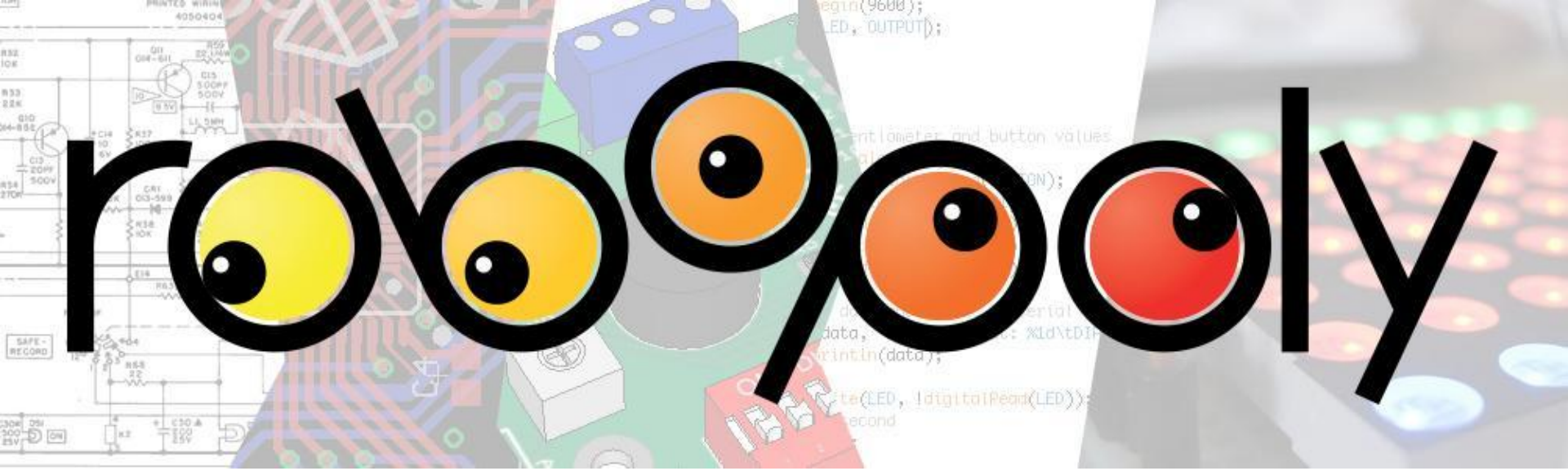


Les entrées/sorties

Module Shield

Programmation

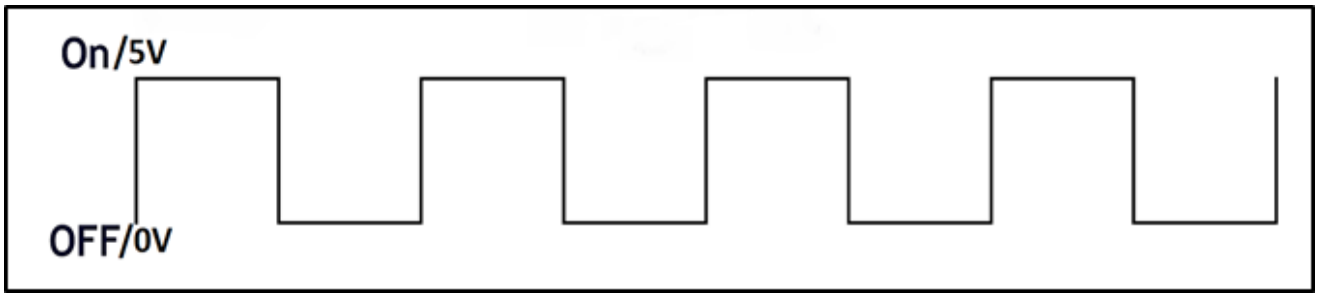
Signal digital

- Dans un ordinateur (et un microcontrôleur), l'information est stockée en binaire sous forme d'une succession de **'0'** et de **'1'**: des **bits**.

- Par exemple '00111010' peut représenter une information, mais son interprétation dépend du contexte!

Signal digital

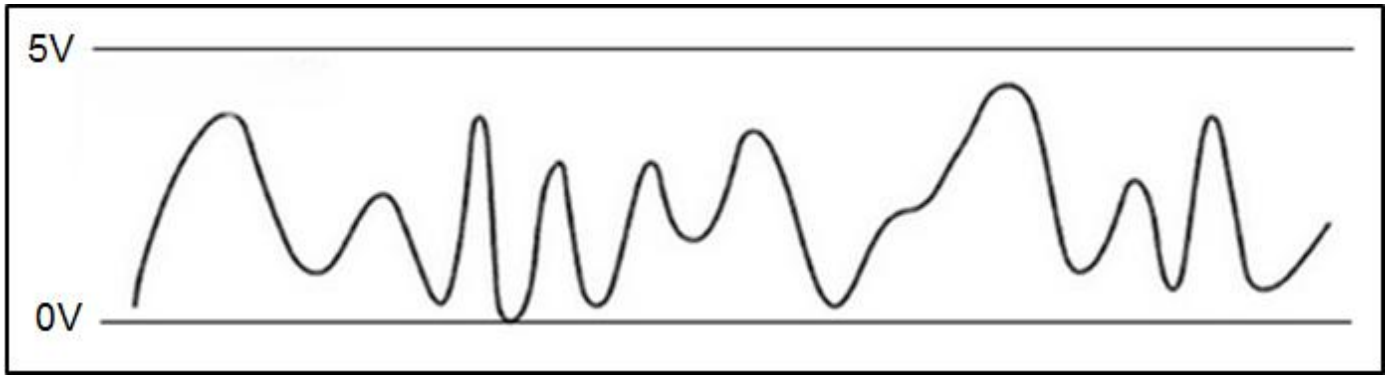
- Le signal est transmis par les valeurs de tension:
  - **‘0’** correspond en général à 0V
  - **‘1’** correspond en général à 5V



- La valeur d’un signal digital **vaut soit ‘0’, soit ‘1’**, mais **pas autre chose!**
  La valeur est **discrète**.
- Exemple: bouton → soit **relâché**, soit **pressé**
  LED → soit **allumée**, soit **éteinte**

Signal analogique

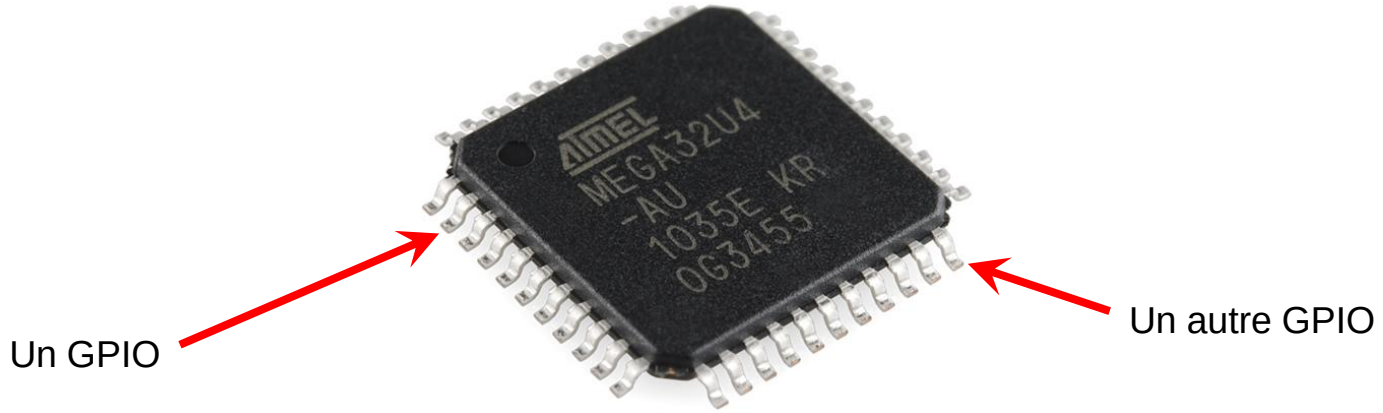
- Au contraire, un signal analogique peut **librement varier** entre 0V et par exemple 5V.



- Le signal analogique (qui peut prendre n'importe quelle valeur) **ne peut pas être directement stocké dans un ordinateur** (qui a une mémoire finie)

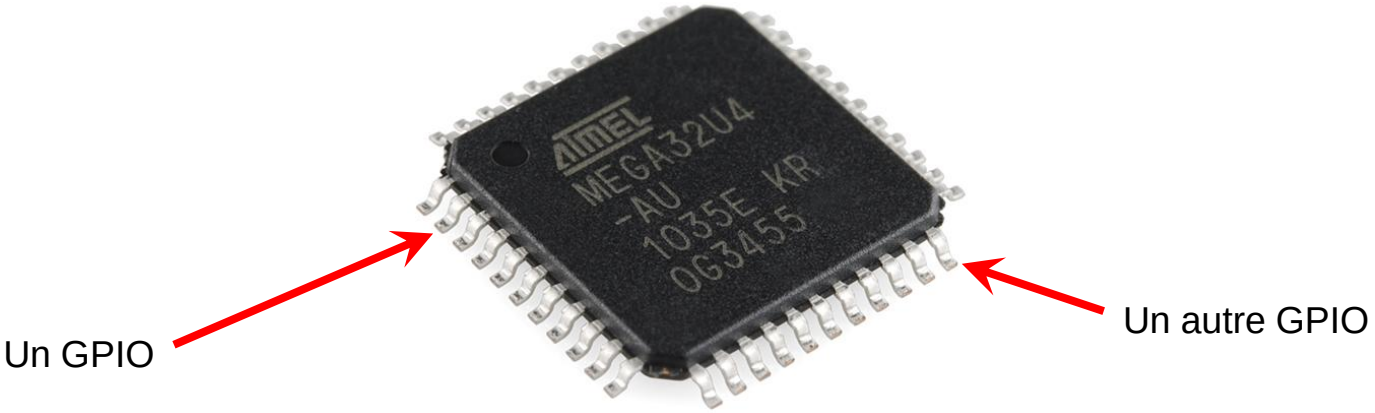
Les pins du microcontrôleur

- Certaines pattes du microcontrôleur correspondent à des **GPIO** (General Purpose Input Output)
- Un GPIO peut être configuré à volonté soit en ***mode entrée***, soit en ***mode sortie***, mais **pas les deux en même temps!**



Les pins du microcontrôleur

- Comment choisir le mode entrée ou sortie dans un programme en cours ?
- On appelle la fonction **pinMode(number, mode)**
  - **number** correspond au numéro du pin en question
  - **mode** vaut généralement **INPUT** ou **OUTPUT**



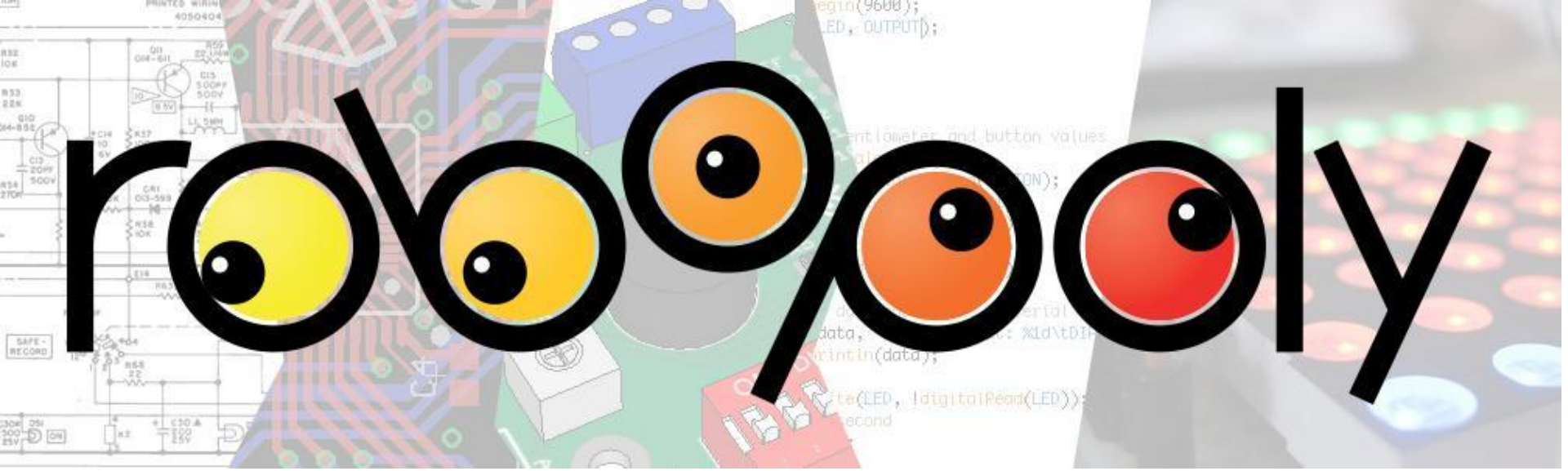
Le mode *entrée*

- Entrée **digitale**
  - fonction **digitalRead**(**number**)
  - retourne:
    - soit **HIGH (‘1’)** pour un pin entre 3V et 5V
    - soit **LOW (‘0’)** pour un pin entre 0V et 2V
    - Entre 2 et 3V: On ne peut pas savoir!
- Entrée **analogique**
  - fonction **analogRead**(**number**)
  - retourne une valeur **entre 0 et 1023** pour une tension entre 0V et 5V

Le mode *sortie*

- Sortie **digitale**
  - permet d'appliquer une tension (soit 0V, soit 5V) sur un pin spécifique
  - fonction **digitalWrite**(**number**, **state**)
  - **state** vaut soit **HIGH (5V)** soit **LOW (0V)**

- Sortie **analogique**
  - permet d'appliquer un signal dont la valeur moyenne est **entre** 0V et 5V
  - fonction **analogWrite**(**numero**, **valeur**)
  - **valeur** entre 0 et 255 (255 correspond à 5V)



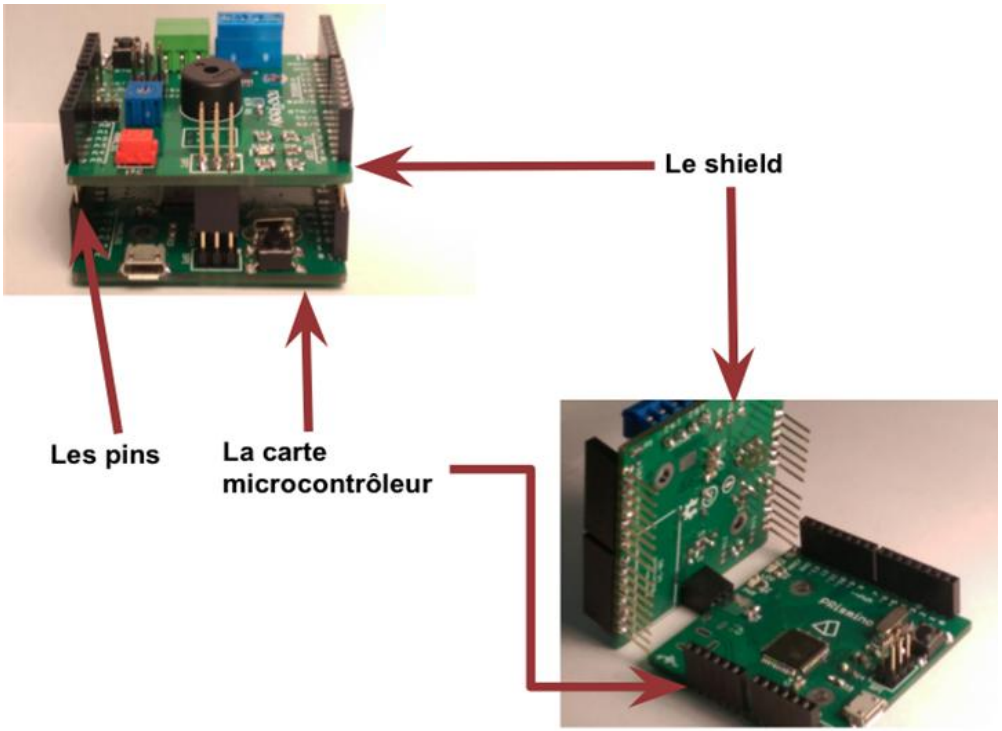
Les entrées/sorties

Module Shield

Programmation

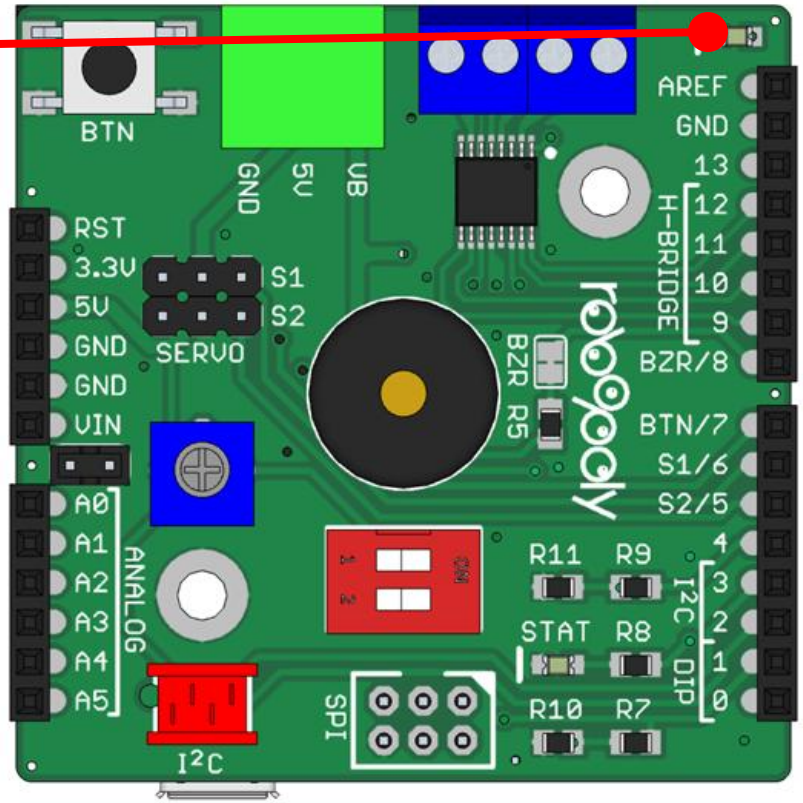
Qu’est-ce que c’est ?

- Une carte d’interface regroupant différents connecteurs et éléments de base
- Se branche directement sur le PRismino



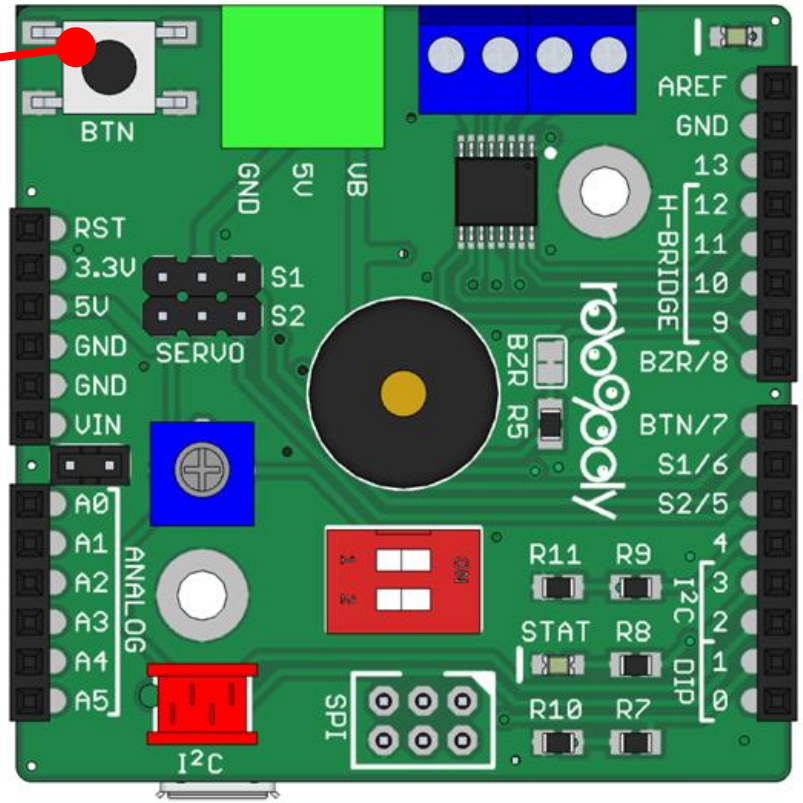
Les éléments du shield

- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules



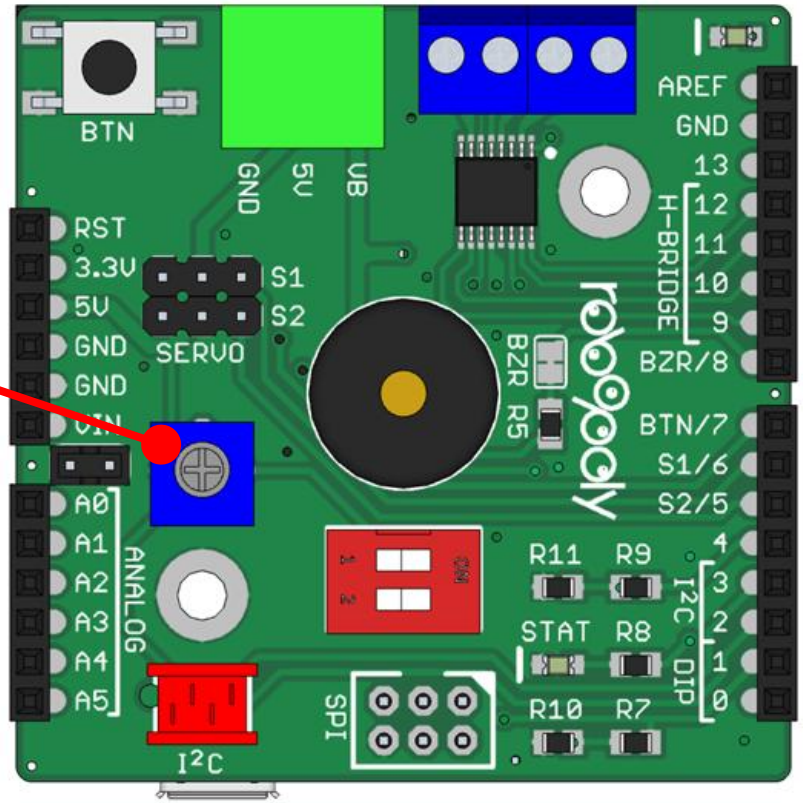
Les éléments du shield

- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules



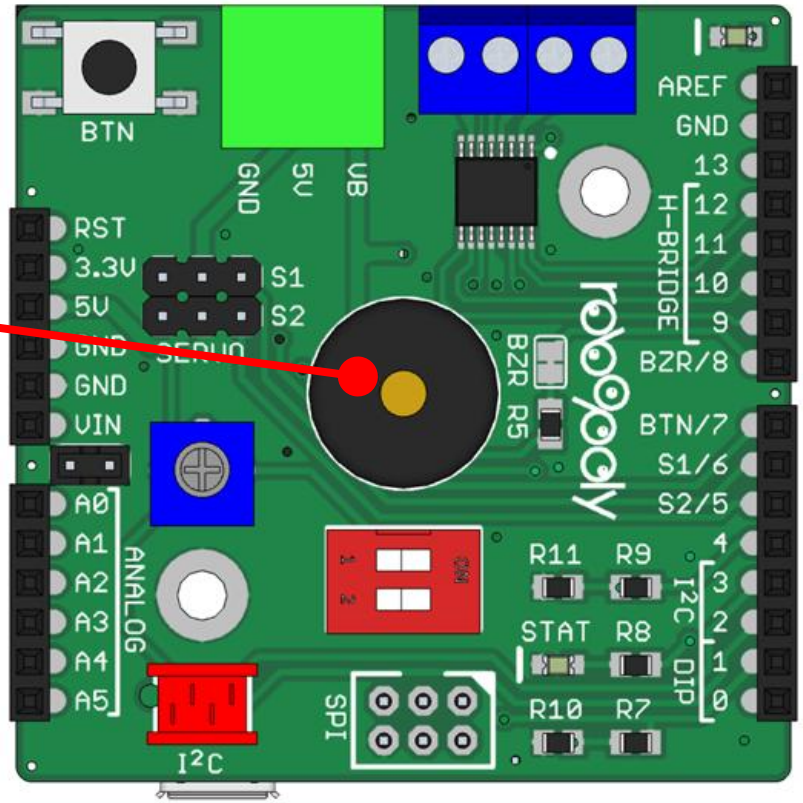
Les éléments du shield

- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules



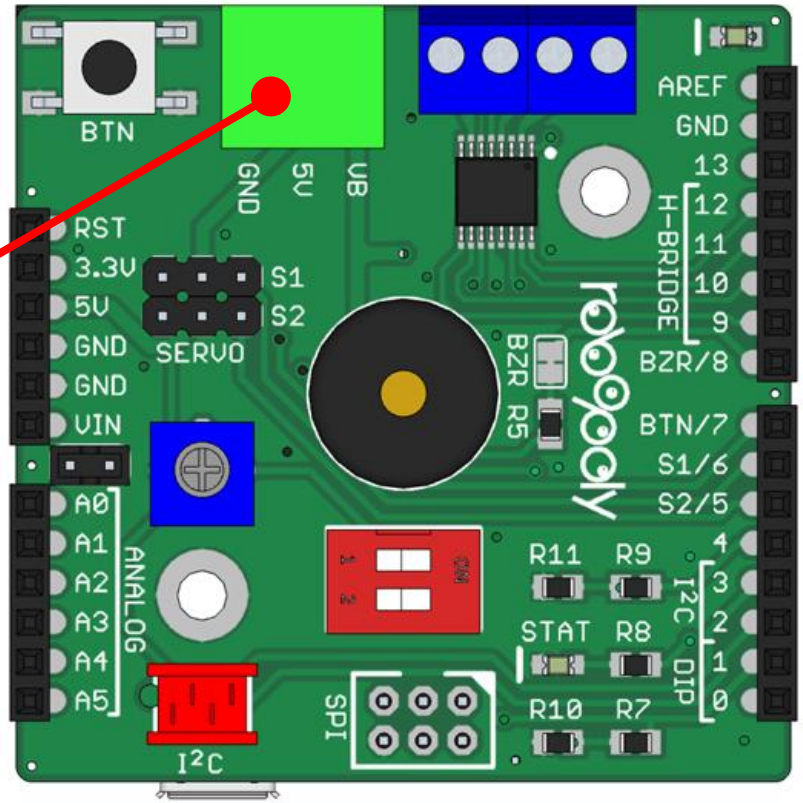
Les éléments du shield

- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules



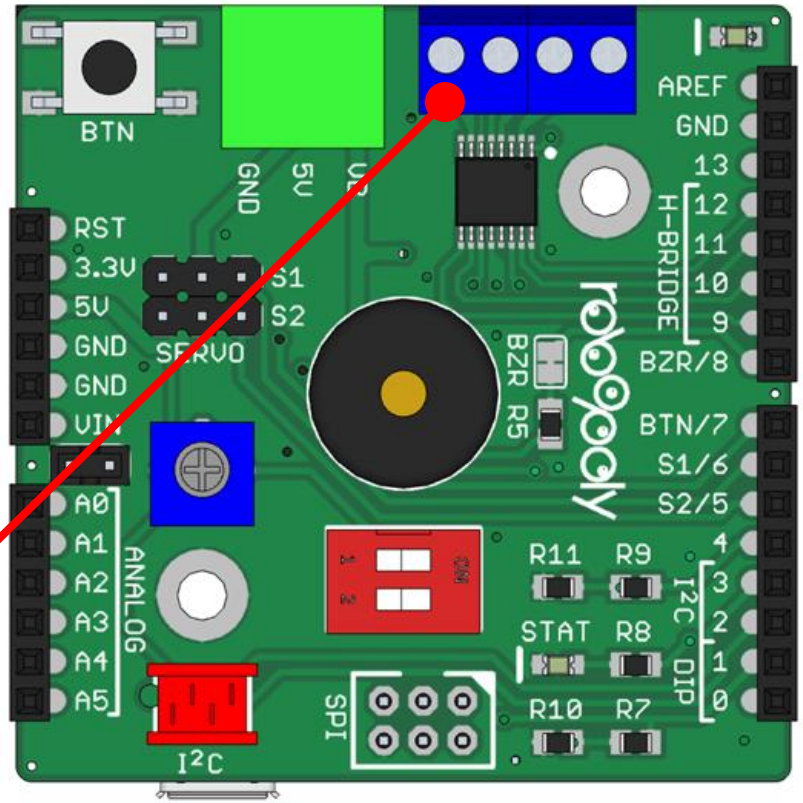
Les éléments du shield

- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules



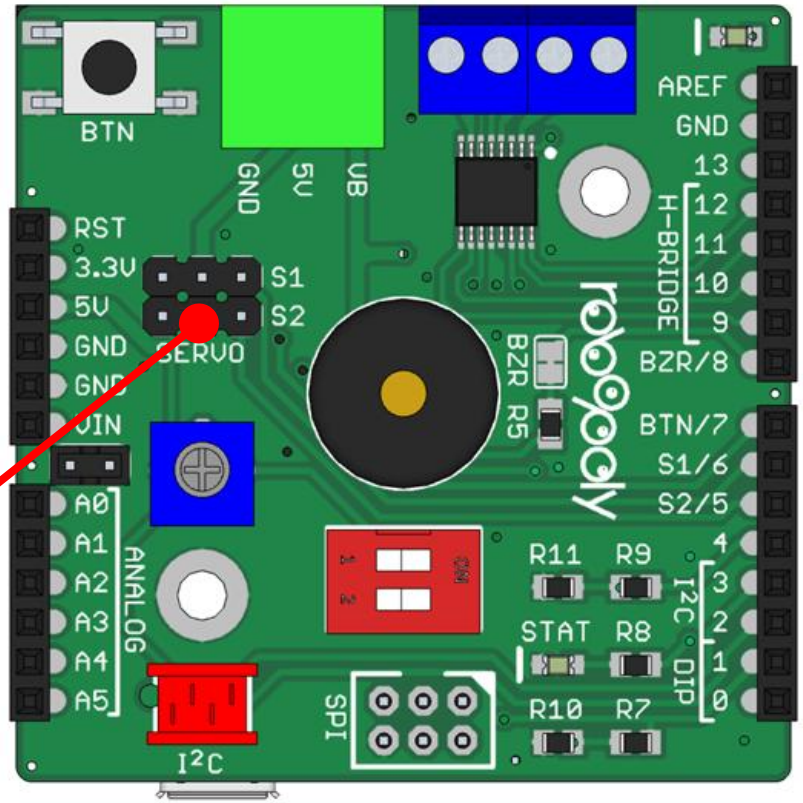
Les éléments du shield

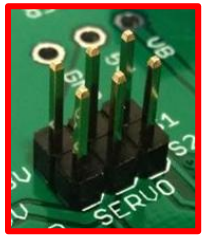
- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules



Les éléments du shield

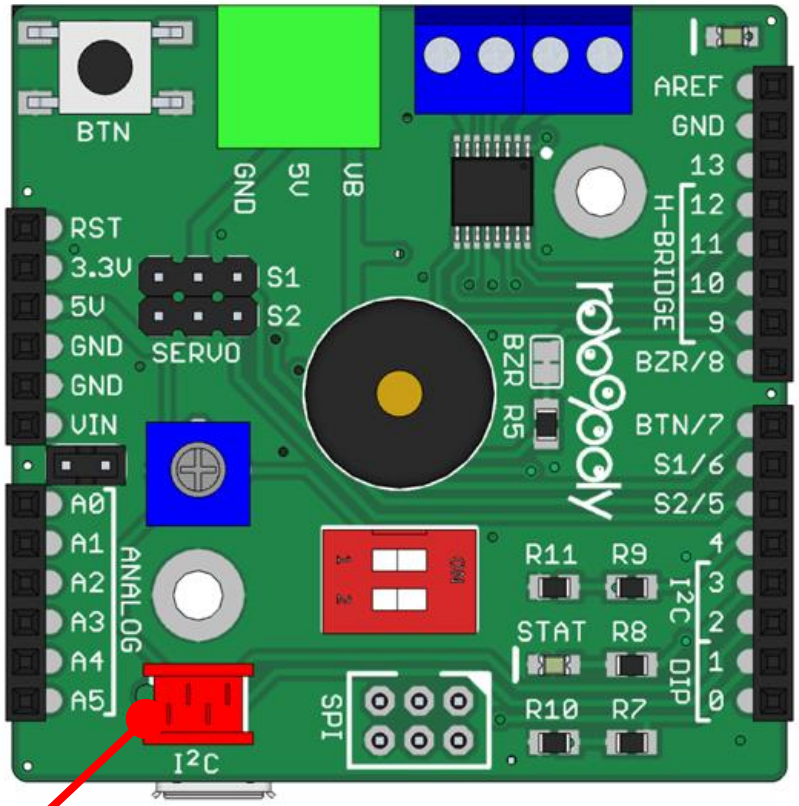
- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules

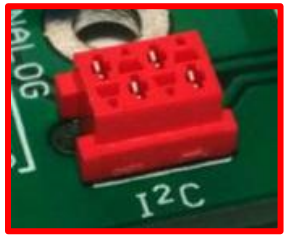




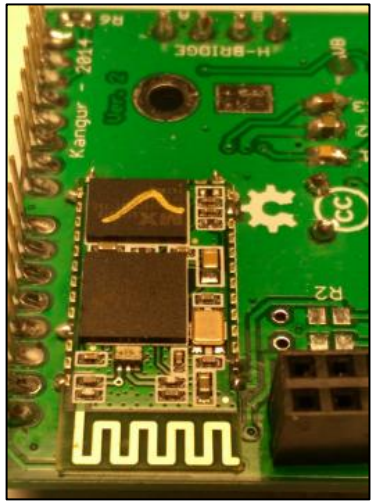
Les éléments du shield

- LED
- Bouton
- Potentiomètre
- Buzzer (pour jouer un son)
- Connecteur vers la PowerBoard
- Connecteur vers les moteurs
- Connecteurs vers les servomoteurs
- Connecteur de communication avec d'autres modules

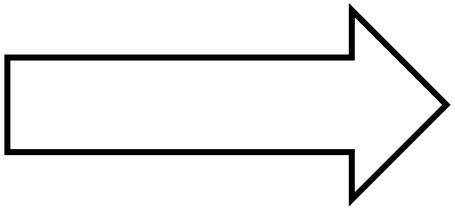




Module Bluetooth



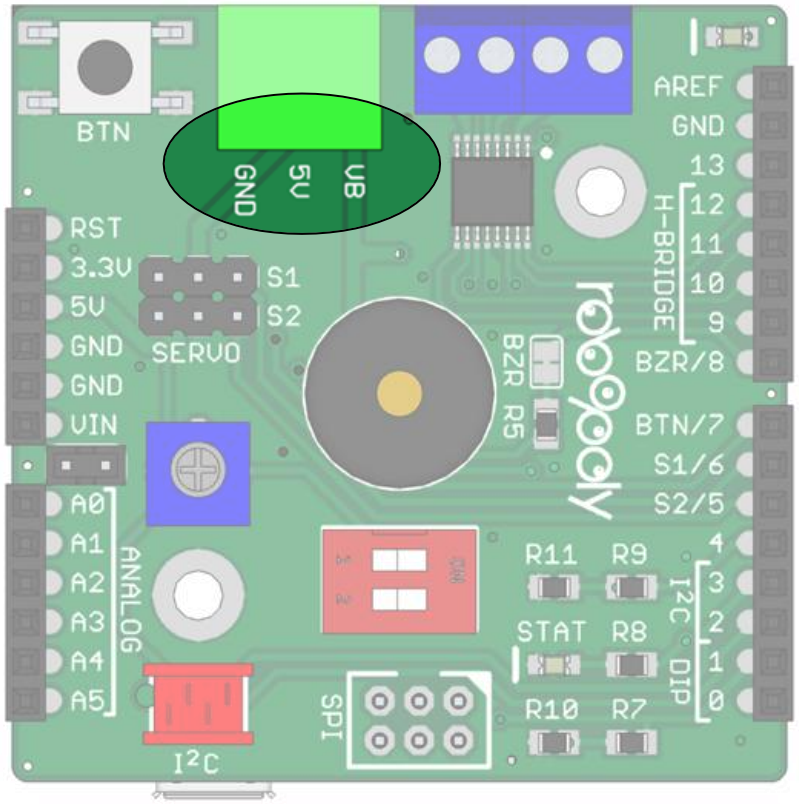
- Peut être soudé derrière le shield
- Ne peut pas être soudé en même temps que le DIP switch

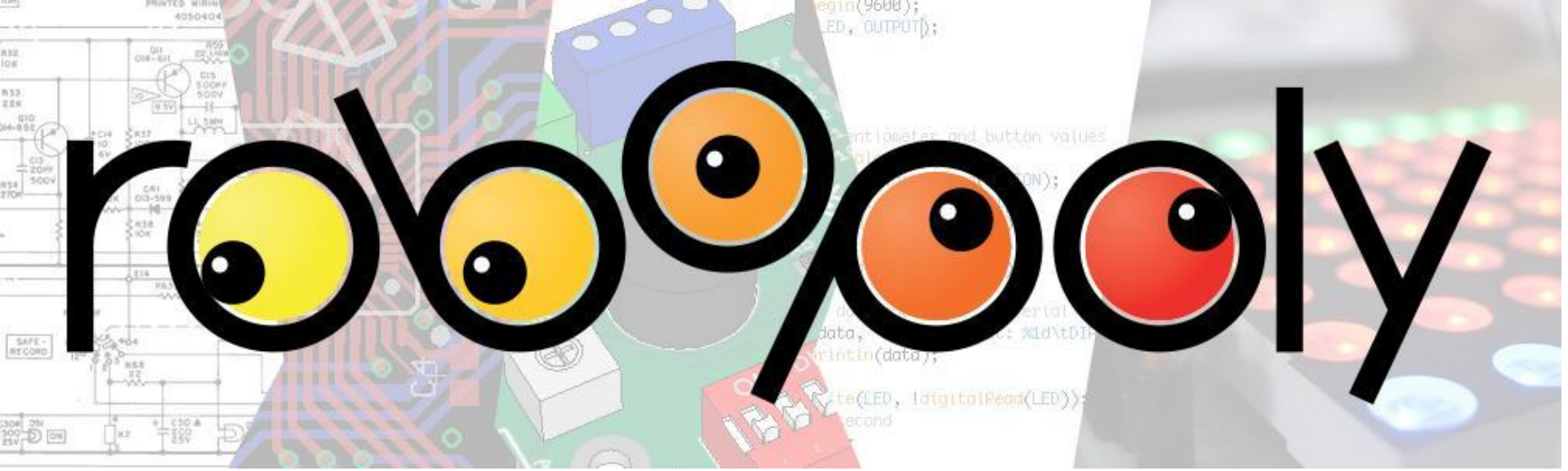


- Son utilisation sera abordée dans un futur démon

Quelques notions de terminologie...

- Les différentes tensions utilisées dans le kit:
  - **GND** = Ground = référence (correspond au 0V)
  - **5V = VCC** = la tension d'alimentation du uC
  - **VB** = la tension des batteries





Les entrées/sorties

Module Shield

Programmation

Structure de base

- Structure de base commune à tous les programmes

```
void setup()
{

}

void loop()
{

}
```

Fonction exécutée une seule fois, au lancement du programme

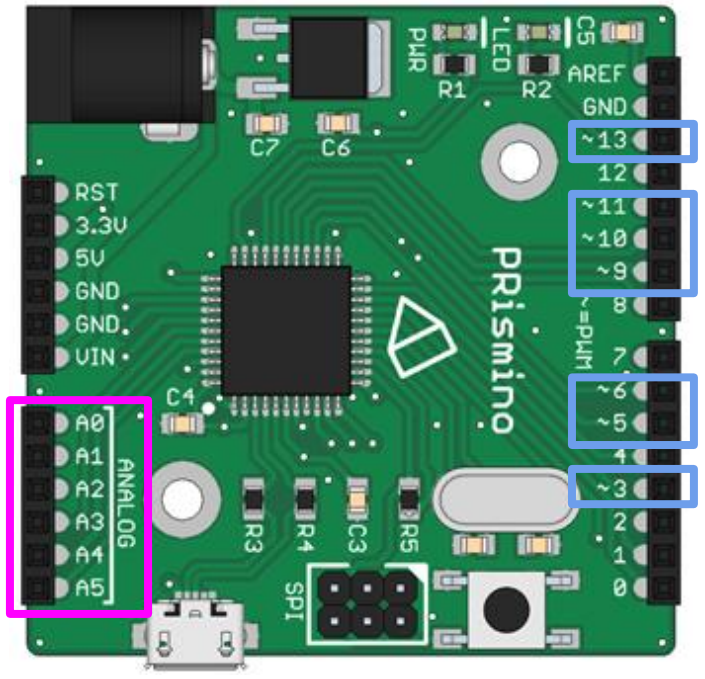
Fonction exécutée en boucle : c'est le code principal

- Pour utiliser la bibliothèque Robopoly, insérer au début du fichier: **#include <prismino.h>**

La numérotation des pins

- **Tous les pins GPIO** peuvent être en entrée/sortie **digitale**
  - de **0 à 13** et de **A0 à A5**

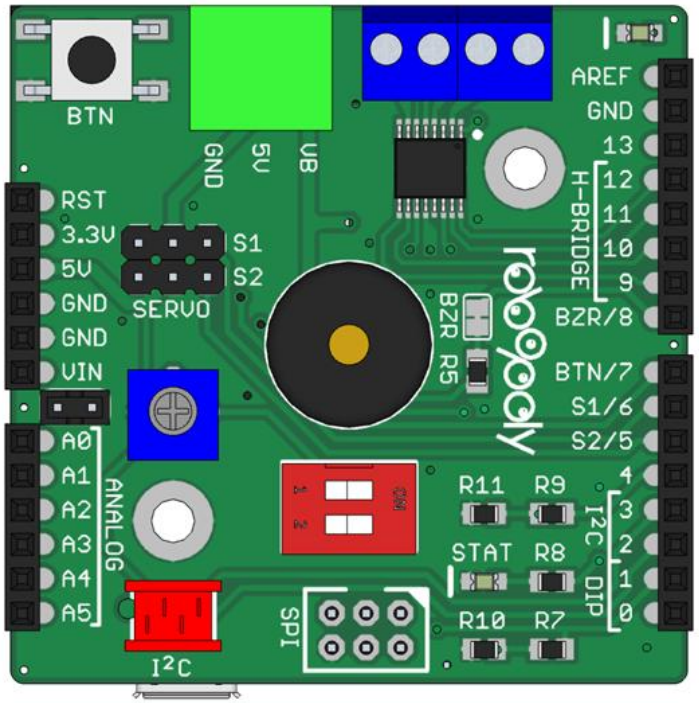
- **Seuls certains pins** peuvent être en entrée ou sortie **analogique**
  - **entrée** analogique: **A0 à A5**
  - **sortie** analogique: **3, 4, 5, 6, 9, 10, 11, 13**



- exemple: **pinMode**(4, **INPUT**);

La connection des éléments du shield

- Chaque élément du Shield est **assigné à un pin du PRismino**:
  - DIP Switch 0 et 1
  - Bouton 7
  - Servomoteurs 5 et 6
  - Buzzer 8
  - Moteurs 9, 10, 11, 12
  - LED 13
  - Potentiomètre A0



- On ne peut pas utiliser en même temps un pin et l'élément du Shield qui lui est associé

La connection des éléments du shield

- Pas besoin de se souvenir des numéros de pins par coeur! La bibliothèque Robopoly contient les définitions suivantes:

Elément	N° de pin	Def
DIP switch 1	0	DIP1
DIP switch 2	1	DIP2
Servo n°1	5	S1
Servo n°2	6	S2

Elément	N° de pin	Def
Bouton	7	BTN
Buzzer	8	BZR
LED	13	LED
Potentiomètre	A0	POT

ex: digitalWrite(**LED**, HIGH) est équivalent à digitalWrite(**13**, HIGH)

LED

- On peut contrôler l'état de la LED
  - **digitalWrite(LED, LOW);** pour l'éteindre
  - **digitalWrite(LED, HIGH);** pour l'allumer

Switch

- On peut lire la valeur du DIP switch 1 avec **digitalRead(DIP1)** qui renvoie 0 ou 1

ATTENTION: **toujours** définir le mode (entrée ou sortie) des pins que l'on va utiliser.

Le cas du bouton

- Intuitivement, on le brancherait de cette manière:

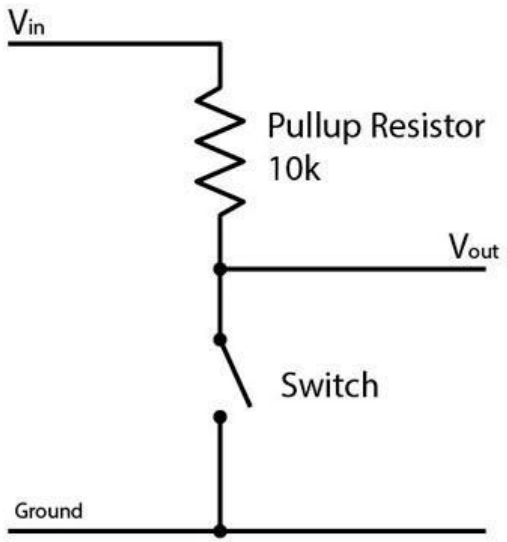
- Mais… quand le bouton n'est pas pressé, l'entrée est déconnectée: le pin est dit flottant.

- Or **la tension d'un pin flottant est imprévisible** et peut valoir **n'importe quelle valeur.**

Le cas du bouton

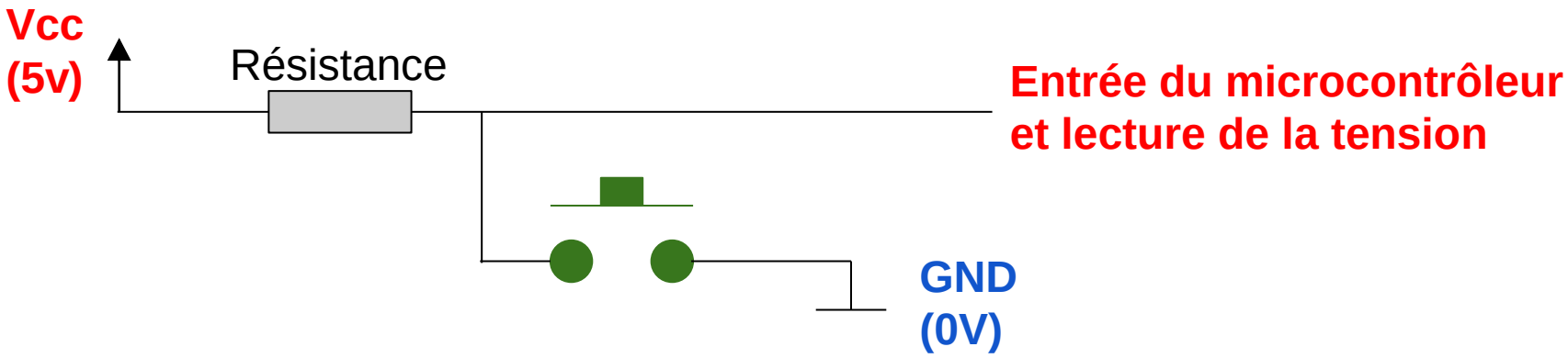
- **La tension d'un pin flottant est imprévisible** et peut valoir **n'importe quelle valeur.**

- On utilise donc une résistance de *pull (tirage)* pour forcer la tension à une valeur connue. La résistance de pull doit avoir une forte valeur pour limiter le courant.

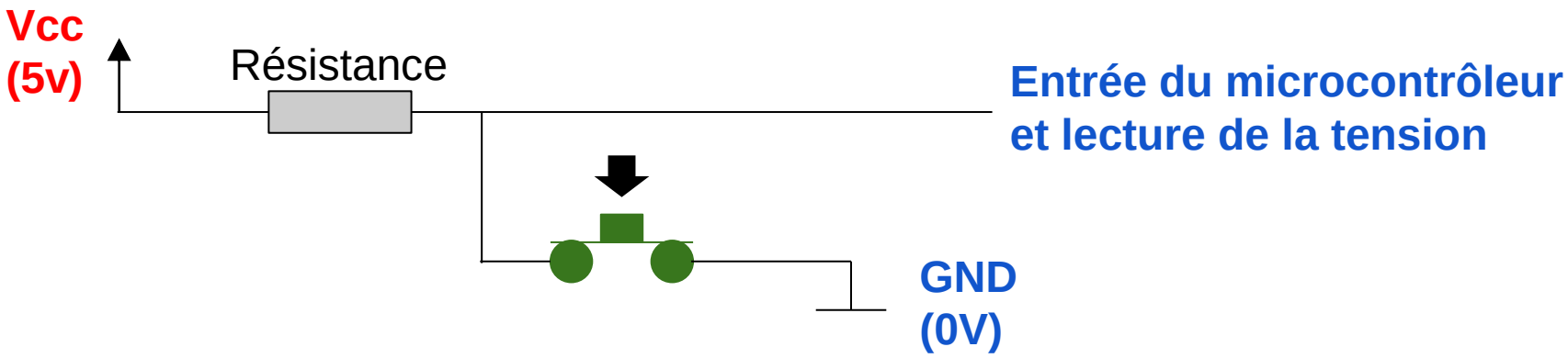


Le cas du bouton

Pull up



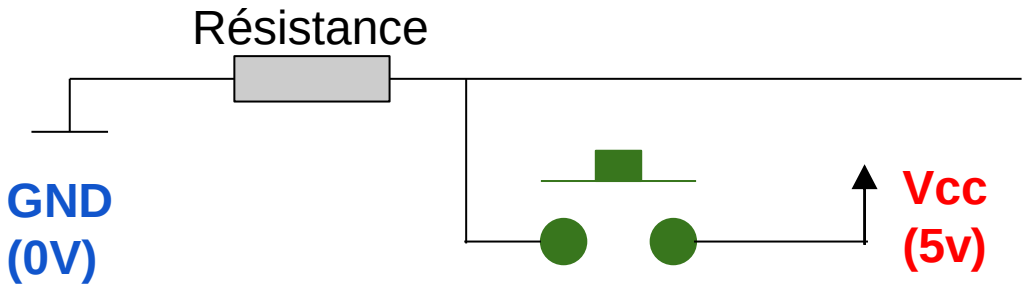
- Bouton relâché: le pin est connecté au **5V**



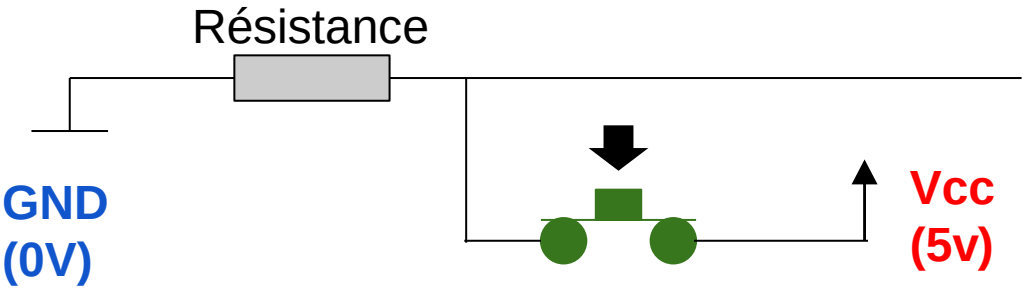
- Bouton pressé: le pin est connecté au **0V**

Le cas du bouton

Pull down



- Bouton relâché: le pin est connecté à **0V**.



- Bouton pressé: le pin est connecté à **5V**.

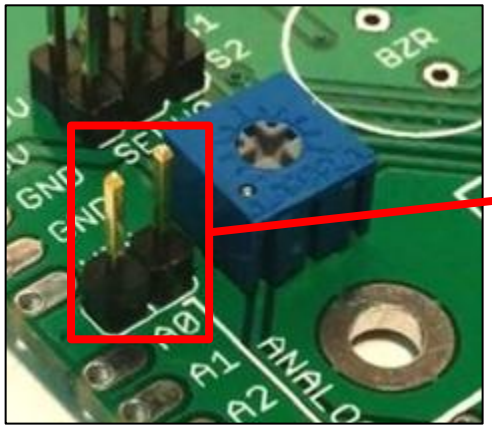
Le cas du bouton

- Le microcontrôleur intègre ce système de pull-up
- Cela peut être activé lorsqu'on définit le mode du bouton: **pinMode**(**BTN**, **INPUT_PULLUP**);

- La lecture de la valeur du bouton est effectuée par une lecture digitale: **digitalRead**(**BTN**);
- Cet appel renvoie **1 si le bouton est relaché et 0 s'il est pressé.**

Potentiomètre

- On fait cette fois-ci une lecture analogique et non digitale: **analogRead**(**POT**);
- Rappel: la fonction renvoie une valeur comprise **entre 0 et 1023**
- Utile pour régler un paramètre
- Un jumper doit être placé sur le Shield pour pouvoir utiliser le potentiomètre

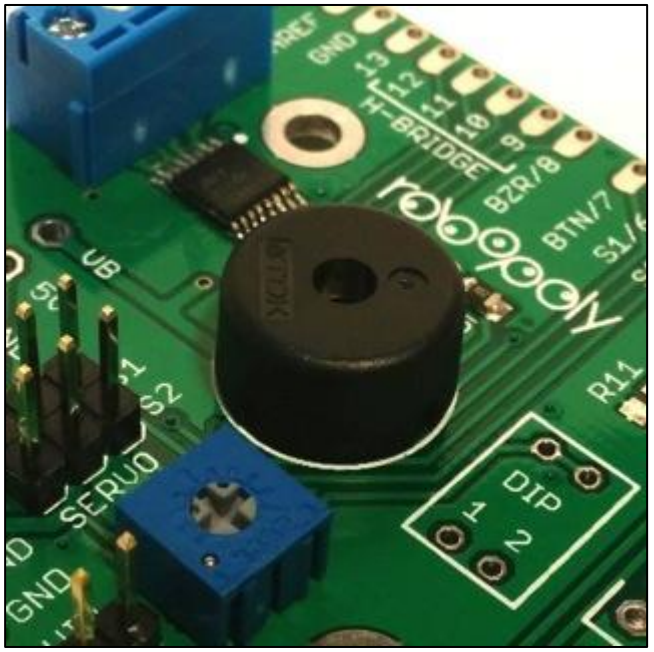


Placer le jumper ici: il fait le lien entre les deux pins

Buzzer

- On contrôle la fréquence sonore en utilisant la fonction **play(frequency, duration);**
  - **frequency** en Hertz
  - **duration** en millisecondes

- Utile pour signaler un évènement dans l'exécution du programme (problème, rencontre d'obstacle par ex.)



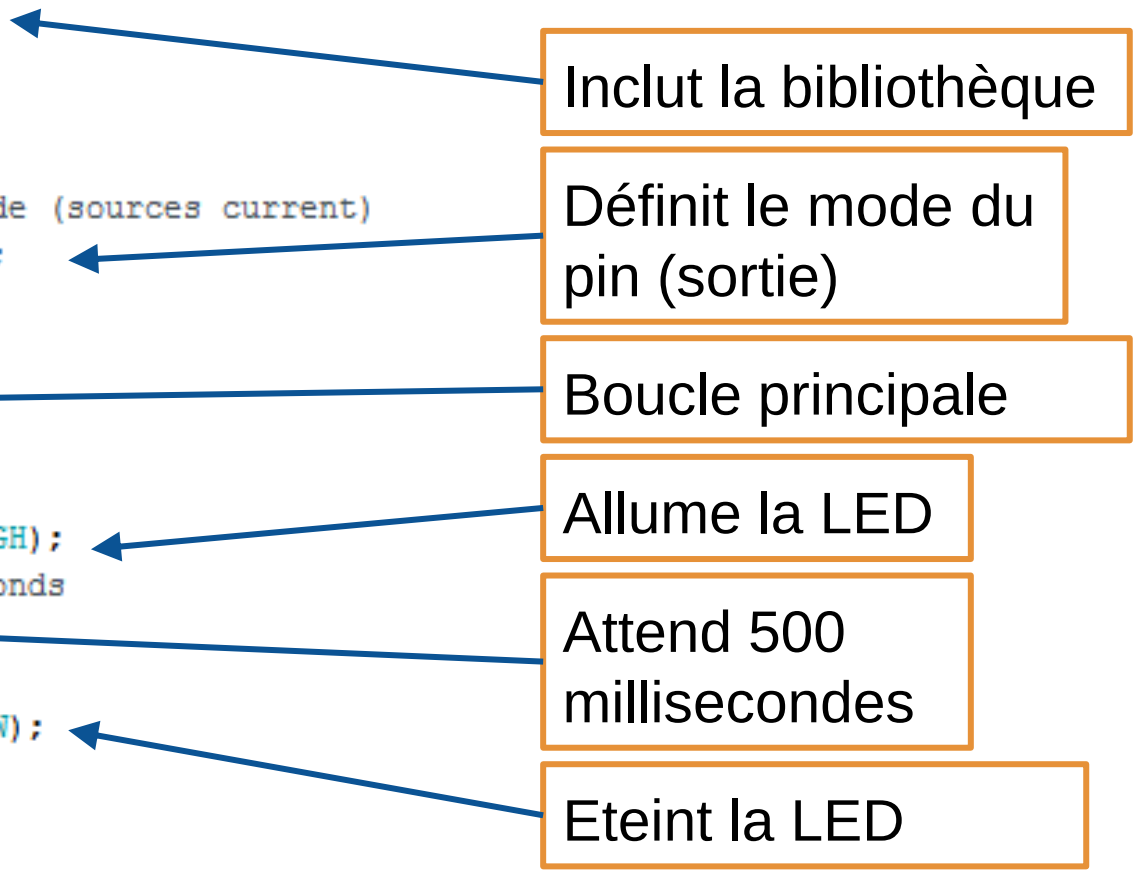
Le programme de base: Blink

- Faire clignoter une LED

```
#include <prismino.h>

void setup()
{
  // set pin output mode (sources current)
  pinMode(LED, OUTPUT);
}

void loop()
{
  // turn LED on
  digitalWrite(LED, HIGH);
  // wait 500 milliseconds
  delay(500);
  // turn LED off
  digitalWrite(LED, LOW);
  delay(500);
}
```



A vous de jouer

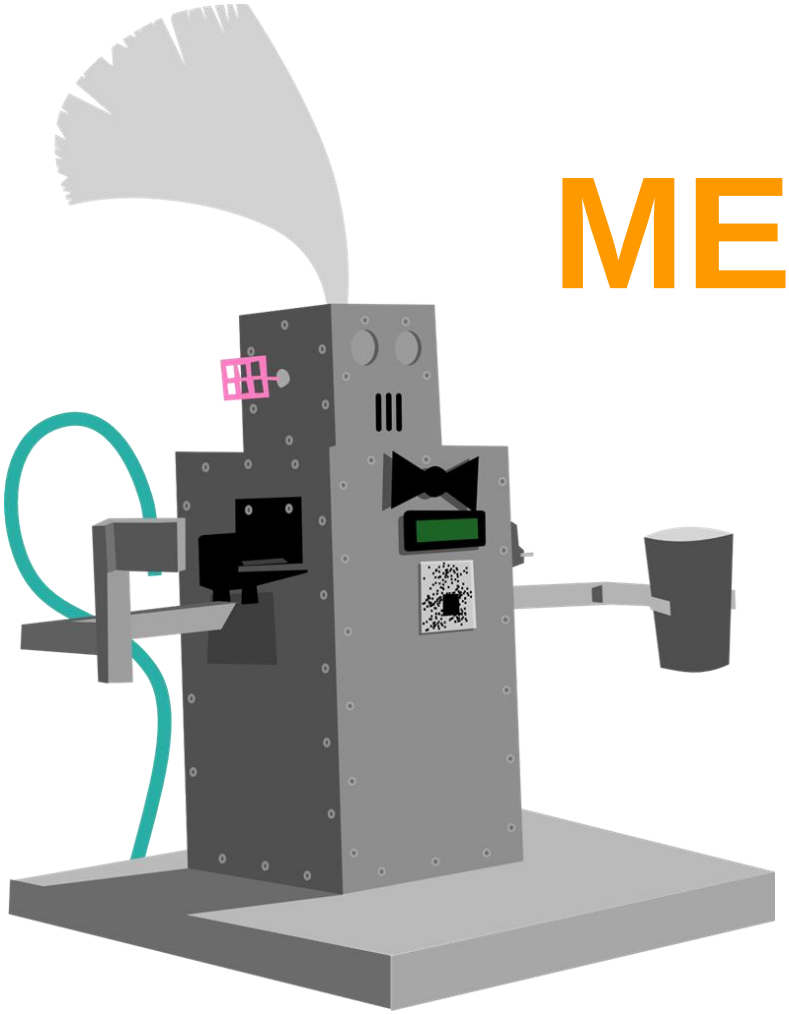
- Petits exercices:
  - Utiliser le **bouton** ou le **DIP switch** pour contrôler la **LED**
  - Augmenter la fréquence de clignotement de la **LED** ou la fréquence du **buzzer** avec le **potentiomètre**
  - Compter le nombre de fois que le **bouton** est pressé et agir sur le clignotement

Workshop

Prochains évènements

- **Prochain Démon**
  - Lundi prochain, 12h15, ELA1
  - La Powerboard et les moteurs

- **Worshop II**
  - Samedi 11 Novembre, 9h-18h
  - En haut du BM!
  - Pour poursuivre ou commencer le montage de ton robot, discuter de tes projets, demander de l'aide au comité!



MERCI!

Questions?