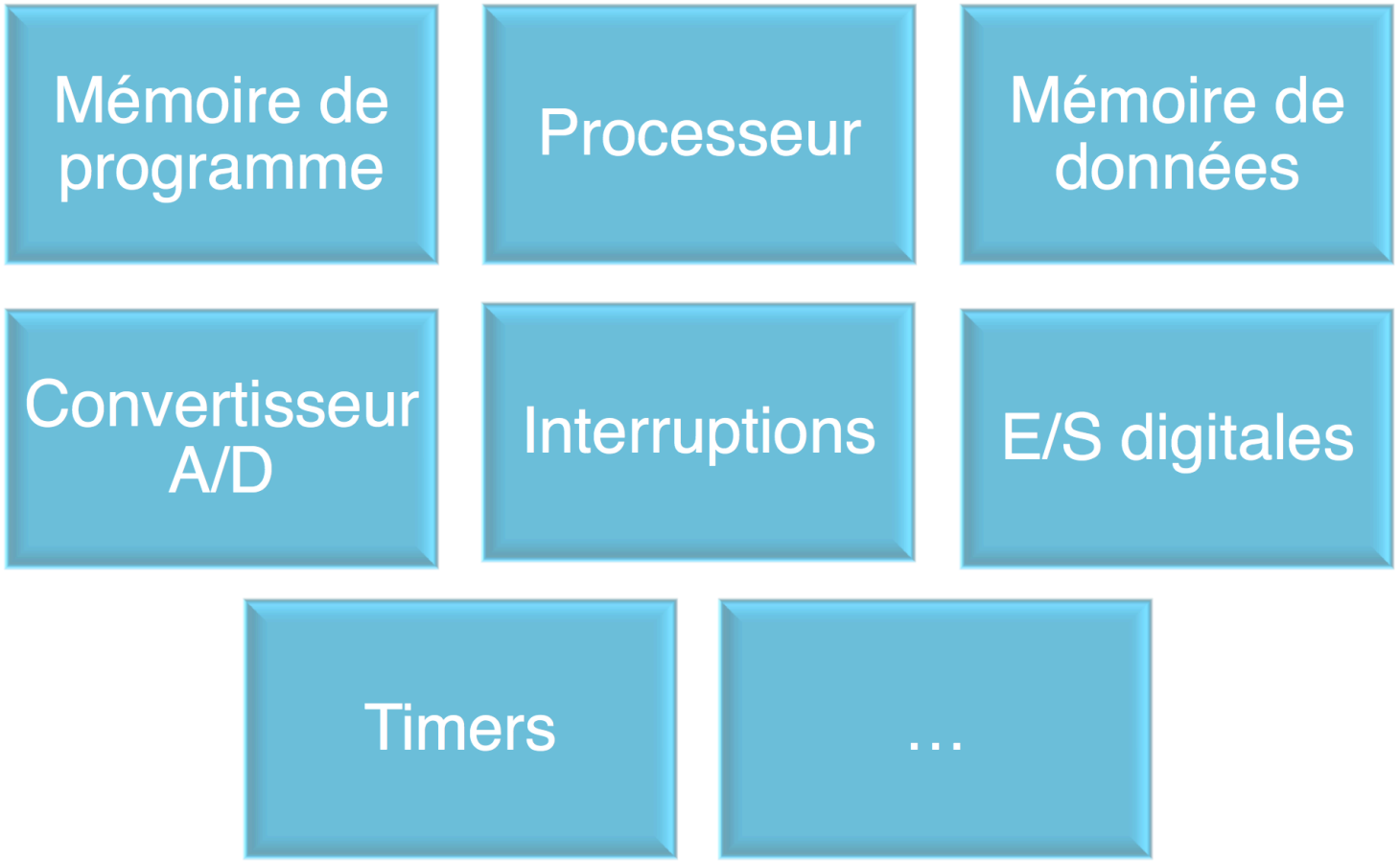
LLSMF2018 – ELEC
Cours 4

Complément d'information préalable à l'utilisation du kit MRK

2017-2018

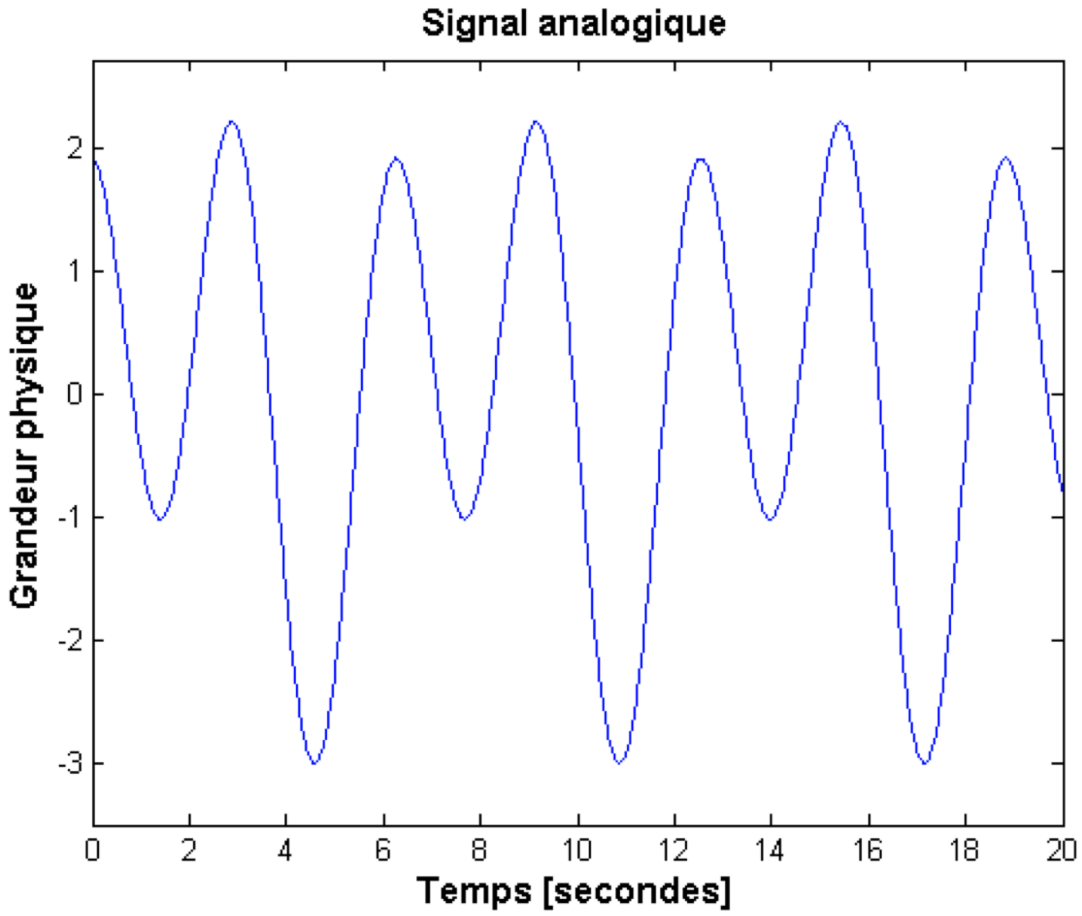
Les périphériques du processeur



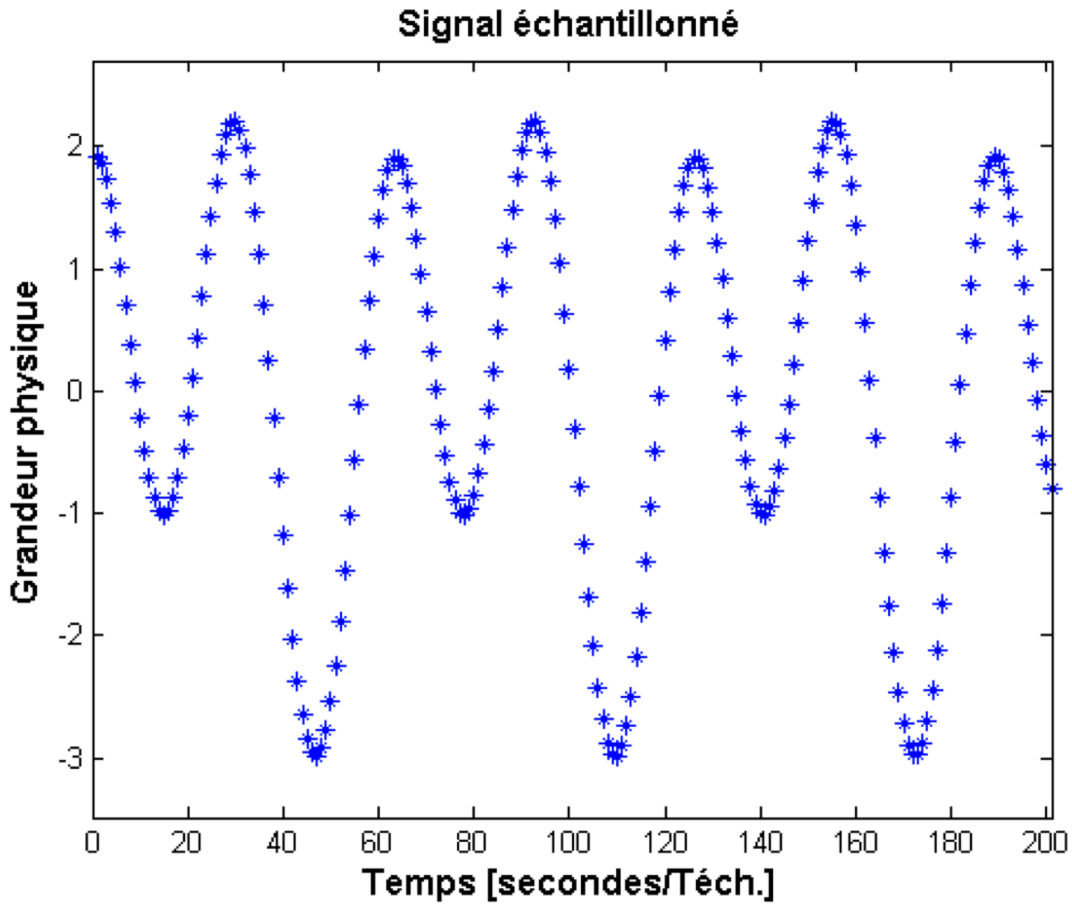
Entrées/Sorties digitale

- Sorties digitales (librairie mx4.h)
  - `initIO(void)`
  - `setLeds(char leds)`
- Entrées digitales (librairie mx4.h)
  - initIO
  - char getButtonX() (X $\in$ [0,1])

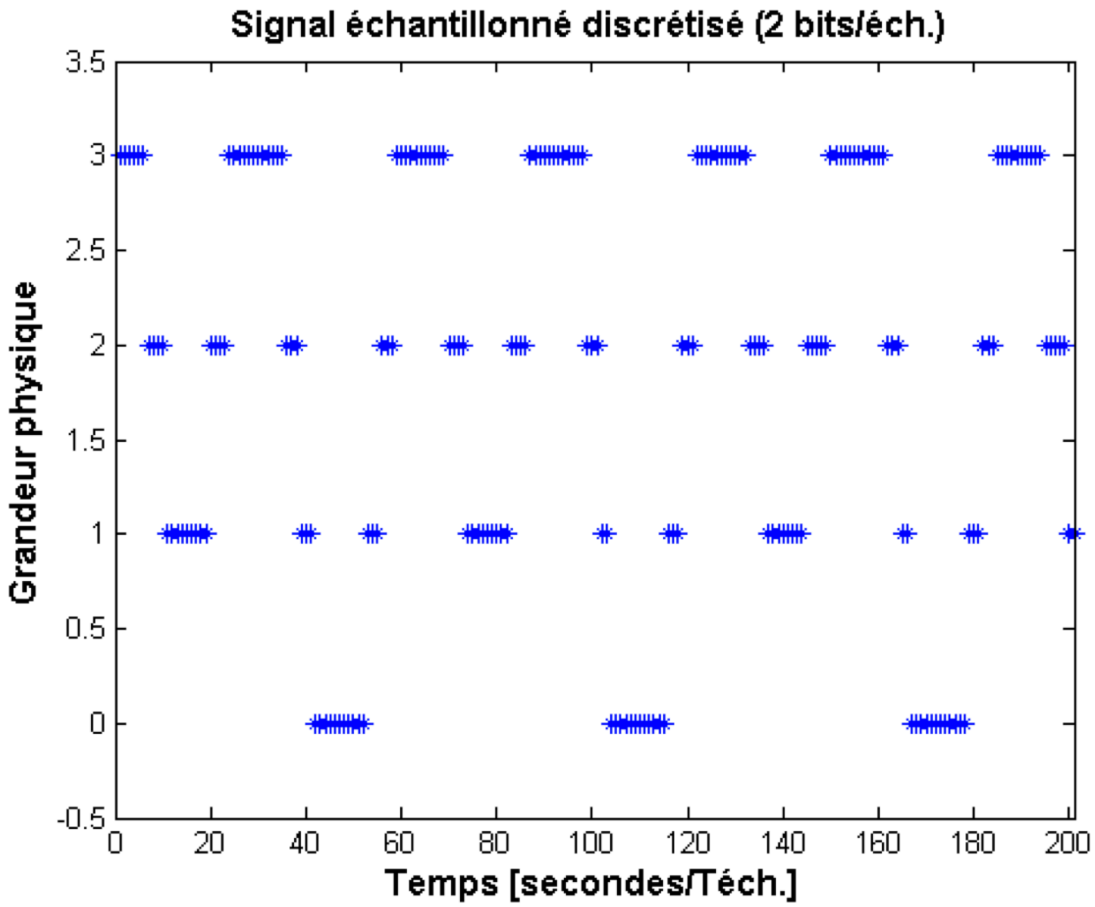
Conversion analogique - digital



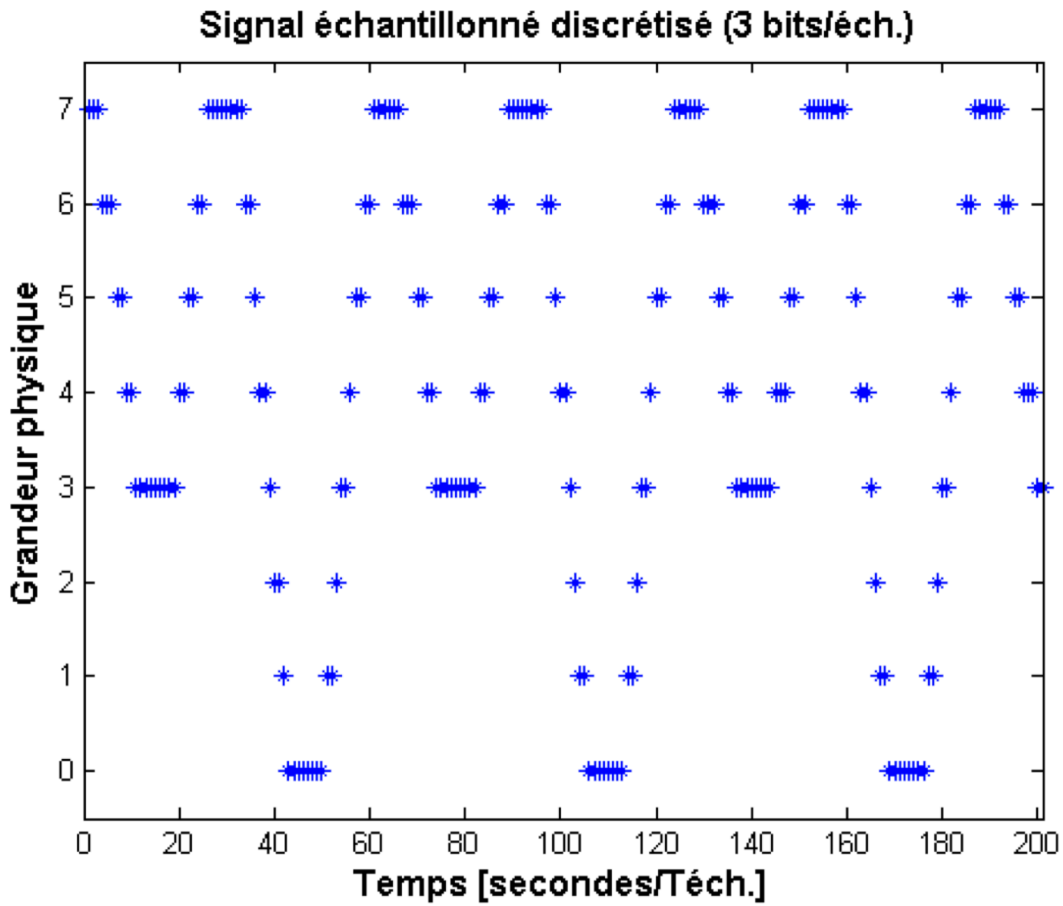
Conversion analogique - digital



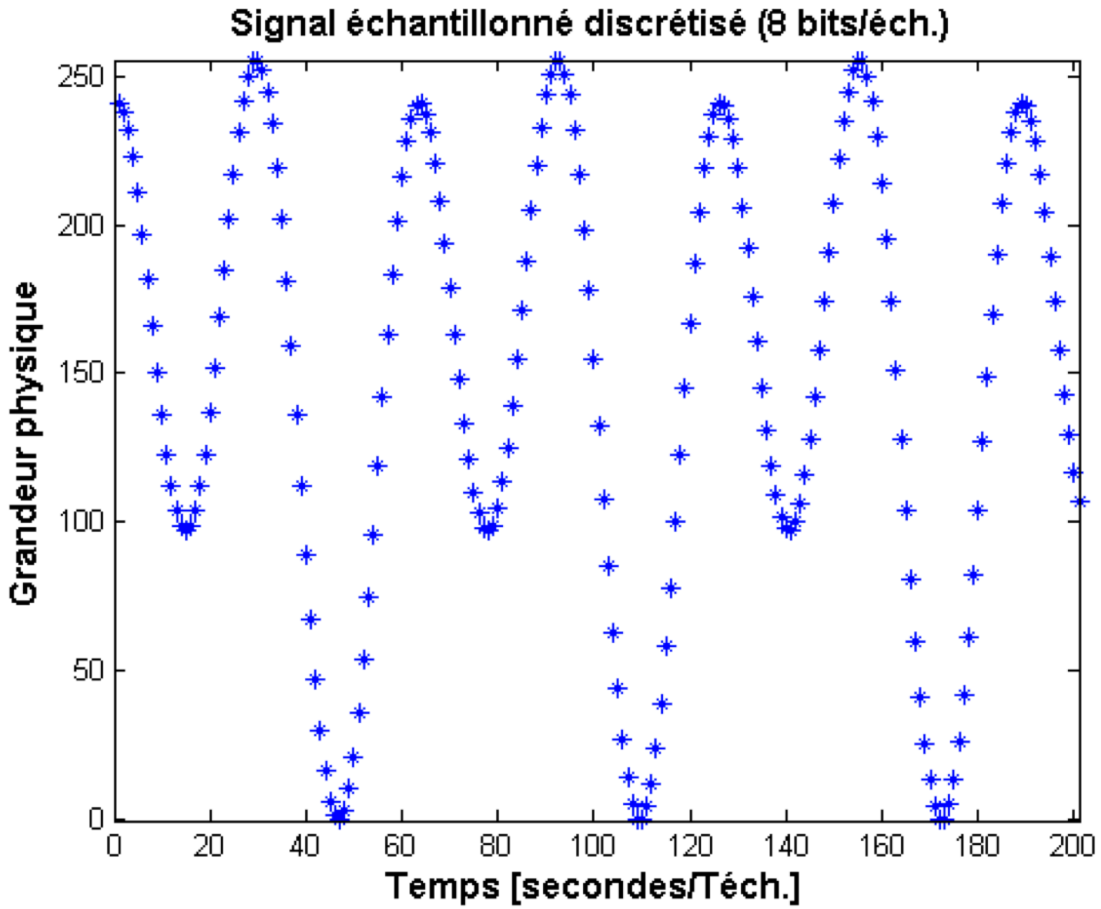
Conversion analogique - digital



Conversion analogique - digital



Conversion analogique - digital



Entrées analogiques

- Entrées analogiques (analogInputs.h)
  - `initAnalogInputs(short analogInputs);`
  - `short readADC(char num);`

- analogInputs : les bits à 1 correspondent aux entrées analogiques que l'on souhaite utiliser.

Entrée analogique	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Position sur la carte	B7	B10	K4	K3	K2	K1	J10	J9	nc	nc	J8	J7	J4	J3	J2	J1

Timers

- Fonction fournies par la librairie interrupts.h

```
void initTimer1(unsigned short period, char ps)
void initTimer2(unsigned short period, char ps)
void timer1InterruptEnable(void)
void timer2InterruptEnable(void)
```

- À compléter dans le fichier main.c

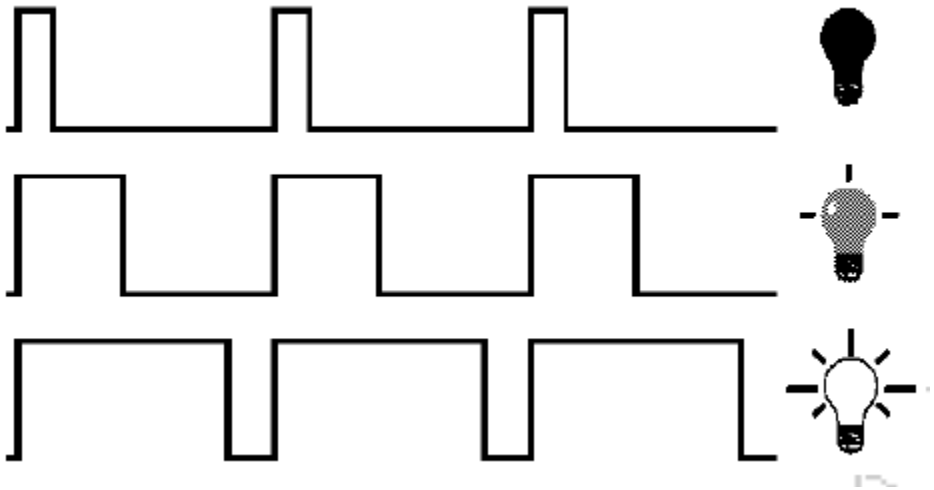
```
void timer1Interrupt(void)
void timer2Interrupt(void)
```

Commande des moteurs

- Nouvelles fonctions fournies par la librairie motor.h:

```
void initMotors();
void initPWM(void);
void setMotor1Direction(unsigned short direction);
void setMotor2Direction(unsigned short direction);
void setMotor1Enable(unsigned short enable);
void setMotor2Enable(unsigned short enable);
void setPWMDC(unsigned int DC1, unsigned int DC2);
void disablePWM(void);
```

Conversion digital - analogique (MLI)



Conversion digital - analogique (MLI)

- Interruption haute fréquence (p. ex. 2kHz)
- pwm est une variable qui donne le rapport cyclique à appliqué en %

```
void timer1Interrupt(void){
     intCount++;
     if(intCount == 100){
            inCount = 0;
            if(pwm != 0)
                  moteurXEnable(1); // on allume
     }
     if (intCount > pwm)
            moteurXEnable(0); // on éteind

}
```

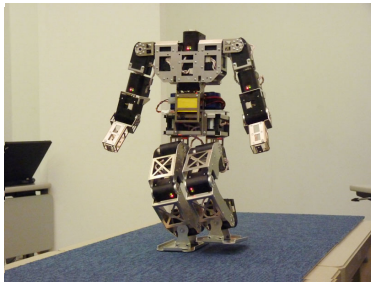
- Résolution : 1%
- Période : 1/(10kHz/100) = 10 ms = 1/100 s

Régulation : Définition

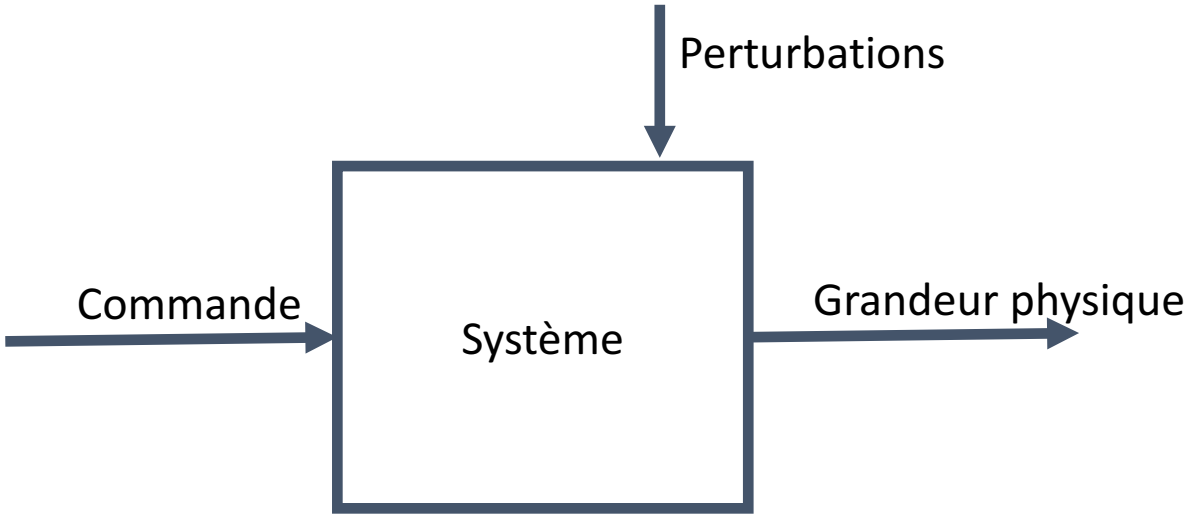
La régulation concerne les moyens mis en oeuvre pour maintenir chaque grandeur physique essentielle égale à une valeur désirée, appelée **consigne**, par action sur une grandeur réglante, appelée **commande**, et ce, malgré l'influence des grandeurs perturbatrices du système.

Régulation : Exemples

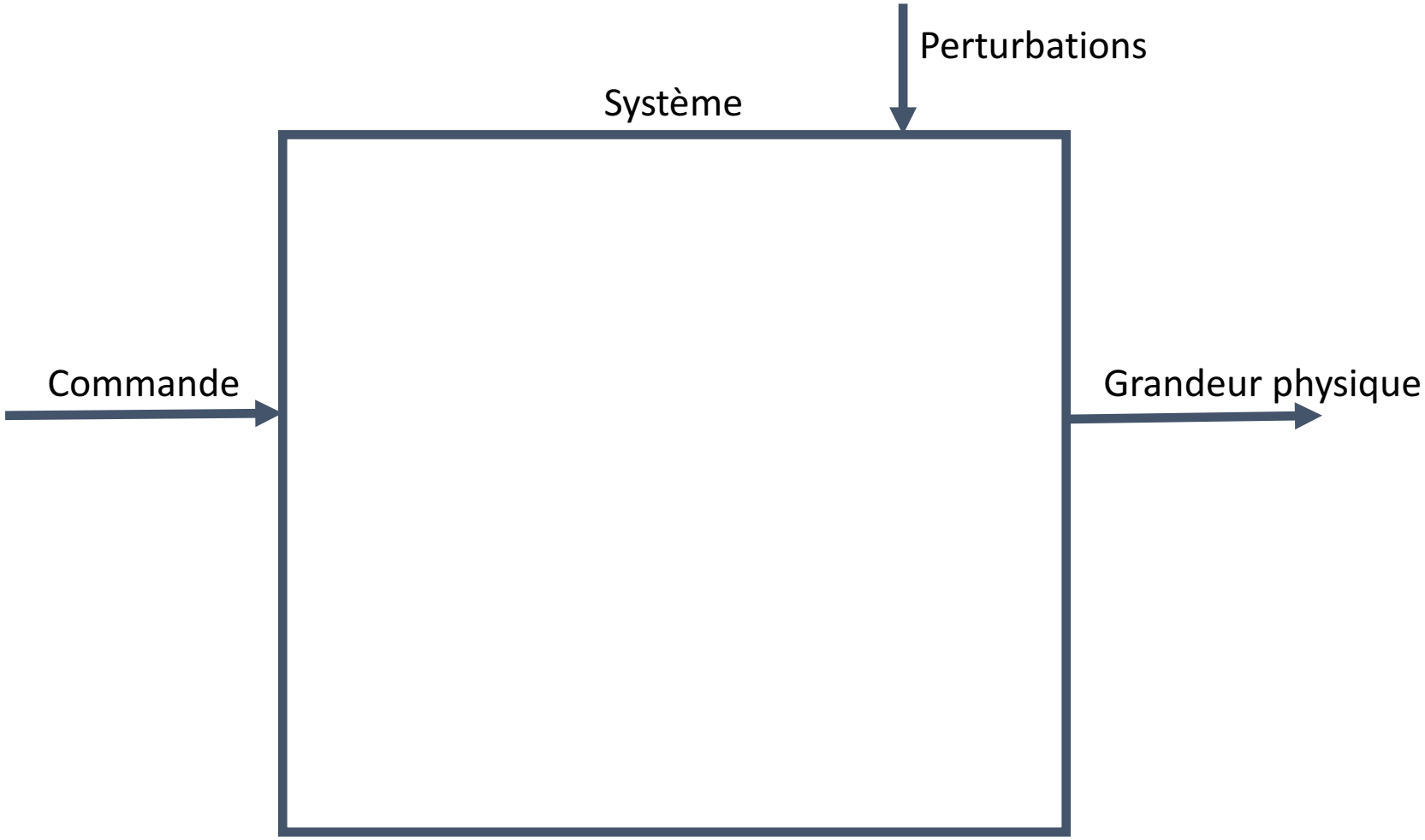
- Température d'un bâtiment
- Niveau d'oxygène, de CO2, d'humidité, …
- Vitesse d'un véhicule (cruise control)
- Puissance produite par une centrale électrique
- Position d'une machine
- Stabilité d'un hélicoptère
- Niveau d'eau dans un bassin



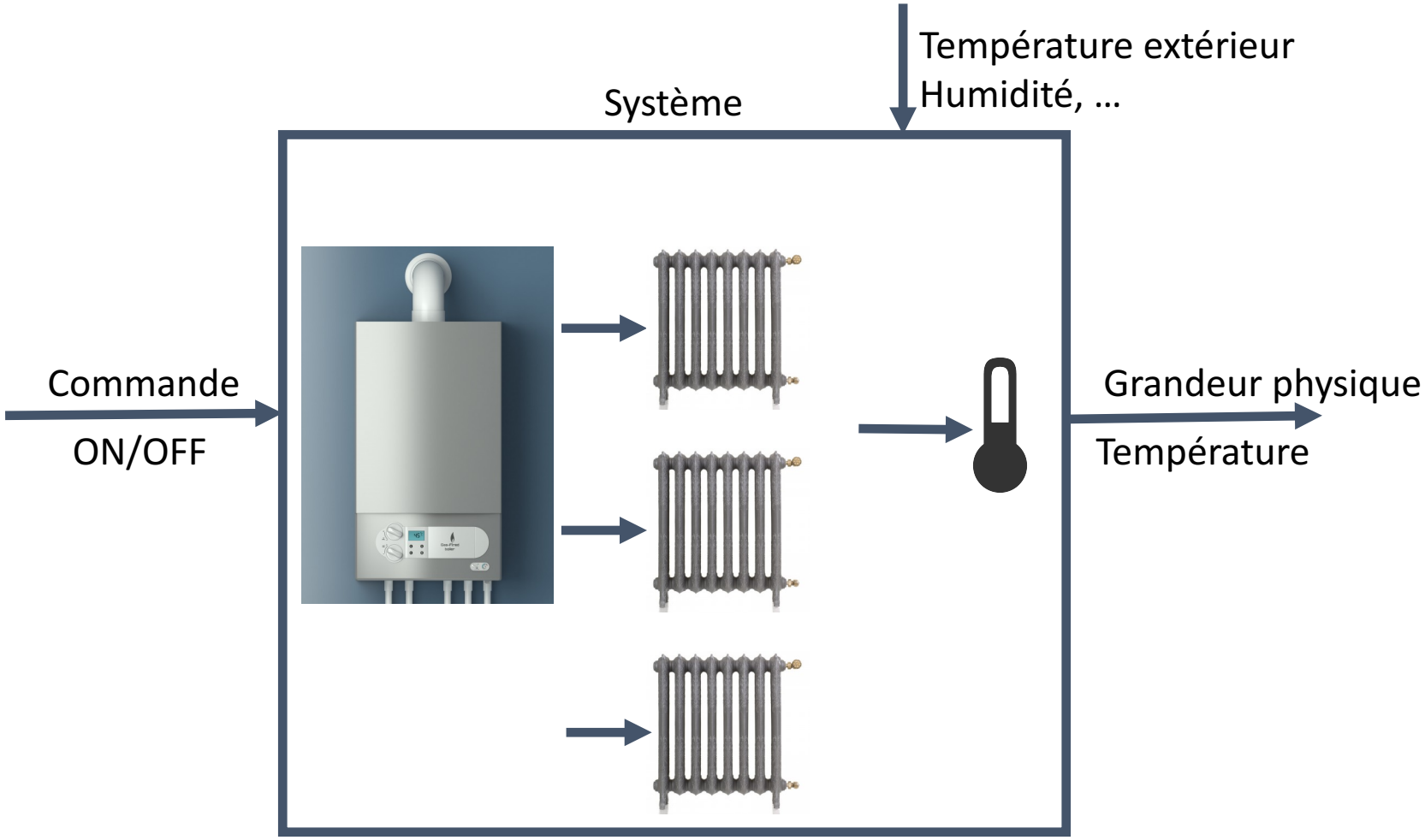
Régulation : système physique



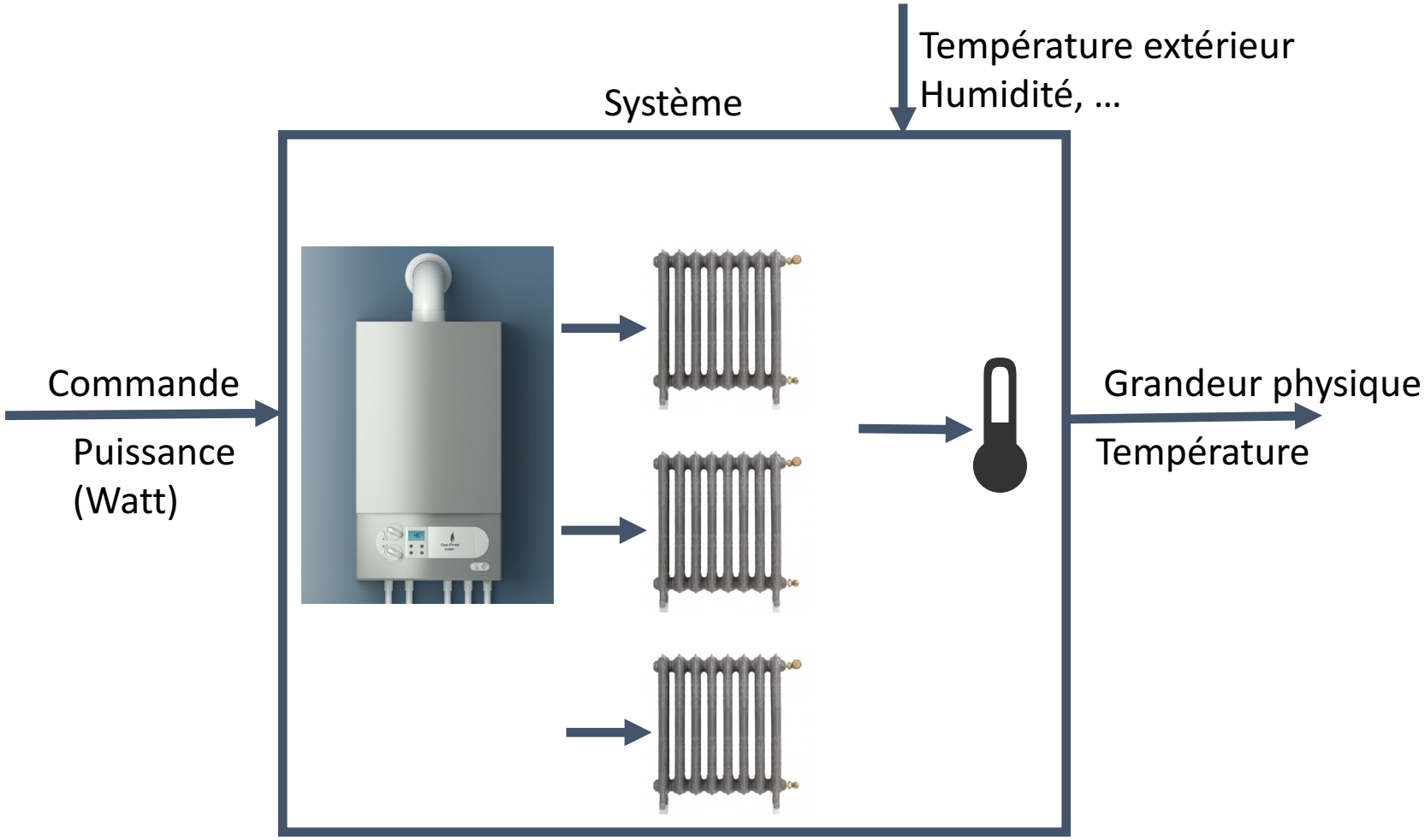
Régulation : système physique



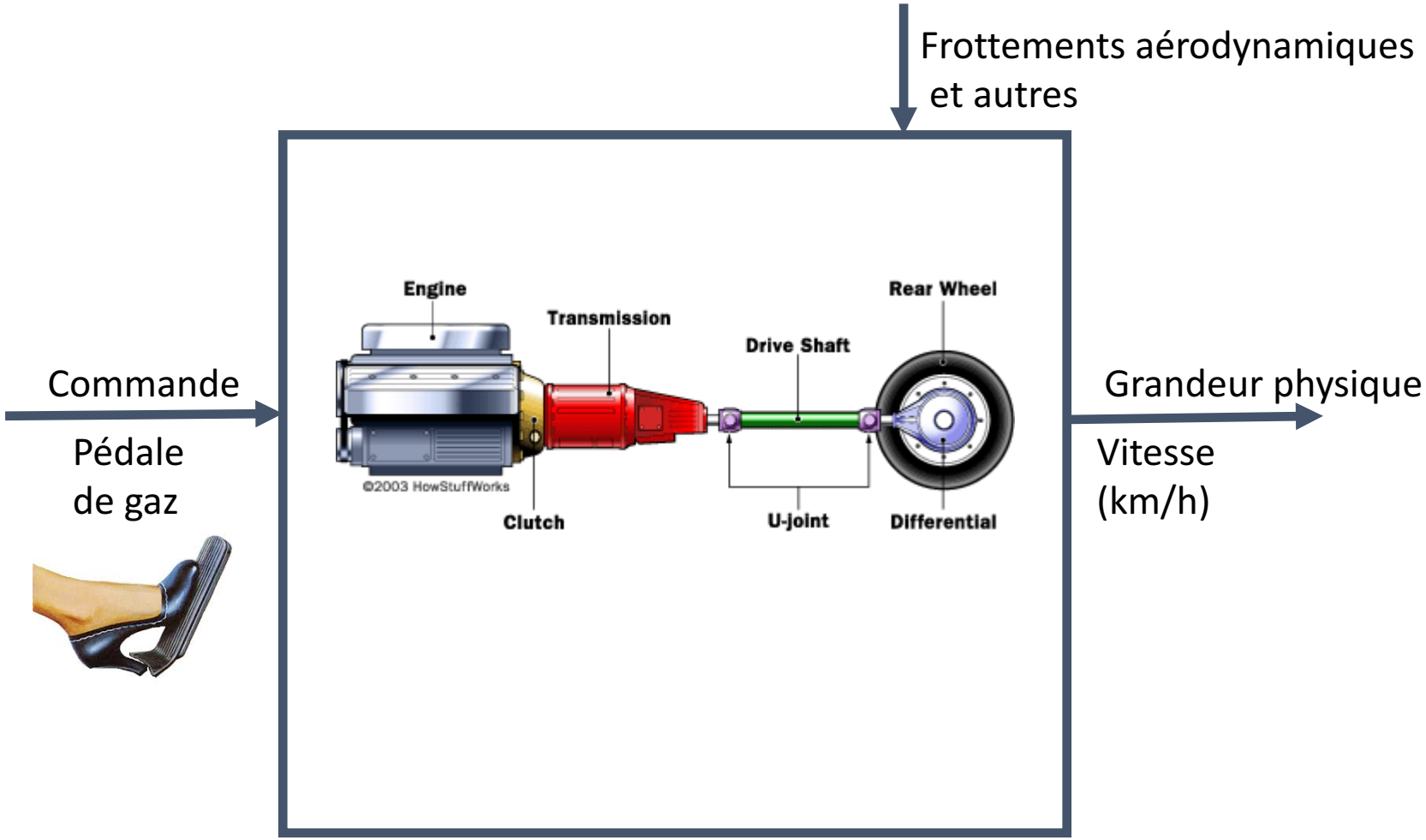
Régulation : système physique



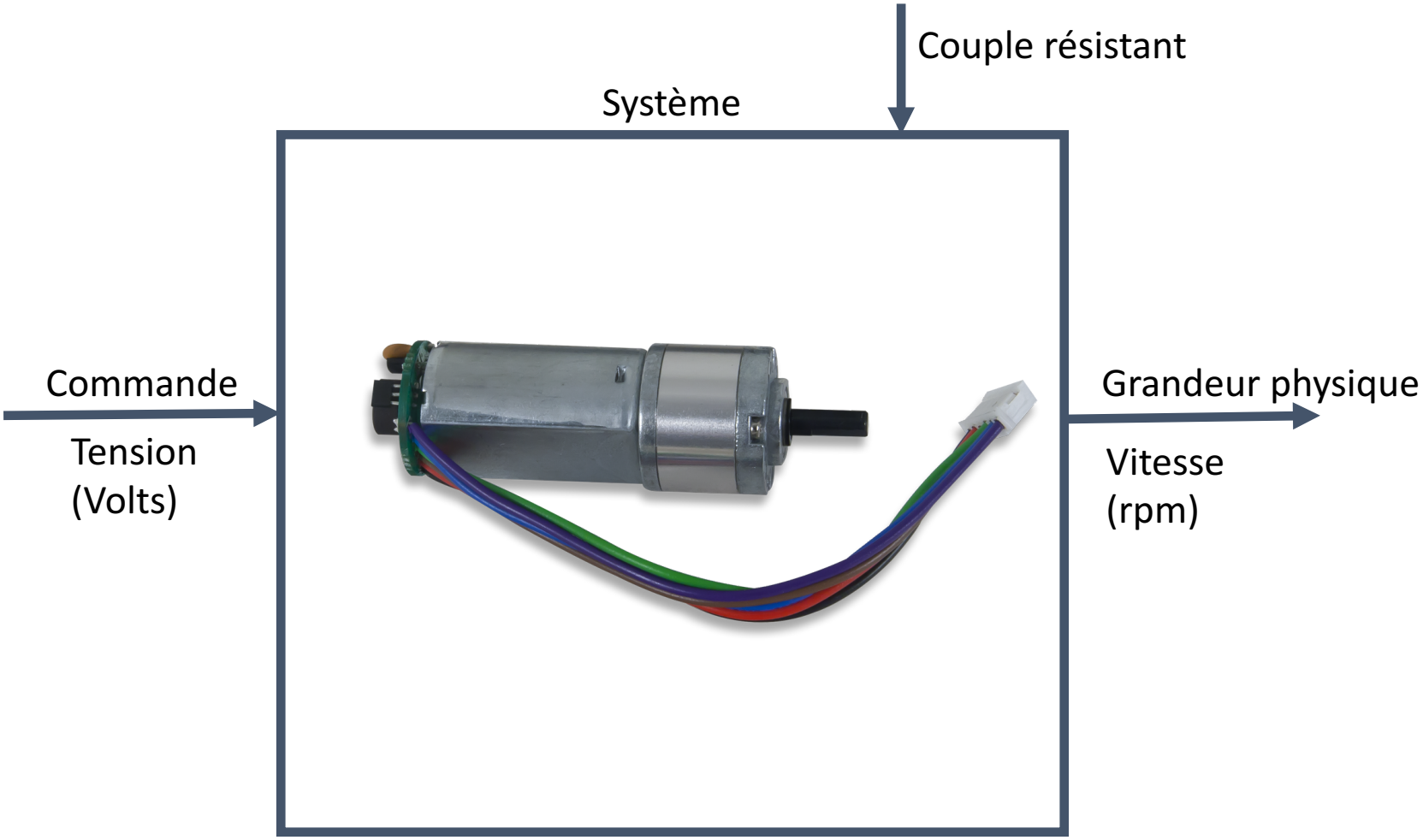
Régulation : système physique



Régulation : système physique



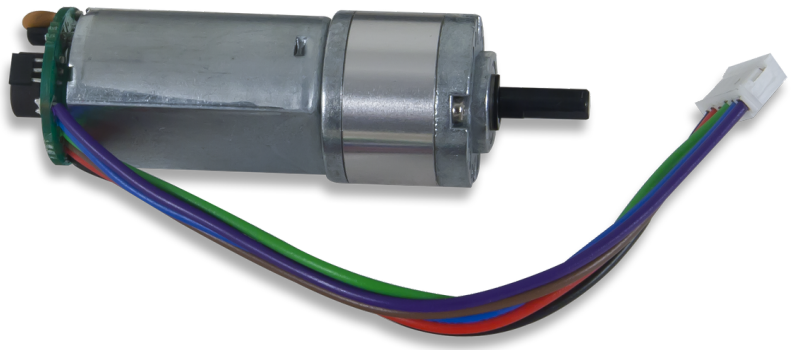
Régulation : système physique



Régulation : système physique

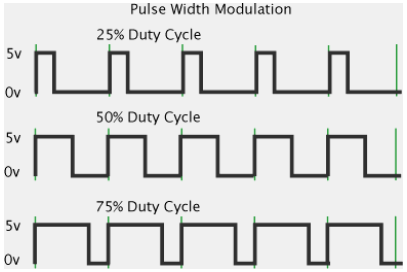
Couple résistant

Système



Commande

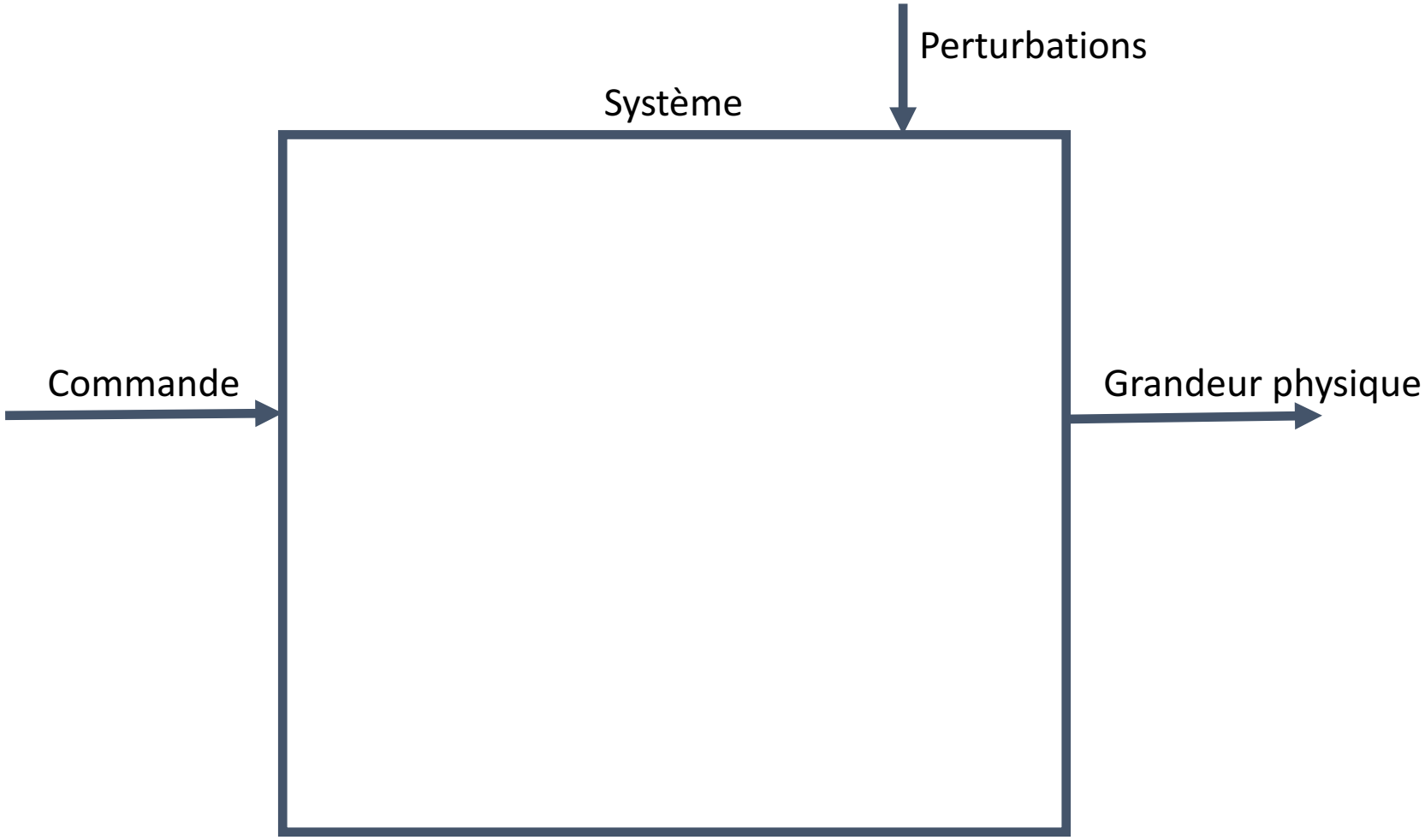
Rapport cyclique (%)



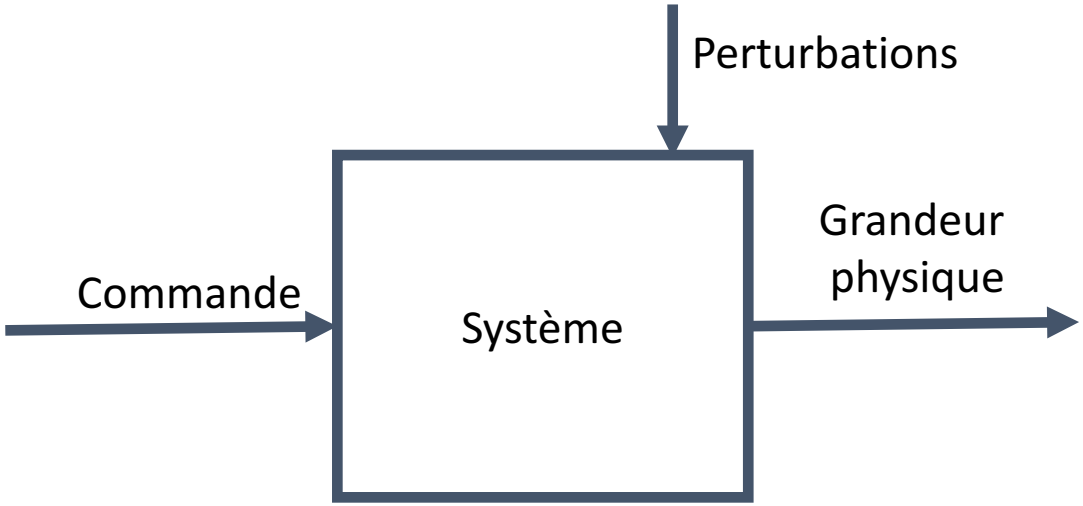
Grandeur physique

Vitesse (rpm)

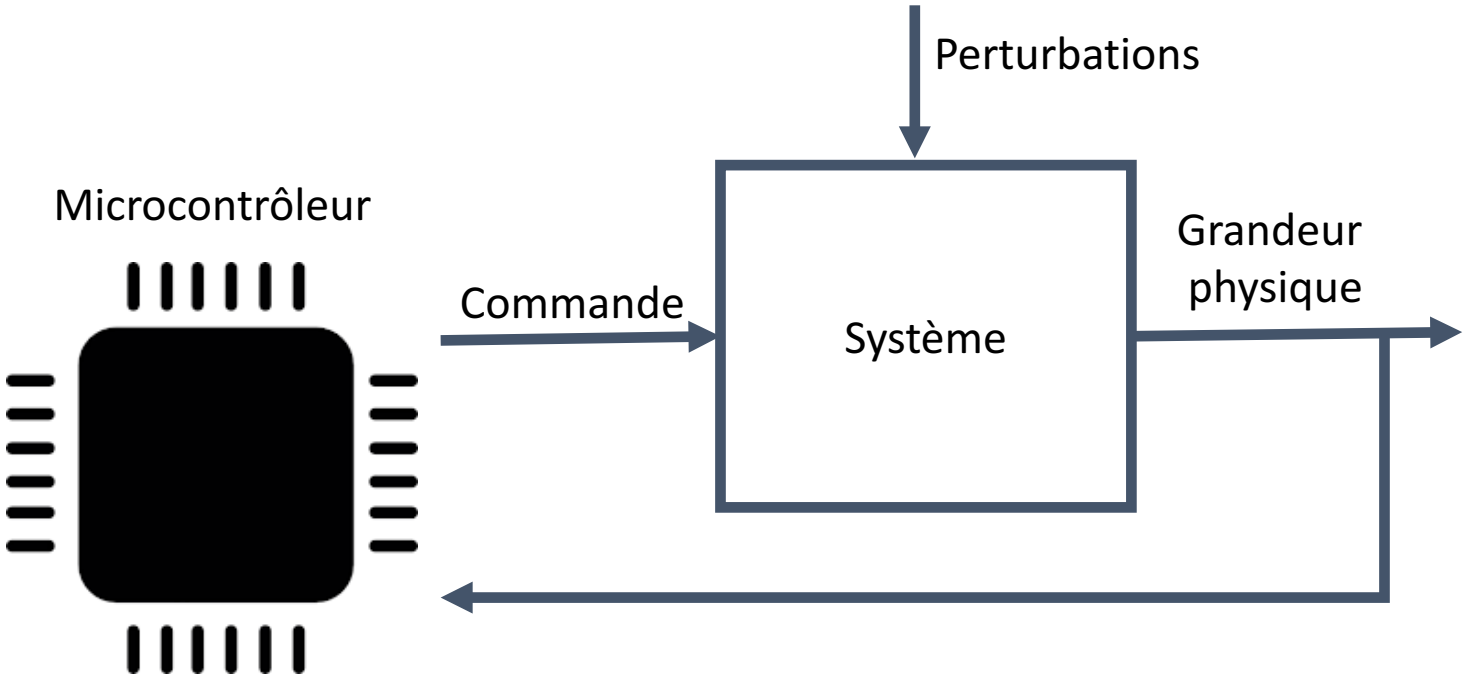
Régulation : système physique



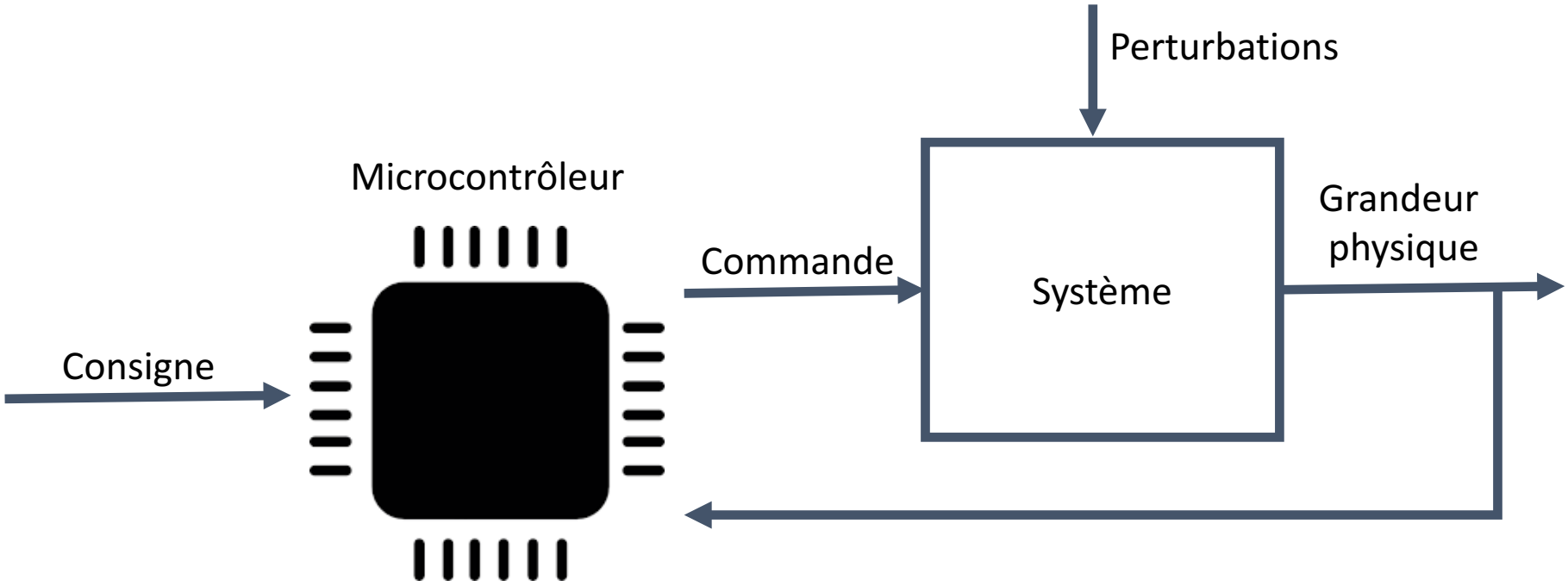
Régulation : système physique



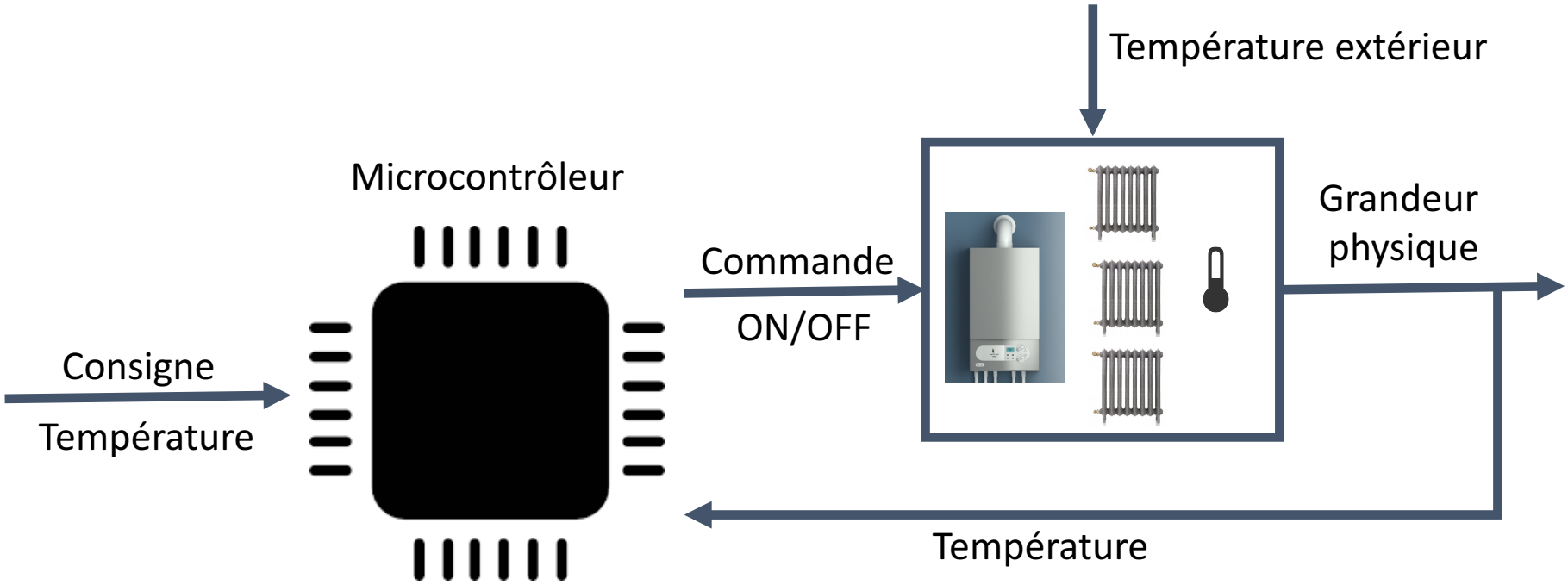
Régulation


Perturbations
Microcontrôleur
Commande
Système
Grandeur
physique

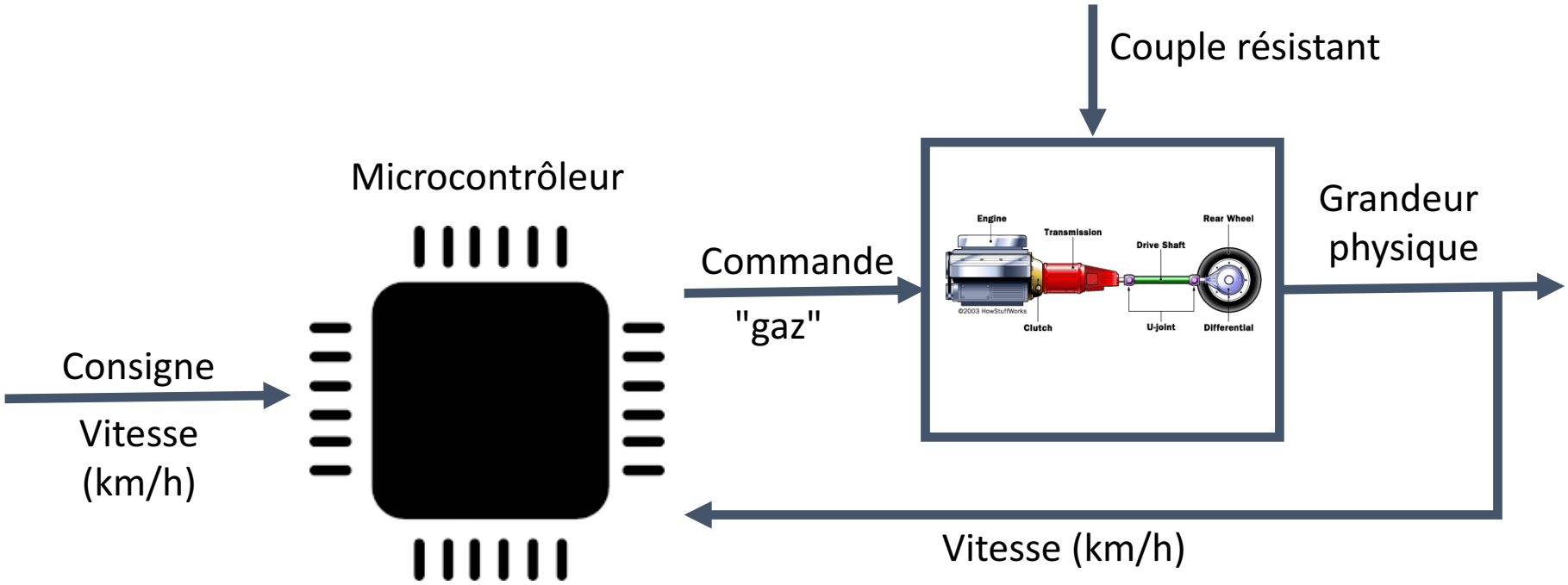
Régulation


Perturbations
Microcontrôleur
Commande
Système
Grandeur physique
Consigne

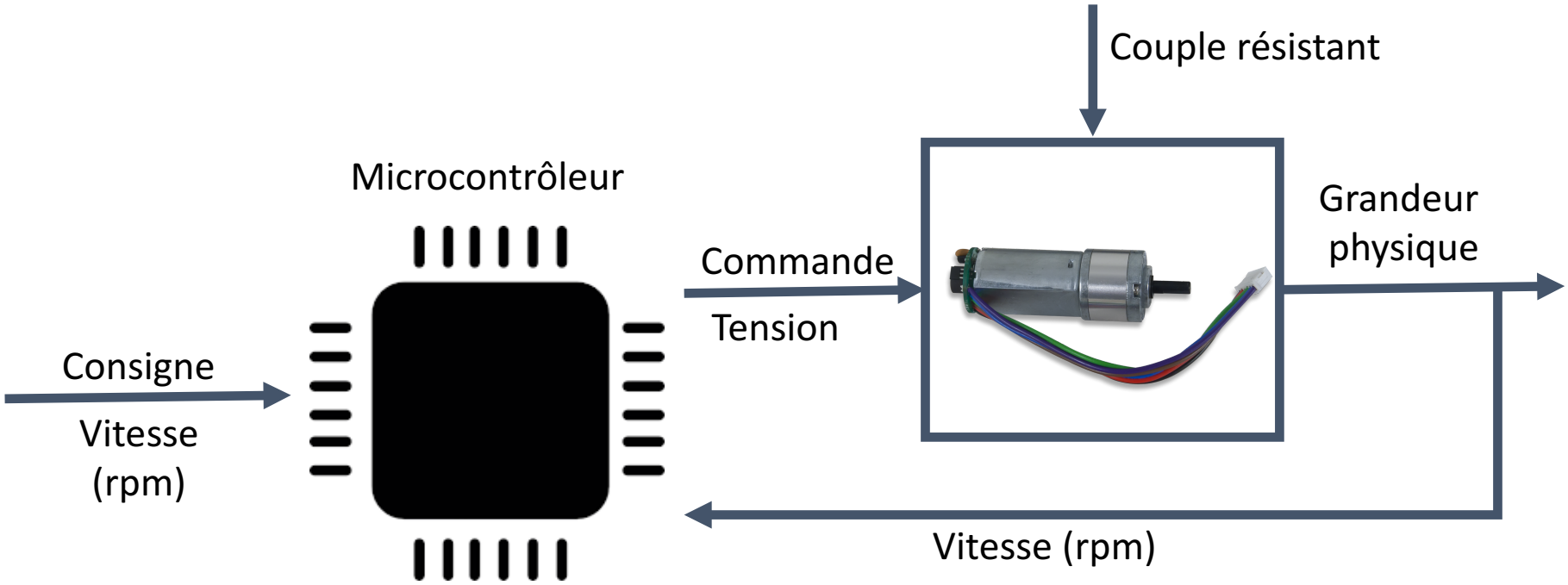
Régulation : Thermostat



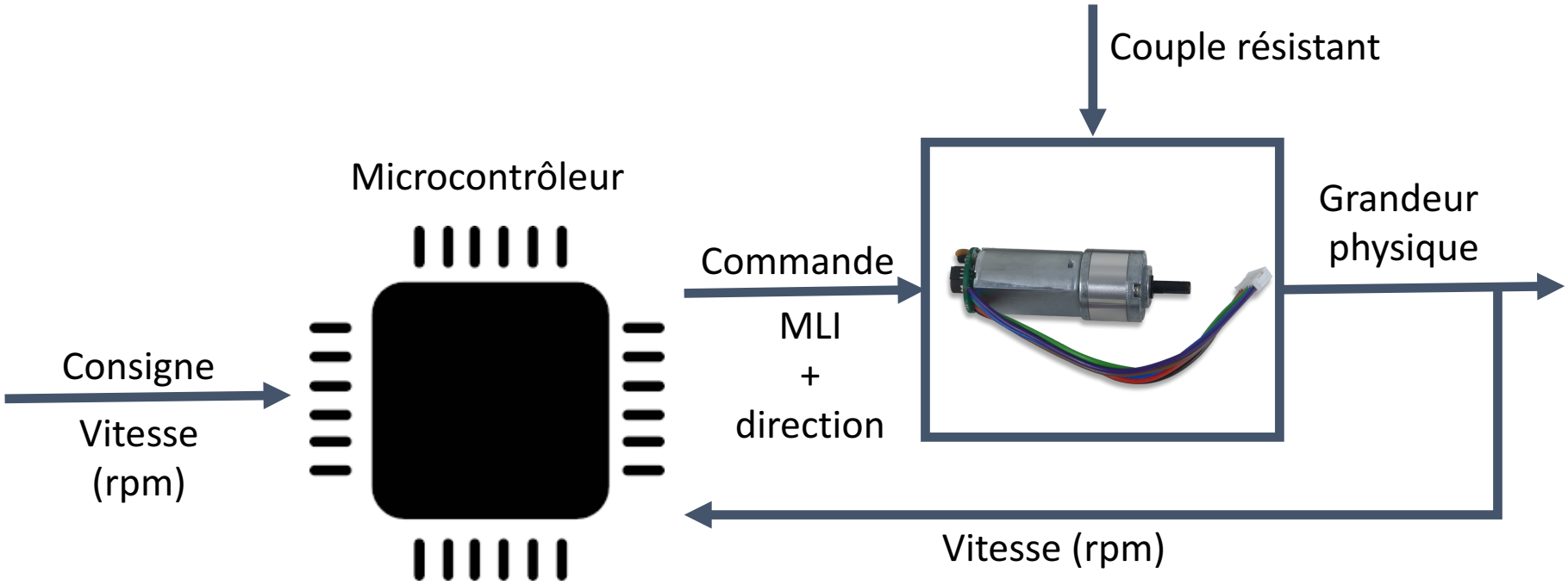
Régulation : Cruise control



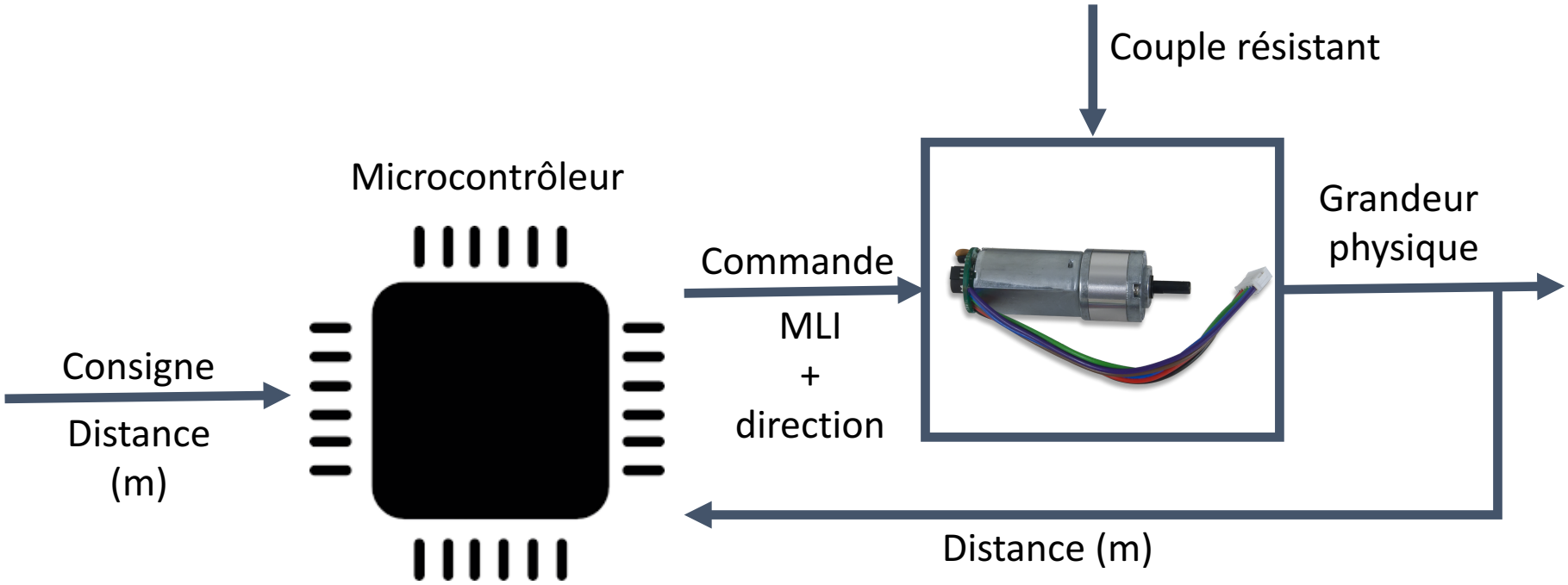
Régulation : Vitesse moteur



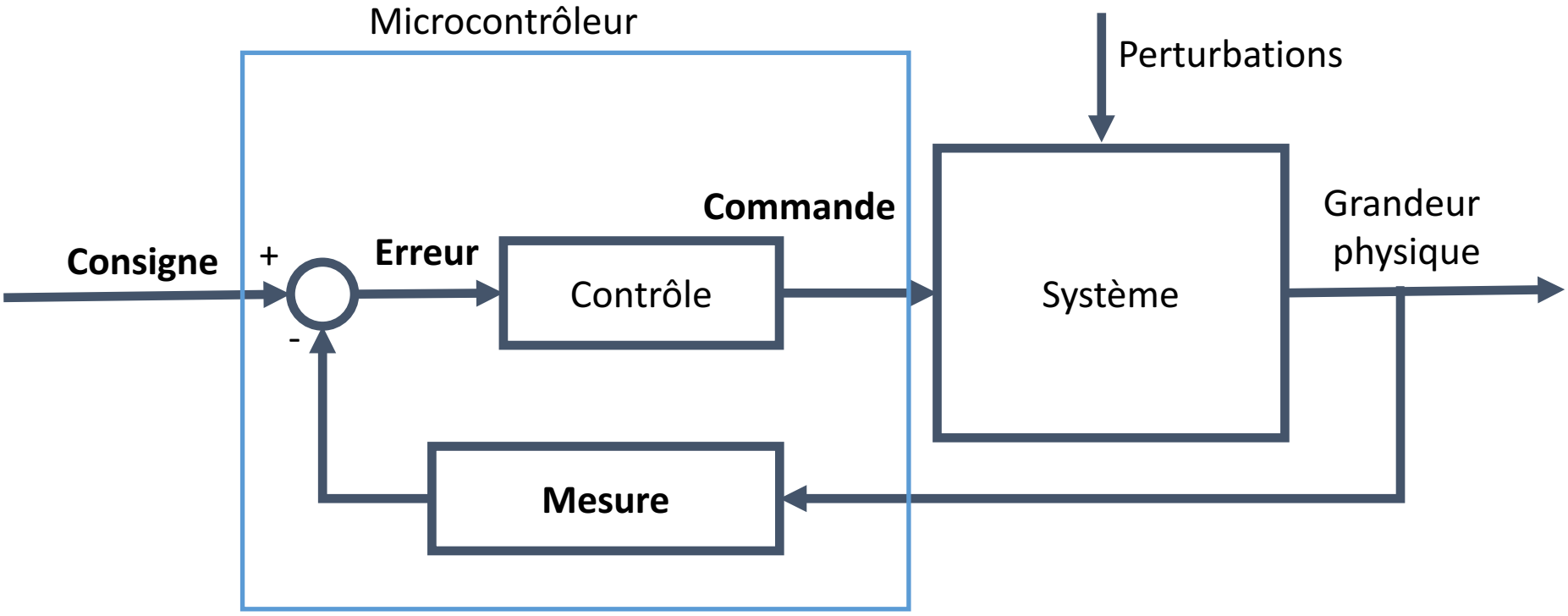
Régulation : Vitesse moteur

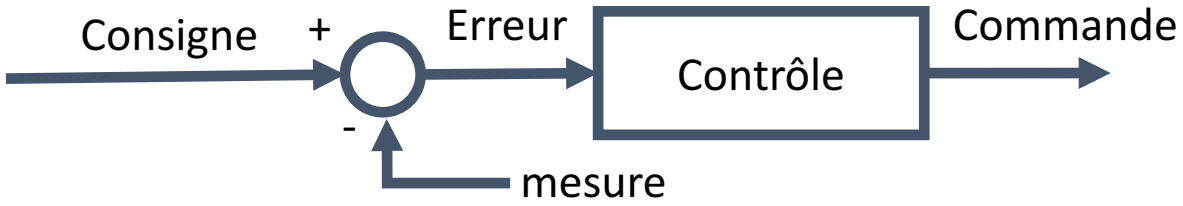


Régulation : Position moteur



Régulation

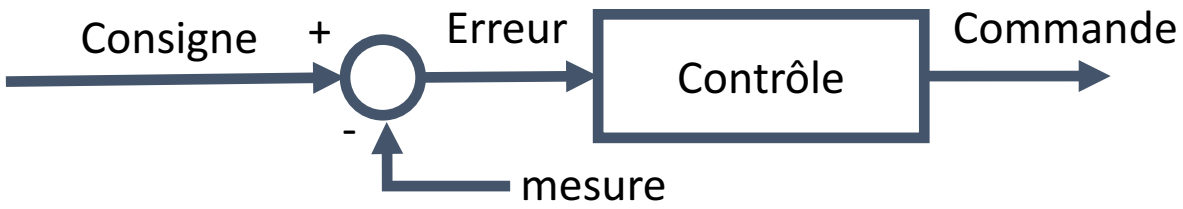

Microcontrôleur
Perturbations
Commande
Grandeur
physique
Consigne
+
Erreur
Contrôle
Système
-
Mesure



- ON/OFF

  commande = erreur > 0

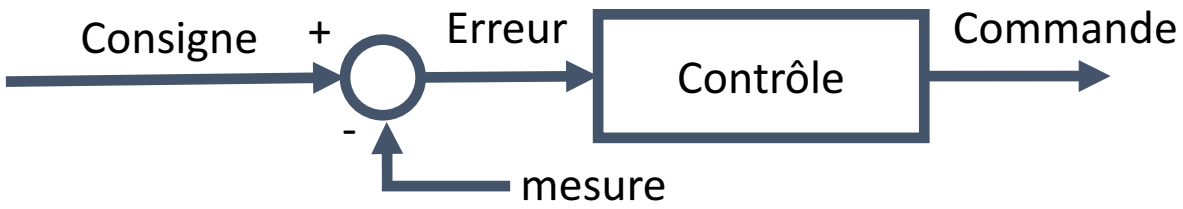
- Exemple : Thermostat
  - Si la température mesurée est inférieur à la consigne, on chauffe
  - Si la température mesurée est supérieur ou égale à la consigne, on ne chauffe pas



- Proportionnel (régulateur P)

  *commande = k_P * erreur*

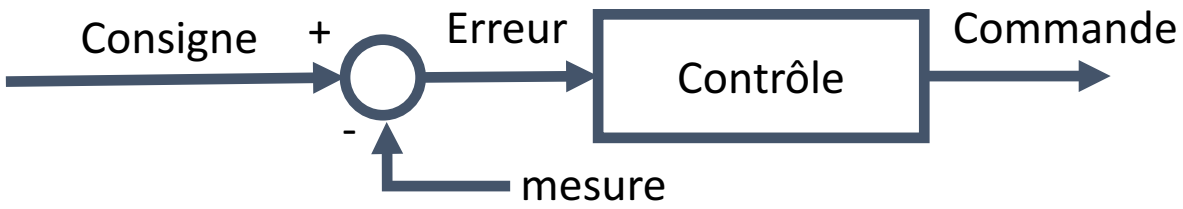
- Exemple : contrôle de position
  - La commande est une tension ou un rapport cyclique (MLI) + une direction (= signe)
  - Plus l'écart entre mesure et consigne est grand plus on applique une commande élevée
  - Le signe est celui de l'erreur



- Proportionnel et intégrale (régulateur PI)

  $commande = k_P * erreur + k_I * \int_{t0}^{t} erreur\ dt$

- Exemple : contrôle de vitesse
  - La commande est une tension ou un rapport cyclique (MLI)
  - Pas de freinage en cas d'erreur négative
  - Plus l'écart entre mesure et consigne est grand plus on applique une commande élevée
  - Sans le terme intégral, on peut avoir une erreur statique



- ON/OFF

  commande = erreur > 0

- Proportionnel (régulateur P)

  *commande = k_P * erreur*

- Proportionnel et intégrale (régulateur PI)

  *commande = k_P * erreur + k_I * $\int_{t0}^{t} erreur\ dt$*

Considérations pratiques : Mesure de vitesse

- Nouvelles fonctions fournies par la librairie motor.h:
  - `unsigned short getMotor1SensorA(void);`
  - `unsigned short getMotor1SensorB(void);`
  - `unsigned short getMotor2SensorA(void);`
  - `unsigned short getMotor2SensorB(void);`
- Chaque moteur est équipé de deux capteurs (A et B) qui renvoient un 1 pendant un demi tour. Ils sont positionnés des telle façon que sur un tour complet, on observe une séquence : (a,b) = {(0,0),(0,1),(1,1),(1,0)} ou l'inverse
- Ceci nous permet de compter les tours et de connaître le sens de rotation.

Considérations pratiques : Mesure de vitesse

- Procédure
  - « Suffisamment souvent » pour ne rater aucun changement, vérifier l'état des capteurs
  - S'il a changé, incrémenter une variable qui représentera le nombre de quarts de tours effectués
  - Pour obtenir une vitesse, il faut pouvoir comparer le nombre de tour avec un temps écoulé.
- Attention : entre le nombre de tour de roue et le nombre de tour de moteur, il y a un rapport de réduction de 19 (ou 53)

Considérations pratiques : PI numérique

- *commande* = k_P * *erreur* + k_I * $\int_{t0}^{t} erreur\ dt$
- On choisit d'appliqué une nouvelle consigne suffisamment souvent par rapport à l'inertie du moteur. Par exemple, 100 fois seconde
- Calcul de l'intégral : il s'agit de la sommes des erreurs passées :

```
void timer1Interrupt(void){
    erreur = consigne - mesure
    integrale = integrale + erreur;
    commande = kp*erreur +
ki*integrale
}
```

Considérations pratiques : PI numérique

- La commande est un rapport cyclique en %, il faut donc veiller à ce que le résultat du calcul reste entre 0 et 100.

```
if(commande>100)
    pwm = 100;
else if(commande<0)
    pwm = 0;
else
    pwm = commande;
}
```

- Le terme intégrale peut devenir très grand. Veillez à l'empêcher de sortir de la plage de son type.