

# Objets connectés

François Roland

19 – 21 Novembre 2018

# Partage d'informations

<http://bit.ly/iot-eecampus3>

# Plan

1 Introduction

2 Arduino

3 Télécommunications

4 Conclusion

## Introduction

## 1 Introduction

- Présentations
  - Définitions
  - Historique

2 Arduino

## 3 Télécommunications

## 4 Conclusion

## Règles de conduite

## Horaire

- 8:30 – 12:30 (pause vers 10:30)
  - 13:30 – 17:30 (pause vers 15:30)

## Prise de parole

- Questions et commentaires bienvenus
  - Parler suffisamment fort pour l'ensemble du groupe
  - Ne pas monopoliser la parole
  - Ne pas interrompre

Respect de chacun

- Commentaires discriminatoires non tolérés
  - Ne pas perturber le cours (gsm, réseaux sociaux)

## François Roland



- Ingénieur Civil en Informatique et Gestion
  - Aujourd'hui
    - Chercheur Télécoms UMONS
    - Chargé de projet Fab-IoT-Lab au FabLab Mons
    - Indépendant complémentaire
  - Dans le passé
    - Consultant en développement logiciel
    - Take-Eat-Easy (startup)

## Mes centres d'intérêt professionnels



- Électronique
  - Programmation
  - Conception mécanique
  - Enseignement

## Et vous ?

- Prénom
  - Attentes
  - Compétences IoT
    - Programmation
    - Électronique
    - Télécommunications
  - Projets personnels IoT



## Qu'est-ce que l'Internet des Objets ?

« L'Internet des objets, ou IdO (en anglais Internet of Things, ou IoT), est l'extension d'Internet à des choses et à des lieux du monde physique.

*Alors qu'Internet ne se prolonge habituellement pas au-delà du monde électronique, l'Internet des objets connectés représente les échanges d'informations et de données provenant de dispositifs du monde réel avec le réseau Internet. »*

[https://fr.wikipedia.org/wiki/Internet\\_des\\_objets](https://fr.wikipedia.org/wiki/Internet_des_objets)

# Qu'est-ce qu'un objet connecté ?

Un objet connecté est constitué de :

- Un capteur ou actionneur pour interagir avec son environnement
- Un micro-contrôleur pour transformer les données
- Un moyen de communication pour communiquer avec le monde
- Une source d'énergie pour alimenter les éléments précédents



# Histoire de l'IoT



**1999** Kevin Ashton invente l'expression « Internet des objets »  
Création d'un labo dédié aux objets connectés au MIT aux États Unis

# Histoire de l'IoT



- 2003** Rafi Haladjian crée la lampe DAL, équipée de 9 LEDs et vendue 790 euros
- 2005** Lancement du Nabaztag, lapin connecté en WiFi qui lit les emails à voix haute, émet des signaux visuels et diffuse de la musique

# Histoire de l'IoT



- 2007** Démocratisation des smartphones  
Sortie du premier iPhone par Apple.
- 2018** La technologie nous permet de connecter Internet à tout objet du quotidien  
Les quantités de données générées par les objets sont colossales et leurs perspectives d'exploitation sans limites  
L'utilisation de ces données suscite des questions éthiques délicates sur le respect de la vie privée

# Arduino

## 1 Introduction

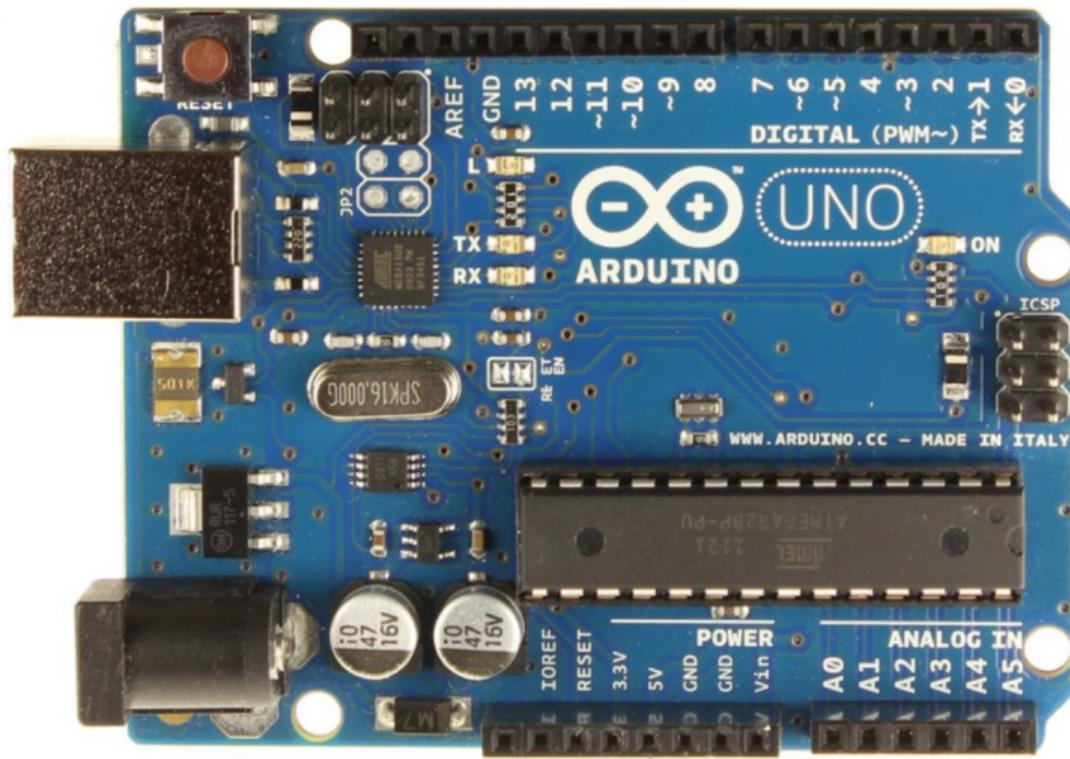
## 2 Arduino

- Découverte du micro-contrôleur
- Programmation Arduino
- Mise en pratique

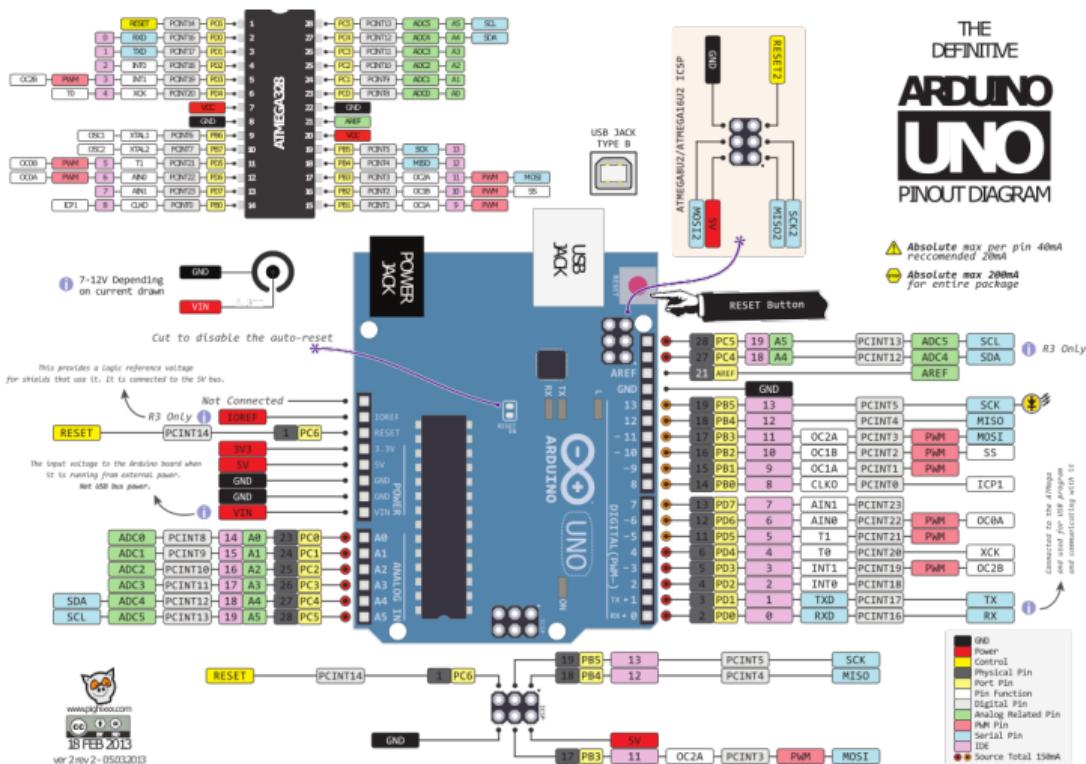
## 3 Télécommunications

## 4 Conclusion

# Arduino



## Arduino PINOUT



# Programmation C pour Arduino

```
1 Code == texte;  
2 Exécution ligne par ligne;  
3 // Commentaire
```

# Variable

```
1 int pin = 13;
```

- Un type
  - int
  - float
  - char []
- Un nom
- Une valeur de départ (*optionnel*)

# Fonction

```
1 int myMultiplyFunction(int x, int y) {  
2     return x * y;  
3 }  
4  
5 void loop() {  
6     int k = myMultiplyFunction(2, 3);  
7  
8     Serial.println(k); // 6  
9 }
```

- Un nom
- Des paramètres (*optionnel*)
- Un type de retour
- Un code d'implémentation

# Fonctions spéciales

```
1 void setup() {  
2     ...  
3 }
```

est exécuté une seule fois au démarrage.

```
4 void loop() {  
5     ...  
6 }
```

est exécuté en boucle après `setup()`.

# Condition

```
1 if (x > 120) {  
2     digitalWrite(LEDpin1, HIGH);  
3     digitalWrite(LEDpin2, HIGH);  
4 }  
5  
6 if (y == 150) {  
7     digitalWrite(LEDpin4, LOW);  
8 } else {  
9     digitalWrite(LEDpin4, HIGH);  
10 }
```

- Une condition à tester
- Un bout de code à exécuter si la condition est vérifiée
- Un bout de code à exécuter si la condition n'est pas vérifiée (*optionnel*)

# Boucle for

```
1 for (int i=0; i <= 255; i++) {  
2     analogWrite(PWMpin, i);  
3     delay(10);  
4 }
```

À utiliser quand on sait combien de fois il faut répéter la boucle.

- Un compteur
- Un bout de code à répéter

# Boucle while

```
1 int value = 0;  
2 while(value < 10) {  
3     digitalWrite(LEDpin3, HIGH);  
4     delay(200);  
5     digitalWrite(LEDpin3, LOW);  
6     value = analogRead(TRIMpin);  
7 }
```

À utiliser lorsque l'on veut répéter une boucle tant qu'une condition est vérifiée

- Une condition à vérifier
- Un code à répéter

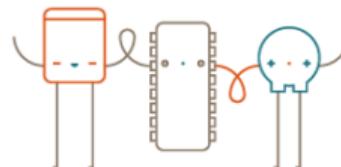
# Découvrons les kits

# Installation de l'IDE Arduino

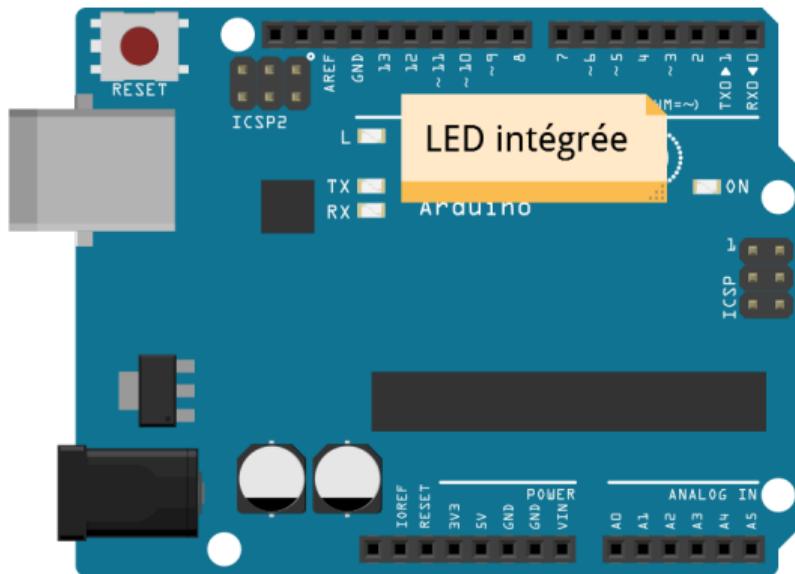


AN OPEN PROJECT WRITTEN, DEBUGGED,  
AND SUPPORTED BY ARDUINO.CC AND  
THE ARDUINO COMMUNITY WORLDWIDE

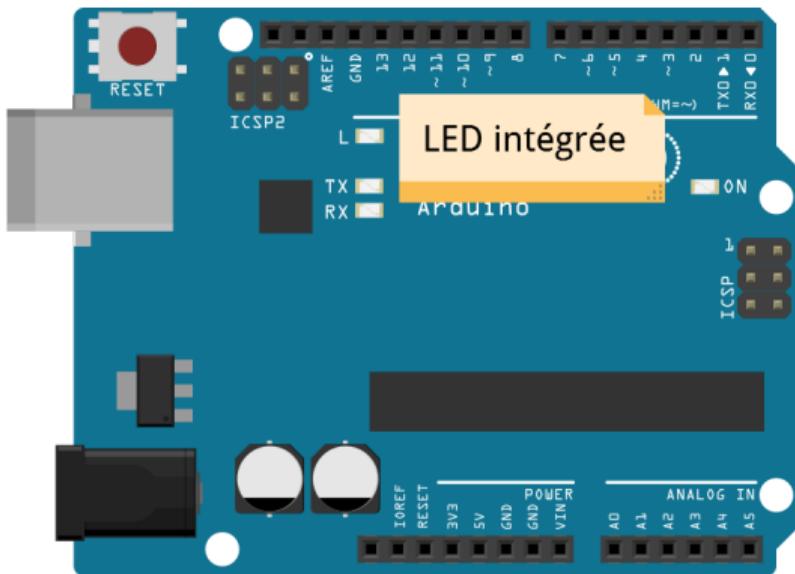
LEARN MORE ABOUT THE CONTRIBUTORS  
OF [ARDUINO.CC](http://ARDUINO.CC) ON [arduino.cc/credits](http://arduino.cc/credits)



# LED interne



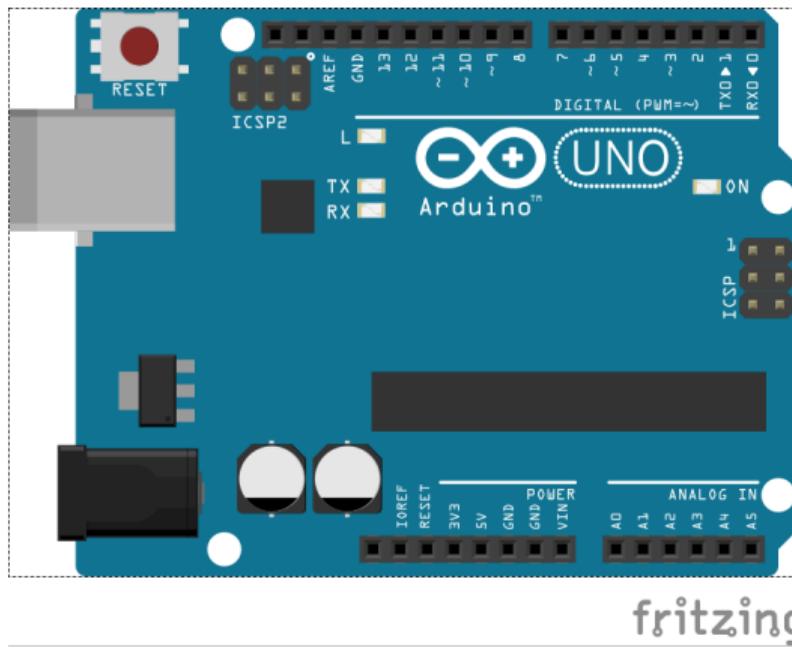
# LED interne



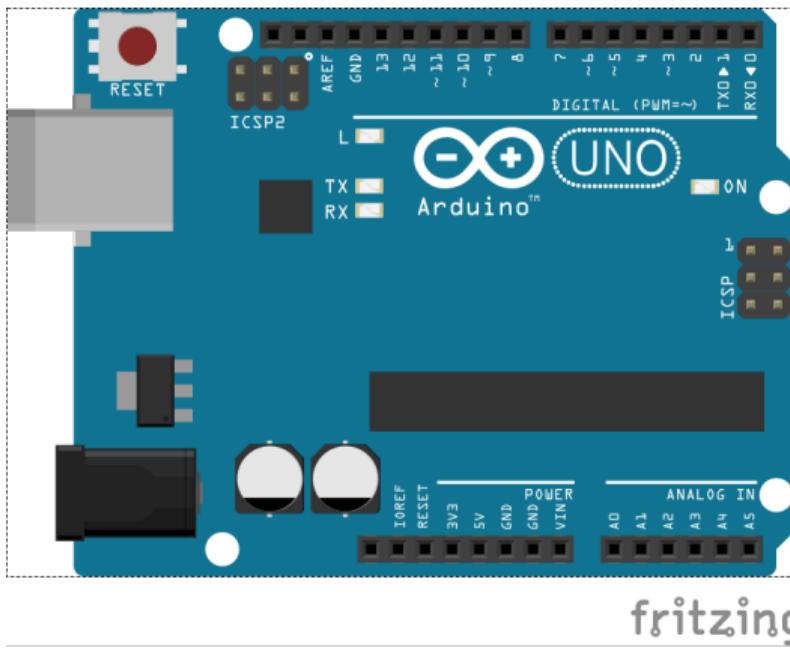
fritzing

```
1 void setup() {  
2     pinMode(LED_BUILTIN, OUTPUT);  
3 }  
4  
5 void loop() {  
6     digitalWrite(LED_BUILTIN, HIGH);  
7     delay(500);  
8     digitalWrite(LED_BUILTIN, LOW);  
9     delay(500);  
10 }
```

# Moniteur série

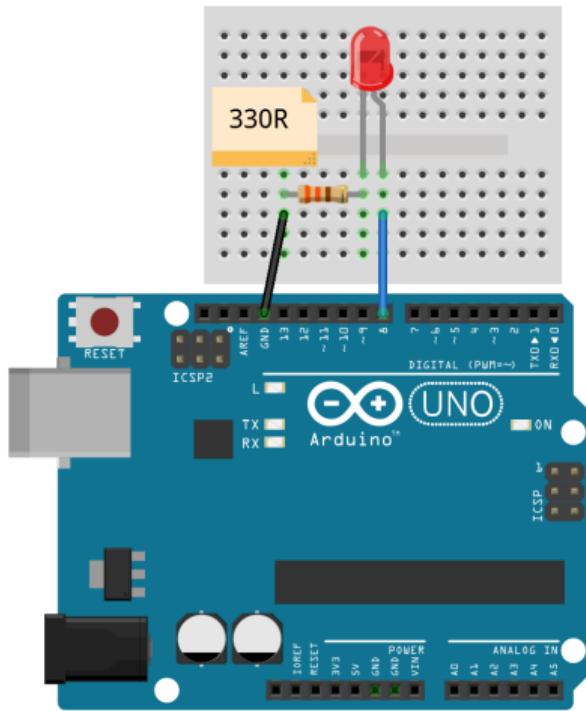


# Moniteur série



```
1 int i = 0;  
2  
3 void setup() {  
4     Serial.begin(115200);  
5     Serial.println("setup\u201dDONE");  
6 }  
7  
8 void loop() {  
9     Serial.print("Loop\u201d#");  
10    Serial.println(i);  
11    i++;  
12    delay(1000);  
13 }
```

# LED externe



fritzing

# Lois électriques

Loi d'Ohm

$$U = R \cdot I$$

$$I = \frac{U}{R}$$

$$R = \frac{U}{I}$$

Puissance électrique

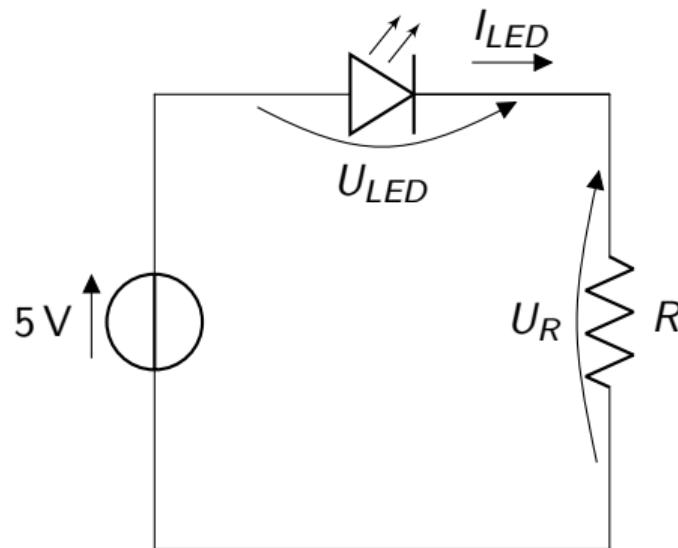
$$P = U \cdot I \quad (1)$$

$$P = R \cdot I^2 \quad (2)$$

$$P = R \cdot I^2 \quad (3)$$

où  $U$  est la tension,  $I$  le courant,  $R$  la résistance et  $P$  la puissance.

# Résistance de limitation du courant



Pour une LED rouge,

$$U_{LED} = 1.7 \text{ V} \quad \text{et} \quad I_{LED} = 10 \text{ mA}$$

donc la tension aux bornes de la résistance vaut

$$U_R = 5 \text{ V} - 1.7 \text{ V} = 3.3 \text{ V}$$

et sa valeur

$$R = \frac{U_R}{I} = \frac{3.3 \text{ V}}{10 \text{ mA}} = 330 \Omega$$

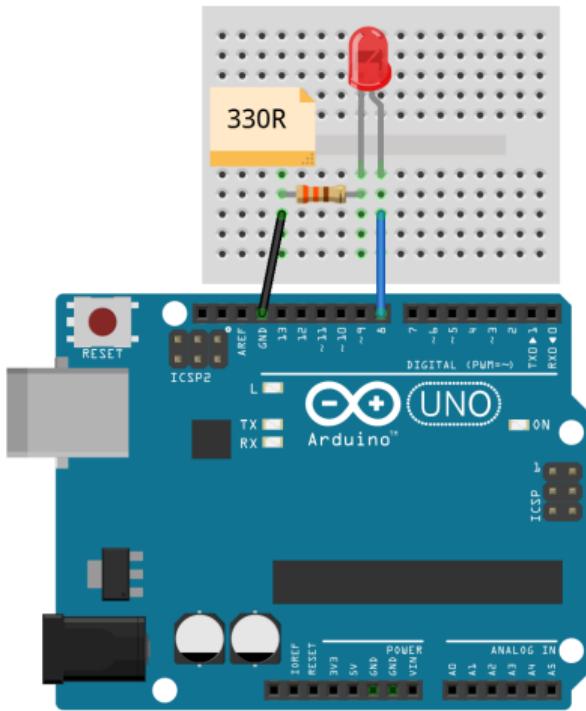
par application de la loi d'Ohm.

# Valeurs standards LED

Les valeurs caractéristiques d'une LED dépendent de sa couleur.

<i>couleur</i>	<i>chute de tension</i>	<i>courant maximal</i>
rouge	1.7 V	10 mA
jaune	2.1 V	10 mA
verte	2.2 V	10 mA
bleue	3.2 V	20 mA
blanche	3.6 V	20 mA

## LED externe



```
1 void setup() {
2     pinMode(8, OUTPUT);
3 }
4
5 void loop() {
6     digitalWrite(8, HIGH);
7     delay(500);
8     digitalWrite(8, LOW);
9     delay(500);
10 }
```

# Multimètre



Permet de mesurer :

**tension** V

**courant**  $\mu$ A mA A

**résistance**  $\Omega$

**continuité**

# Multimètre



Permet de mesurer :

**tension** V

**courant**  $\mu$ A mA A

**résistance**  $\Omega$

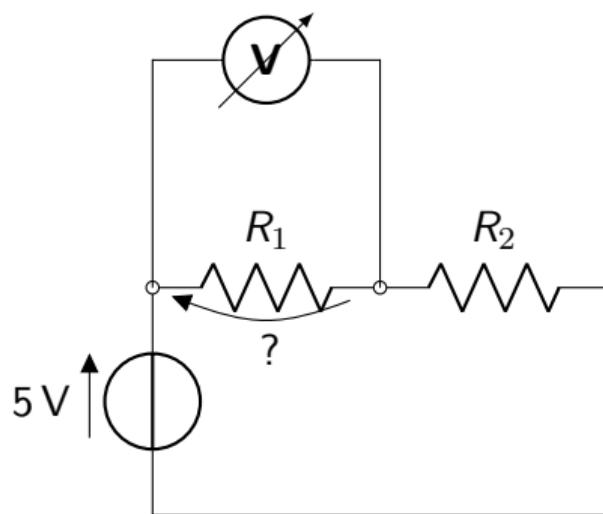
**continuité**

## Risque de court-circuit

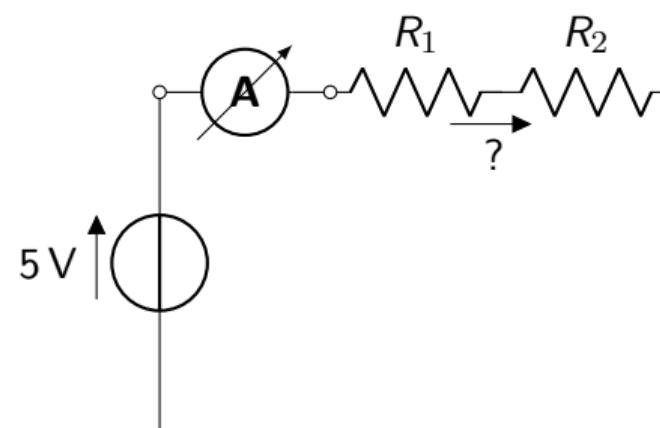
Pour les mesures de courant, il y a court-circuit entre les sondes. Le risque est d'appliquer une tension inadéquate ou un courant trop important sur un composant fragile.

# Mesures de base

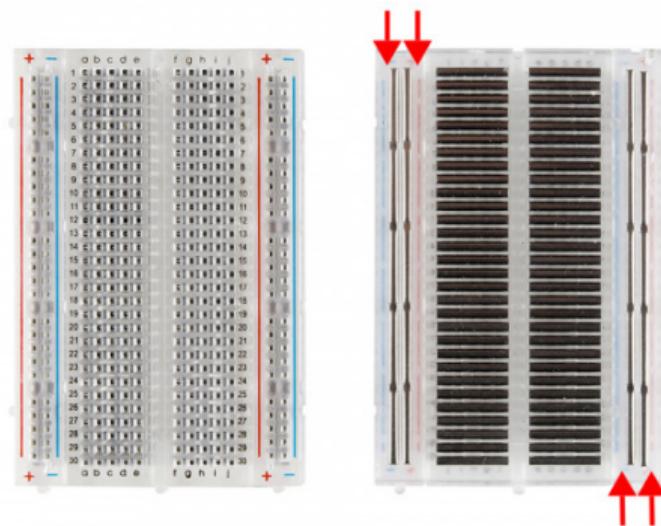
Mesure de tension



Mesure de courant



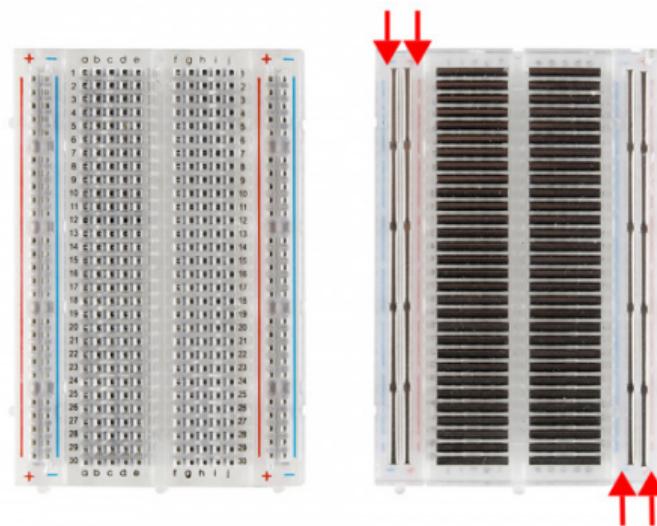
# Breadboard



Une breadboard se compose :

- de rangées horizontales
- de rails d'alimentation
- d'un ravin

# Breadboard



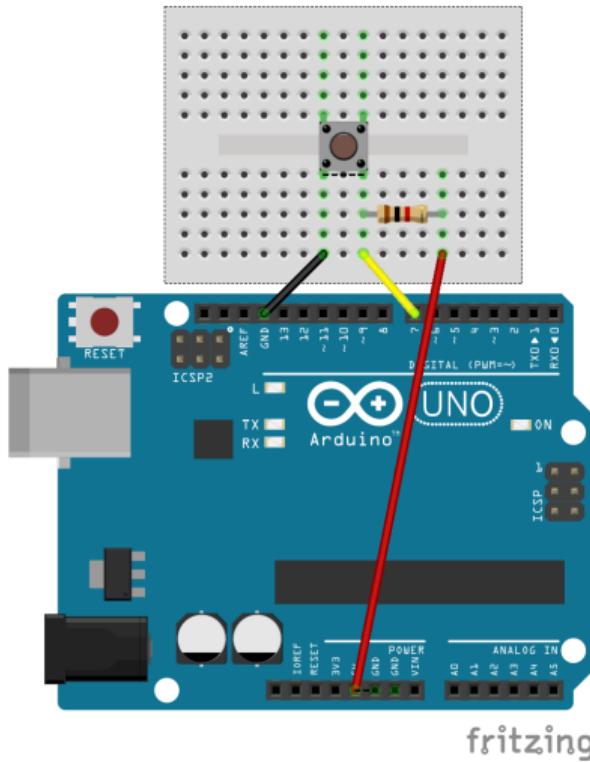
Une breadboard se compose :

- de rangées horizontales
- de rails d'alimentation
- d'un ravin

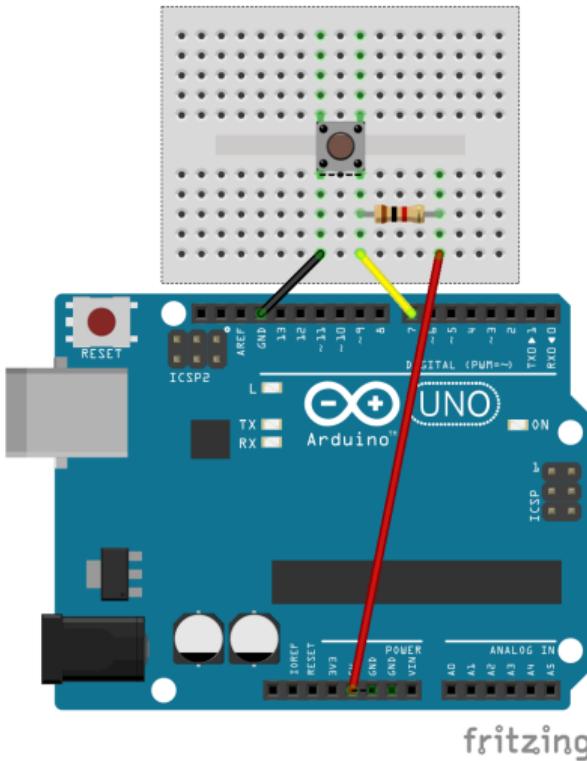
## Rails d'alimentation

Les rails d'alimentation de vos breadboards sont interrompus en leur milieu.

## Bouton poussoir



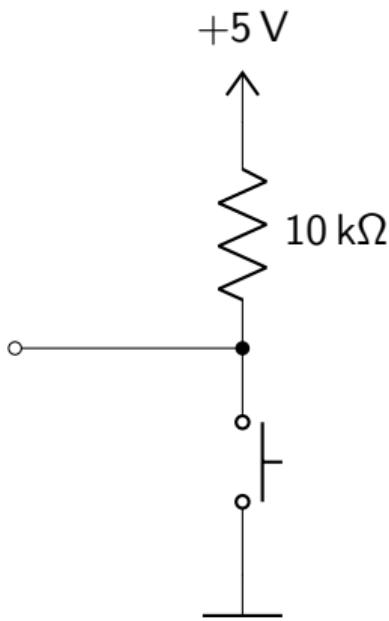
# Bouton poussoir



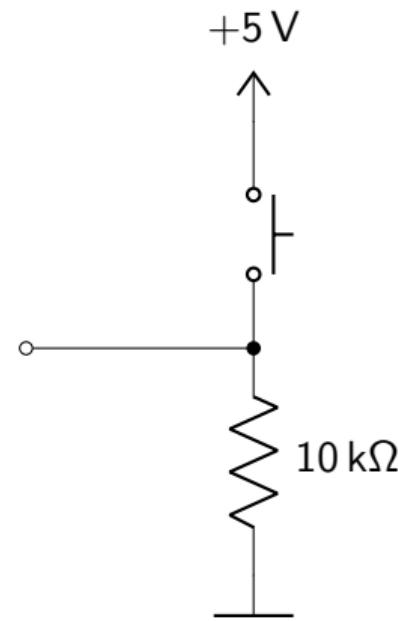
```
1 void setup() {  
2     Serial.begin(115200);  
3     pinMode(7, INPUT);  
4 }  
5  
6 void loop() {  
7     if (digitalRead(7) == LOW) {  
8         Serial.println("button pushed");  
9     }  
10    delay(200);  
11 }
```

# Résistance de tirage

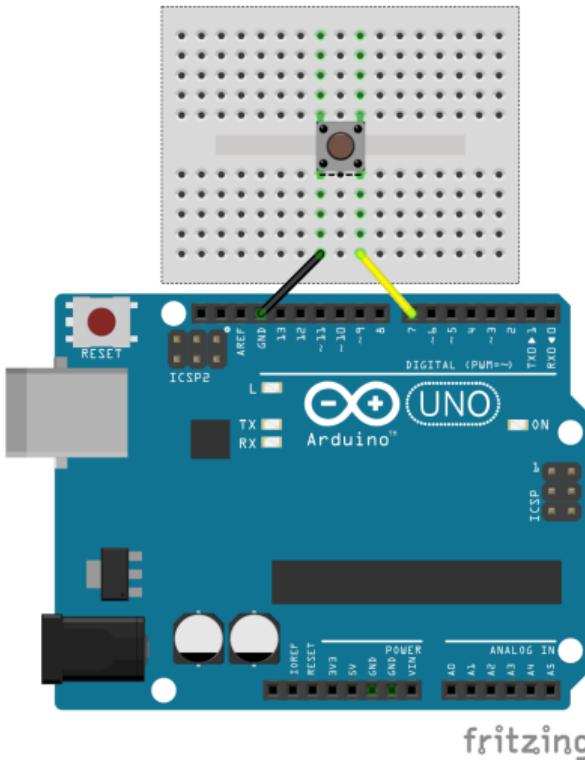
Résistance pullup



Résistance pulldown



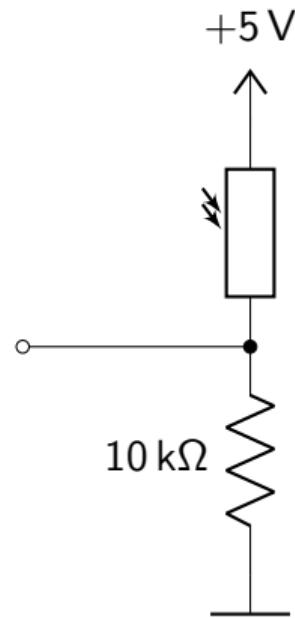
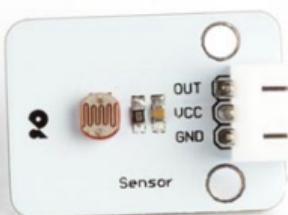
# Bouton poussoir avec pullup interne



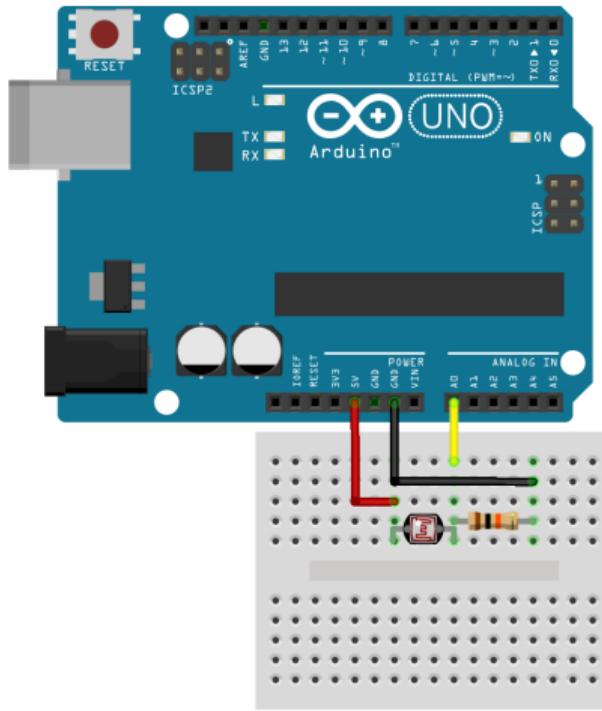
```
1 void setup() {
2     Serial.begin(115200);
3     pinMode(7, INPUT_PULLUP);
4 }
5
6 void loop() {
7     if (digitalRead(7) == LOW) {
8         Serial.println("button pushed");
9     }
10    delay(200);
11 }
```

# Photorésistance

VMA407

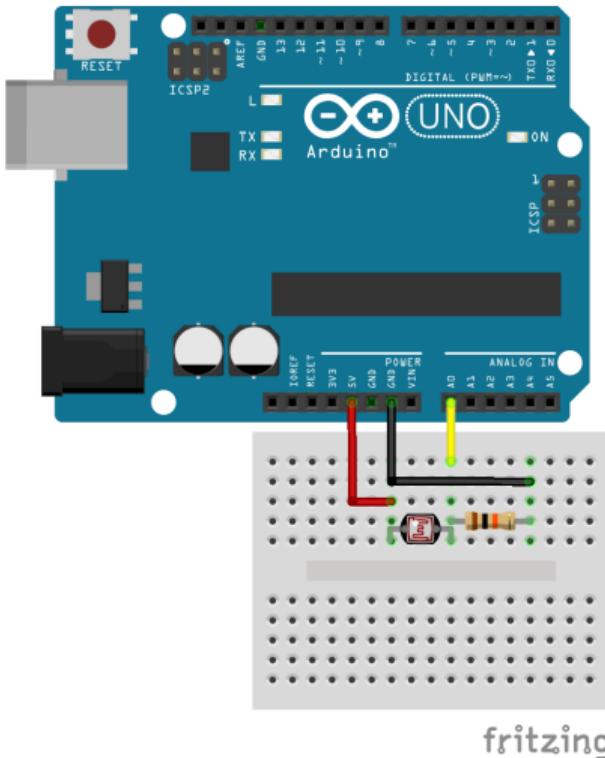


# Photorésistance



fritzing

# Photorésistance



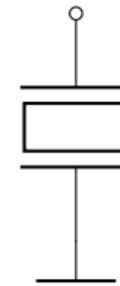
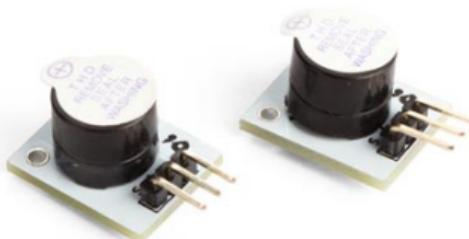
```
1 void setup() {  
2     Serial.begin(115200);  
3 }  
4  
5 void loop() {  
6     Serial.print("Light:");  
7     int lightValue = analogRead(A0);  
8     Serial.println(lightValue);  
9     delay(1000);  
10 }
```

## Valeur analogique

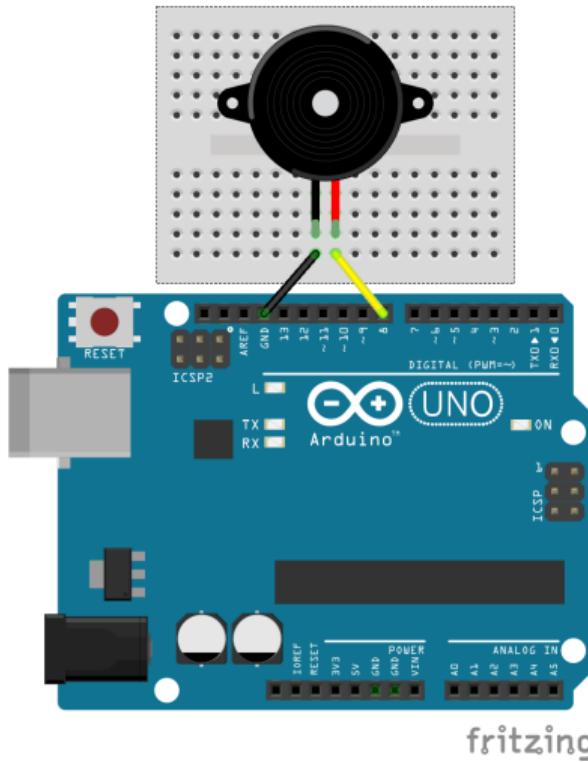
analogRead() retourne un entier entre 0 et 1024. 0 correspond à 0 V et 1024 à 5 V.

# Buzzer

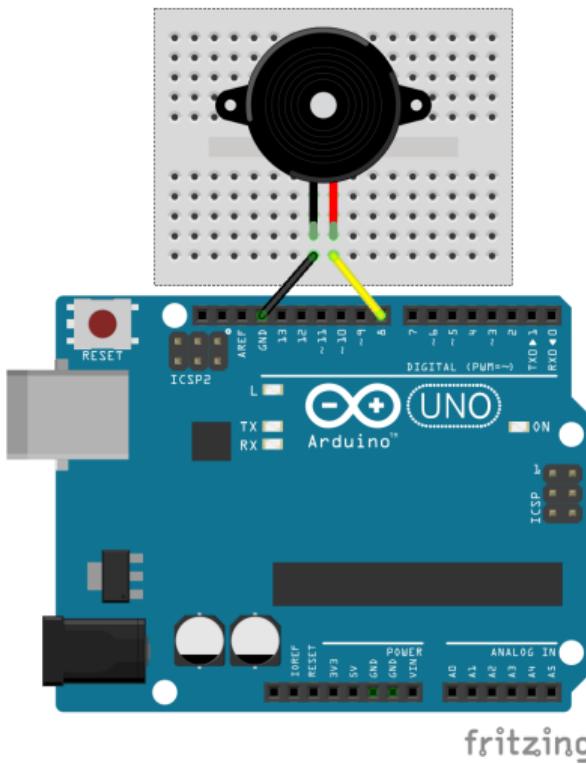
VMA319



# Buzzer



# Buzzer



```
1 void setup() {  
2     pinMode(8, OUTPUT);  
3 }  
4  
5 void loop() {  
6     tone(8, 2000, 300);  
7     delay(1000);  
8 }
```

# Télémètre à ultrason

VMA306



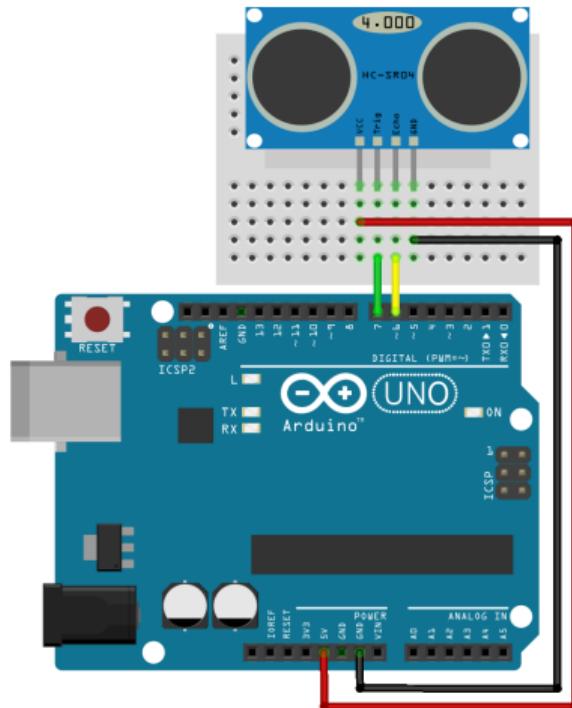
Librairie recommandée : NewPing

<https://bitbucket.org/teckel12/arduino-new-ping>

## Librairie externe

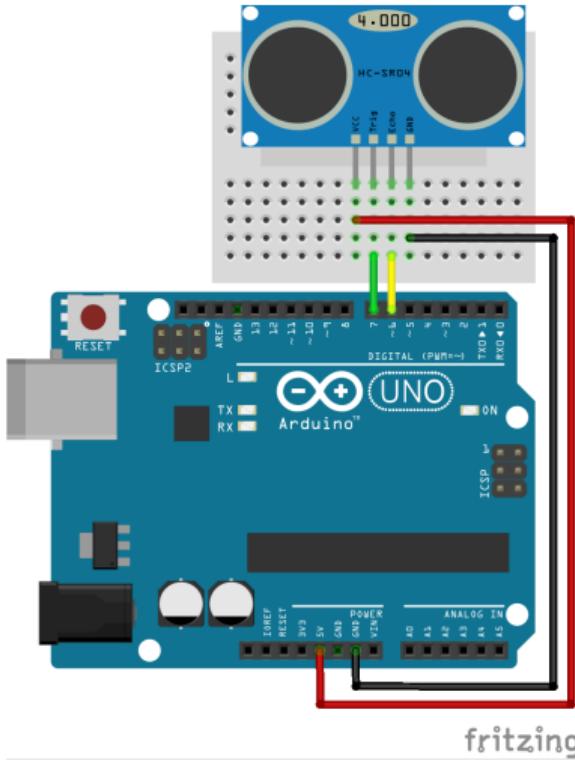
Téléchargez les librairies externes à l'aide du gestionnaire de librairie avant la première utilisation.

## Télémètre à ultrason



fritzing

# Télémètre à ultrason



```
1 #include <NewPing.h>
2 #define TRIG_PIN 7
3 #define ECHO_PIN 6
4 #define MAX_DISTANCE 200
5
6 NewPing sensor(TRIG_PIN, ECHO_PIN,
7                         MAX_DISTANCE);
8
9 void setup() {
10     Serial.begin(115200);
11 }
12
13 void loop() {
14     Serial.print("Distance: ");
15     Serial.print(sensor.ping_cm());
16     Serial.println(" cm");
17     delay(1000);
18 }
```

# Capteur de température et d'humidité

# Affichage à cristaux liquide

# Lecteur RFID

# Télécommunications

## 1 Introduction

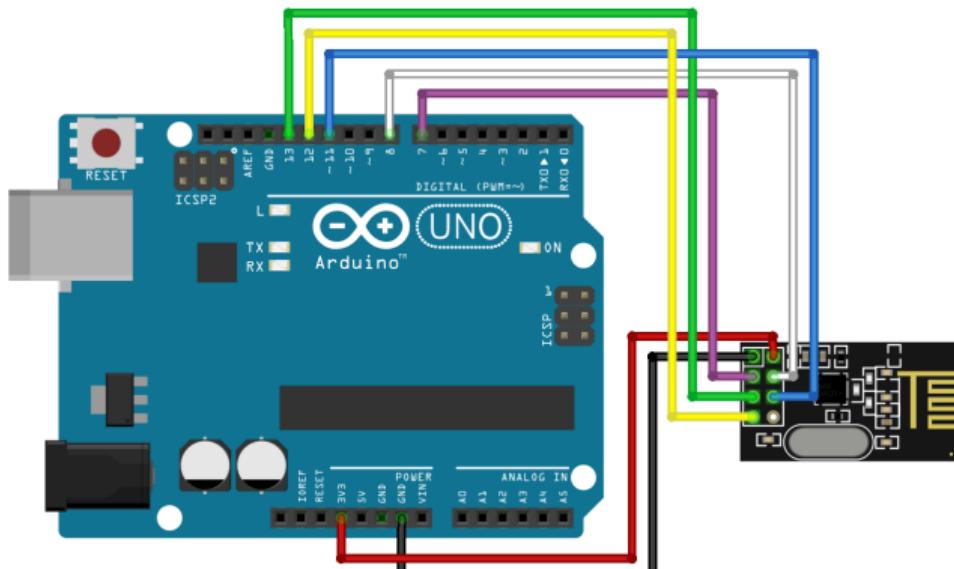
## 2 Arduino

## 3 Télécommunications

- Communications sans fil
- Outils de développement IoT

## 4 Conclusion

# Communication point-à-point



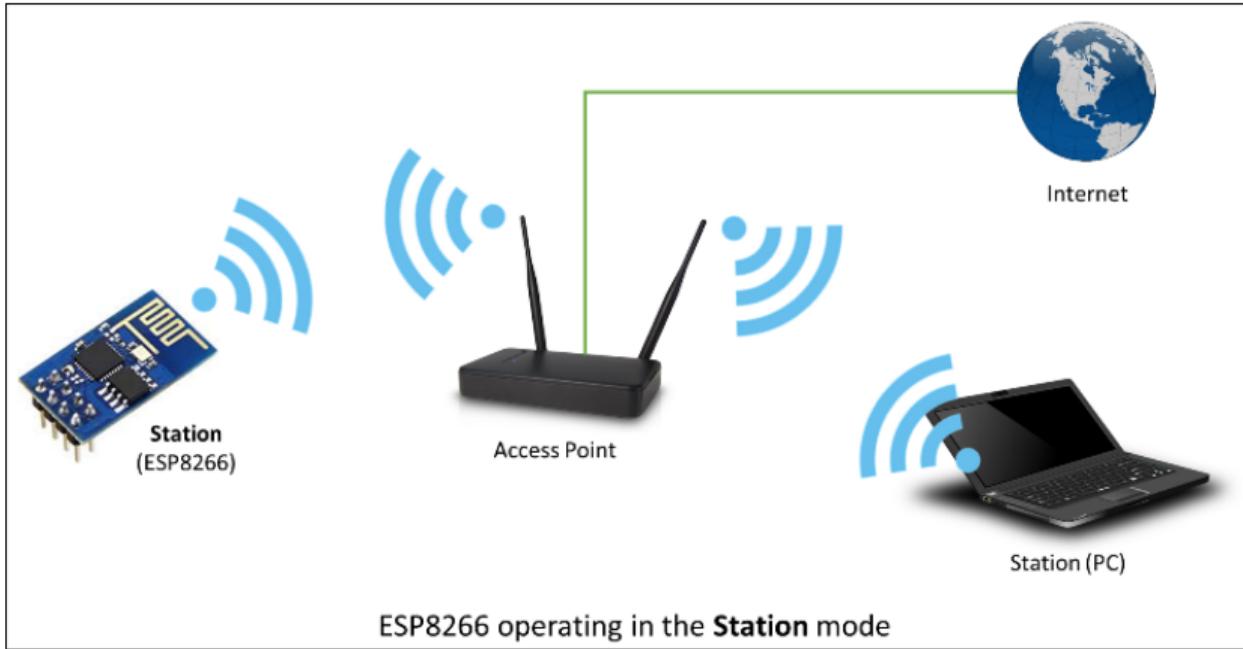
fritzing

<http://tmrh20.github.io/RF24/>

3.3 V

Ce module ne supporte pas une tension de plus de 3.3 V

# Communication WiFi



<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

# WiFi

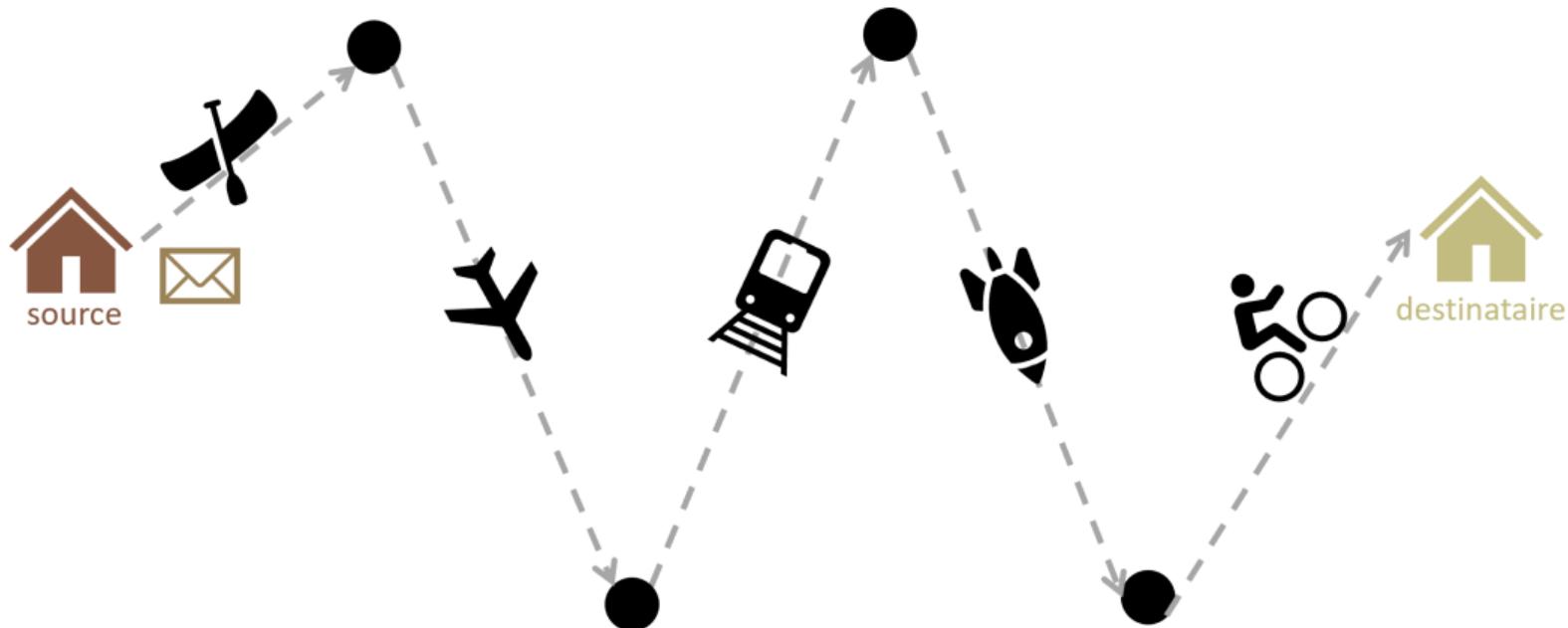
## Avantages

- Vitesse de transfert ++
- Portée +
- Coût des transmetteurs –

## Inconvénients

- Consommation d'énergie ++
- Facilité de configuration –

# Protocole Internet (IP)



# Adresse IP

## Adresse statique

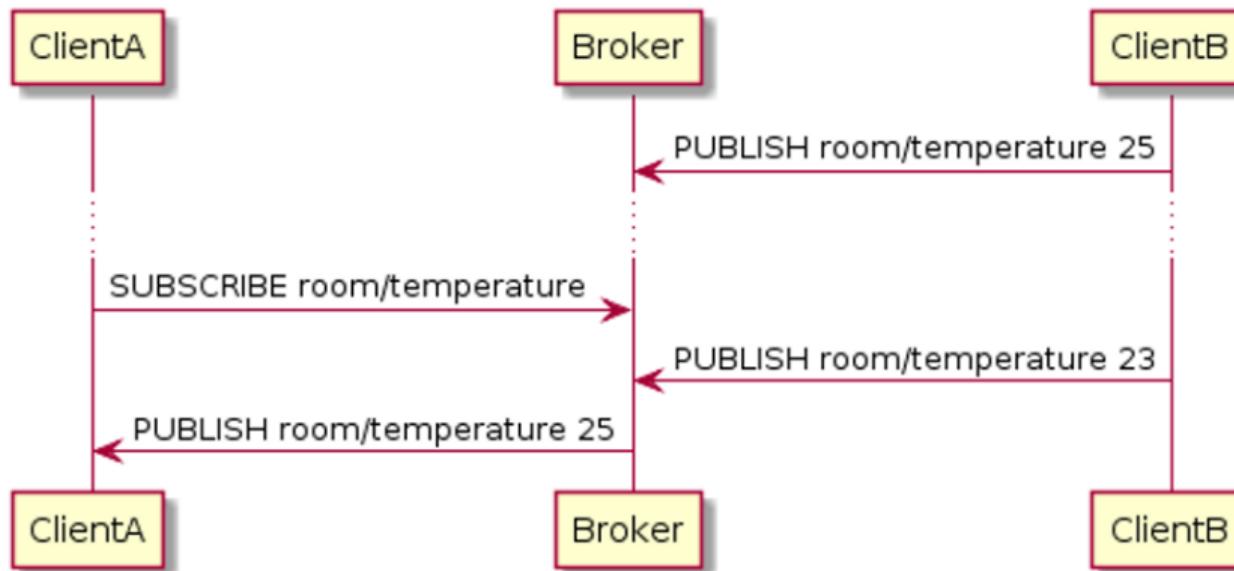
- Adresse : 192.168.243.11
- Passerelle : 192.168.243.1
- Masque de sous-réseau : 255.255.255.0

## Adresse dynamique (DHCP)

- Attribution automatique
- Configuration réseau



# MQTT



# Node-RED

The screenshot shows the Node-RED interface with a flow diagram on the left and a detailed view of a node on the right.

**Flow Diagram:**

- An "inject" node (purple) is connected to a "Filter dups" node (orange).
- The output of "Filter dups" is connected to a "msg.payload" node (green).
- A "Node-RED GitHub Hooks" node (purple) is connected to a "home/knolesley/github\_hooks.json" node (orange).
- The output of "home/knolesley/github\_hooks.json" is connected to a "Timestamp" node (blue).
- The output of "Timestamp" is connected to another "msg.payload" node (green).

**Node Properties:**

**Node:**  
Name: Slackhook  
Type: http in  
ID: 40c9104d.b08e4

**Properties:**

Provides an input node for http requests, allowing the creation of simple web services.

The resulting message has the following properties:

- msg.req : [http request](#)
- msg.res : [http response](#)

For POST/PUT requests, the body is available under `msg.req.body`. This uses the [Express bodyParser](#) middleware to parse the content to a JSON object.

By default, this expects the body of the request to be url encoded:

```
foo=bar&hi=that
```

To send JSON encoded data to the node, the content-type header of the request must be set to [application/json](#).

Note: This node does not send any response to the http request. This should be done with a subsequent HTTP Response node.

# Conclusion

1 Introduction

2 Arduino

3 Télécommunications

4 Conclusion

# Ce que nous avons vu

- Programmation Arduino
- Capteurs et actionneurs
- Multimètre et console série
- Communication sans fil

# Ce que nous n'avons pas abordé

- Sécurité
- Mise à jour
- Gestion d'énergie
- Communication sans fil basse consommation
- Packaging (chaleur, vibration)