



roboonly

Les entrées/sorties

Module Shield

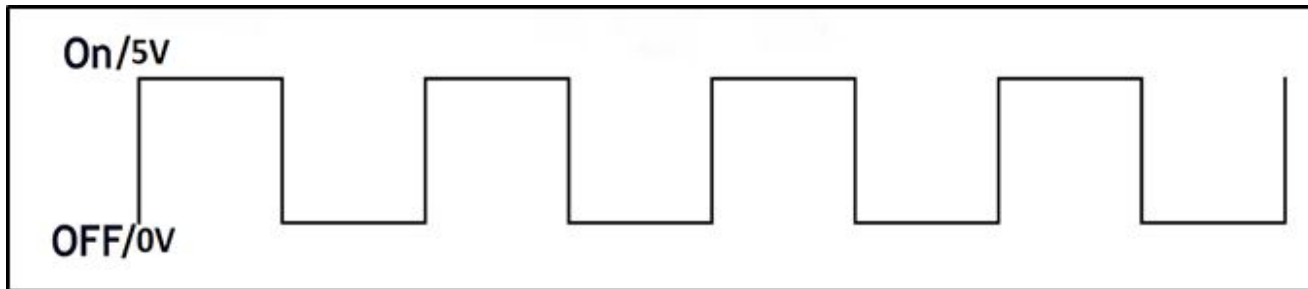
Programmation

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

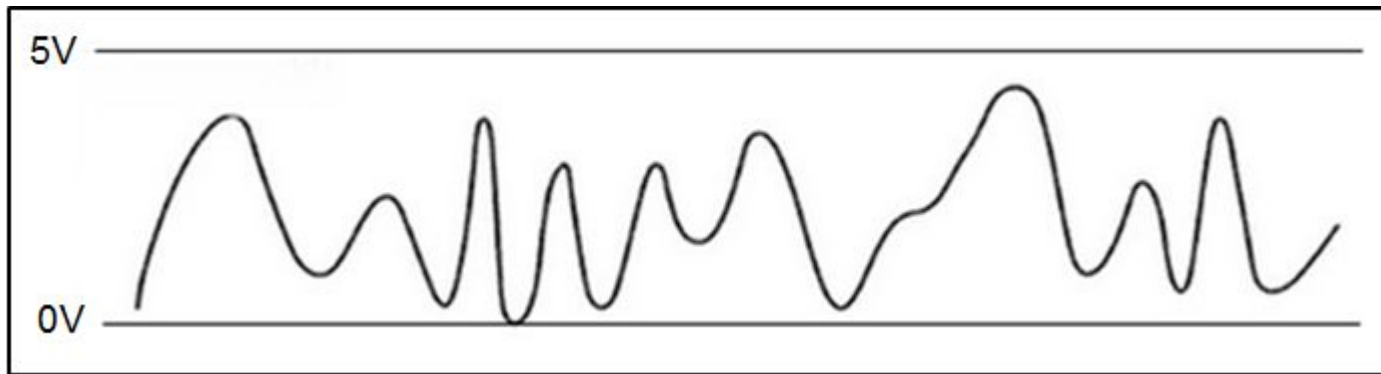
- Le signal est transmis par les valeurs de tension:
 - '0' correspond à 0V
 - '1' correspond en général à 5V



- La tension d'un signal digital **vaut soit '0', soit '1'**, mais **pas autre chose!**
La valeur de la tension est donc **discrète**.
- Exemple: bouton → soit **relaché**, soit **pressé**
LED → soit **allumée**, soit **éteinte**

Signal analogique

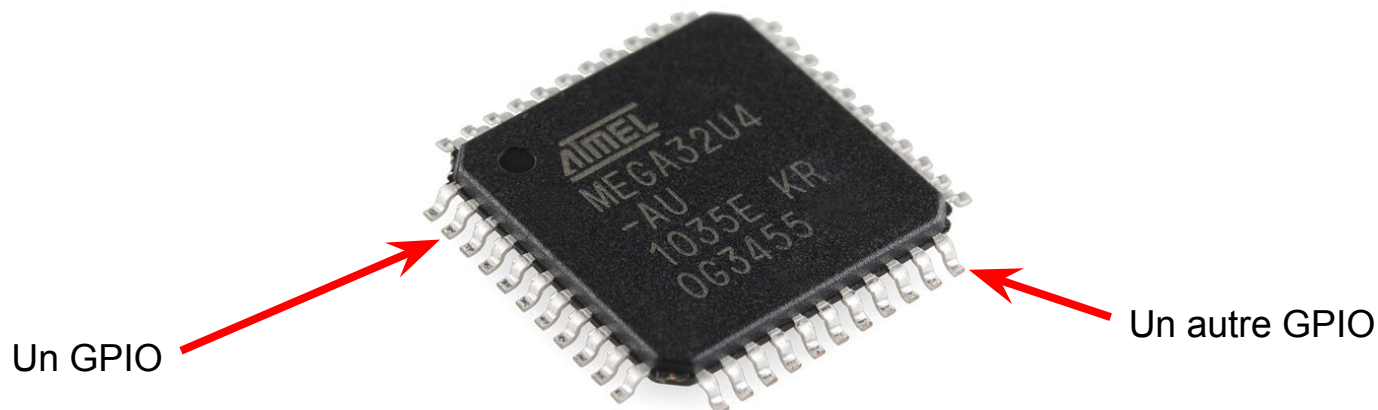
- Au contraire, un signal analogique peut **librement varier** entre 0V et par exemple 5V.



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

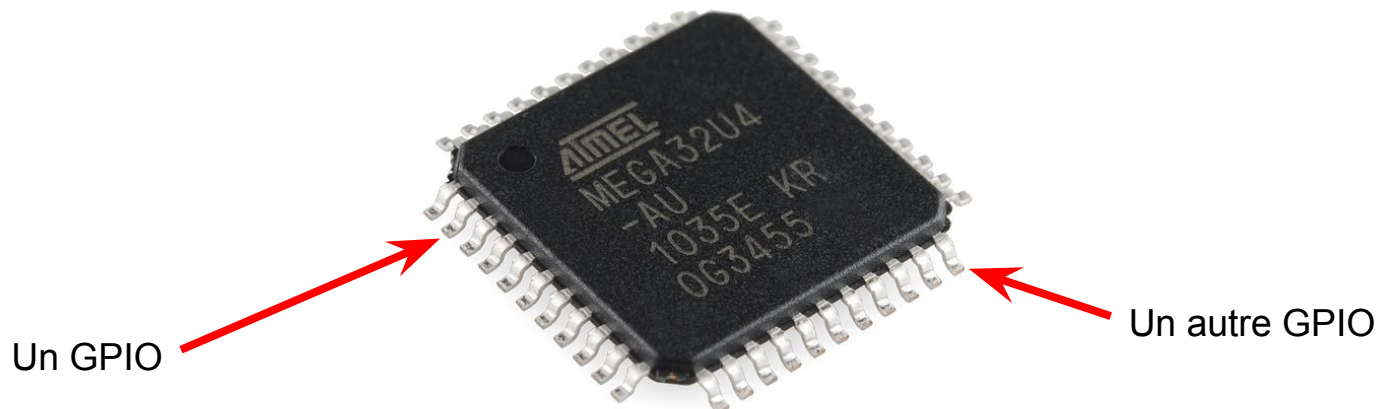
Les pins du microcontrôleur

- Certaines pattes du microcontrôleur correspondent à des **GPIO** (General Purpose Input Output)
- Une 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(numero, mode)**
 - **numero** correspond au numéro du pin en question (on verra les numéros plus loin)
 - **mode** vaut soit **INPUT** soit **OUTPUT**



Le mode *entrée*

- Entrée **digitale**
 - permet de lire la tension sur le pin en question
 - fonction **digitalRead(numero)**
 - retourne:
 - soit **HIGH ('1')** pour un pin entre 3V et 5V
 - soit **LOW ('0')** pour un pin entre 0V et 2V
- Entrée **analogique**
 - permet de lire la tension sur le pin en question
 - fonction **analogRead(numero)**
 - 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(numero, état)**
 - **état** vaut soit **HIGH (5V)** soit **LOW (0V)**

- Sortie **analogique**
 - permet d'appliquer une tension **entre** 0V et 5V
 - fonction **analogWrite(numero, valeur)**
 - **valeur** entre 0 et 255

roboonly

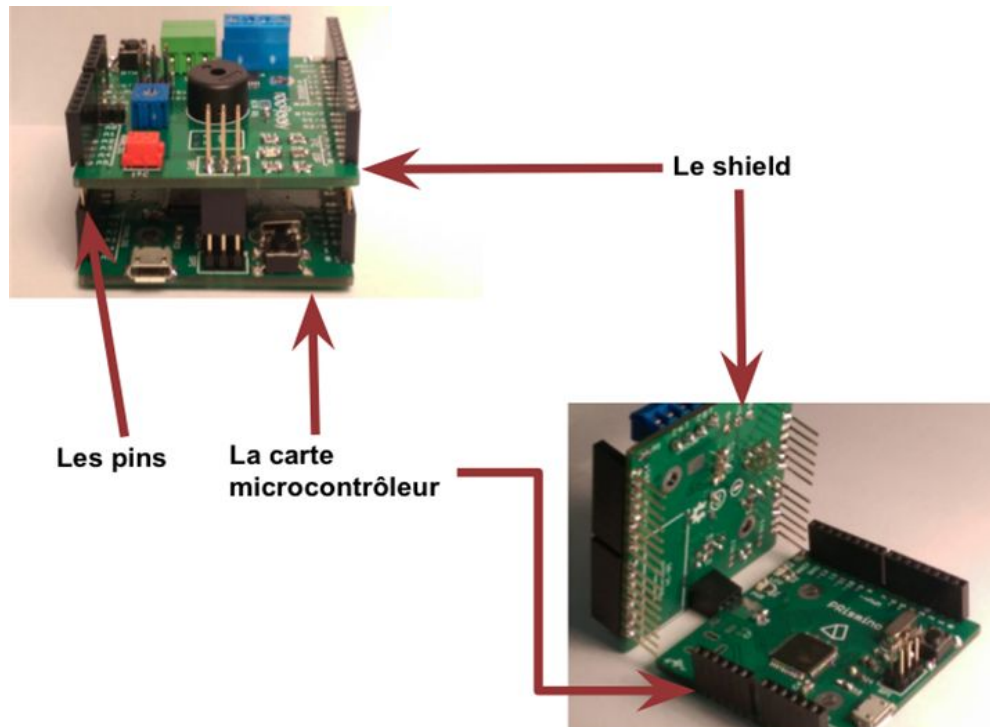
Les entrées/sorties

Module Shield

Programmation

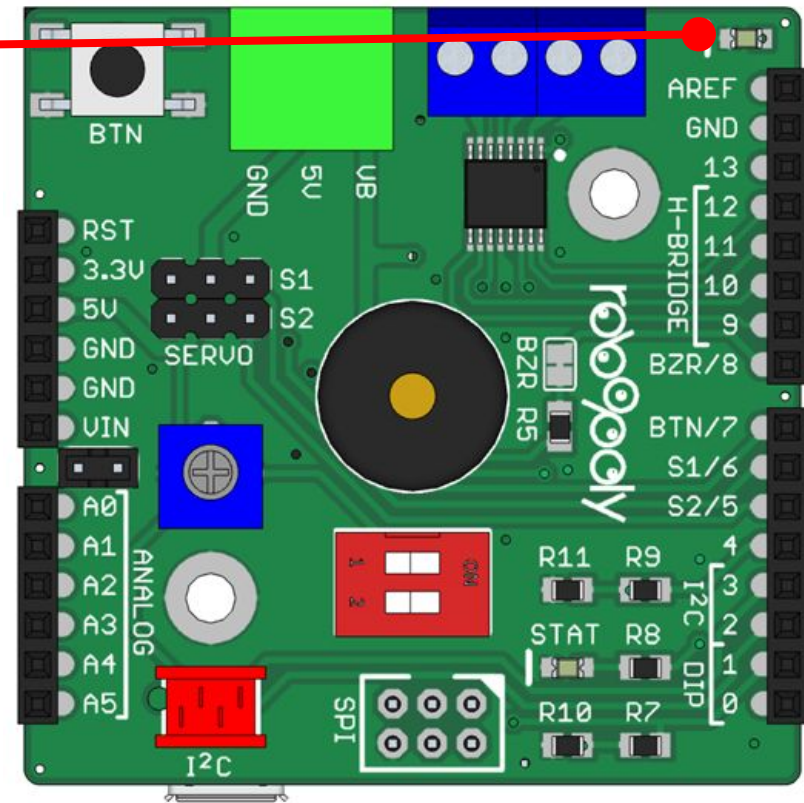
Qu'est-ce que c'est ?

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



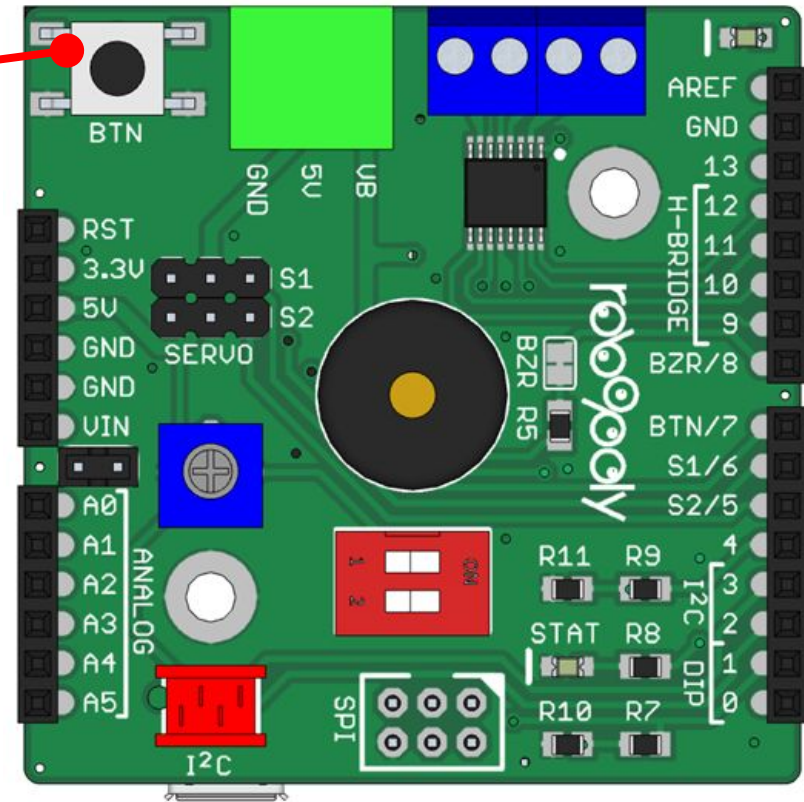
Les éléments du shield

- LED
- Bouton
- Switch
- 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



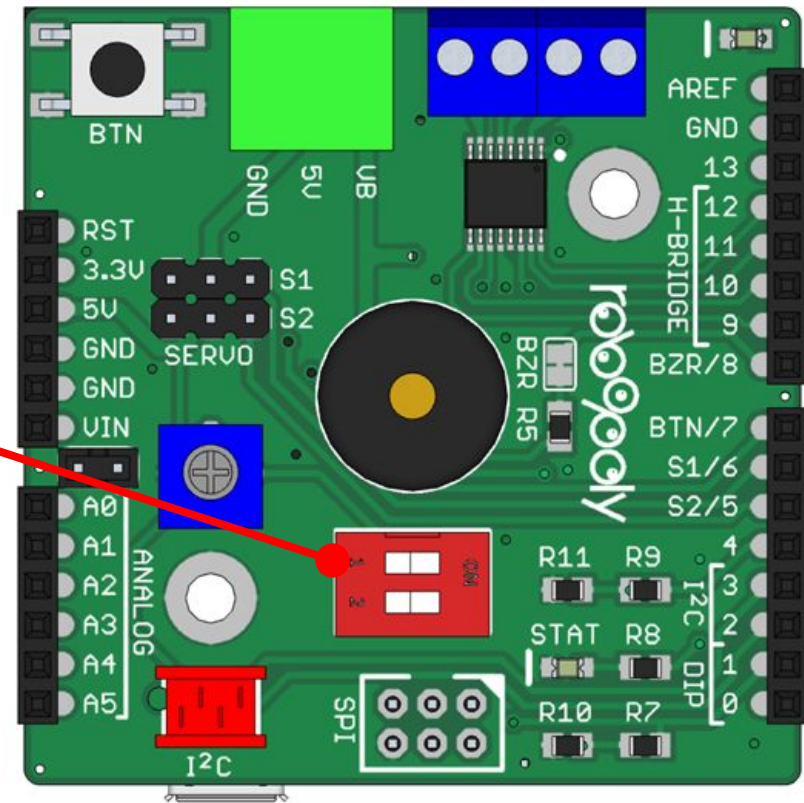
Les éléments du shield

- LED
- **Bouton**
- Switch
- 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



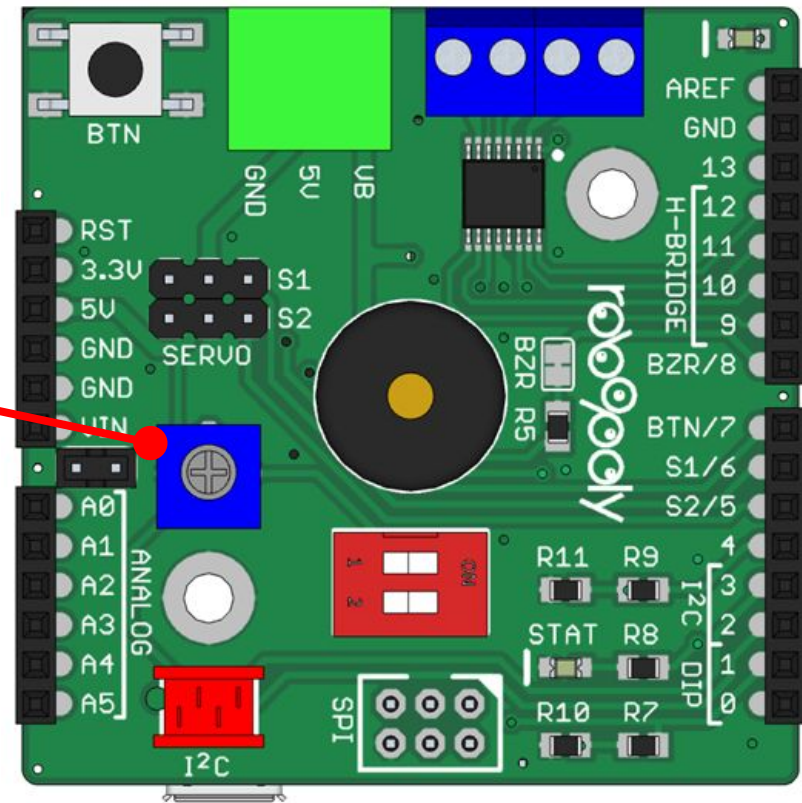
Les éléments du shield

- LED
- Bouton
- **Switch**
- 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



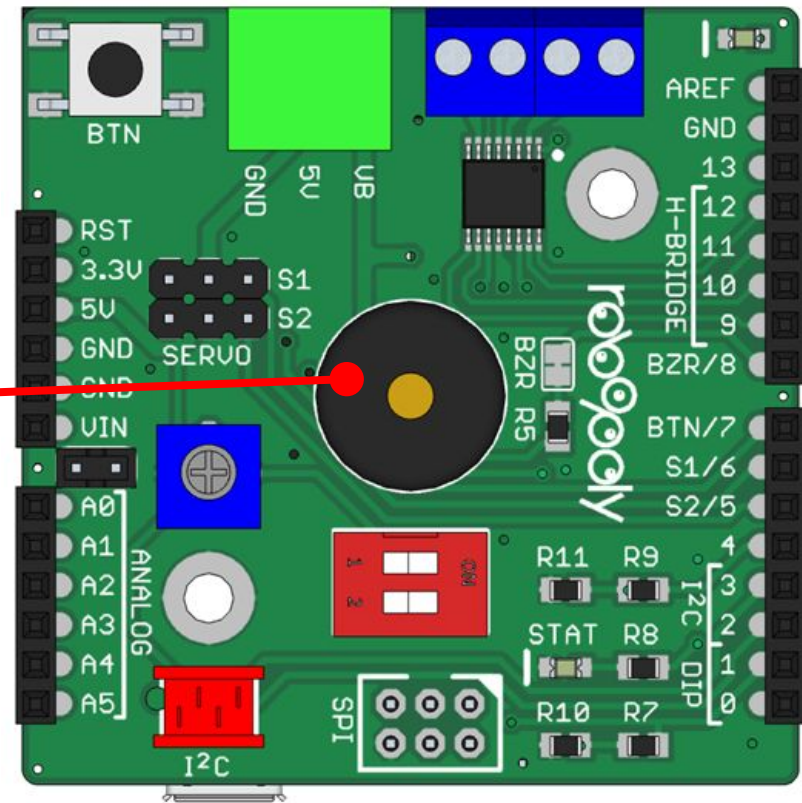
Les éléments du shield

- LED
- Bouton
- Switch
- **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



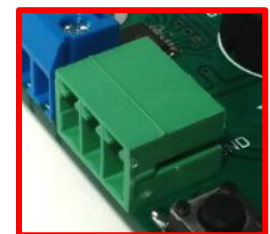
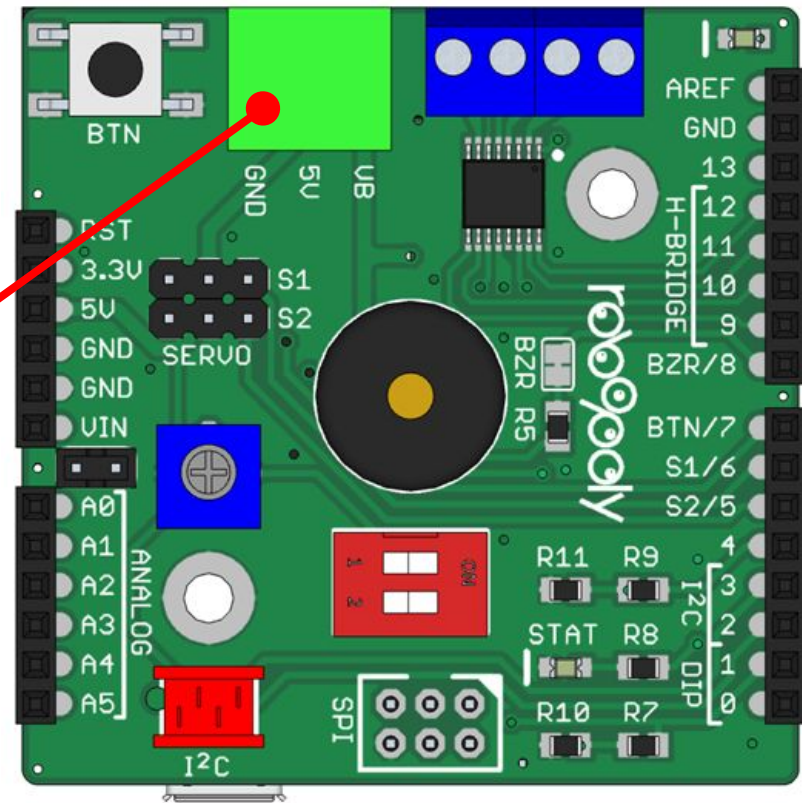
Les éléments du shield

- LED
- Bouton
- Switch
- 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



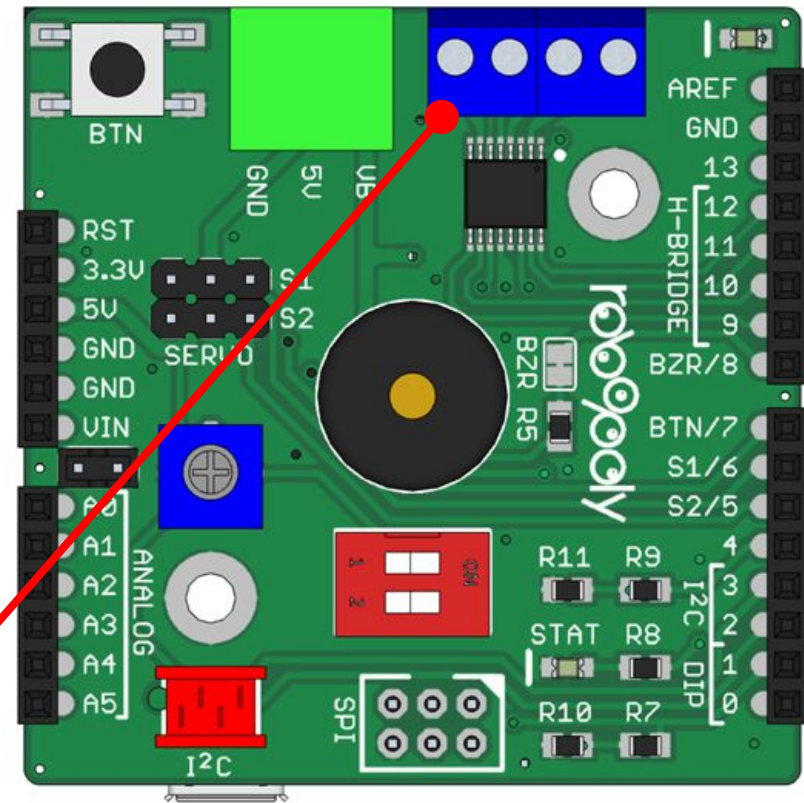
Les éléments du shield

- LED
- Bouton
- Switch
- 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



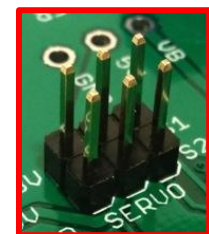
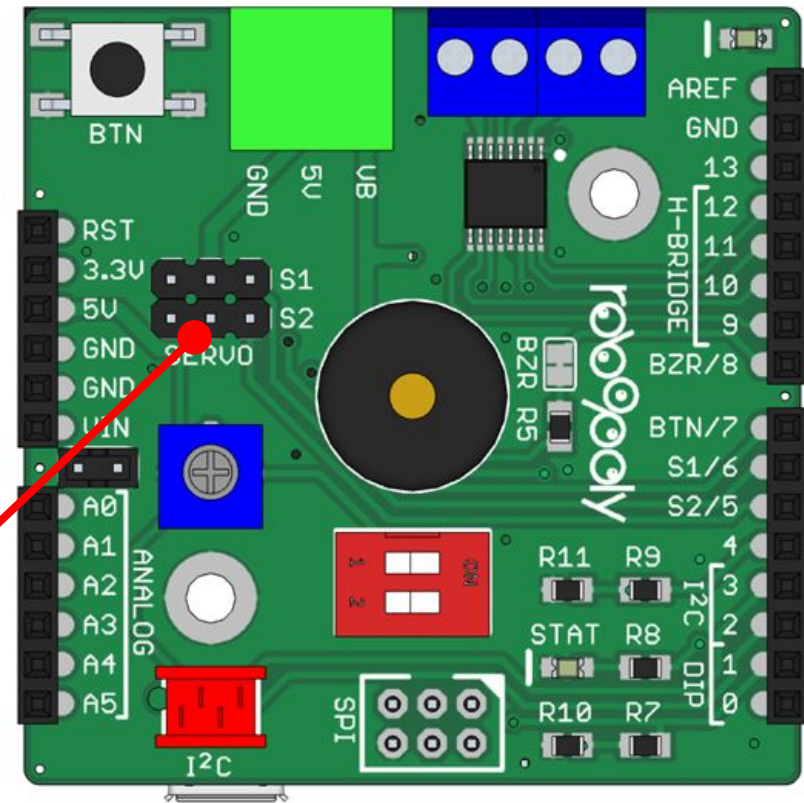
Les éléments du shield

- LED
- Bouton
- Switch
- 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



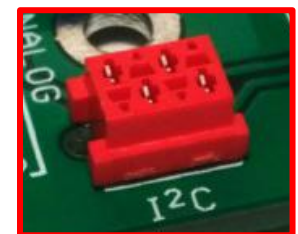
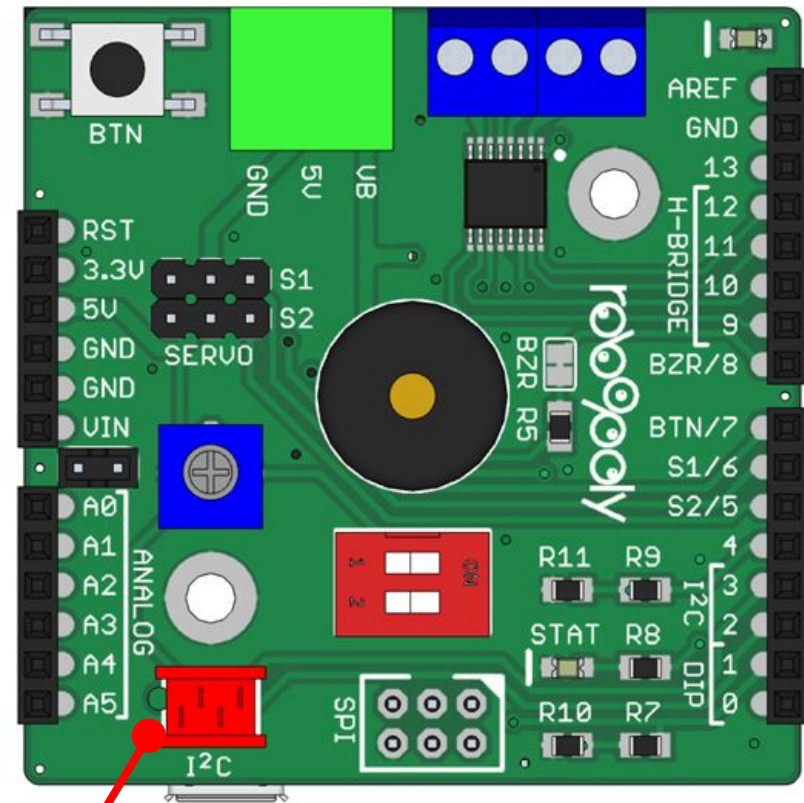
Les éléments du shield

- LED
- Bouton
- Switch
- 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



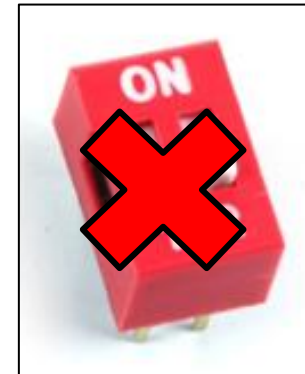
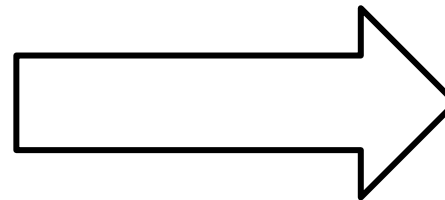
Les éléments du shield

- LED
- Bouton
- Switch
- 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

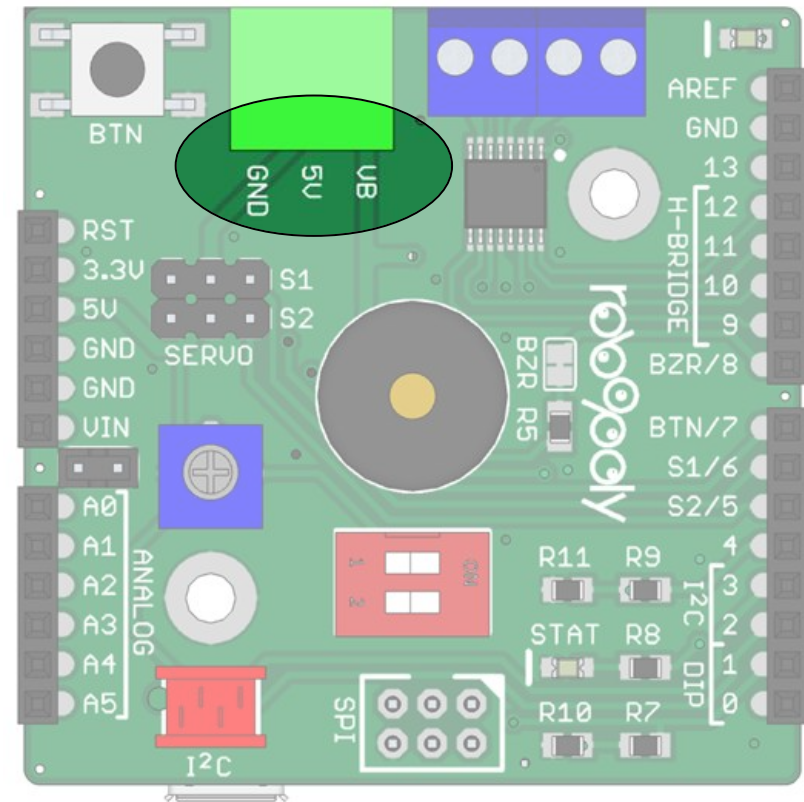
- Peut être soudé derrière le shield
- Ne peut pas être soudé en même temps que le Switch



- Son utilisation sera abordée dans un futur démon

Quelques notions de terminologie...

- Les différentes tensions du microcontrôleur:
 - **GND** = Ground = la terre (correspond au 0V)
 - **5V = VCC** = la tension d'alimentation du uC
 - **VB** = la tension des batteries





roboonly

Les entrées/sorties

Module Shield

Programmation

Structure de base

- Structure de base commune à tous les programmes

```
void setup()
```

```
{  
  
}
```



Fonction exécutée qu'une seule fois, au lancement du programme

```
void loop()
```

```
{  
  
}
```

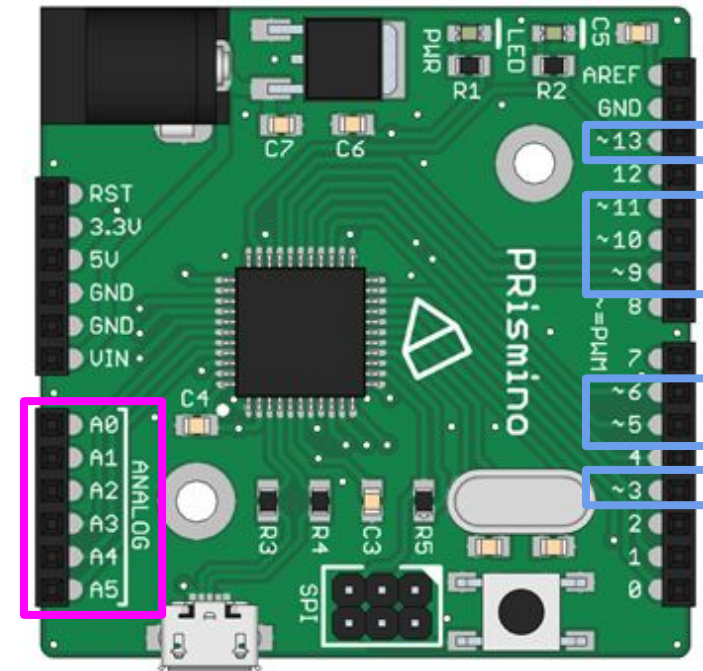


Fonction exécutée en boucle : c'est le code principal

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

La numérotation des pins

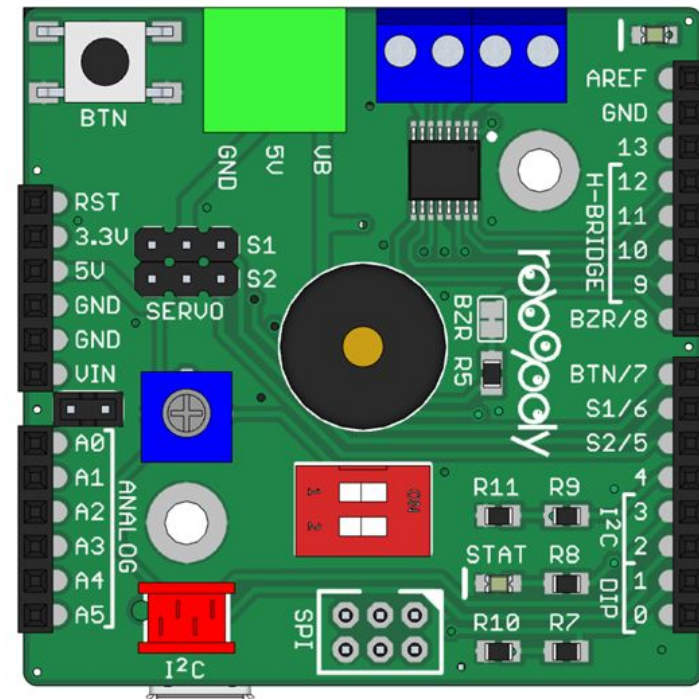
- **Tous les pins GPIO** peuvent être en entrée/sortie **digitale**
 - de **0 à 13** et de **A0 à A5**
- **Seuls certains pins** peuvent être en entrée ou sortie **analogique**
 - **entrée** analogique: **A0 à A5**
 - **sortie** analogique: **3, 4, 5, 6, 9, 10, 11, 13**
- exemple: `pinMode(4, INPUT);`



La connection des éléments du shield

- Chaque élément du Shield est **assigné à un pin du PRismino:**

- 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 connexion des éléments du shield

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

Élément	N° de pin	Macro	Élément	N° de pin	Macro
Switch n°1	0	DIP1	Bouton	7	BTN
Switch n°2	1	DIP2	Buzzer	8	BZR
Servo n°1	5	S1	LED	13	LED
Servo n°2	6	S2	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 switch à tout moment avec `digitalRead(DIP1)`; qui renvoie 0 ou 1

ATTENTION: toujours définir dans le `setup()` le mode (entrée ou sortie) des pins que l'on va utiliser

Le cas du bouton

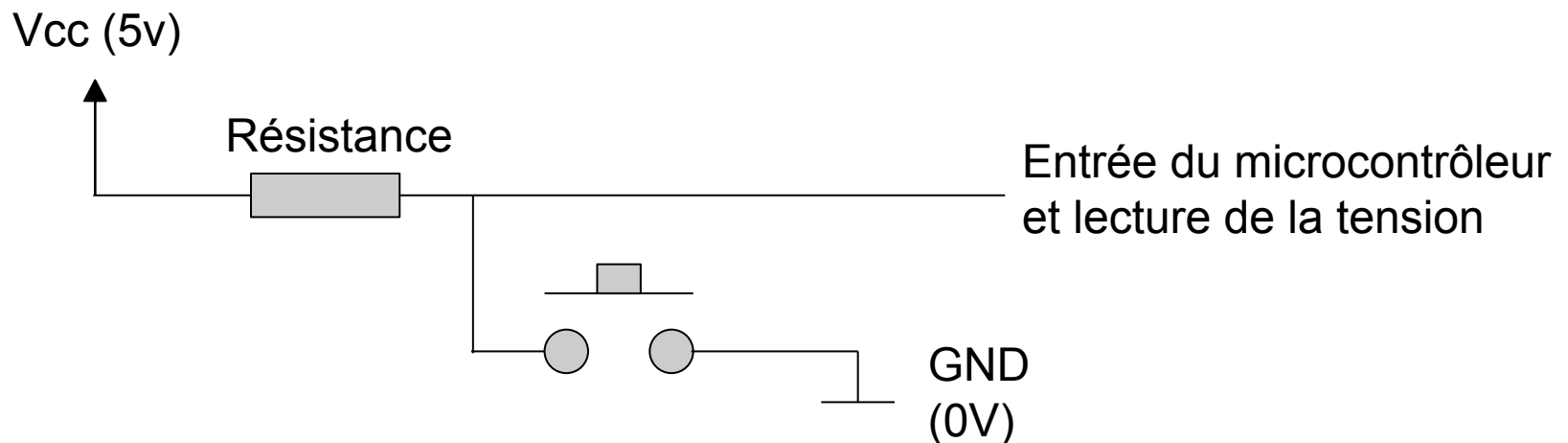
- Intuitivement, on le brancherait de cette manière:



- 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 totalement imprévisible** et peut valoir **n'importe quelle valeur**

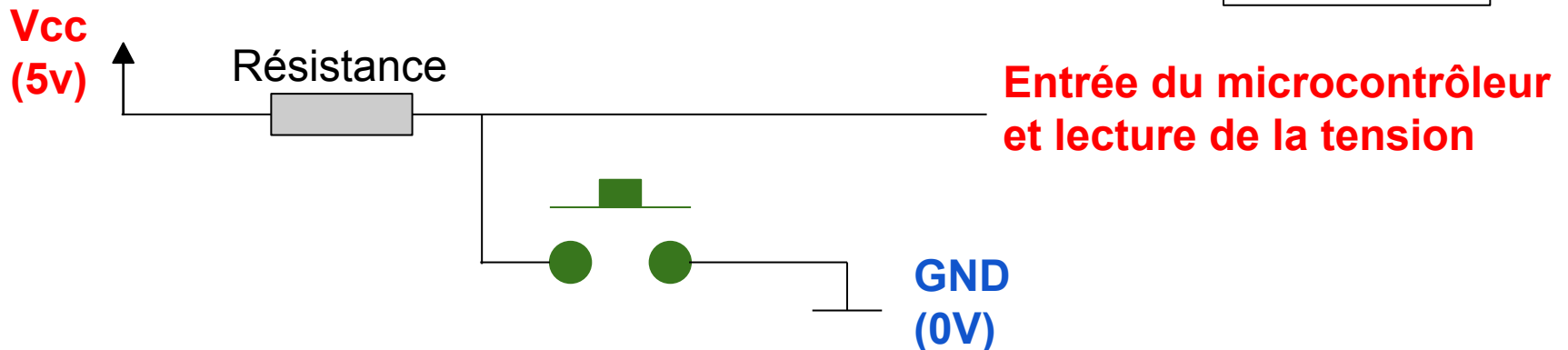
Le cas du bouton

- Or **la tension d'un pin flottant est totalement imprévisible** et peut valoir **n'importe quelle valeur**
- Une lecture de la tension du pin peut donc donner 5V alors que le bouton est relâché...
- On utilise donc le branchement suivant:

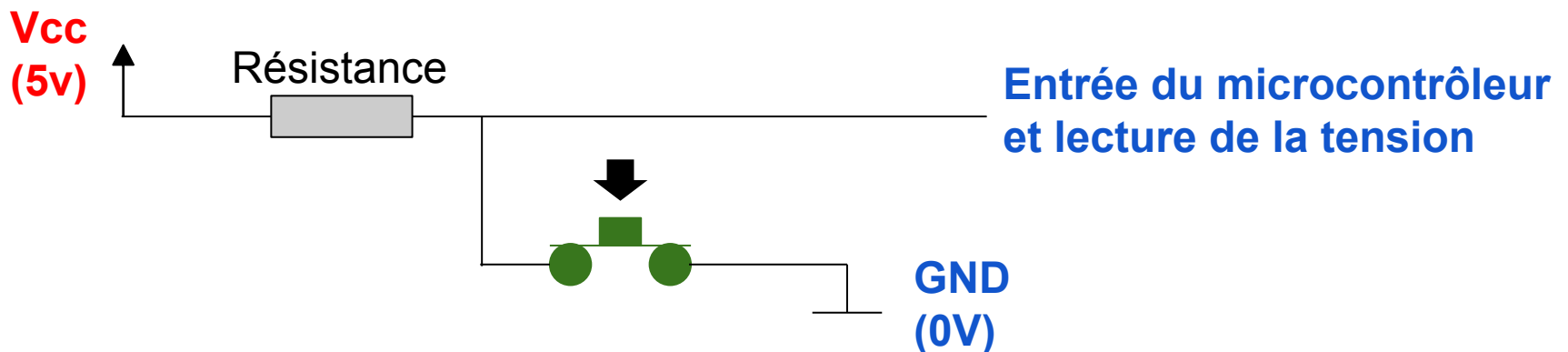


Le cas du bouton

Pullup



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



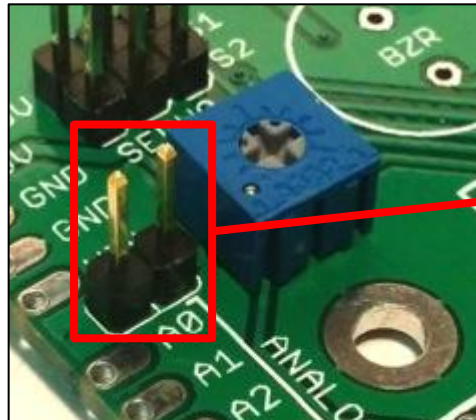
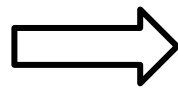
- Bouton pressé: le pin est connecté au **0V**

Le cas du bouton

- Le microcontrôleur intègre ce système de pullup
- Cela peut être activé lorsqu'on définit le mode du bouton: **pinMode(BTN, INPUT_PULLUP);**
- Attention: ceci n'est à utiliser que dans le cas du bouton!
- Le lecture de la valeur du bouton est effectuée par une lecture digitale: **digitalRead(BTN);**
- Cet appel renvoie **1 si le bouton est relâché et 0 si il est pressé!**

Potentiomètre

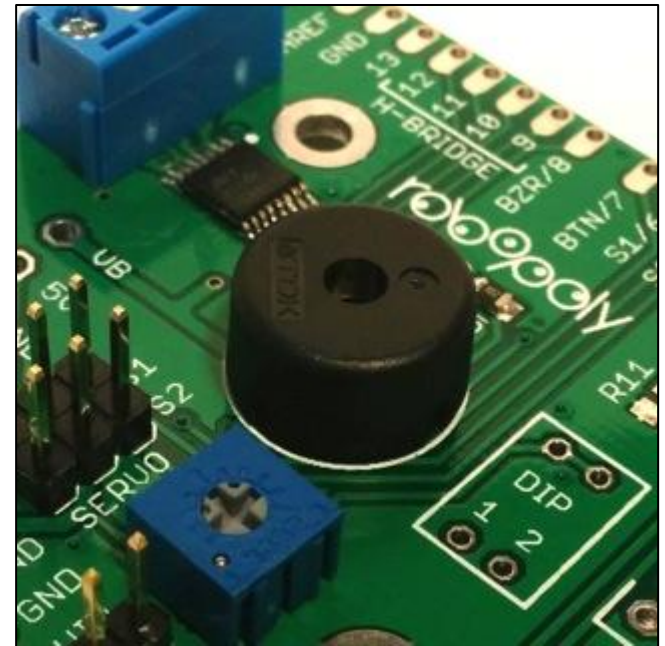
- On fait cette fois-ci une lecture analogique et non digitale: **analogRead(POT)**;
- Rappel: la fonction renvoie une valeur comprise **entre 0 et 1023**
- Utile pour régler un paramètre
- Un jumper doit être placé sur le Shield pour pouvoir utiliser le potentiomètre



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

Buzzer

- On le contrôle directement par sa fréquence en utilise l'appel `play(fréquence, durée);`
 - **fréquence** en Hertz
 - **durée** 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>
```

Inclus la bibliothèque

```
void setup()
```

```
{
```

```
  // set pin output mode (sources current)
```

```
  pinMode(LED, OUTPUT);
```

Définit le mode du pin (entrée)

```
}
```

```
void loop()
```

Entre dans la boucle

```
{
```

```
  // turn LED on
```

```
  digitalWrite(LED, HIGH);
```

Allume la LED

```
  // wait 500 milliseconds
```

```
  delay(500);
```

Attend 500 millisecondes

```
  // turn LED off
```

```
  digitalWrite(LED, LOW);
```

```
  delay(500);
```

Eteint la LED

```
}
```

A vous de jouer

- Petits exercices:
 - Utiliser le **bouton** ou le **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 1



Prochains évènements

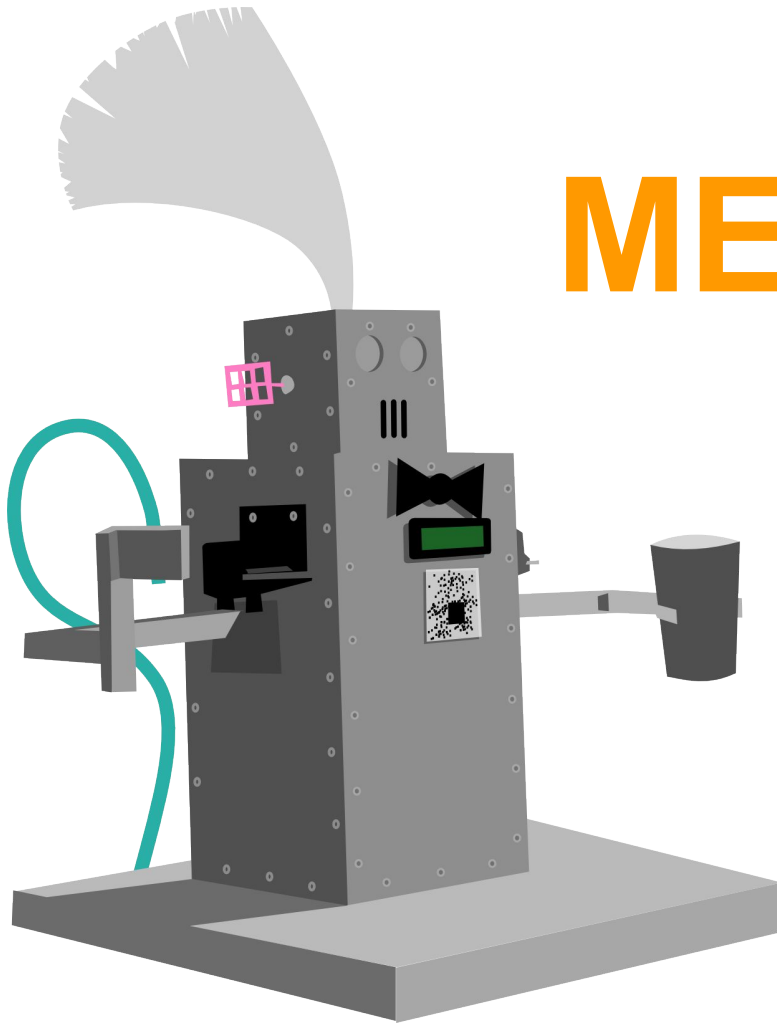
- **Prochain Démon**

- Lundi prochain, 12h15, ELA1
- La **PowerBoard** et les **moteurs**

- **Worshop 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!

MERCI!



Questions?