

# Bouton et capteur IR

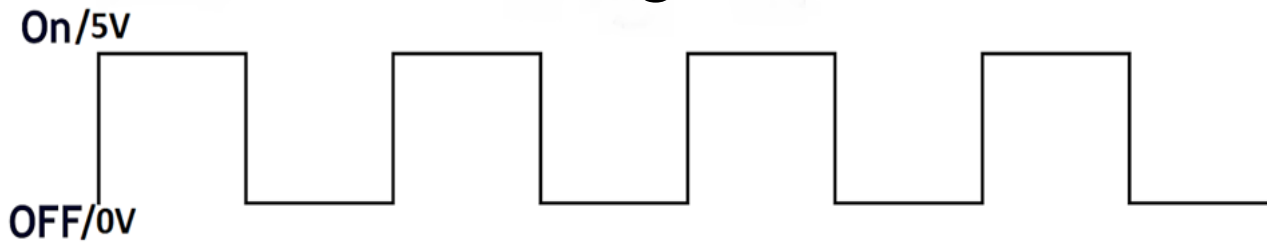
Comment “voir et sentir” le monde  
qui nous entoure

# Qu'est ce que l'on apprend ?

- Monde digital / analogue
- Bouton
- Capteur IR
- Principes de programmation et applications

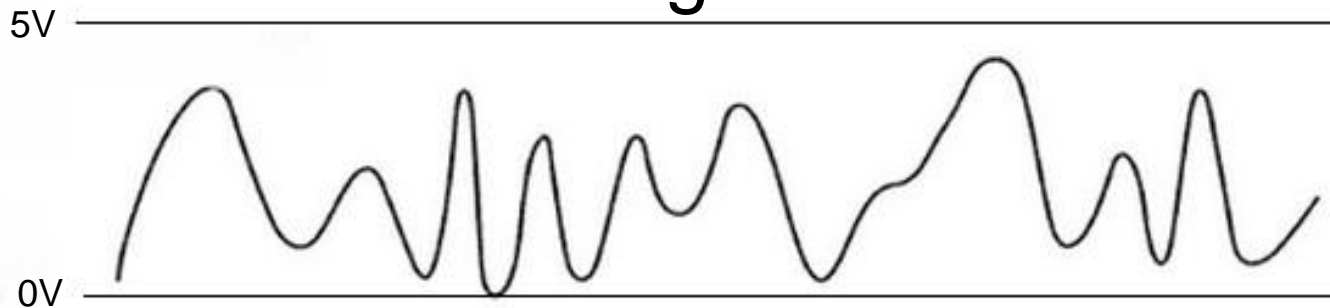
# Signal digital/analogue

## Digital



Faux = 0V  
Vrai = 5V

## Analogue

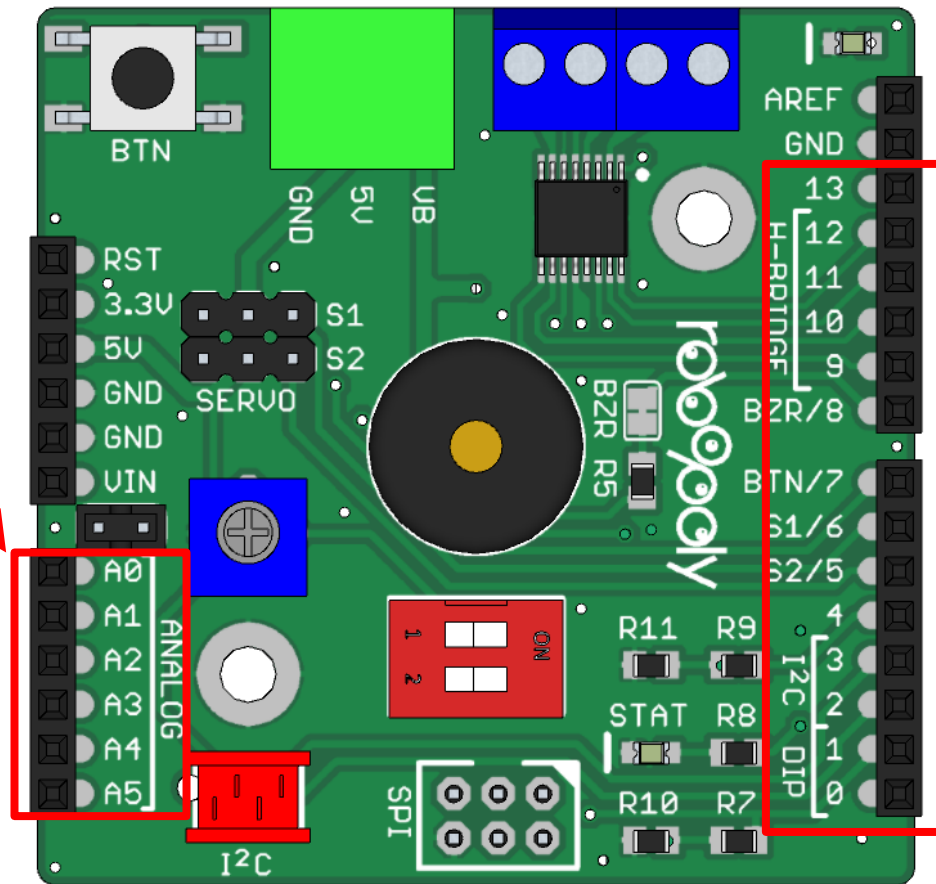


n'importe  
quelle  
valeur

# Pin digital/analogique

**Pins  
“analogiques”  
(A1-A5)**

peuvent aussi être utilisé  
pour lire en digital, mais  
du coup on prends de la  
place sur les pin analog.  
pour rien



**Pins  
“digitals”  
(0-13)**

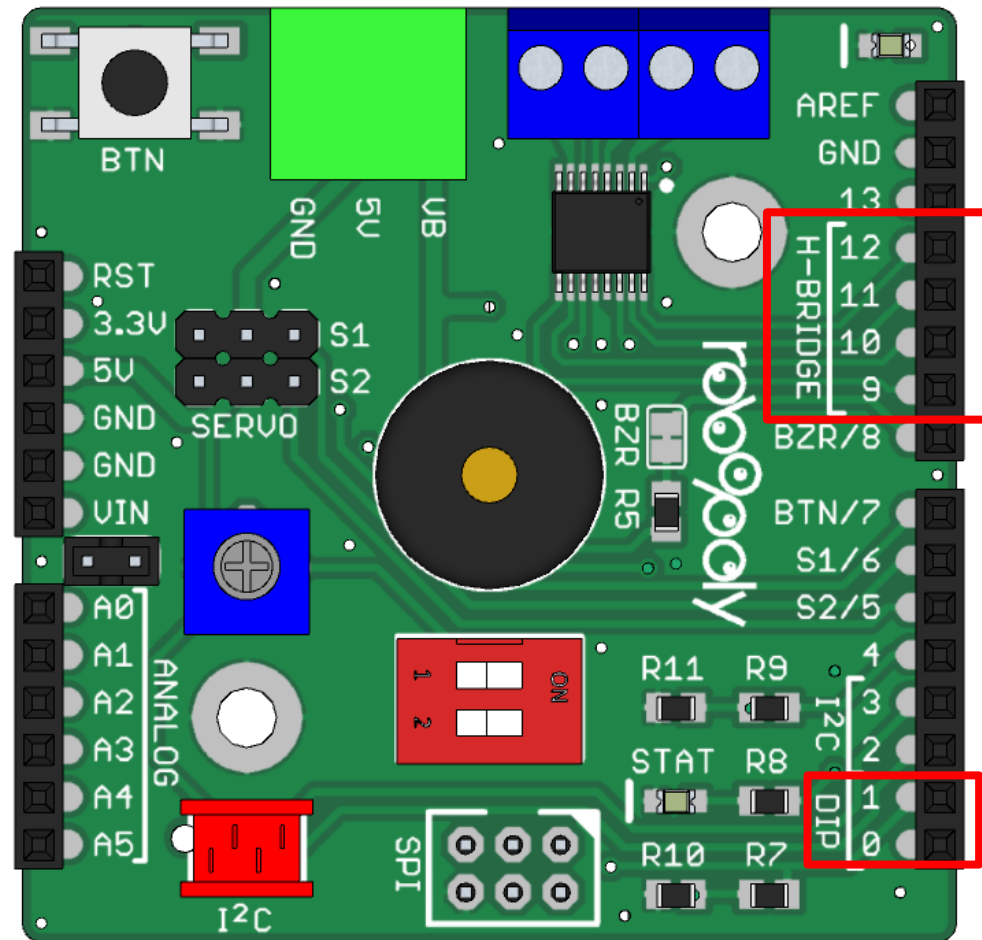
# Avant de faire des bêtises

vérifier que le pin soit libre !

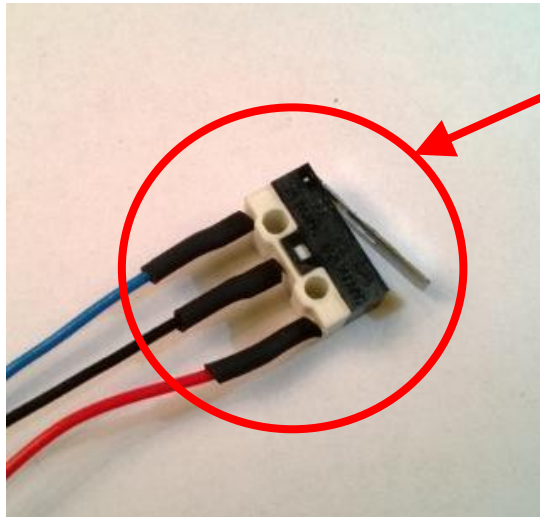
Comme le shield comporte plusieurs accessoires (servo, pont-H,...) certains pins sont justement utilisés pour leur fonctionnement et ne peuvent pas être utilisés pour brancher n'importe quoi ! On peut p.ex. voir que les pistes du pont-H utilisent les pins de 9 à 12, donc pas touche !

Voici la répartition des pin utilisables :

- A0 : potentiomètre
- A1, A2, A3 : capteur IR
- A4, A5 : bouton poussoir
- 0 – 1 : DIP switch
- 2 – 3: I2C
- 13 : rail d'alim des capteurs

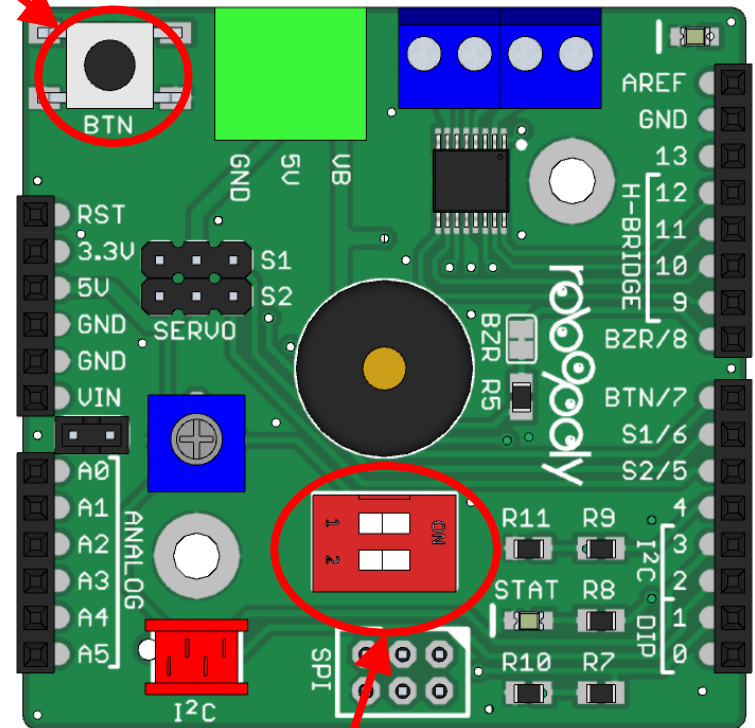
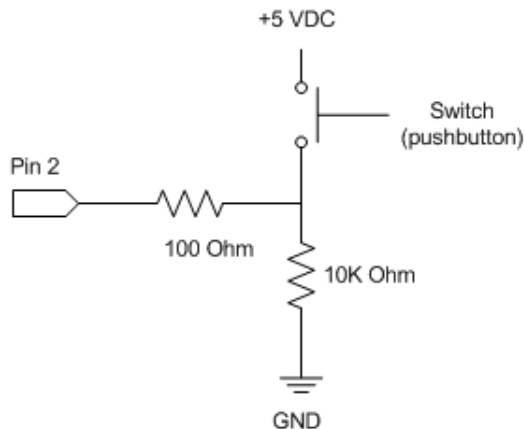


# Buttons/Switches



Bouton

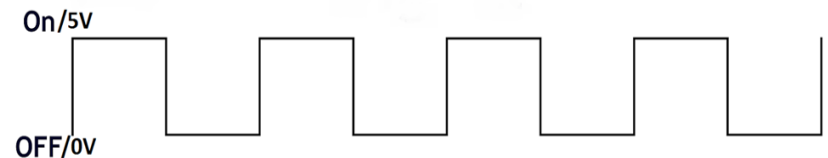
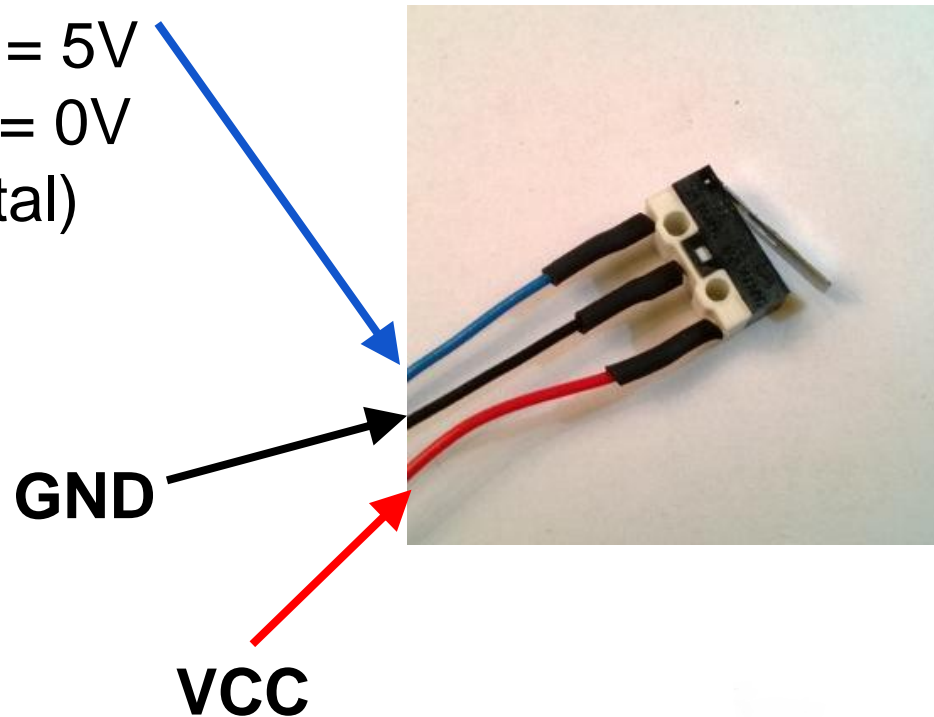
Pushbutton Schematic



DIP-Switch

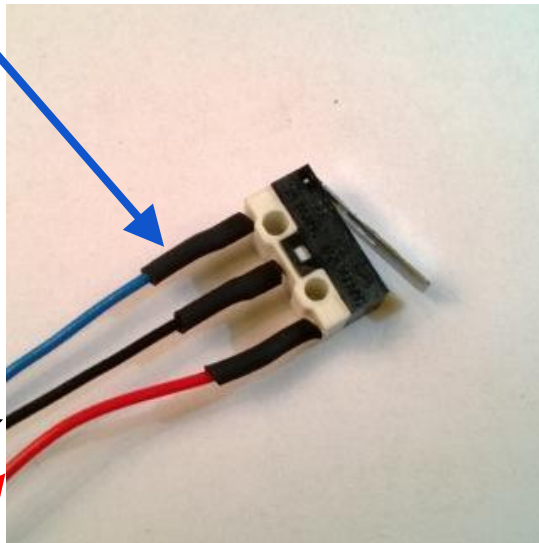
# Fonctionnement du bouton

**Signal**  
 ouvert = 5V  
 fermé = 0V  
 (digital)



# Fonctionnement du bouton

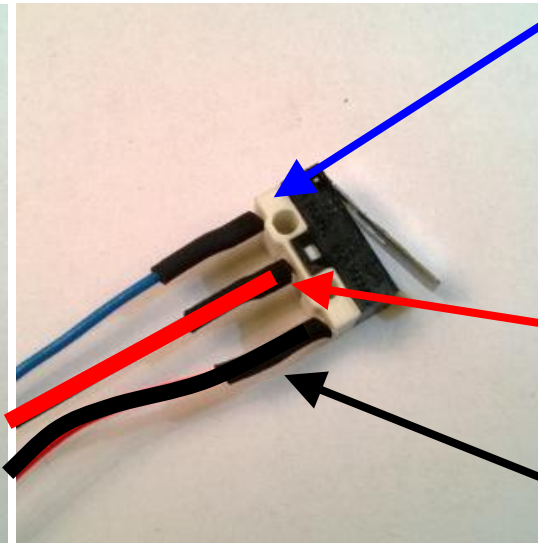
**Signal**  
 ouvert = 5V  
 fermé = 0V  
 (digital)



**GND**

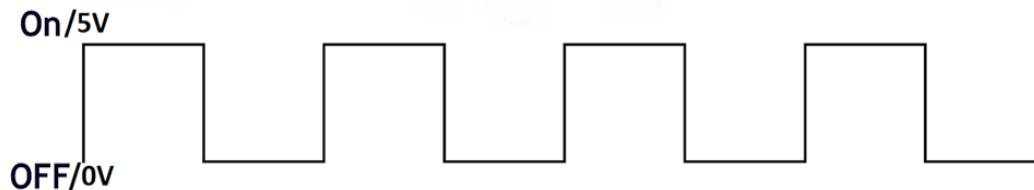
**VCC**

**Signal**  
 ouvert = 0V  
 fermé = 5V  
 (digital)



**VCC**

**GND**





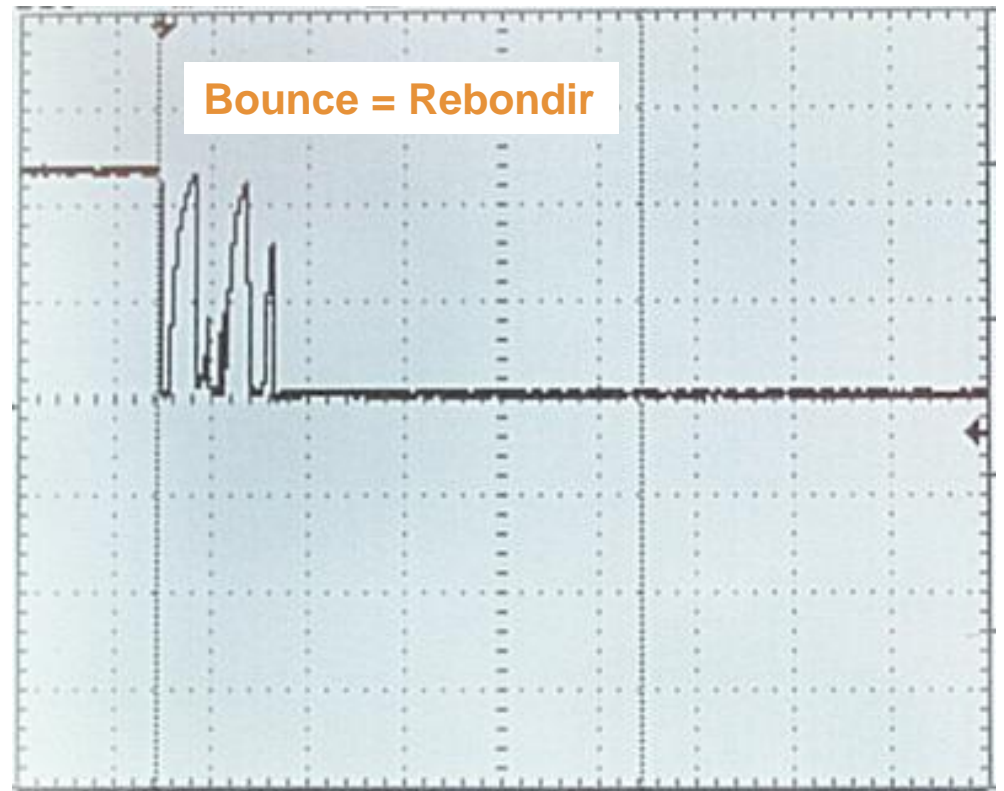
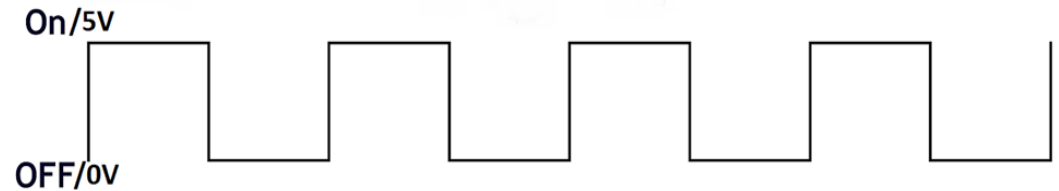
# Debounce

Quand on appuie sur un bouton, le passage de 0 à 1 ne se fait pas instantanément, il y a une oscillation avant de changer d'état.

```

if(capteur == 1)
    // allumer LED
else if(capteur == 0)
    // éteindre LED
    
```

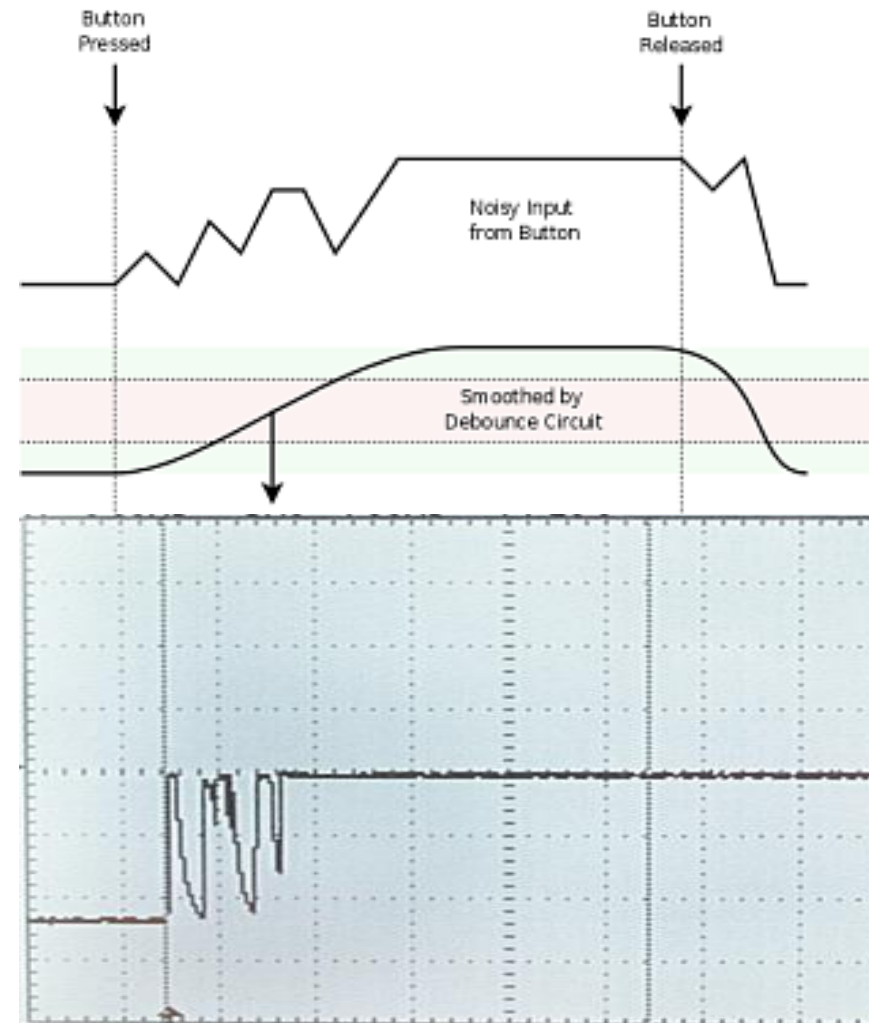
En effectuant le code ci-dessus, la LED risque de clignoter avant de rester allumée quand le capteur passe de 0 à 1. (peut-être que ça ne sera pas visible à l'œil nu mais si c'est autre chose qu'une LED, ça risque de poser problème)



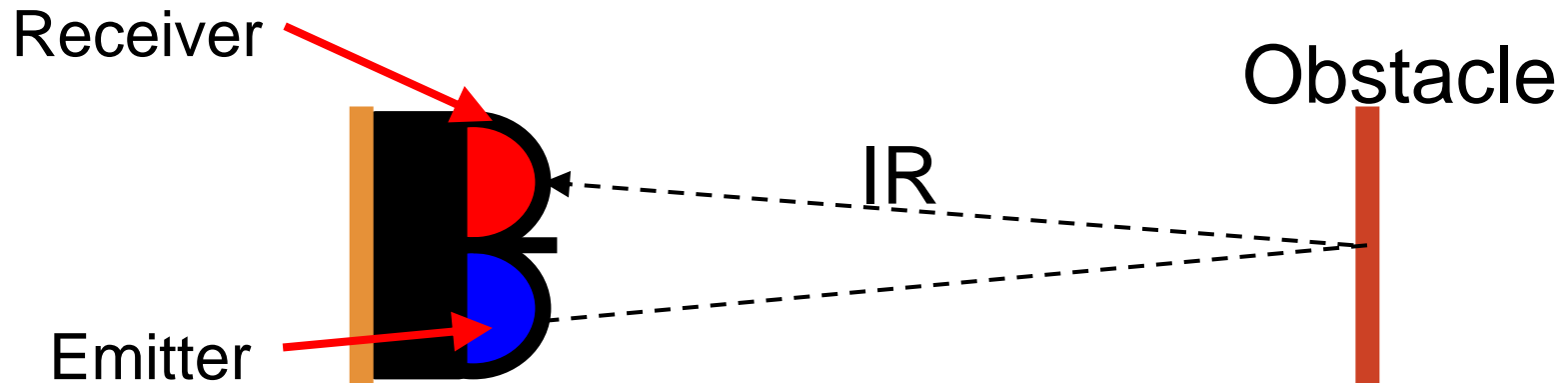
# Debounce

Résolution :

- 1) Ajouter une capacité
  - 2) Faire deux mesures sur un intervalle de temps pour voir si la valeur à changé (il y a une fonction arduino pour ça)
- 1) Mettre un temps de pause avant chaque mesure à la fin de la boucle avec `delay(100);`

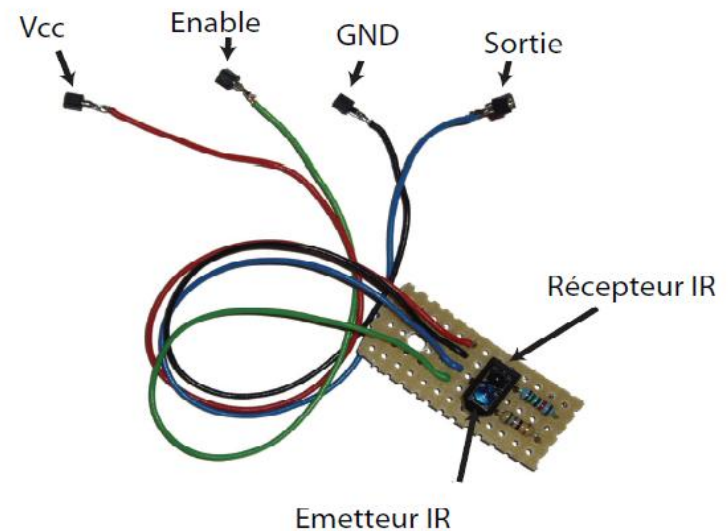


# Capteur IR (infrared)

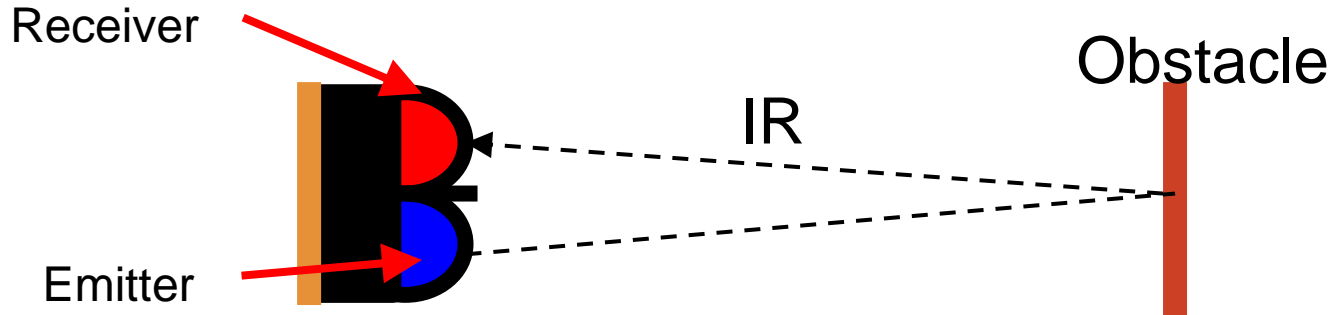


deux modes de fonctionnement:

- digital (0 ou 1)
- analogue (0 - 1023)

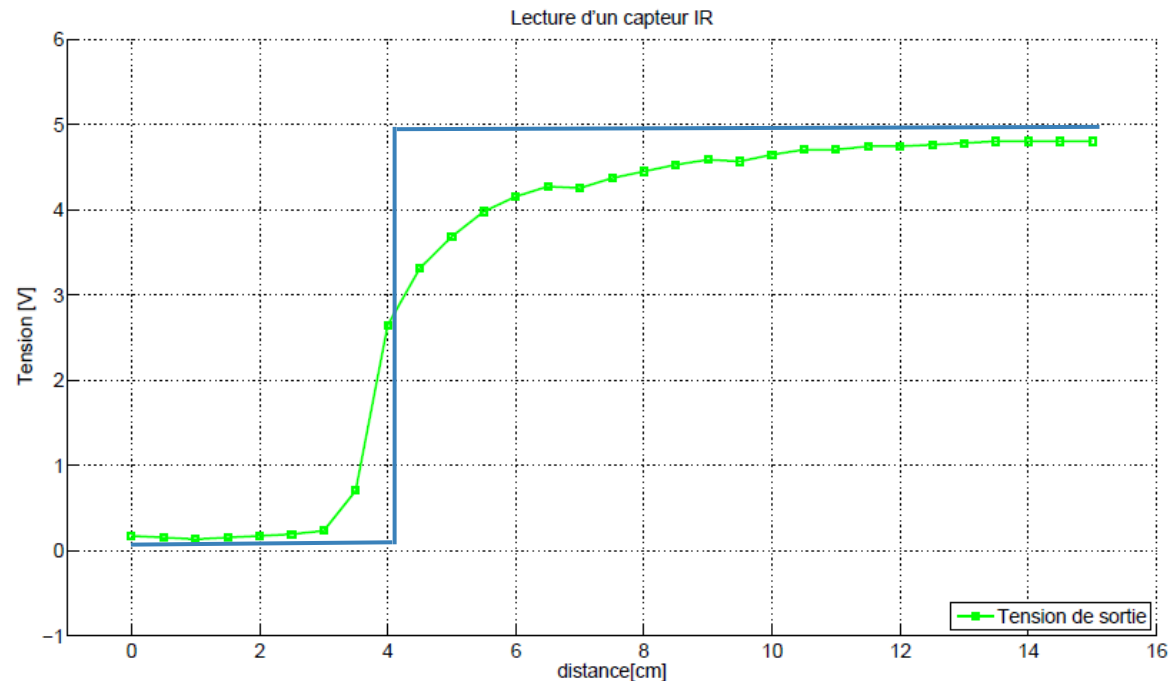


# Capteur IR - fonctionnement



deux modes de fonctionnement:

- digital (0 ou 1)
- analogue (0 - 1023)



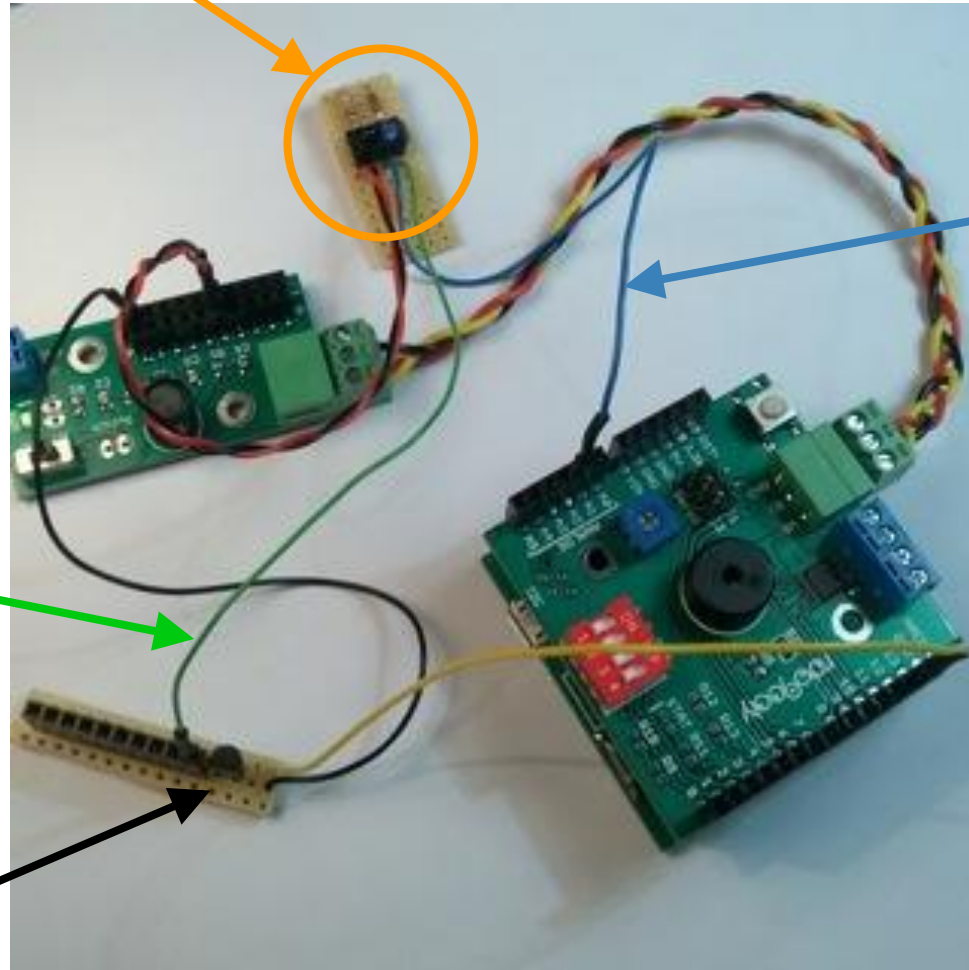
# Capteur IR - branchement

VCC  
GND

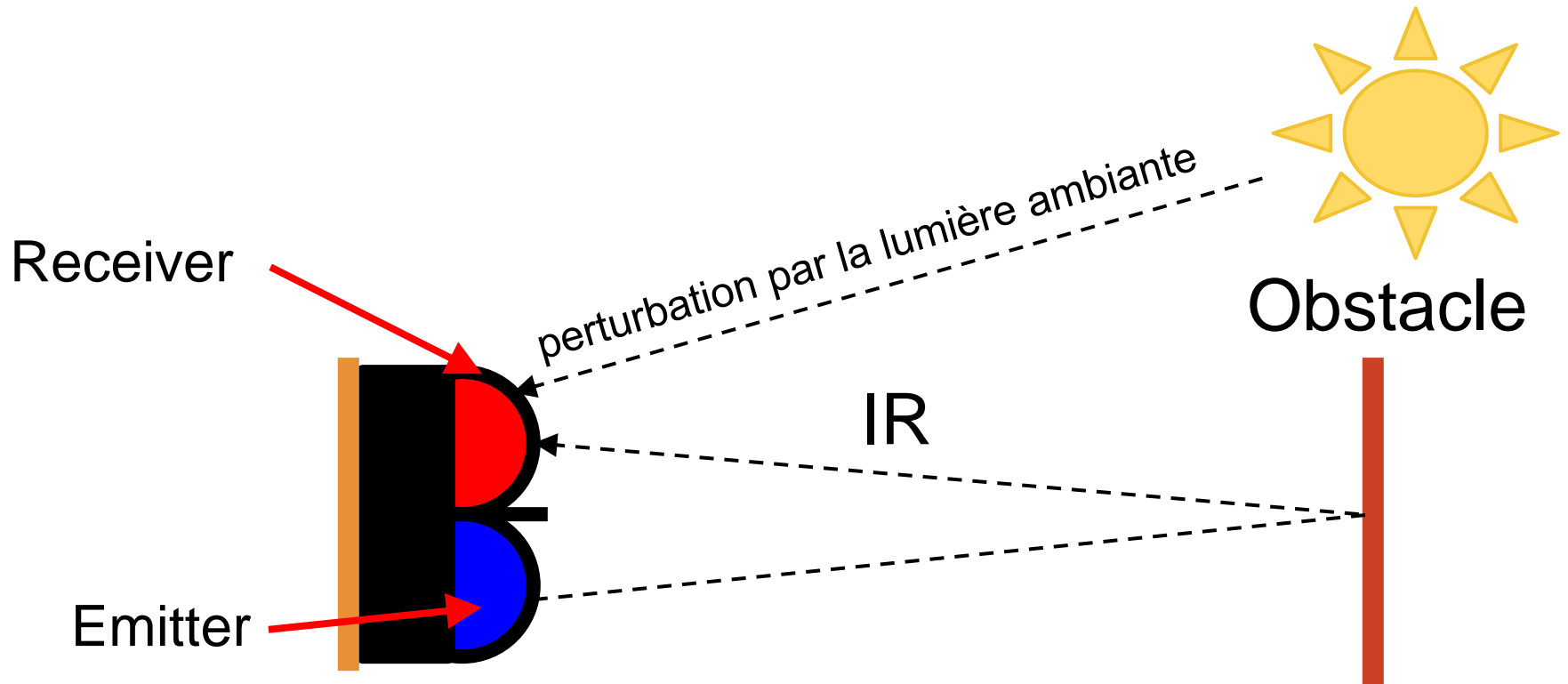
Signal

Enable

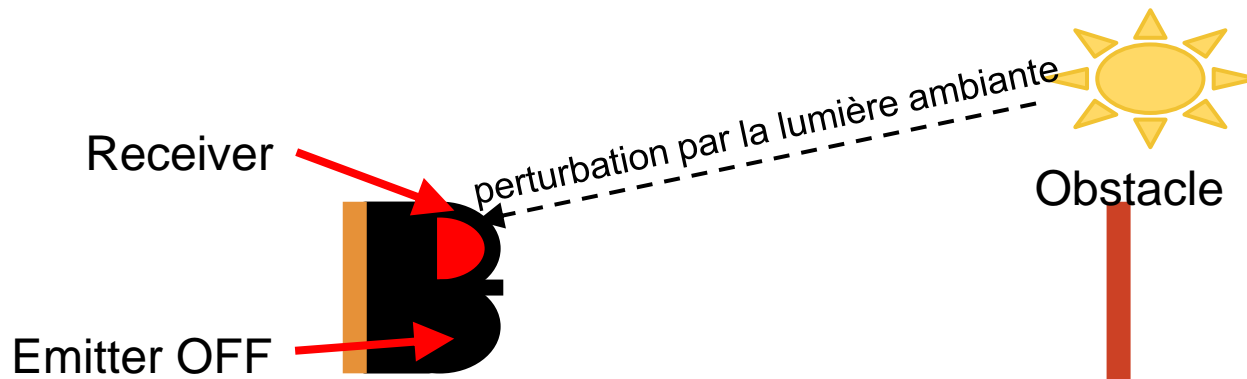
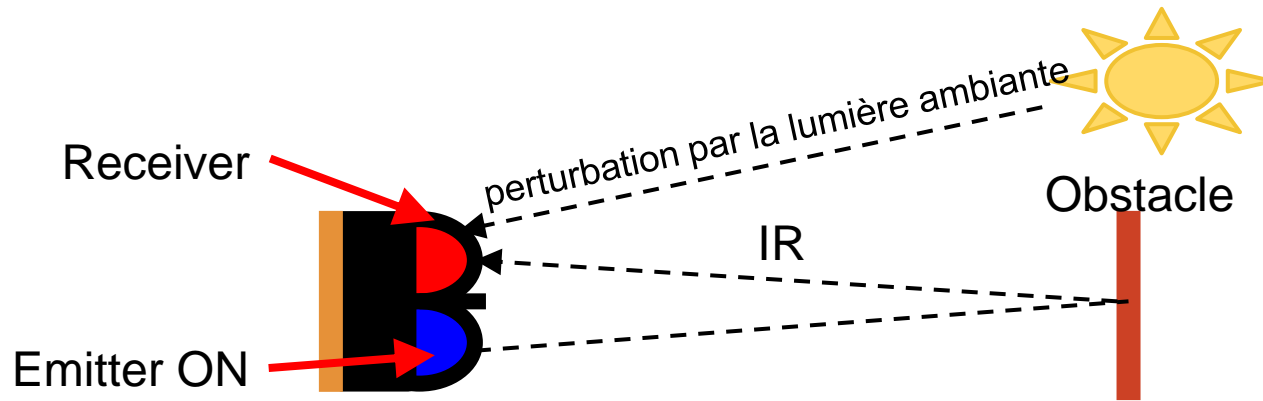
Transistor



# Capteur IR - perturbations

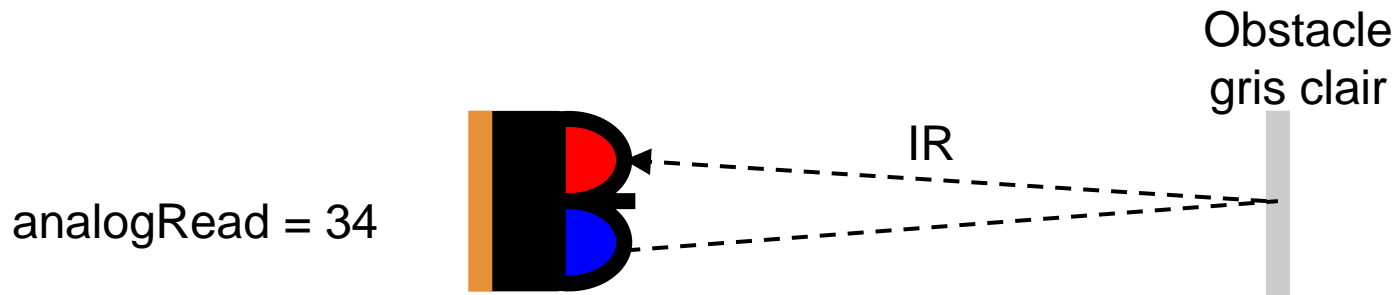
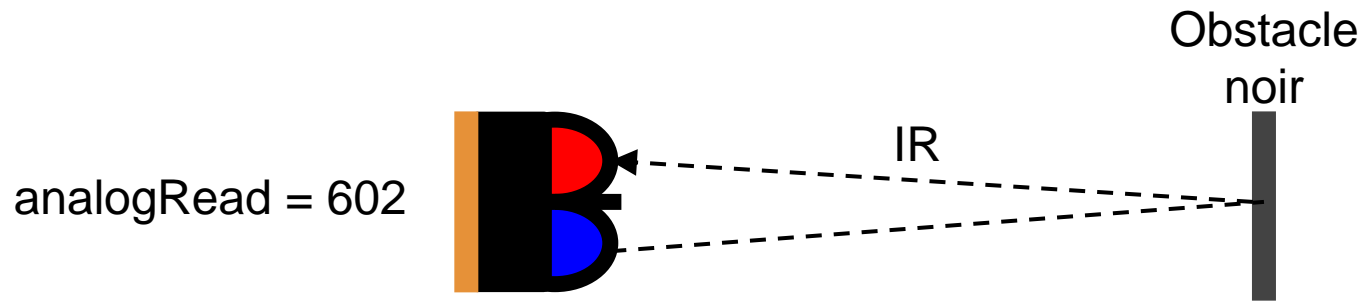


# Capteur IR - perturbations

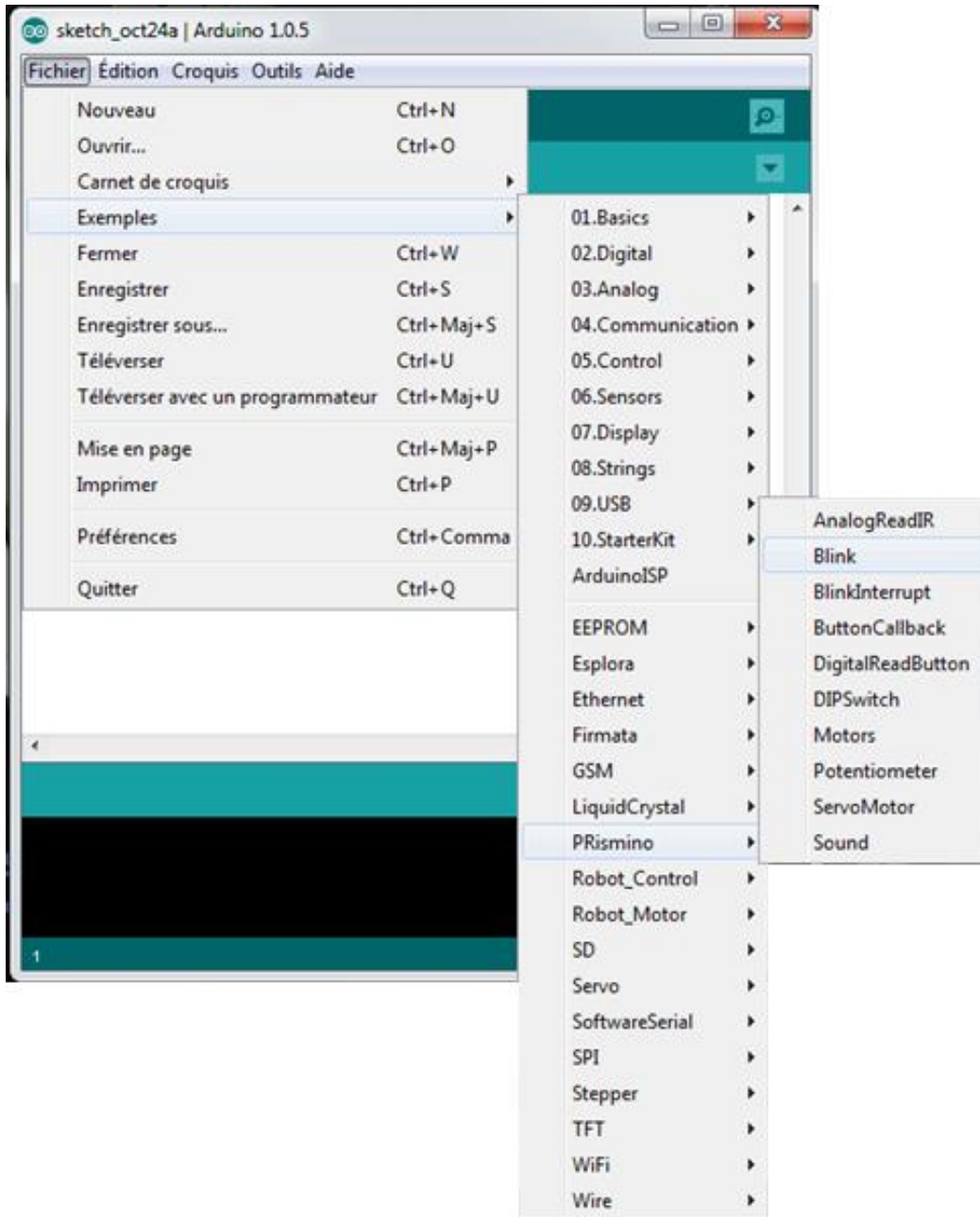


# Capteur IR - surfaces

analogRead [0,1023]







Regarder les exemples PRismino, c'est un bon moyen de se souvenir comment utiliser une fonction.

- Blink
- DigitalReadButton
- AnalogReadIR



Search the Arduino Website

Reference [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

# Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

## Structure

- `setup()`
- `loop()`

### Control Structures

- `if`
- `if...else`

## Variables

### Constants

- `HIGH` | `LOW`
- `INPUT` | `OUTPUT` | `INPUT_PULLUP`
- `true` | `false`
- integer constants
- floating point constants

## Functions

### Digital I/O

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

### Analog I/O

# mode INPUT / OUTPUT

INPUT : lecture

OUTPUT : écriture

```
pinMode(pin, mode);
```

numéro du pin

INPUT ou OUTPUT

- INPUT : le pin est utilisé pour lire / récupérer des données (capteur, bouton,...)

- OUTPUT : le pin est utilisé pour communiquer / faire fonctionner des accessoires (LED, moteur, servo,...)

Pins configured as outputs can also be damaged or destroyed if short circuited to either ground or 5 volt power rails.

# mode INPUT / OUTPUT

```
#include <prismo.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);
}
```

On met la LED en mode sortie (OUTPUT) pour lui donner une valeur (HIGH ou LOW).  
On ne va quand même pas essayer de récupérer la valeur de la LED...

# digitalWrite

écrire en digital

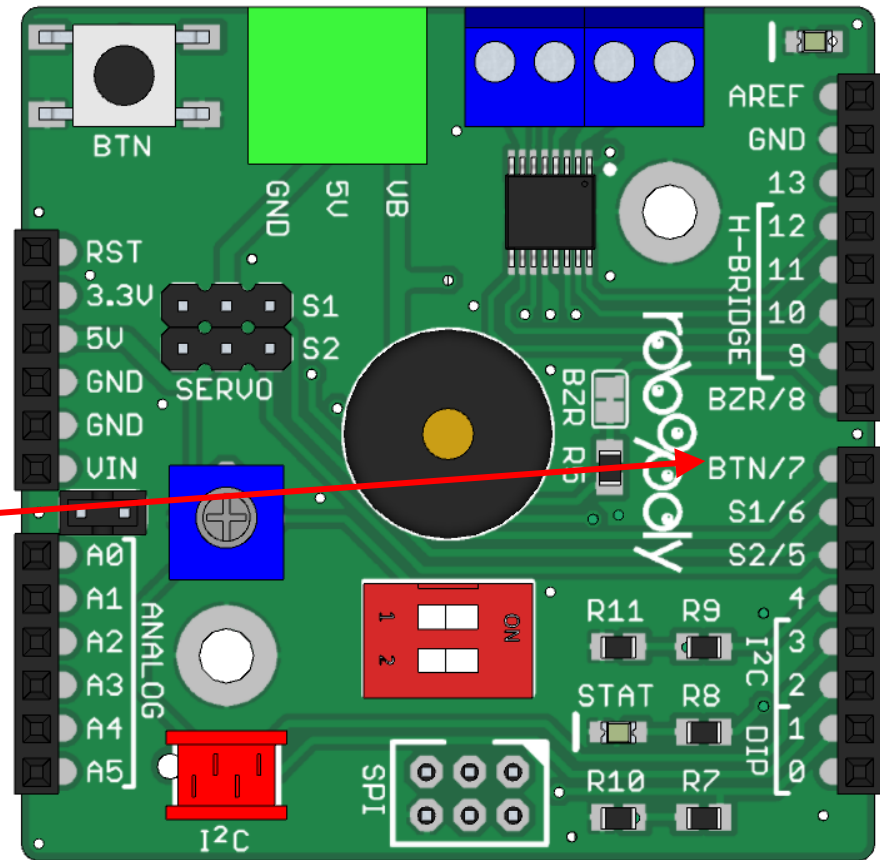
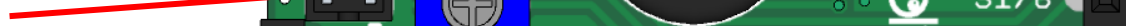
HIGH / LOW



`digitalWrite(pin, value);`



numéro du pin  
p. ex. 7



# digitalRead

lire en digital

`digitalRead(pin);`



numéro du pin  
p. ex. 3

```
#include <prismino.h>

// the button value can be 0 or 1
unsigned char button_value;

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

void loop()
{
    // connect lever button to pin A4
    button_value = digitalRead(A4);

    if(button_value)
    {
        digitalWrite(LED, HIGH);
    }
    else
    {
        digitalWrite(LED, LOW);
    }
    delay(100);
}
```

# analogRead

analogRead(pin);



lire en analogique,  
principalement utilisé  
pour lire une valeur  
analogique des  
capteurs IR  
valeur retournée entre  
**0 et 1023**

```
#include <prismo.h>

// the analog value is between 0 and 1023 so it needs a 16-bit variable
unsigned int ir_value;

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

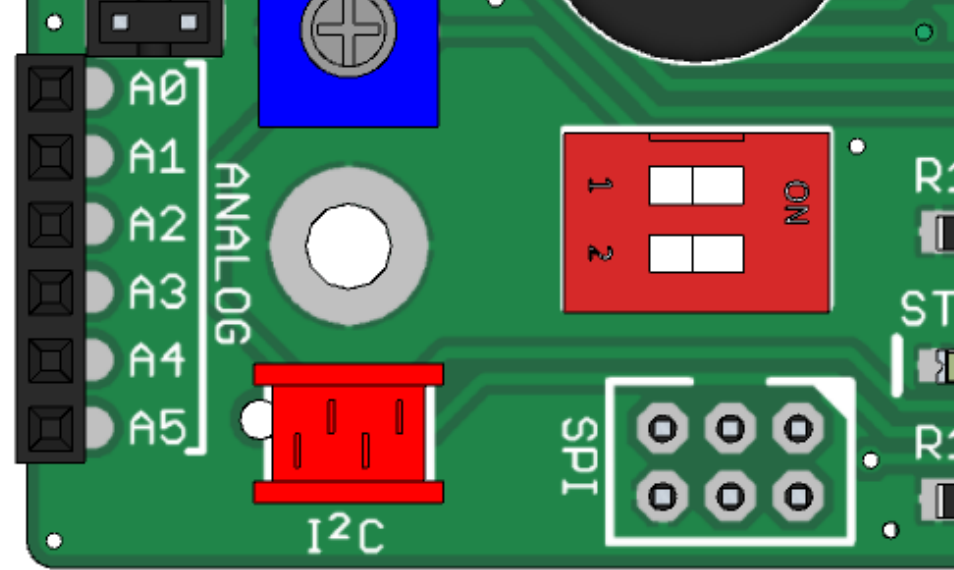
void loop()
{
  // connect sensor to pin A1, A0 is used by the potentiometer on the shield
  ir_value = analogRead(A1);

  if(ir_value < 512)
  {
    // a low value means something (an obstacle) reflects emitted IR to the receiver
    digitalWrite(LED, HIGH);
  }
  else
  {
    // no obstacle is detected, turn LED off
    digitalWrite(LED, LOW);
  }
  delay(100);
}
```

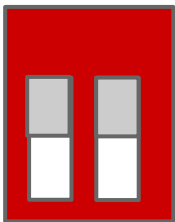
# DIP switch

$$2^2=4$$

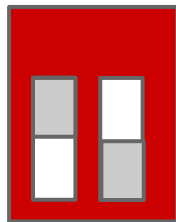
Le DIP switch est un moyen d'interagir avec le code en lui communiquant des informations.



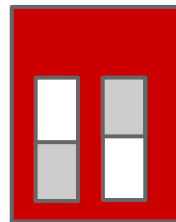
On a 2 switch avec 2 positions, donc on peut au maximum communiquer  $2^2$  positions soit 4 !  
On peut donc soit les utiliser “un par un”, ce qui fait 2 boutons, soit comme un nombre binaire !



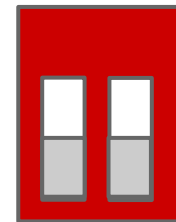
00 -> 0



01 -> 1



10 -> 2



11 -> 3

“un par un”, on utilise seulement le bouton voulu

binaire : on “écrit” un nombre binaire qui sera entre 0 et 3



# switch case

## tester plusieurs cas

On peut tester la valeur retournée par un capteur IR en analogique ou la valeur du DIP-switch p.ex.

```
switch (variable)
{
  case 42:
    //je fais ça
    break;
  case 3:
    //ou plutôt ceci
    break;
  case -273:
    //en fait non, c'est mieux ça
    break;
  case obi-wan kenobi:
    //ouais non, ce cas est mieux
    break;
}
```

# Agenda

- Workshop II:

**Samedi 1 novembre, de 8h à 17h**

- En haut du BM (Cafétéria): Soudure et la prog
- Au local: pour les bases mécaniques

Gratuite !!!

Repas (non obligatoire): 5 CHF

Inscription à la journée/repas:

<http://doodle.com/gn2uccpy9m9vw3n2>

- Prochain démon : ???

# Informations

- **Rappel**

- Inscription 20.-
- Kit PRisme 65.-
- Batteries 15.-
- Notice de montage:

<http://robopoly.epfl.ch/prisme/assembleage>

**FIN**

Questions?