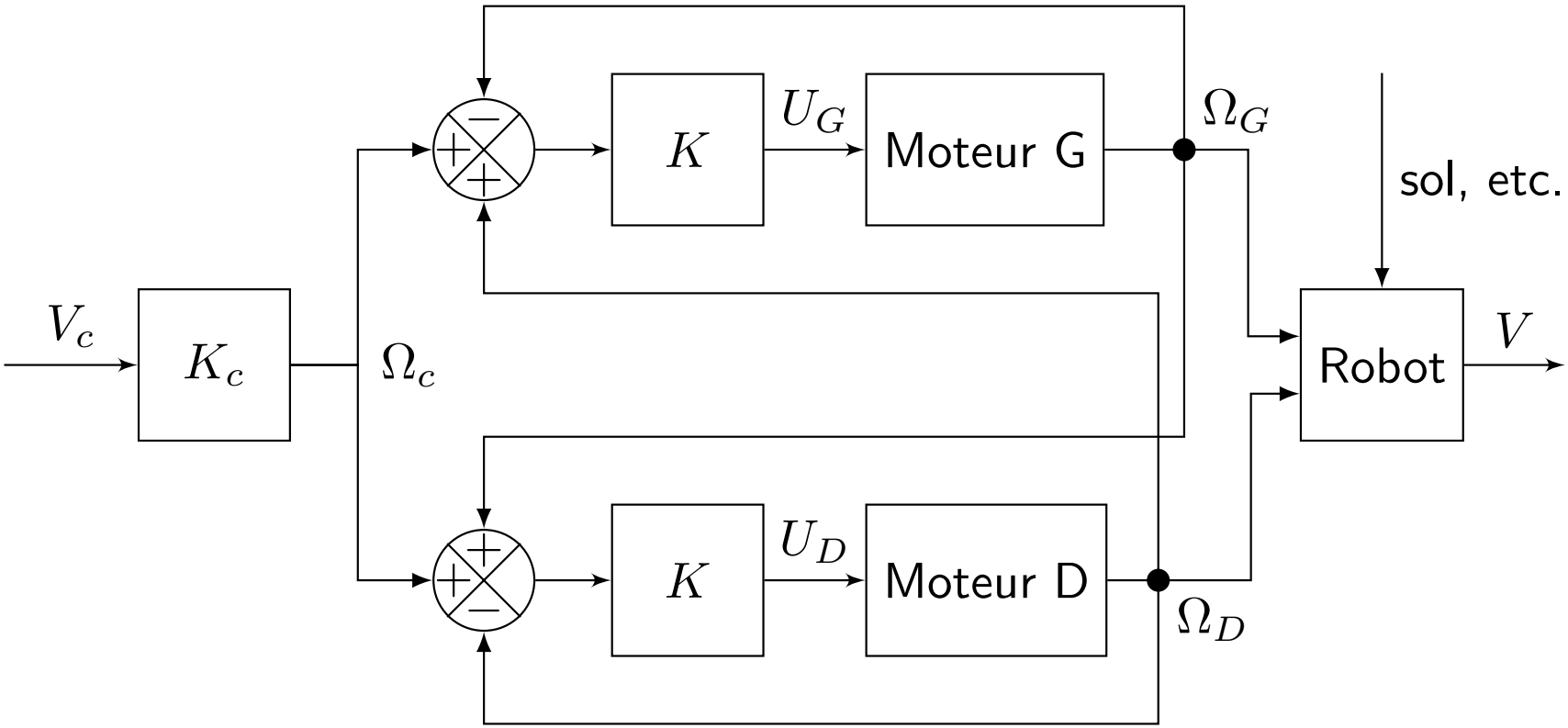
- trouver une consigne pour chaque moteur
  de sorte que le* robot aille en ligne droite
  $\star$ *à chaque robot son jeu de valeurs (...)* $\Longrightarrow$ bidouille $\neq$ science !
- mettre en place un asservissement.

Principe de l'asservissement



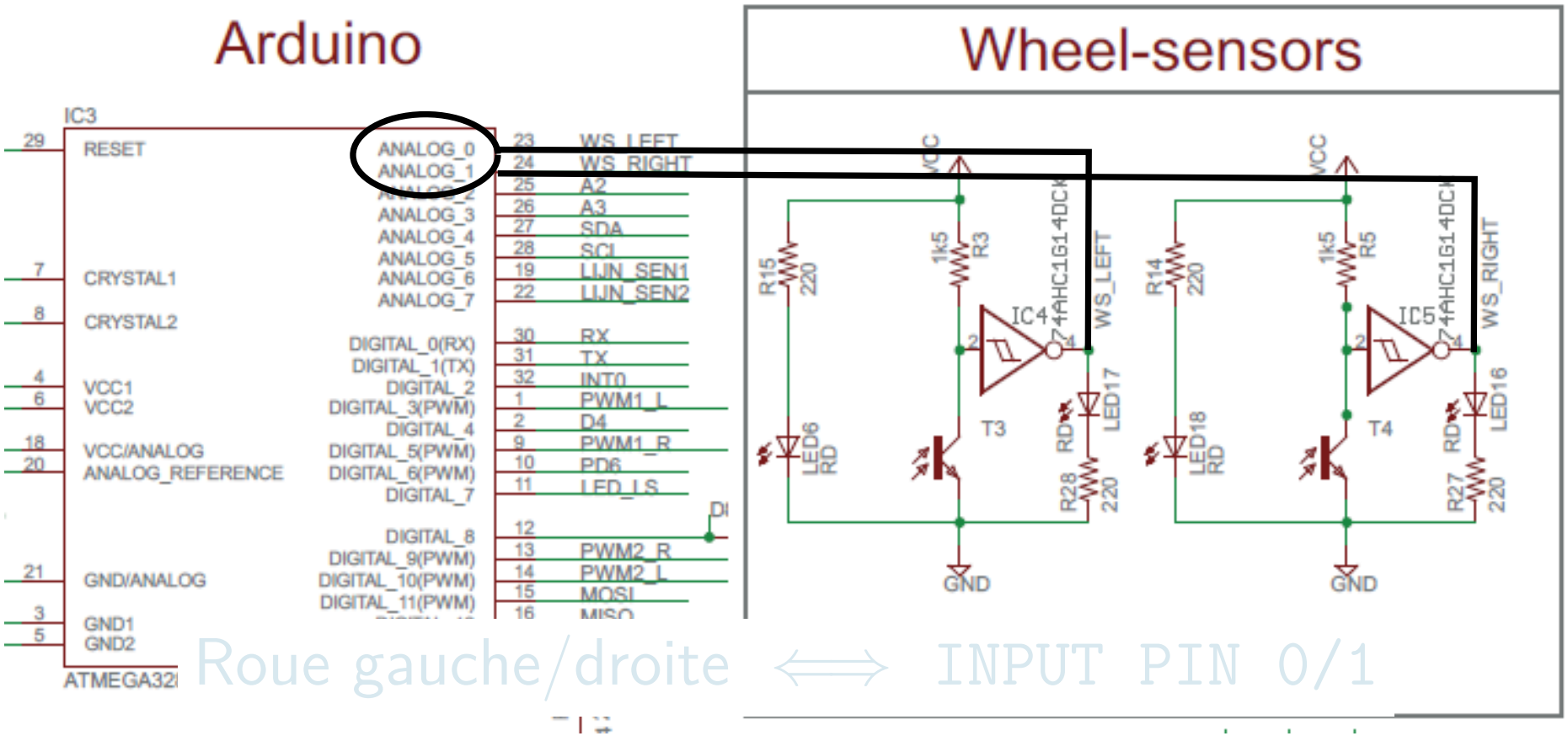
Capteurs de vitesse des roues ?

Principe de l'asservissement

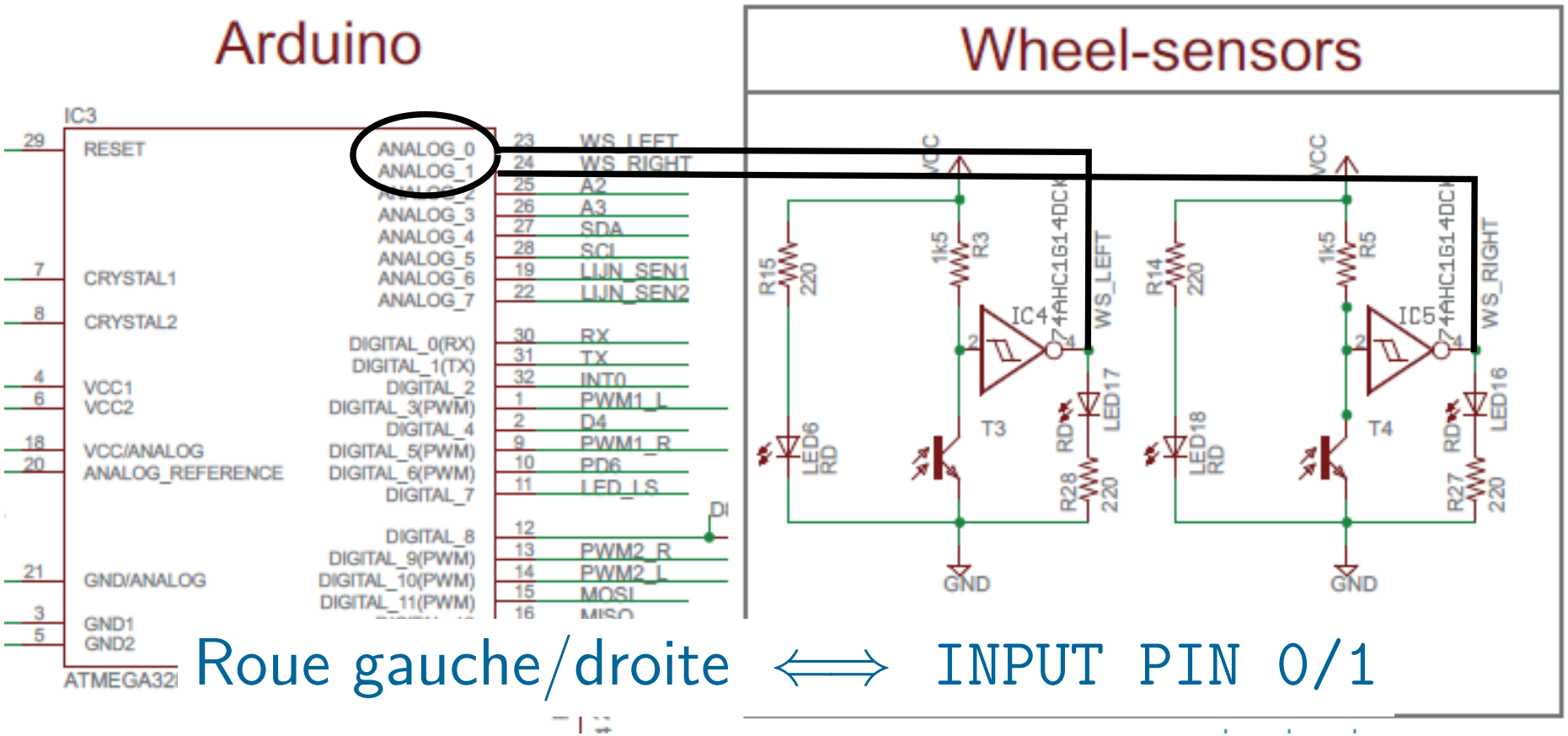


Capteurs de vitesse des roues ?

- Schéma électrique : fouches optiques



- Schéma électrique : fouches optiques



Acquisition des signaux

- essai avec une roue, à la main ;
- afficher les informations dans un terminal série :

```
void setup(){
    Serial.begin(9600);
    ...
}
void loop(){
    ...
    Serial.println(...);
    ...
}
```

Acquisition des signaux

- essai avec une roue, à la main ;
- afficher les informations dans un terminal série ;
- tester la nature du signal (cas du capteur gauche `pin 0`) :
  - signal logique (binaire) ?

```
...
Serial.println(digitalRead(0));
...
```

Renvoie toujours 0 $\Longrightarrow$ non !

  - signal analogique ?

```
...
Serial.println(analogRead(0));
...
```

Renvoie 0 ou ~ 940 $\Longrightarrow$ signal analogique quantifié sur 10 bits !

Acquisition des signaux

- essai avec une roue, à la main ;
- afficher les informations dans un terminal série ;
- tester la nature du signal ;
- fabriquer un signal logique quantifié manuellement :
  (cas du capteur gauche `pin 0`)

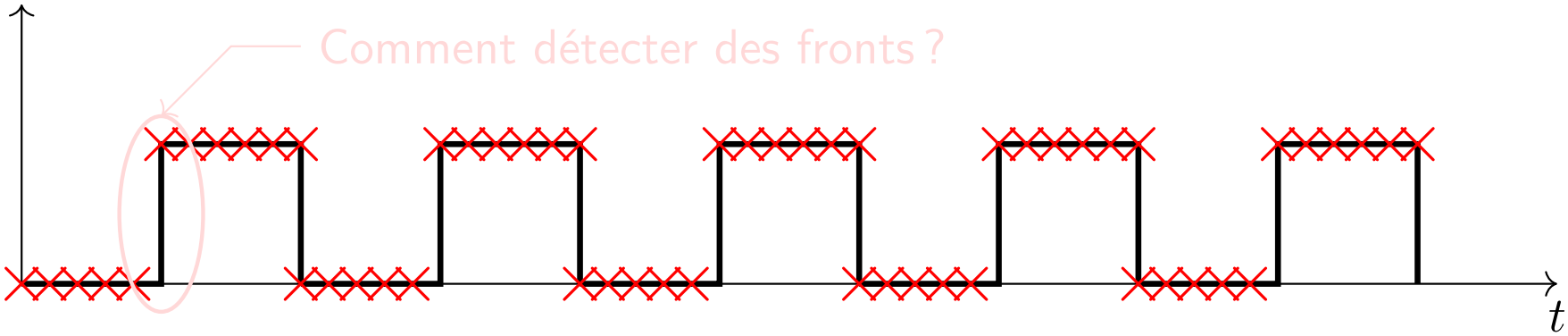
```
...
Serial.println(analogRead(0)>900);
...
```

Renvoie 0 ou 1

selon que le phototransistor reçoit la lumière de la LED ou non.

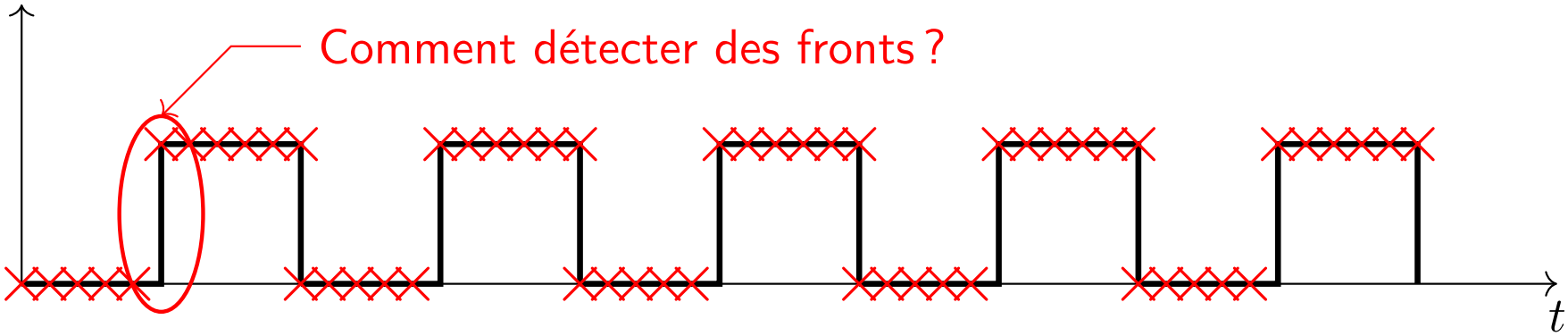
Acquisition des signaux

- essai avec une roue, à la main ;
- afficher les informations dans un terminal série ;
- tester la nature du signal ;
- fabriquer un signal logique quantifié manuellement ;
- mettre en place un compteur :



Acquisition des signaux

- essai avec une roue, à la main ;
- afficher les informations dans un terminal série ;
- tester la nature du signal ;
- fabriquer un signal logique quantifié manuellement ;
- mettre en place un compteur :



Mise en place du compteur

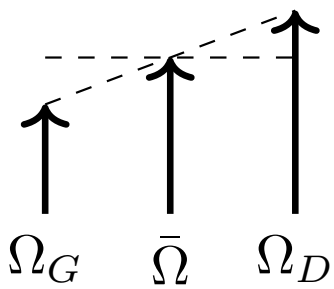
- détection des fronts :

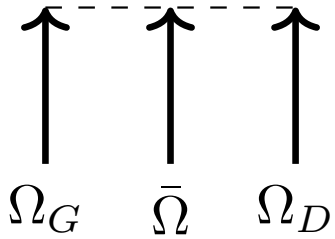
```
void loop(){
    ...
    cptG = analogRead(0)>900; // bit du capteur gauche
    if (cptG > cptG_old){     // front montant
        compteurG++;          // incrémentation du compteur
    }
    ...
    cptG_old = cptG;          // mise àjour du prédécesseur
}
```

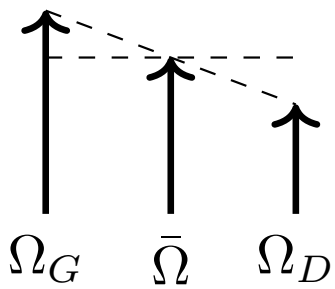
- essai avec une roue, à la main, en affichant les informations dans un terminal série.

Idée de l'asservissement

- vérifier tous les t les signaux des capteurs des roues
  et incrémenter le compteur de chaque roue (si besoin) ;
- modifier tous les $T \gg t$ la commande des moteurs
  avec une correction proportionnelle à vitesse moyenne $\bar{\Omega}$ constante







`compteurG` $>$ `compteurD`

`compteurG` $<$ `compteurD`

$$\Omega_G = \bar{\Omega} + K\,(\texttt{compteurD} - \texttt{compteurG})$$
$$\Omega_D = \bar{\Omega} - K\,(\texttt{compteurD} - \texttt{compteurG})$$

Mise en place de l'asservissement

```
void loop(){
    if (t < T){// boucle de comptage
        cptG = analogRead(0)>900; // bit du capteur gauche
        cptD = analogRead(1)>900; // bit du capteur droit
        if (cptG > cptG_old){    // front montant
            compteurG++;         // incrémentation du compteur
        }
        if (cptD > cptD_old){    // front montant
            compteurD++;         // incrémentation du compteur
        }
        cptG_old = cptG; // mise àjour des prédécesseurs
        cptD_old = cptD;
        delay(100);
        t+=0.1;
    }
    else{
        ...
    }
}
```

Mise en place de l'asservissement

```
void loop(){
    if (t < T){...}// boucle de comptage
    else{// changement de commande des moteurs
        diff = compteurG - compteurD;
        if (diff>0){
            commandeG -= K*diff;
            commandeD += K*diff;
        }
        else{
            commandeG += K*diff;
            commandeD -= K*diff;
        }
        compteurG=0;                    // initialisation des compteurs
        compteurD=0;
        t=0;
    }
}
```

Paramètres de l'asservissement à régler

- fréquence de mesure associée à l'incrémentation de `t`
  (par pas de $0,1$ dans le code)
- durée de comptage `T` :
  - si trop grande le robot risque de trop dévier de sa trajectoire ;
  - si trop faible le robot risque d'avoir un comportement trop saccadé
    (changements de vitesses trop rapides) ;
- gain proportionnel `K`.

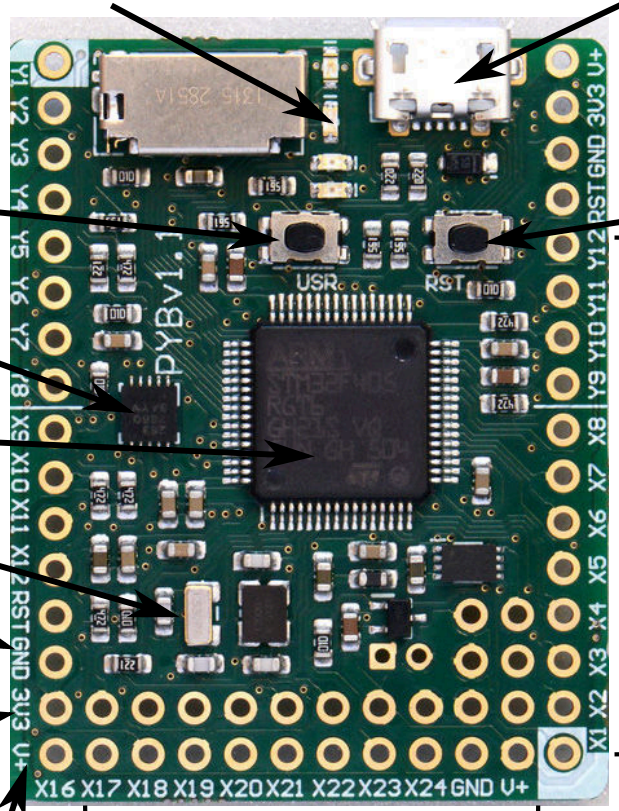
1 2

Axe seul & vérin électrique

Axe seul & vérin électrique

Axe seul & vérin électrique

Carte pyboard



Leds test

Port micro USB : utile pour la communication et l'alimentation

Bouton user USR

Bouton Hard Reset

Accéléromètre

Microcontrôleur

Quartz

GND : Masse

3V3 : sortie Alimentation 3,3 V pour alimenter des périphériques (technologie CMOS)

V+ : Alimentation externe si non connectée USB en permanence

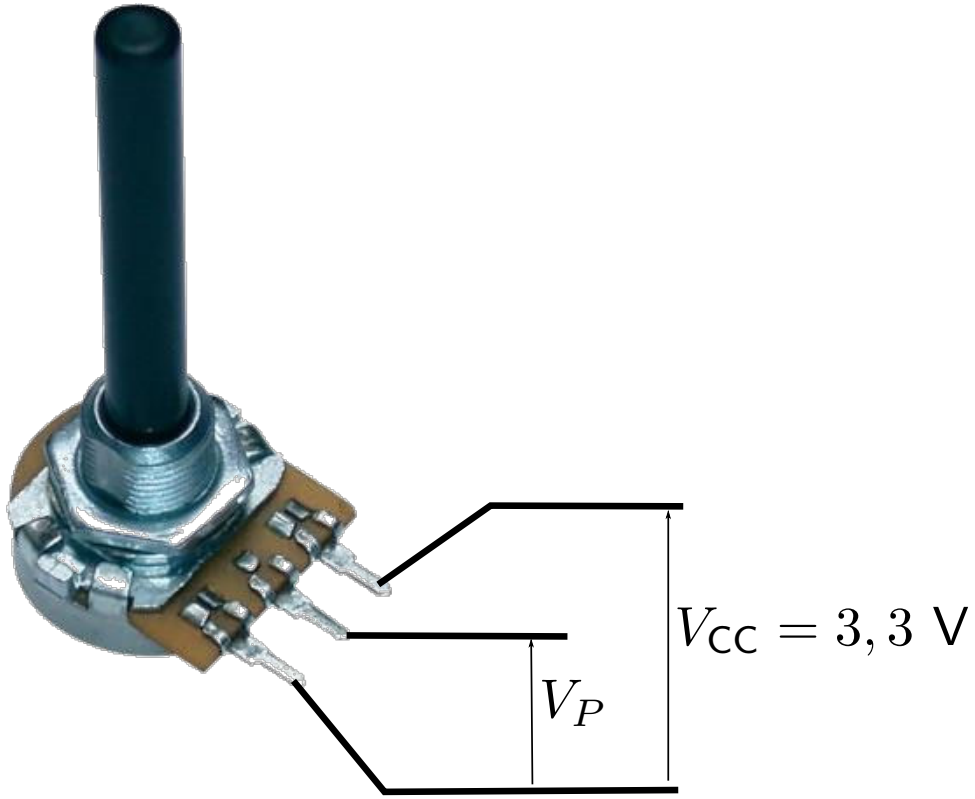
Entrées/sorties binaires

Ou Entrées analogiques si branchées au convertisseur analogique numérique (12 bits) ADC

Zone non exploitée ici

Axe seul & vérin électrique

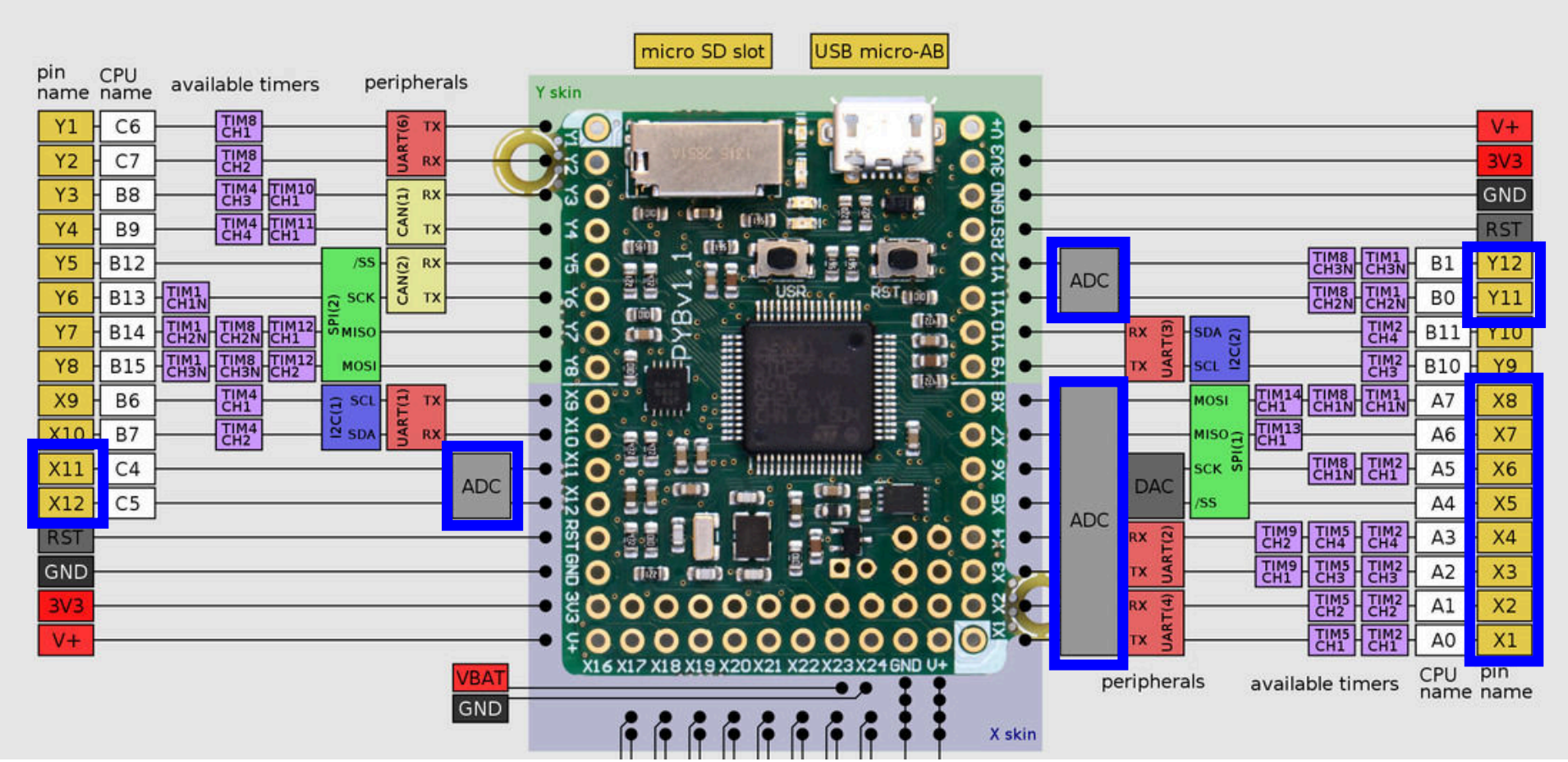
- Acquisition de la consigne à partir du potentiomètre



Principe du pont diviseur de tension $\Longrightarrow$ $V_p \in [0; 3,3 \text{ V}]$

Axe seul & vérin électrique

Acquisition de la consigne à partir du potentiomètre



Acquisition du signal puis conversion numérique avec un CAN (ADC en anglais)

Axe seul & vérin électrique

Acquisition de la consigne à partir du potentiomètre

Déclaration du potentiomètre :

```
import pyb
pot = pyb.ADC('X7')
```

Lecture de la valeur :

```
pot.read()
```

`pot.read()` $\in [\![0, 4095]\!] \implies$ quantification sur 12 bits

Axe seul & vérin électrique

Acquisition de la consigne à partir du potentiomètre

Choix du 0 « au milieu » :

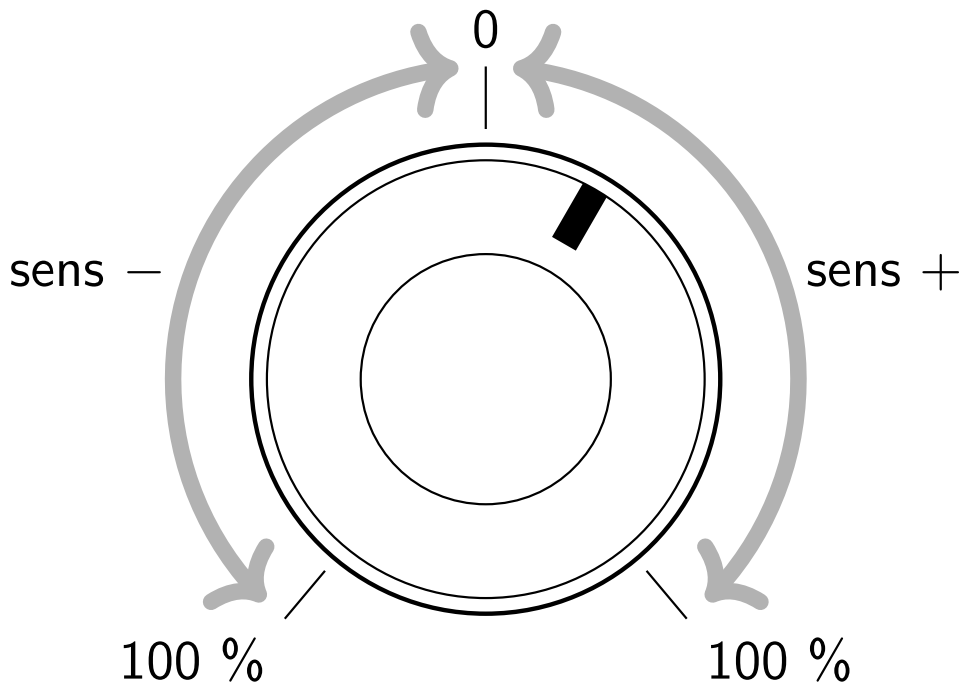
$$2^{11} = 2048 \rightarrow 0\ \%$$

tel que

$$0 \rightarrow -100\ \%$$

et

$$2^{12} - 1 = 4095 \rightarrow \sim 100\ \%$$

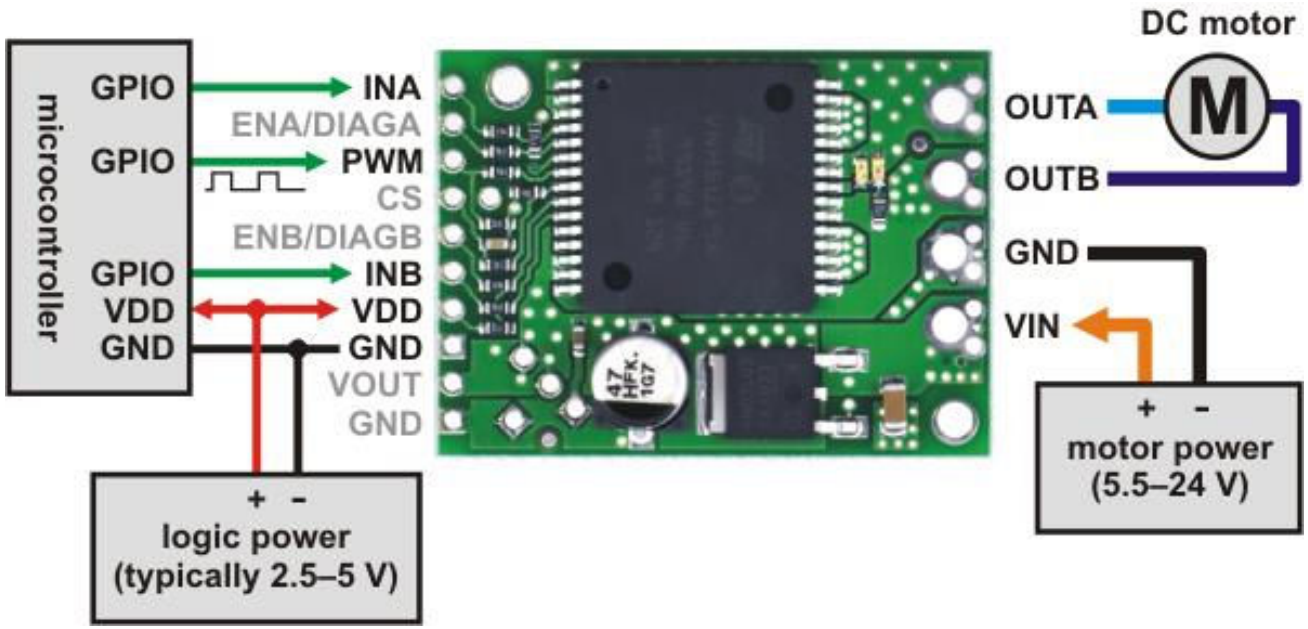


d'où la consigne :

```
consigne = 100*(pot.read()-2**11)/2**11
```

Axe seul & vérin électrique

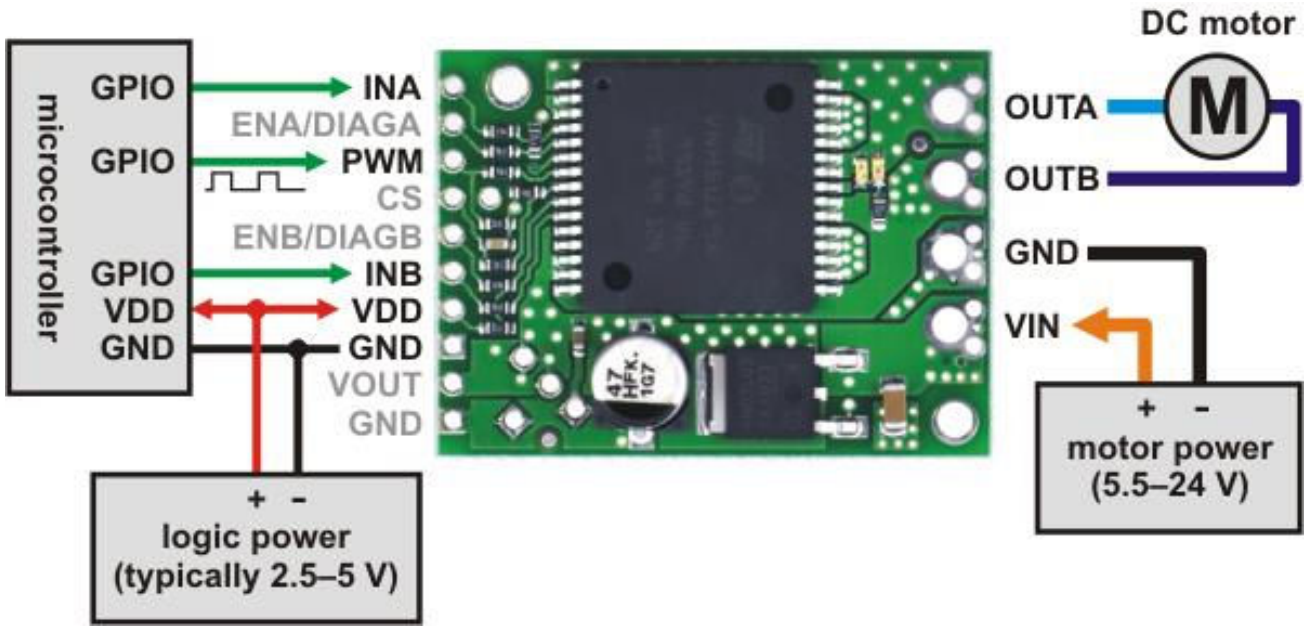
Commande du moteur



- circuit VNH5019 : hacheur quatre quadrants
- logique (de commande) à gauche ;
- puissance à droite.

Axe seul & vérin électrique

Commande du moteur

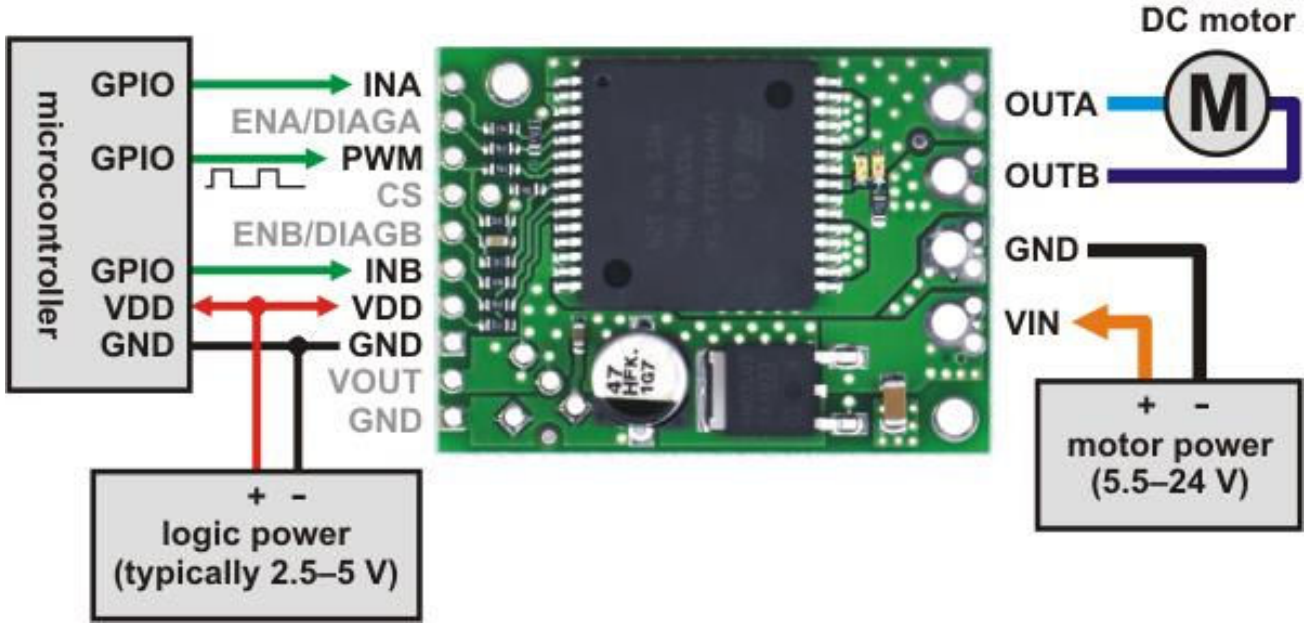


Coté puissance :

- bornes de puissance à visser ;
- alimentation polarisée !

Axe seul & vérin électrique

Commande du moteur



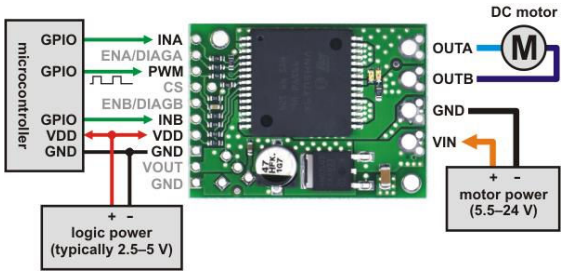
Coté logique :

- `VDD` et `VND` pour le niveau de logique (3,3 V) ;
- `INA` et `INB` pour le sens de rotation ;
- `PWM` : signal logique permettant de moduler la tension entre `OUTA` et `OUTB`.

- U_E : tension d'alim
- u_S : tension entre `OUTA` et `OUTB`

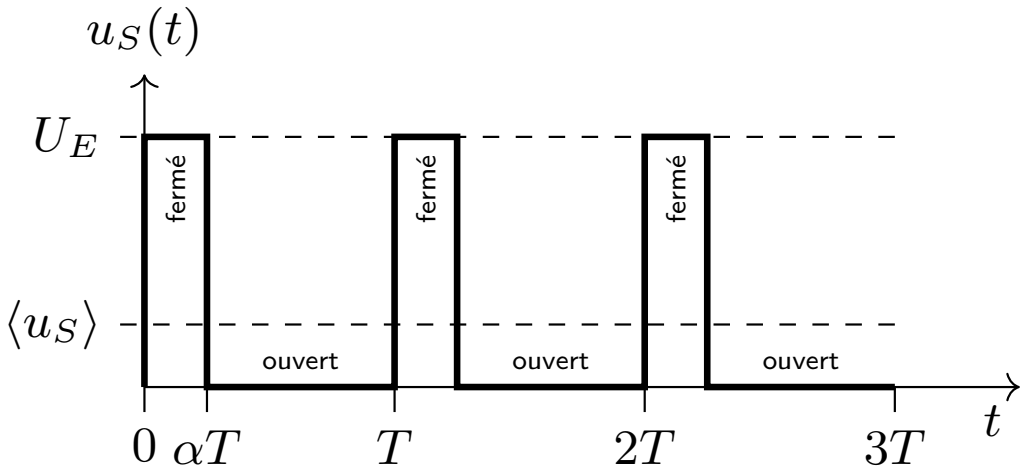
Axe seul & vérin électrique

Commande du moteur



Coté logique :

- `VDD` et `VND` pour le niveau de logique (3,3 V) ;
- `INA` et `INB` pour le sens de rotation ;
- `PWM` : signal logique permettant de moduler la tension entre `OUTA` et `OUTB`.



- U_E : tension d'alim
- u_S : tension entre `OUTA` et `OUTB`
- T période de hachage
- $\alpha \in [0;1]$: rapport cyclique

Axe seul & vérin électrique

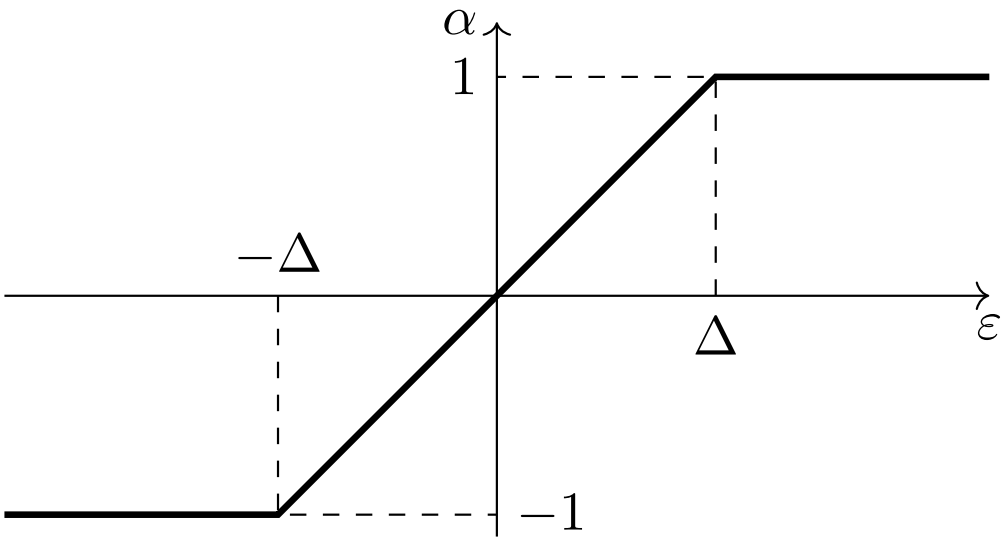
Commande du moteur

```
from pyb import Pin, Timer
pot = pyb.ADC('X7')   # potentiomètre de consigne
m = Pin('X1') # pin X1, PWM
tim1 = Timer(2, freq=1000)# fréquence d'horloge à1 kHz
ch1 = tim1.channel(1, Timer.PWM, pin=m)# définition du PWM
sens_a = Pin('X2', Pin.OUT_PP) # pin X2, sens +
sens_b = Pin('X3', Pin.OUT_PP) # pin X3, sens -
while True:# boucle infinie (...)
    alpha = (pot.read()-2**11)/2**11 # rapport cyclique
    seuil=.2 # si |alpha|<0,2 pas de mouvement mais sifflement
    if alpha<-seuil:
        sens_a.low()
        sens_b.high()
        ch1.pulse_width_percent(abs(alpha*100))
    elif alpha>seuil:
        sens_b.low()
        sens_a.high()
        ch1.pulse_width_percent(abs(alpha*100))
    else:
        ch1.pulse_width_percent(0)
```

Axe seul & vérin électrique

Asservissement de l'axe

- capteur : potentiomètre angulaire sur système pignon-crémaillère,
- acquisition avec ADC : 0 tige rentrée, 4095 tige sortie ;
- idée de l'asservissement :



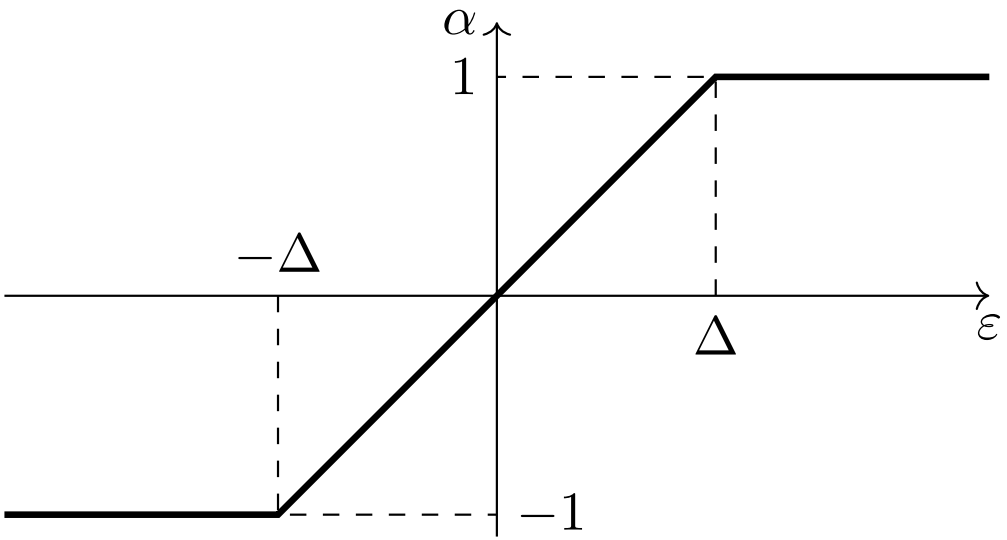
- ε : erreur de position (bit) ;
- α : rapport cyclique signé pour le sens ;
- Δ : distance max d'action proportionnelle (bit) ;

Commande proportionnelle dans un voisinage à $\pm\Delta$, saturée en vitesse sinon.

Axe seul & vérin électrique

Asservissement de l'axe

- capteur : potentiomètre angulaire sur système pignon-crémaillère,
- acquisition avec ADC : 0 tige rentrée, 4095 tige sortie ;
- idée de l'asservissement :



- ε : erreur de position (bit) ;
- α : rapport cyclique signé pour le sens ;
- Δ : distance max d'action proportionnelle (bit) ;

$$\alpha = \begin{cases} \dfrac{\varepsilon}{\Delta} & \text{si } |\varepsilon| \leqslant \Delta \\ \dfrac{\varepsilon}{|\varepsilon|} & \text{sinon.} \end{cases}$$

Axe seul & vérin électrique

Asservissement de l'axe

```
from pyb import Pin, Timer
pot = pyb.ADC('X7')   # potentiomètre de consigne
pos = pyb.ADC('X4')   # potentiomètre de position (capteur)
... # déf PWM et sens
while True:# boucle infinie (...)
    delta = 2**3 # 1/8 de la course
    erreur = pot.read() - pos.read()
    # calcul du rapport cyclique
    if abs(erreur)<=delta:# cas proportionnel
        alpha = erreur/delta
    else:# cas saturé
        alpha = erreur/abs(erreur)
    # commande du moteur
    if alpha<0:
        sens_a.low(); sens_b.high()
    elif alpha>0:
        sens_b.low(); sens_a.high()
    else:
        sens_a.low(); sens_b.low()
    ch1.pulse_width_percent(abs(alpha*100))
```

N. Mesnier, lycée Jean Perrin, Lyon