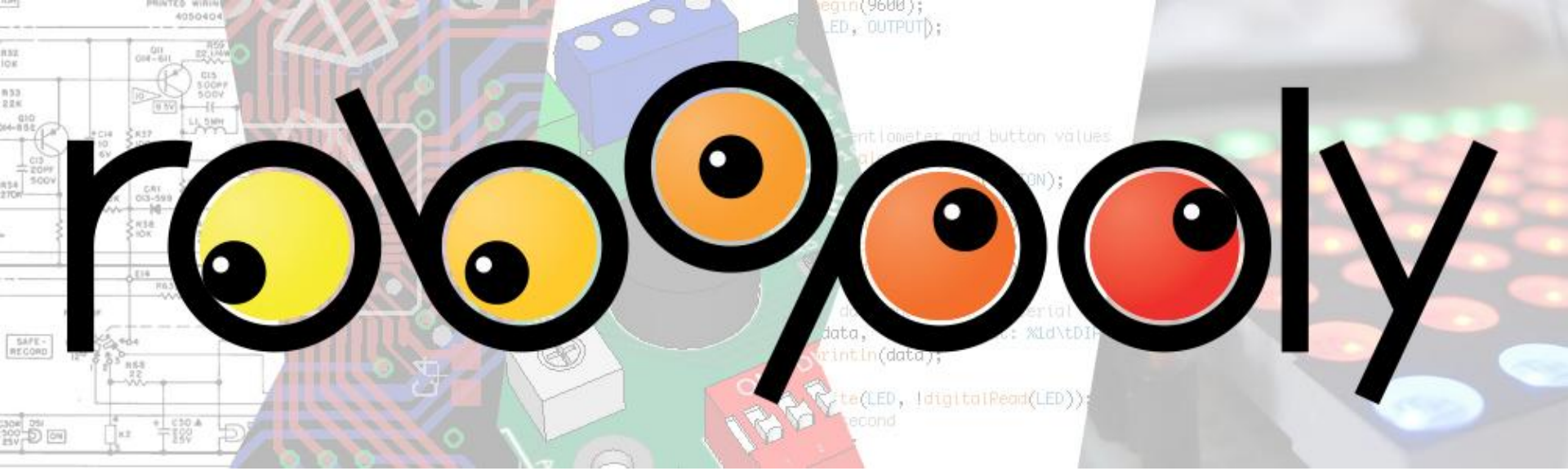


Power board

PWM

Moteurs et servomoteurs

Pourquoi une Power Board ?

- Problème n°1: Comment rendre notre **robot totalement autonome ?**
- Solution: Utiliser des **batteries** (rechargeables)

- Problème n°2: Les batteries n'ont **jamais une tension stable et fixe**, pourtant c'est nécessaire:
  - Le PRismino et la logique fonctionnent à **5V**
  - Besoin d'une référence pour la lecture analogique
- Solution: **Convertir la tension** des batteries en 5V

C'est le role de la Power Board

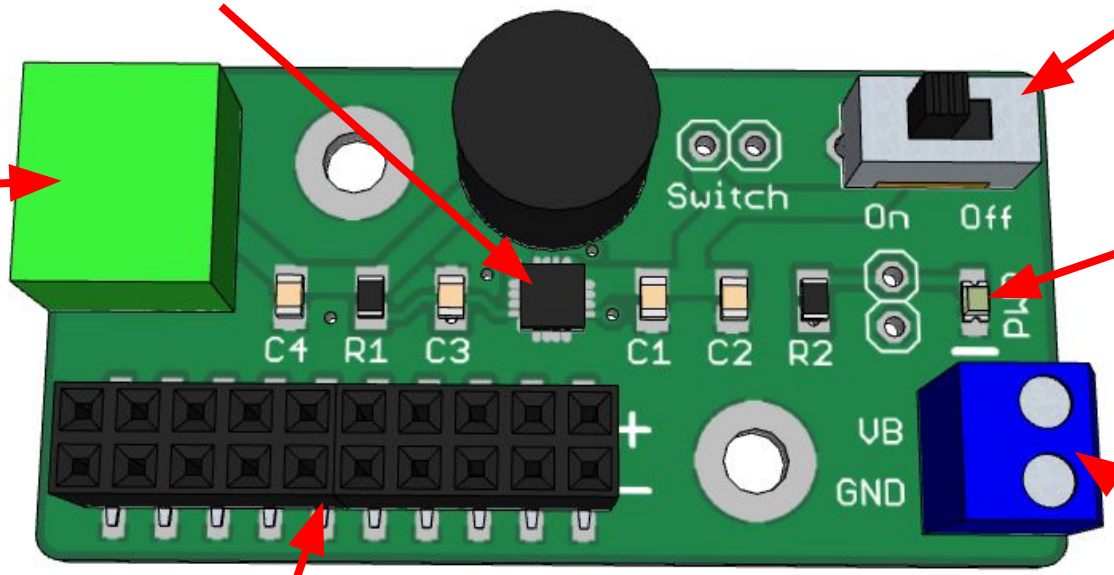
Power Board - Caractéristiques

- Tension d'entrée: **5V à 17V***
- Tension de sortie: **5V stable**
- Délivre un courant jusqu'à **3A**

- Protection contre les courts-circuits
- Interrupteur marche/arrêt
- Multiples **sorties 0V et +5V**
  - pour capteurs, LEDs, tensions de référence

*Si connectée au Shield: ***max = 11.8V*** (voir plus loin)

Power Board



Régulateur de tension

Switch

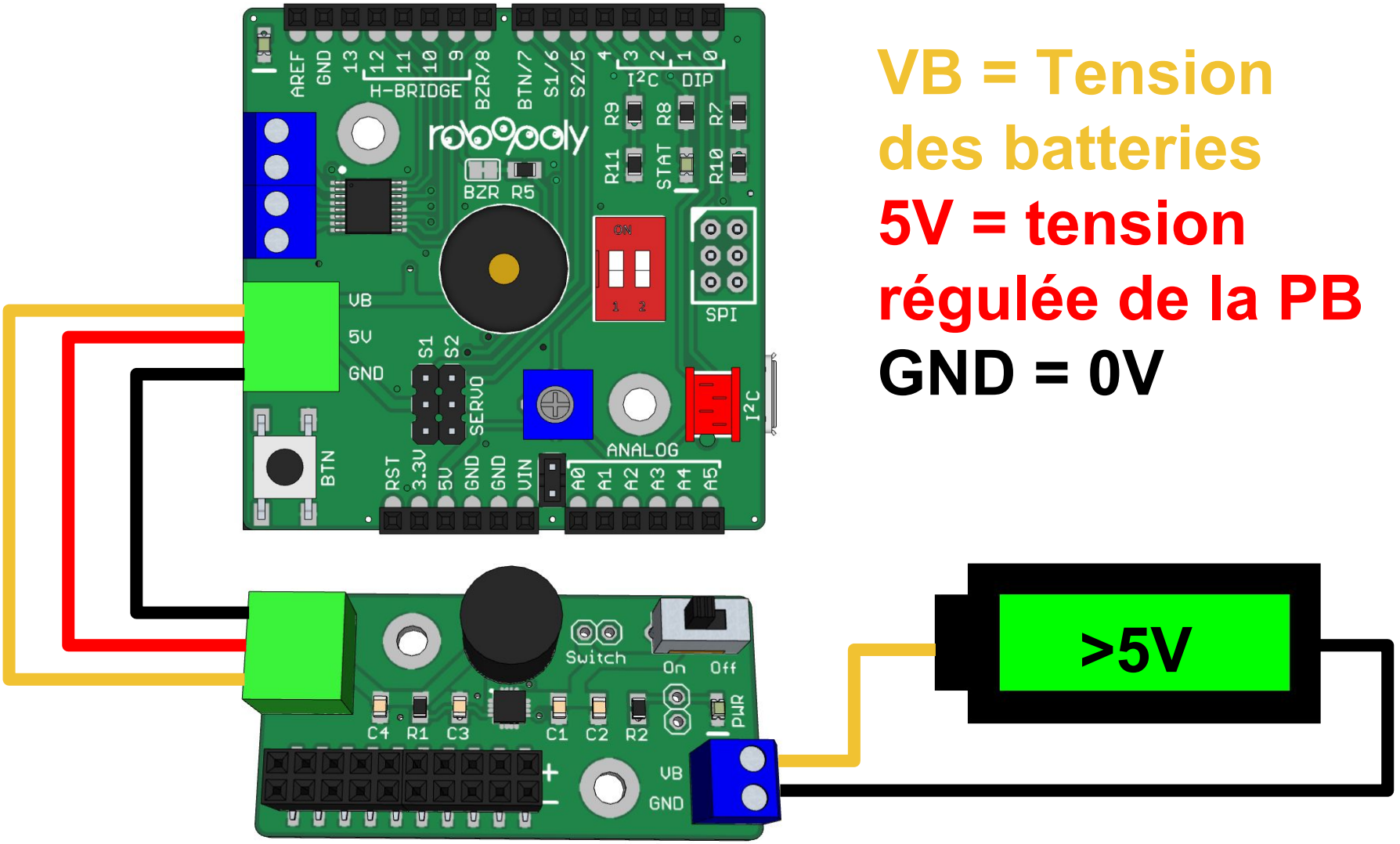
Sortie vers le shield

LED de status

Entrée batterie

10 sorties à 5V pour d'autres élements (capteurs…)

Power Board - Branchements

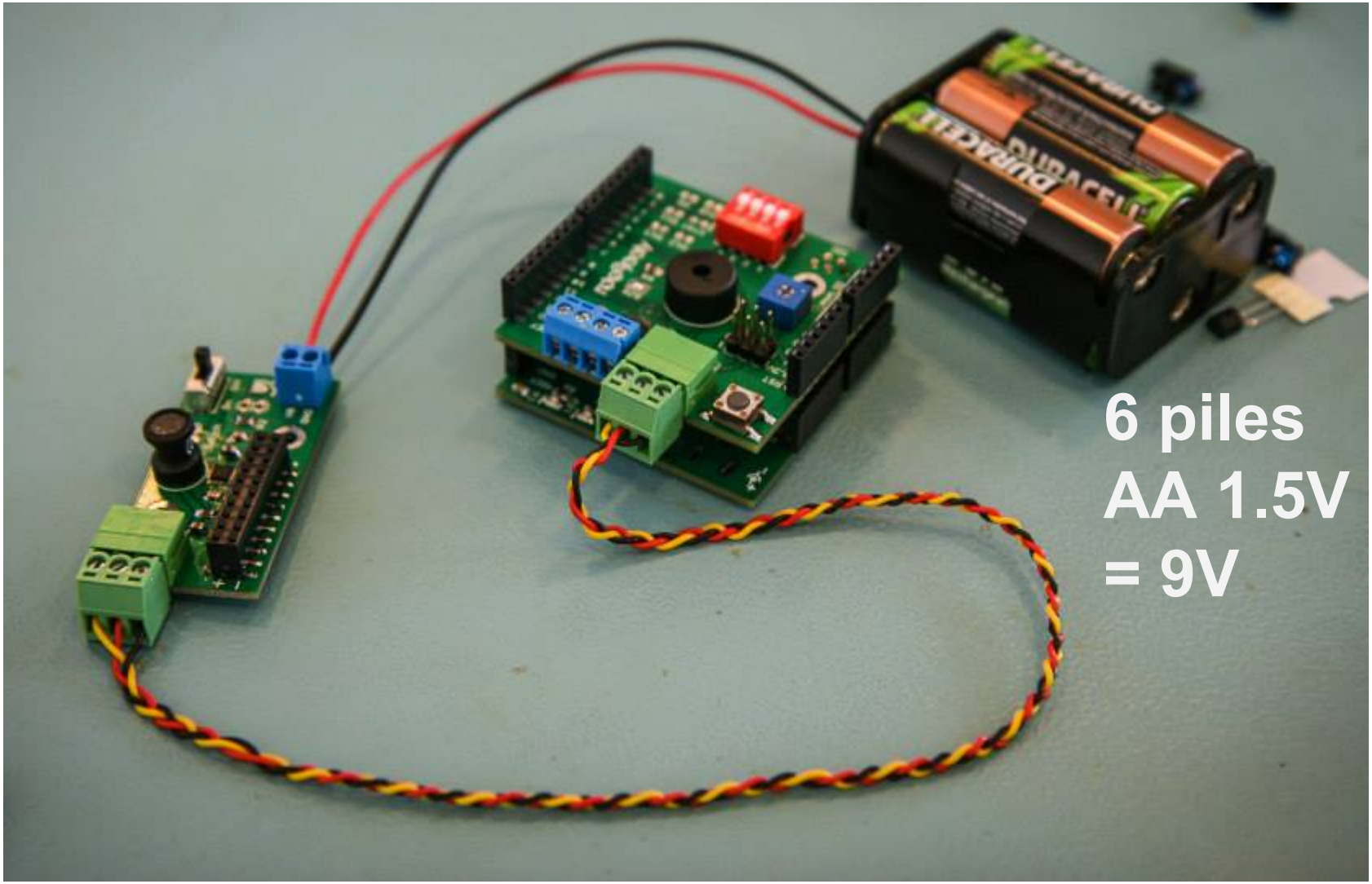


VB = Tension des batteries

5V = tension régulée de la PB

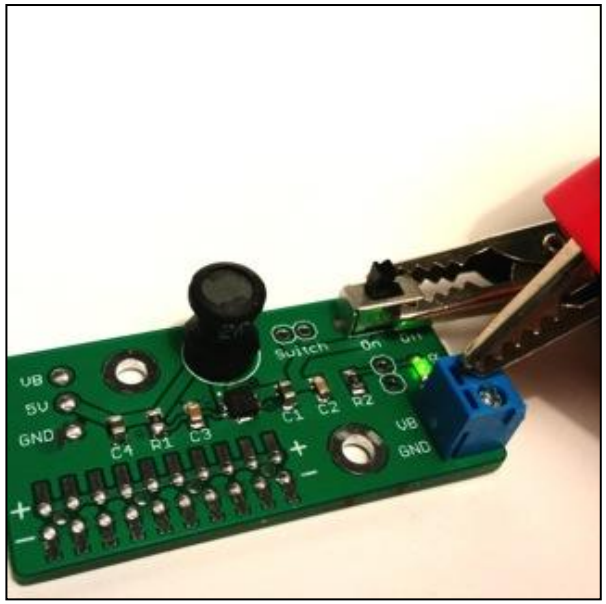
GND = 0V

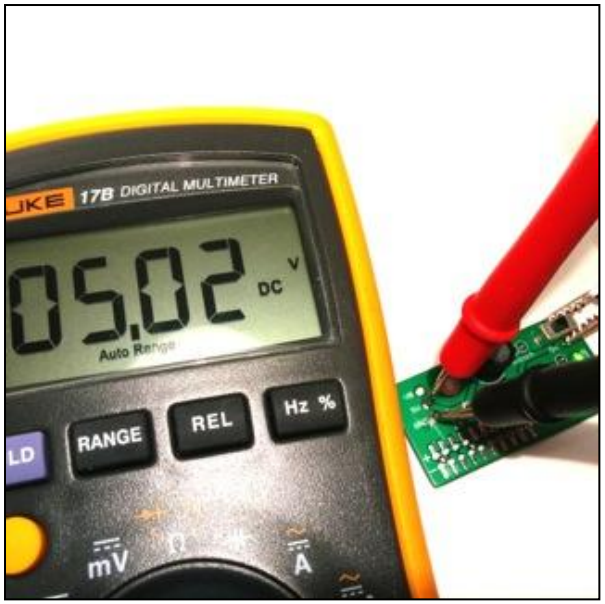
Power Board - Rack de piles

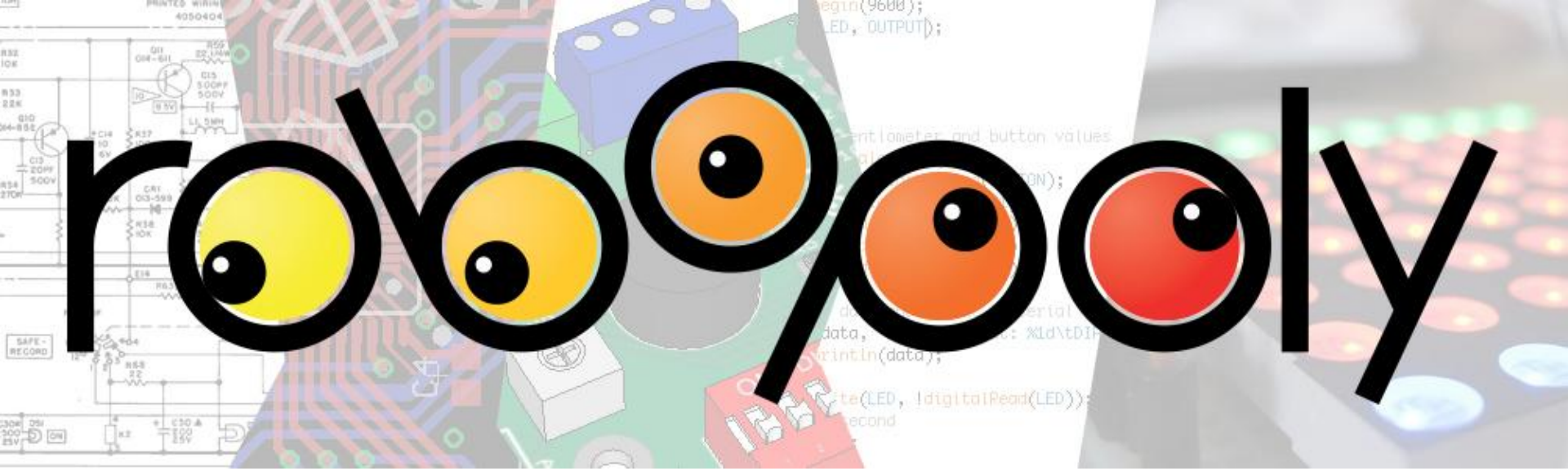


Attention

- Avant de souder les connecteurs
  - **alimenter la PowerBoard** (avec une source de tension au local par exemple)
  - **tester la sortie 5V** avec un multimètre
- Si il n'indique pas 5V, **c'est qu'il y a un problème!**







Power board

PWM

Moteurs et servomoteurs

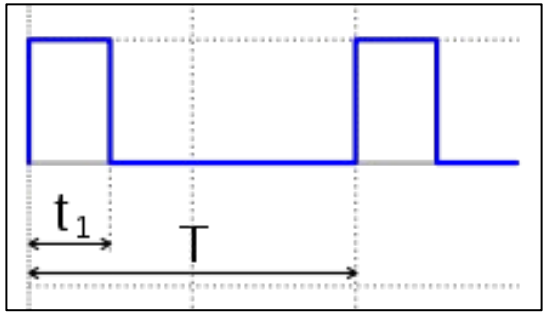
Pourquoi du PWM ?

- Problème : le PRismino ne peut sortir que des **tensions fixes**: **0V ou 5V.**
  Comment **transmettre une information** comme on le ferait en faisant **varier la tension de sortie** ?

- Solution: Utiliser le **PWM** (Pulse Modulation Width ou Modulation de Largeur d'Impulsion).

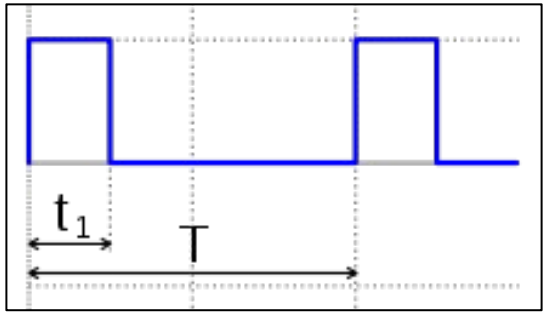
Rapport cyclique (Duty cycle)

- **Rapport** de la **durée de l'état haut** (**t1**) sur la **période** (**T**).

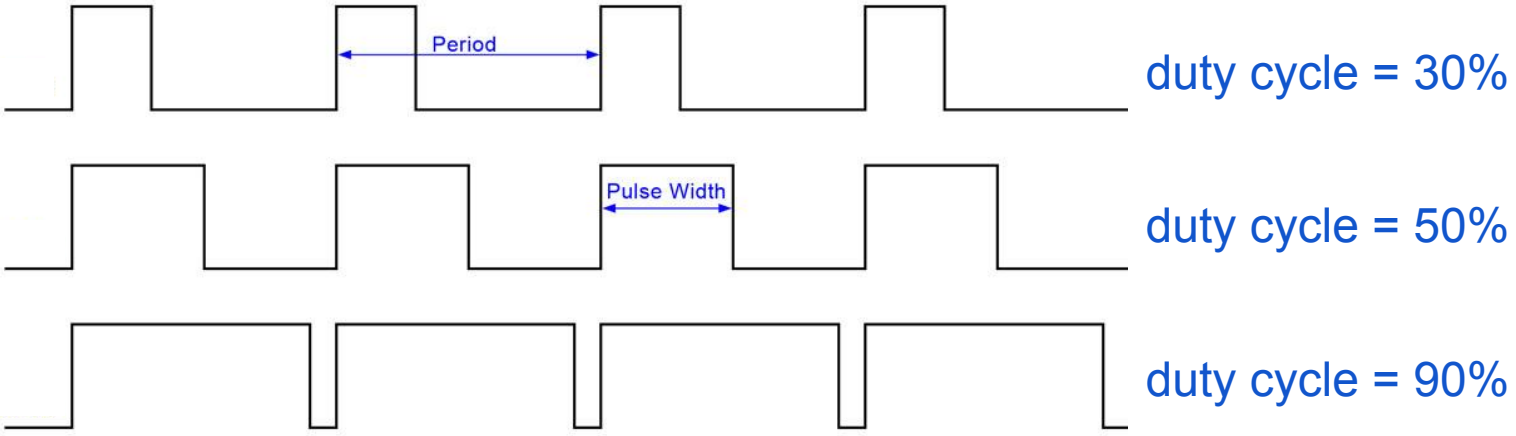


Rapport cyclique (Duty cycle)

- **Rapport** de la **durée de l'état haut** (**t1**) sur la **période** (**T**).

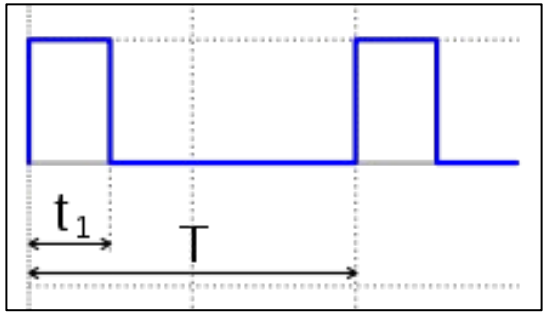


- Varie de **0 à 100%**.

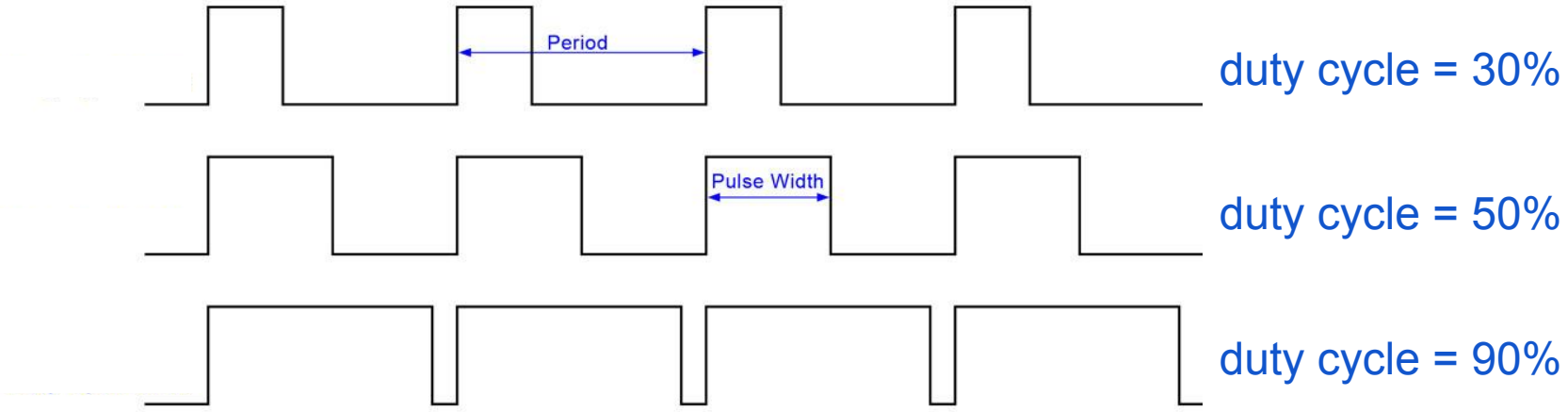


Rapport cyclique (Duty cycle)

- **Rapport** de la **durée de l'état haut** (**t1**) sur la **période** (**T**).



- Varie de **0 à 100%**.

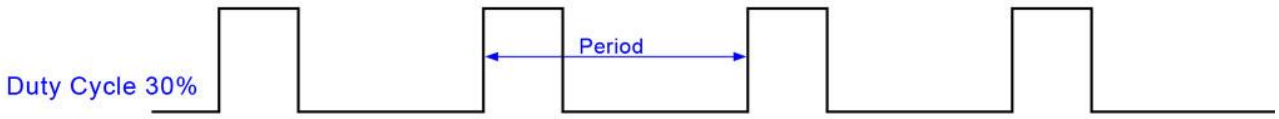


Exemple : Pour un **rapport cyclique de 30%**, le signal **reste à l'état haut pendant 0.3*T**.

Principe du PWM

- **Succession d'états logiques** (0/+5V)

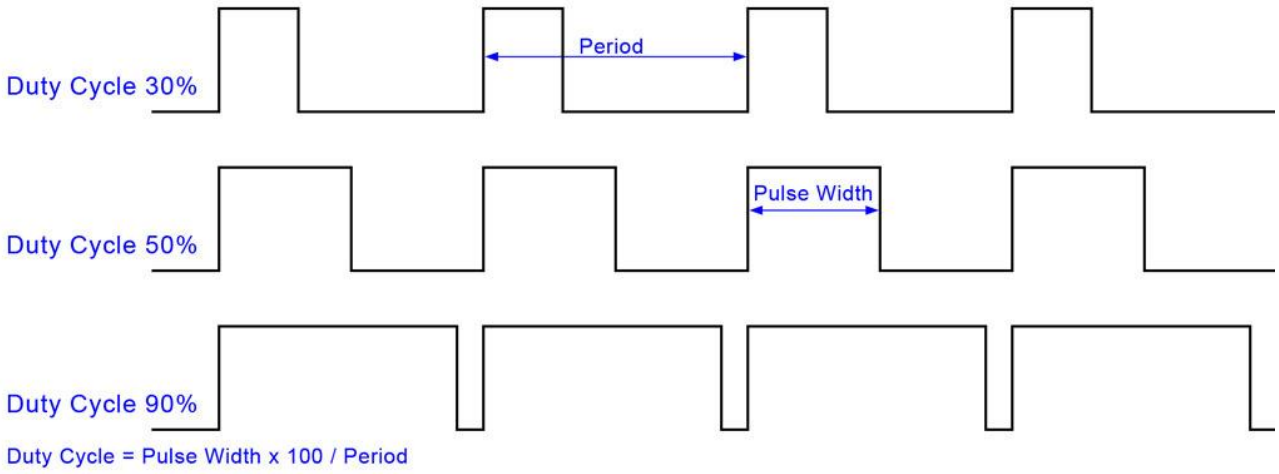
- Signal PWM



Principe du PWM

- **Succession d'états logiques** (0/+5V) dont on fait **varier le rapport cyclique** (0 à 100%)

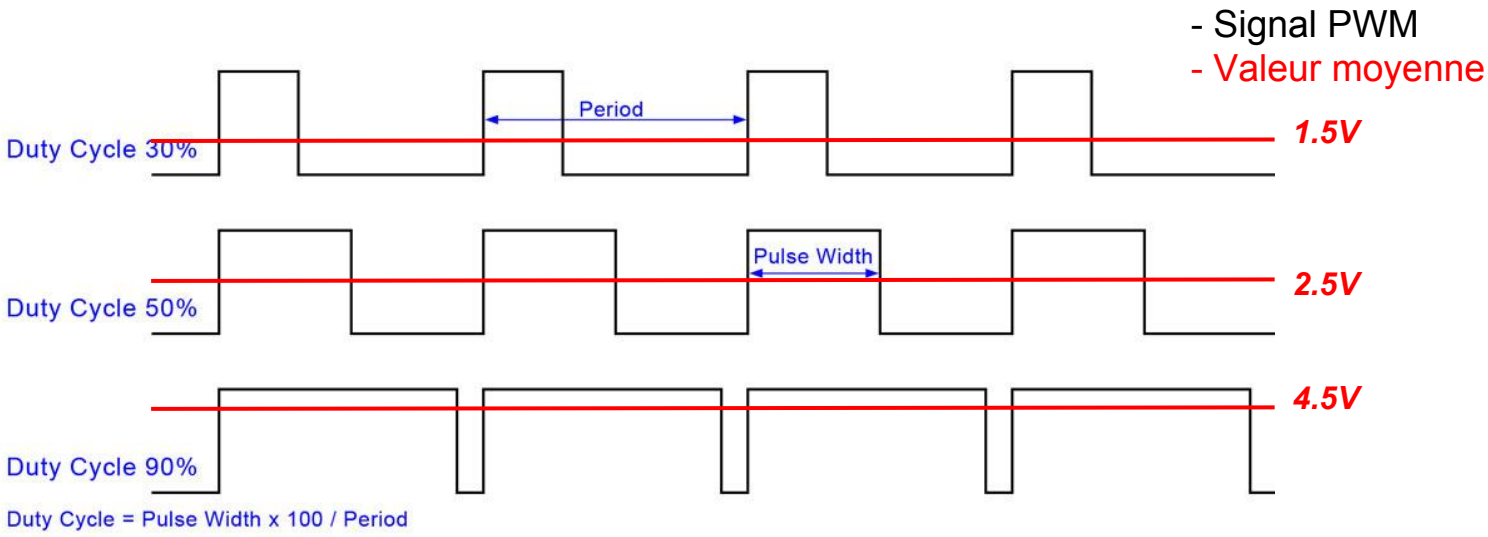
- Signal PWM



Principe du PWM

- **Succession d'états logiques** (0/+5V) dont on fait **varier le rapport cyclique** (0 à 100%)
  - **Tension moyenne** entre 0 et 5V

Energie reçue **proportionnelle** à cette tension



PWM - En Pratique

- **On vous a menti**: analogWrite n'envoie pas une fonction continue mais un PWM...
- Utiliser la fonction **analogWrite**(**pin**, **valeur**)
  - **pin** parmi 3,5,6,9,10,11,13.
  - **valeur** entre 0 et 255.

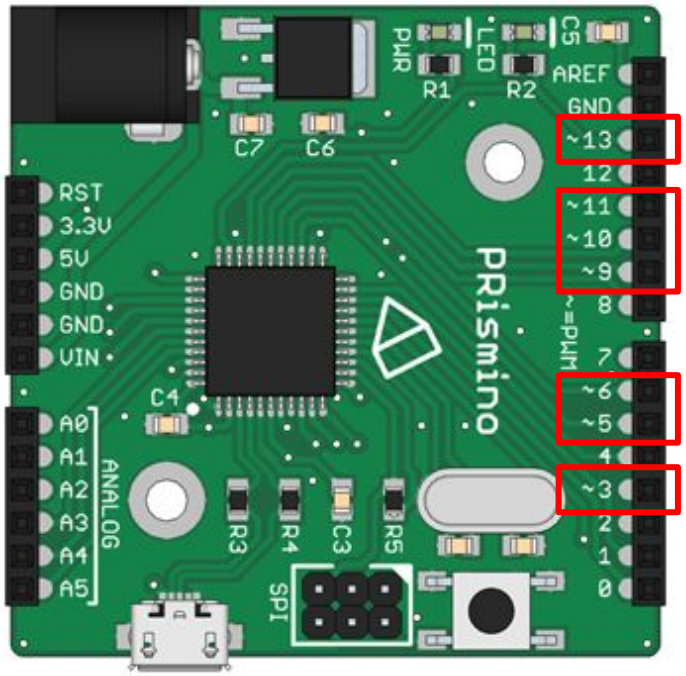
- Exemple :

analogWrite(3, 127)

envoie sur le **pin 3** un **signal PWM** de **rapport cyclique 50%** (~127/255)

⟺

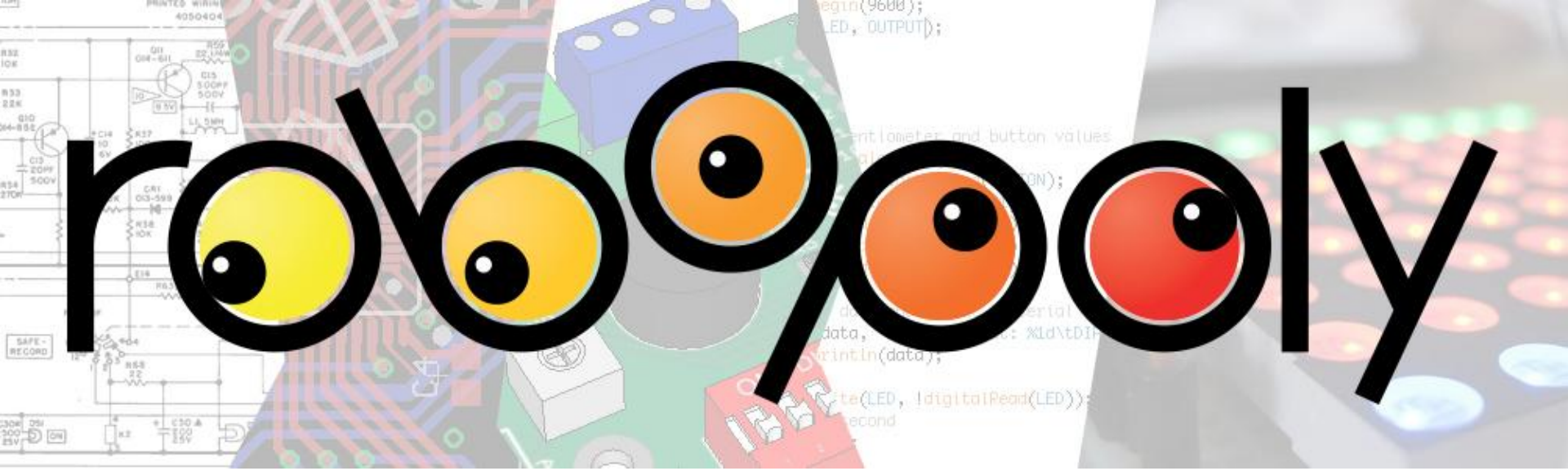
envoie sur le **pin 3** une **énergie equivalente** à celle d'une **tension continue** de **2.5 V**.



PWM - Quelle utilité ?

Exemples d'applications :

- Contrôler la **vitesse** d'un **moteur DC**
- Contrôler l'**angle** des **servomoteurs**
- Contrôler la **luminosité** d'une **LED**



Power board et Branchements

PWM

Moteurs et servomoteurs

Moteur à Courant Continu (DC*)

- Nécessite une **tension continue**
  (= pas alternative)
- Vitesse **proportionnelle** à la tension (3-6V)

**DC = Direct Current = Courant Continu*

Moteur à Courant Continu (DC*)

- Nécessite une **tension continue**
  (= pas alternative)
- Vitesse **proportionnelle** à la tension (3-6V)
- Deux types de moteurs proposés par Robopoly:

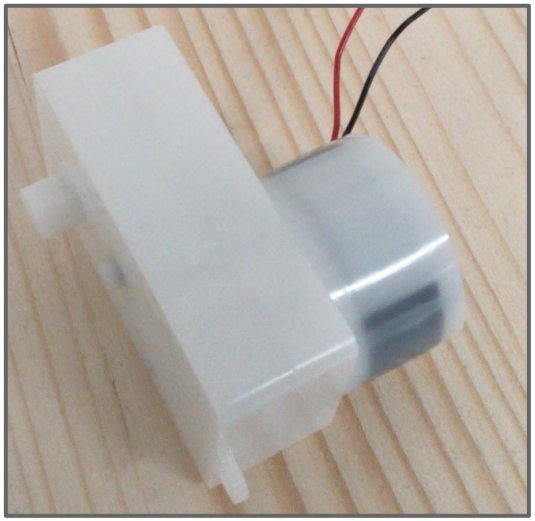
Moteurs standards avec reducteurs

**DC = Direct Current = Courant Continu*

Moteur à Courant Continu (DC*)

- Nécessite une **tension continue**
  (= pas alternative)
- Vitesse **proportionnelle** à la tension (3-6V)
- Deux types de moteurs proposés par Robopoly:

Moteurs standards avec reducteurs



Moteurs à axe excentré avec reducteurs (pour encodeurs)

**DC = Direct Current = Courant Continu*

Parenthèse: un encodeur, c'est quoi ?

- Un dispositif permettant de **compter les tours de roues**
- En effet on connait la tension envoyée au moteur, mais pas sa position ni sa vitesse exacte

- Un encodeur permet de savoir tout ça!
  - Connaitre et réguler la **vitesse du robot**
  - Connaitre la **position exacte du robot dans un repère défini** (x;y) et son orientation θ
- Robopoly dispose de modules d'encodeurs
- **Documentation + démon disponibles très bientôt**

Pourquoi un pont-H ?

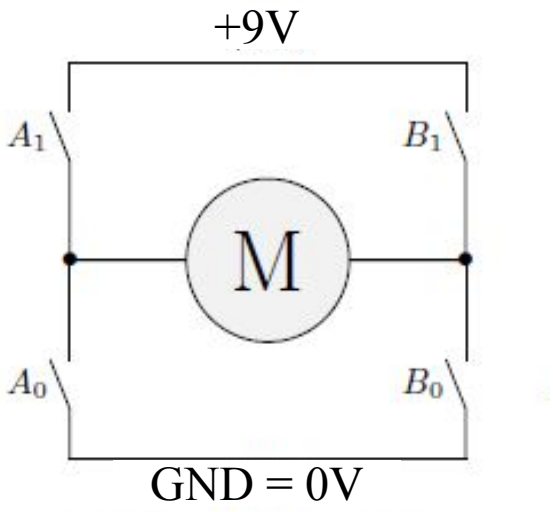
- Le microcontrôleur ne peut pas **fournir directement un courant élevé** (>20 mA). **Le pont-H le peut !**

Pourquoi un pont-H ?

- Le microcontrôleur ne peut pas **fournir directement un courant élevé** (>20 mA). **Le pont-H le peut !**
- Choisir le **sens de rotation** du moteur.

Pourquoi un pont-H ?

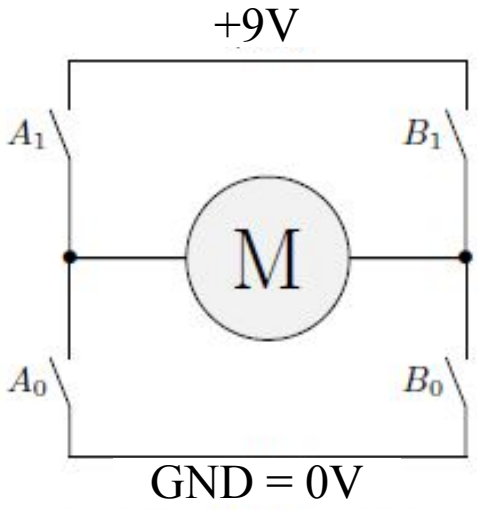
- Le microcontrôleur ne peut pas **fournir directement un courant élevé** (>20 mA). **Le pont-H le peut !**
- Choisir le **sens de rotation** du moteur.



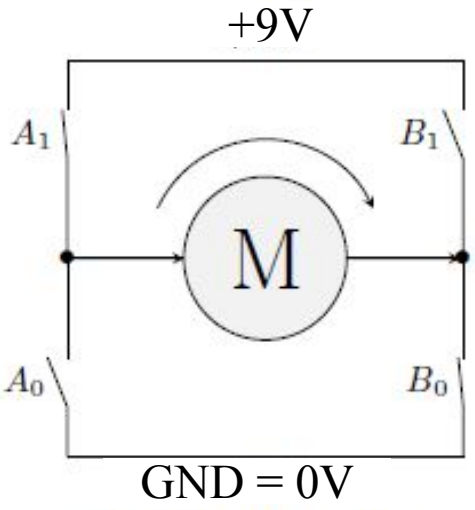
Moteur à l'arrêt

Pourquoi un pont-H ?

- Le microcontrôleur ne peut pas **fournir directement un courant élevé** (>20 mA). **Le pont-H le peut !**
- Choisir le **sens de rotation** du moteur.



Moteur à l'arrêt

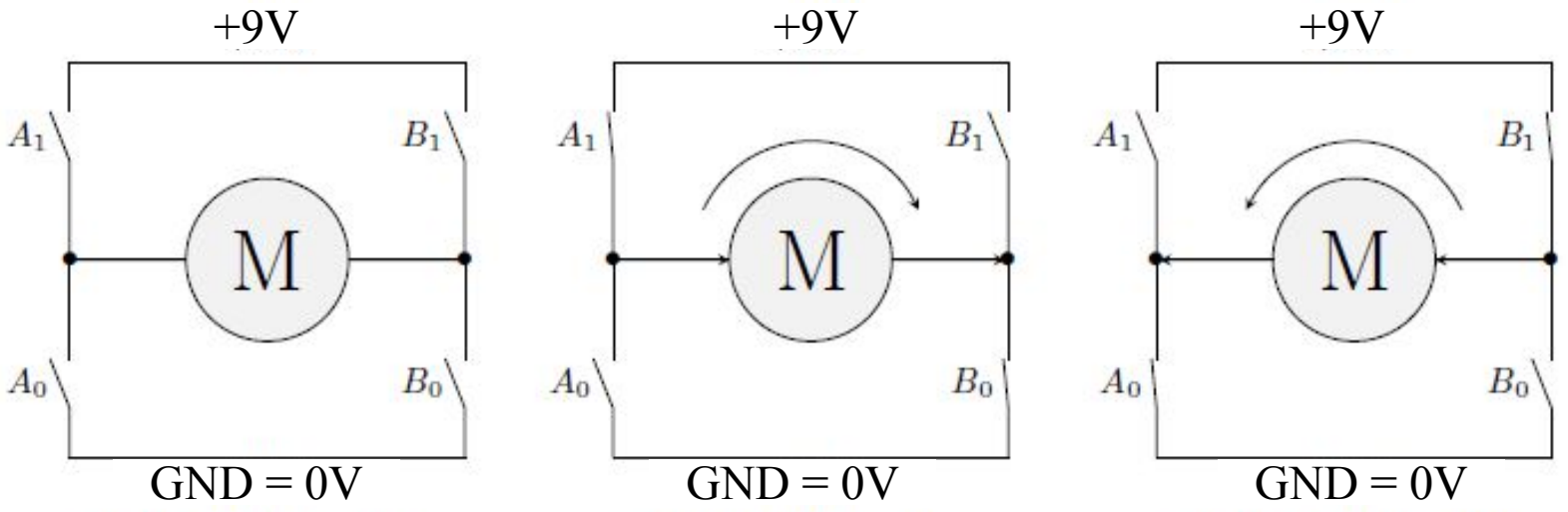


Sens horaire

M reçoit une tension +9V

Pourquoi un pont-H ?

- Le microcontrôleur ne peut pas **fournir directement un courant élevé** (>20 mA). **Le pont-H le peut !**
- Choisir le **sens de rotation** du moteur.



Moteur à l'arrêt

Sens horaire

M reçoit une tension +9V

Sens antihoraire

M reçoit une tension -9V

Attention

Même si la **Power Board** peut régulter une tension jusqu'à 17V, le **Pont-H** accepte une **tension maximale de 11.8V**.

Au delà:

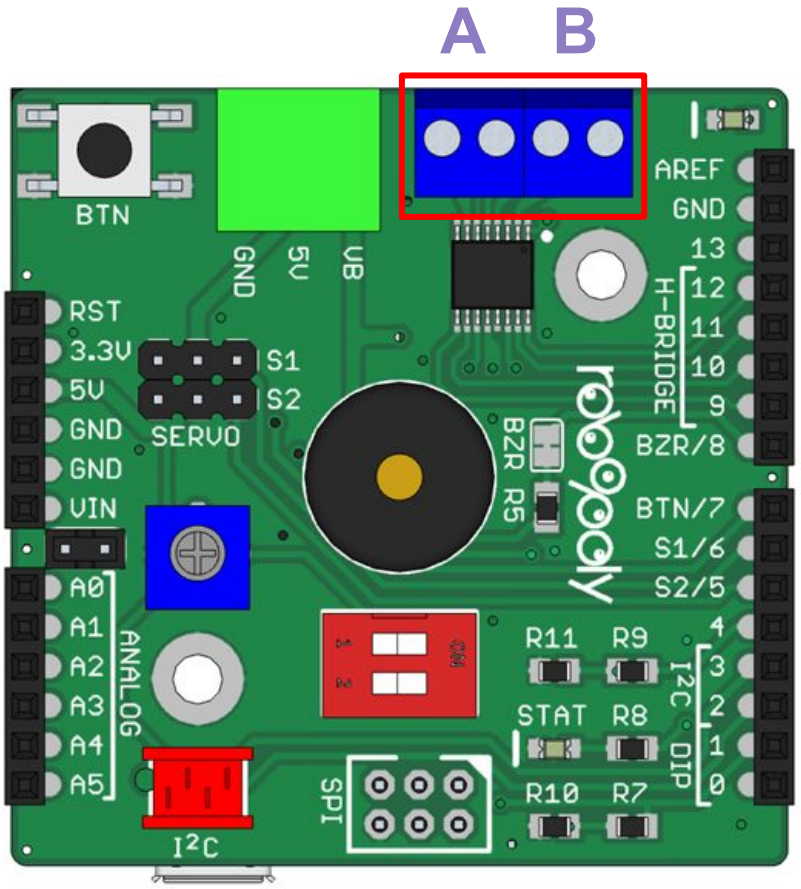
Pont-H à 12V

<u>Conclusion:</u> Ne pas alimenter la **Power Board** au delà de 11.8V.

VB < 11.8V

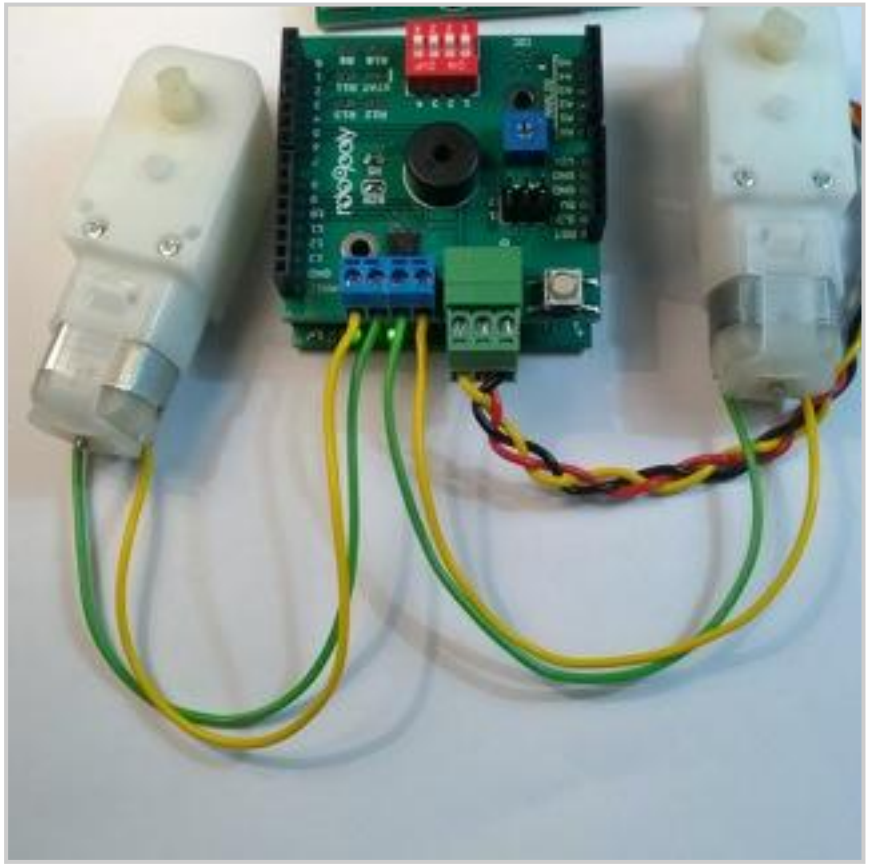
Moteurs - En pratique

- Côté Hardware :
  - Brancher les **moteurs** au shield :
    - **Moteur 1** sur **A**
    - **Moteur 2** sur **B**
  - Quand on utilise les moteurs, **ne rien connecter** sur les pins **9 à 12**.



Moteurs - En pratique

- Côté Hardware :
  - Brancher les **moteurs** au shield :
    - **Moteur 1** sur **A**
    - **Moteur 2** sur **B**
  - Quand on utilise les moteurs, **ne rien connecter** sur les pins **9 à 12**.



Moteurs - En pratique

- **Partie Software** :

Importer la librairie <prismino.h>

setSpeed(vMot1, vMot2)

Ne pas oublier
les delay(ms)

```
#include <prismino.h>

void setup(){
}

void loop(){
  setSpeed(20, 20);
  delay(500);
  setSpeed(-20, -20);
  delay(500);
}
```

Moteurs - En pratique

- **Partie Software** :

Importer la librairie
<prismino.h>

setSpeed(vMot1, vMot2)
vMot entre **-100 et 100**

Ne pas oublier
les delay(ms)

```
#include <prismino.h>

void setup(){
}

void loop(){
  setSpeed(20, 20);
  delay(500);
  setSpeed(-20, -20);
  delay(500);
}
```

En avant

En arrière

Moteurs - En pratique

- **Partie Software** :

Importer la librairie
<prismino.h>

setSpeed(vMot1, vMot2)
vMot entre **-100 et 100**

Ne pas oublier les delay(ms)

ms doit être supérieur au temps d'execution des moteurs (100ms)

```
#include <prismino.h>

void setup(){
}

void loop(){
  setSpeed(20, 20);
  delay(500);
  setSpeed(-20, -20);
  delay(500);
}
```

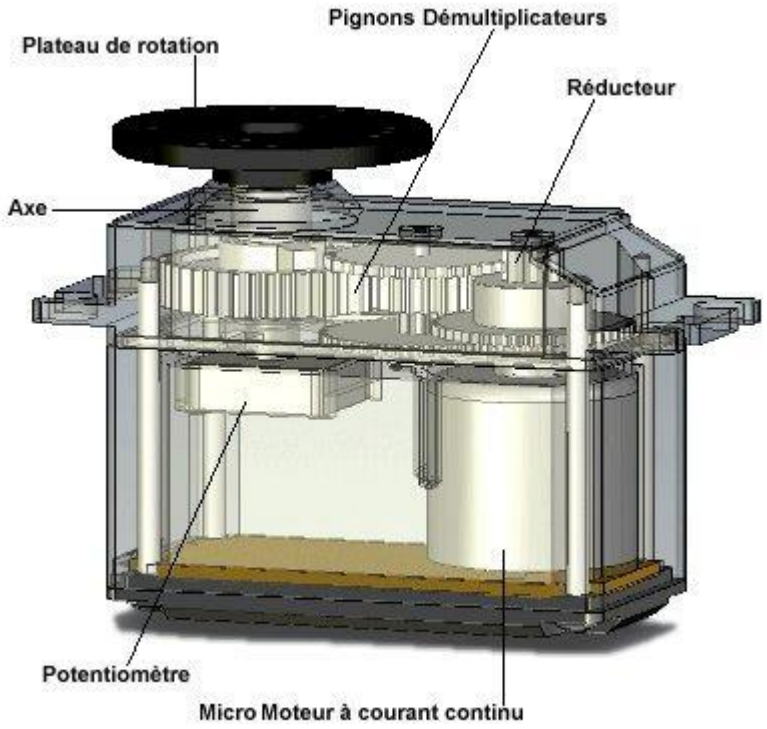
Moteurs - À vous de jouer !

- Petits exercices :
  - Faire **varier la vitesse** d'un **moteur** à l'aide d'un **potentiomètre**.
  - Réaliser un **robot mobile** (dans le plan) à l'aide de **deux moteurs** fixés à une **base**.

Servomoteurs - Principe et structure

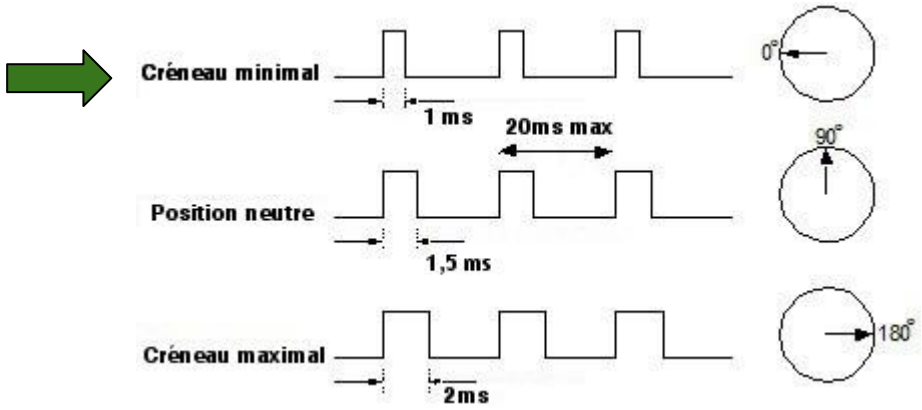
- **Contrairement aux moteurs DC contrôlés en vitesse de rotation.**

- **Les servomoteurs sont asservis en position angulaire** (0 à 180°)
  = **conserve l'angle** donné en consigne.



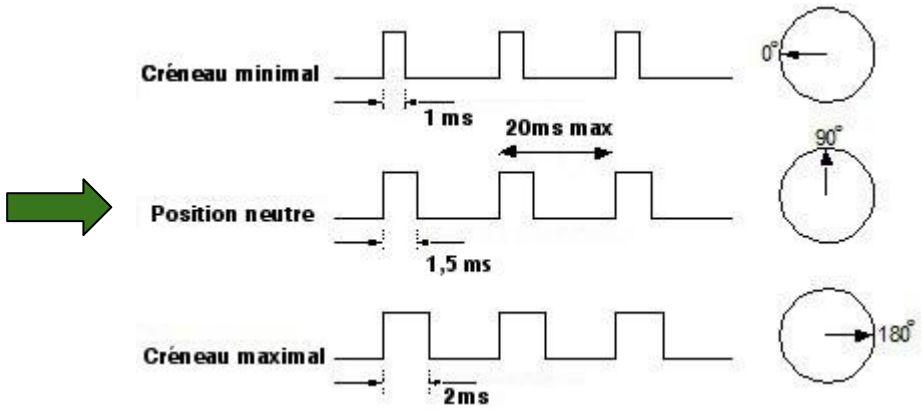
Servomoteurs - Contrôle PWM

- Le **rapport cyclique** définit **l'angle du servomoteur** :
  - **PWM de 5%** = **angle de 0°**



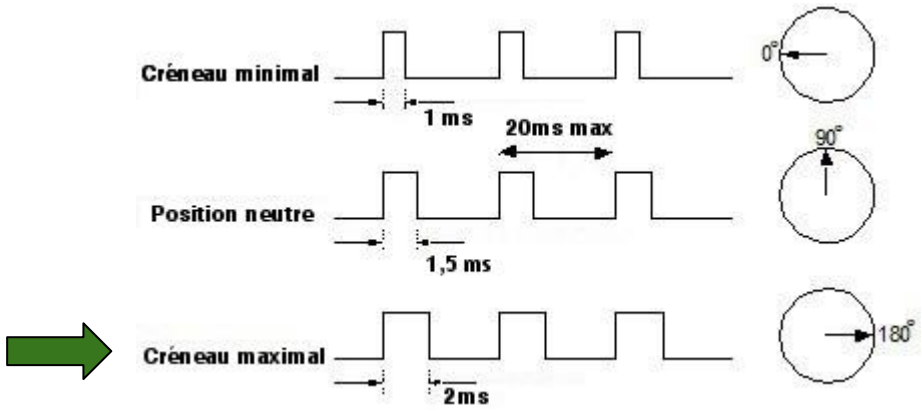
Servomoteurs - Contrôle PWM

- Le **rapport cyclique** définit **l'angle du servomoteur** :
  - **PWM de 5%** = **angle de 0°**
  - **7.5%** => **90°**



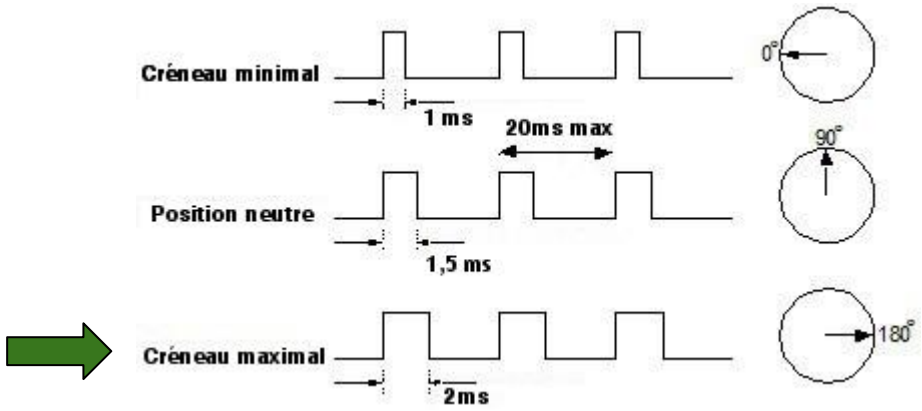
Servomoteurs - Contrôle PWM

- Le **rapport cyclique** définit **l'angle du servomoteur** :
  - **PWM de 5%** = **angle de 0°**
  - **7.5%** => **90°**
  - **10%** => **180°**



Servomoteurs - Contrôle PWM

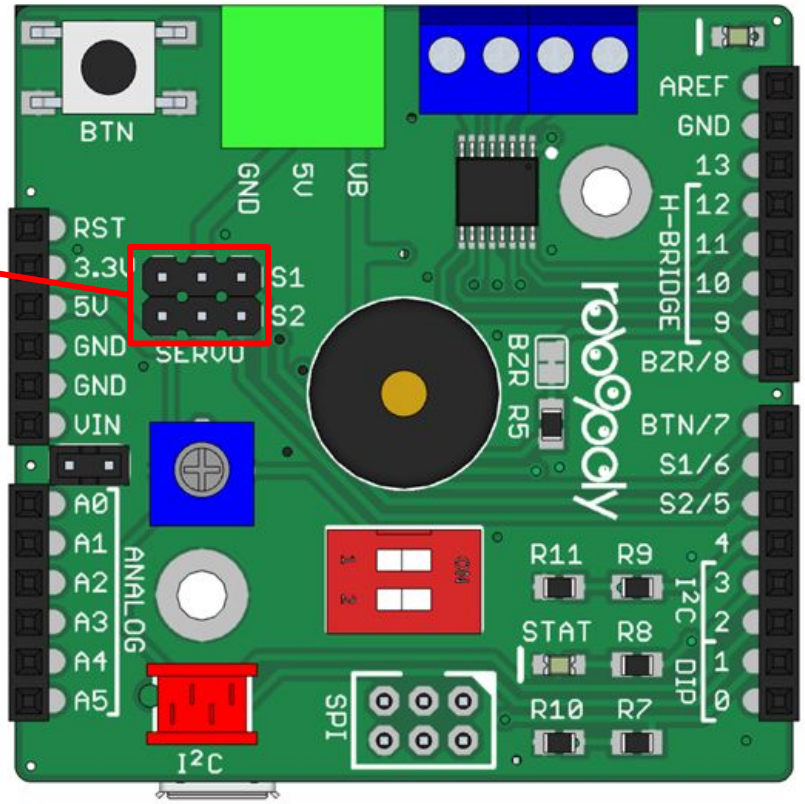
- Le **rapport cyclique** définit **l'angle du servomoteur** :
  - **PWM de 5%** = **angle de 0°**
  - **7.5%** => **90°**
  - **10%** => **180°**



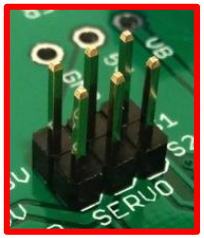
Grâce à <servo.h>, pas besoin d'apprendre la correspondance PWM-angle !

Servomoteurs - Branchements

- Le shield possède **2 connexions** destinées aux **servomoteurs**.



- **<u>Sens de connexion</u> :**
  - **Fil noir** (GND) **vers l'extérieur** de la carte.



Servomoteurs - En pratique

Importer la librairie Servo.h

Créer un objet servomoteur

Indiquer le pin du servo

Imposer un angle au servomoteur

```
#include <Servo.h>

// Définit l'objet monServo
Servo monServo;

void setup()
{
  monServo.attach(5); // Indique le pin du servo.
  monServo.write(90);
}

void loop(){

}
```

Servomoteurs - En pratique

Importer la librairie Servo.h

Créer un objet servomoteur

Indiquer le pin du servo

Imposer un angle au servomoteur

```
#include <Servo.h>

// Définit l'objet monServo
Servo monServo;

void setup()
{
  monServo.attach(5); // Indique le pin du servo.
  monServo.write(90);
}

void loop(){

}
```

Servomoteurs - En pratique

Importer la librairie Servo.h

Créer un objet servomoteur

Indiquer le pin du servo

Imposer un angle au servomoteur

```
#include <Servo.h>

// Définit l'objet monServo
Servo monServo;

void setup()
{
  monServo.attach(5); // Indique le pin du servo.
  monServo.write(90);
}

void loop(){

}
```

Servomoteurs - En pratique

Importer la librairie Servo.h

Créer un objet servomoteur

Indiquer le pin du servo

Imposer un angle au servomoteur

```
#include <Servo.h>

// Définit l'objet monServo
Servo monServo;

void setup()
{
  monServo.attach(5); // Indique le pin du servo.
  monServo.write(90);
}

void loop(){

}
```

Servomoteurs - À vous de jouer !

- **<u>Petits exercices</u> :**
  - Commander un **servo** avec un **bouton poussoir** (**0° à 90° puis 90° à 0°**).
  - Commander un **servo** avec un **potentiomètre** (**0° à 180° selon la valeur du pot.**).

À vos projets !

Pas d'idée ? Visitez notre page projet !
robopoly.epfl.ch/projets

Trop d'idées ? Venez nous en faire part !
projects@robopoly.ch

Prochains événements

- **Prochain Demon**
  - Lundi prochain, 12h15, ELA 1
  - Communication Serial (USB, Bluetooth)

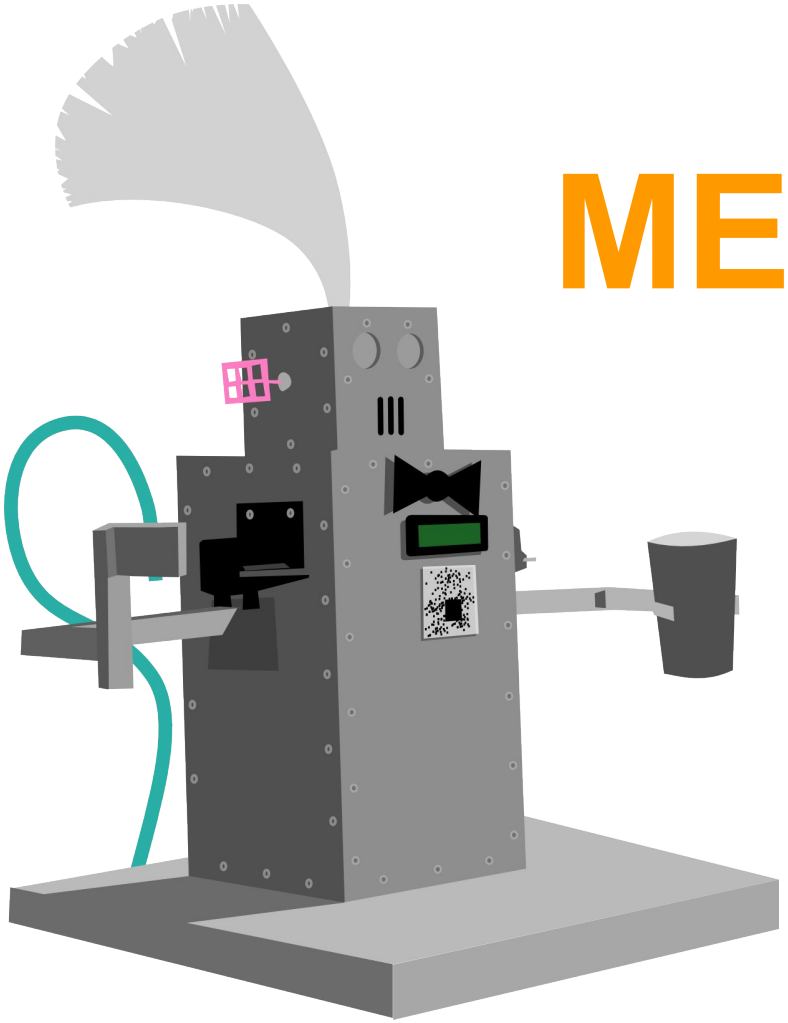
- **Workshop II**
  - Samedi 7 novembre, 9h-18h
  - En haut du BM et au local !
  - Pour poursuivre le montage de ton robot, discuter de tes projets, demander de l'aide au comité !

Contact/Infos

Contact principal
robopoly@epfl.ch

Site officiel - toutes les infos et slides sont la!
robopoly.epfl.ch

Facebook - pour suivre l'actualité du club!
www.facebook.com/robopoly



MERCI!

Questions?