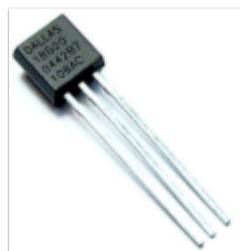


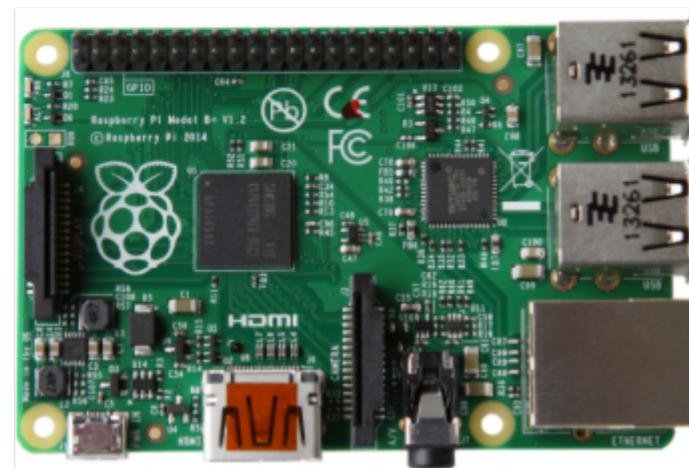


Construire une station météo en Node.js avec un Raspberry Pi et un Arduino

Vue d'ensemble



Le tout sans écrire une ligne de C !



Qui suis-je ?



- Consultant et formateur chez **Zenika Nantes**
- Mise en place de démarche dev-ops
- Développeur Java depuis 10 ans, en transition sur du grails
- Mes hobbies : DIY, Raspberry Pi, Arduino

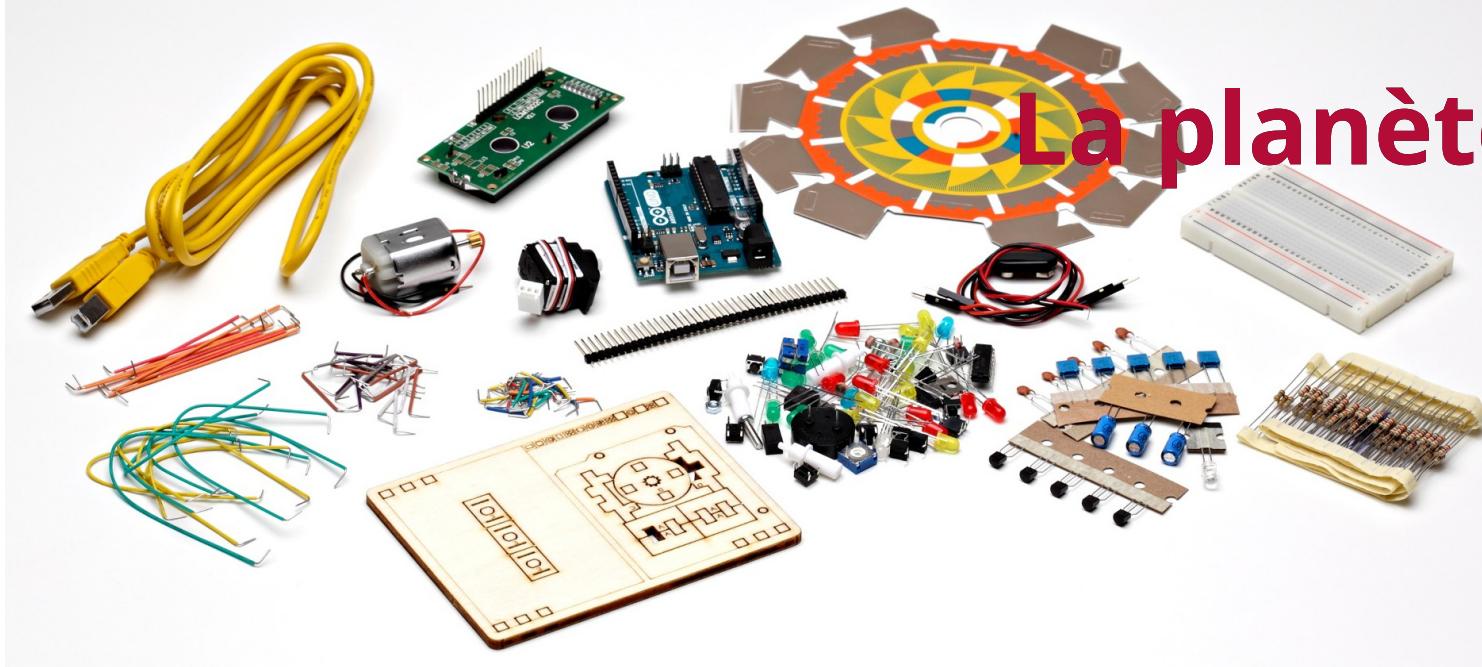
<http://www.monbook.tech>

<https://github.com/gmembre-zenika/>

<https://gitlab.com/coliss86/>



La planète DIY



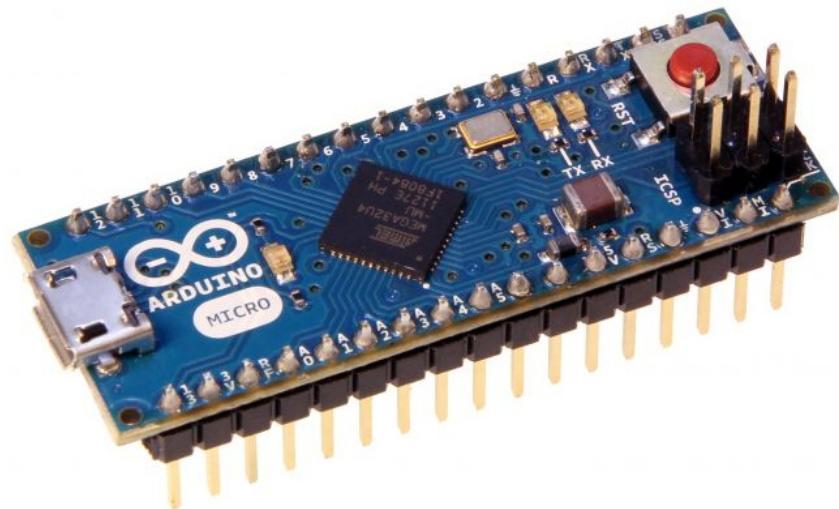
Arduino : qu'est ce ?



<http://www.arduino.cc/> et <http://www.arduino.org/>

Arduino is an open-source prototyping platform based on easy-to-use hardware and software.

- Fondé en 2008
- Caractéristiques d'un Arduino Micro
 - Microcontrôleur ATME^L 8 bit @ 16 MHz
 - 32 **Ko** de Flash
 - 2.5 **Ko** de RAM + 1 **Ko** de EEPROM
 - 20x E/S numériques
 - 12x E/S analogiques



Logiciels



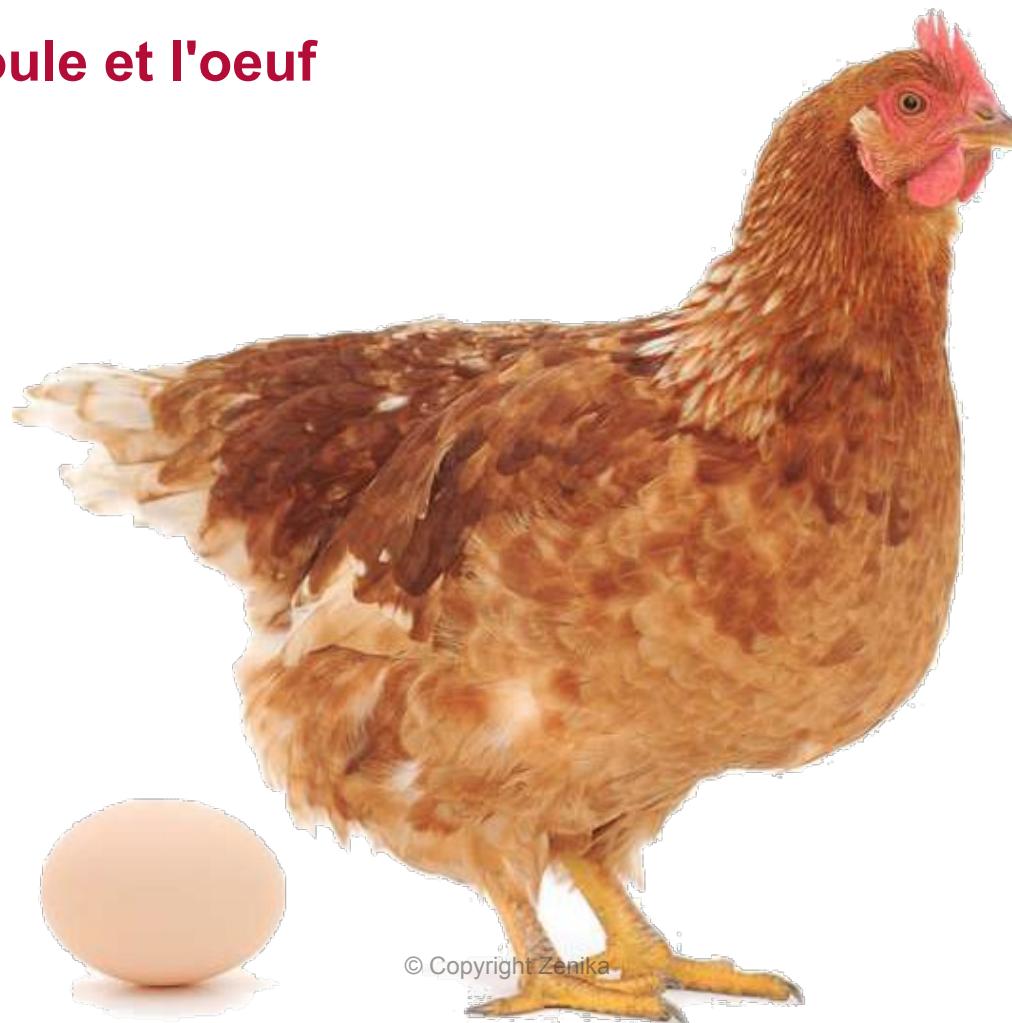
- Pas de système d'exploitation
- Uniquement le programme s'exécute dessus.
- Gestion des interruptions à coder si nécessaire
- Le debug se fait à coup de reboot et à la LED...
- Pas de BSOD :(

Raisons du succès ?



*Pour programmer un micro-contrôleur,
il faut un micro-contrôleur programmé...*

Problème de la poule et l'oeuf





Solutions apportées

- Arduino intègre sur une même platine :
 - un programmateur (déjà programmé...) + un microcontrôleur de "run"
 - une prise USB + des connecteurs multi-fonctions
- Schéma électrique libre
- Logiciels libres et gratuits :
 - bootloader (≈ "bios/UEFI d'un PC")
 - SDK + chaîne de cross compilation + IDE simple
 - Multi plateforme
 - Tarif : ≈ 5 → 30 €



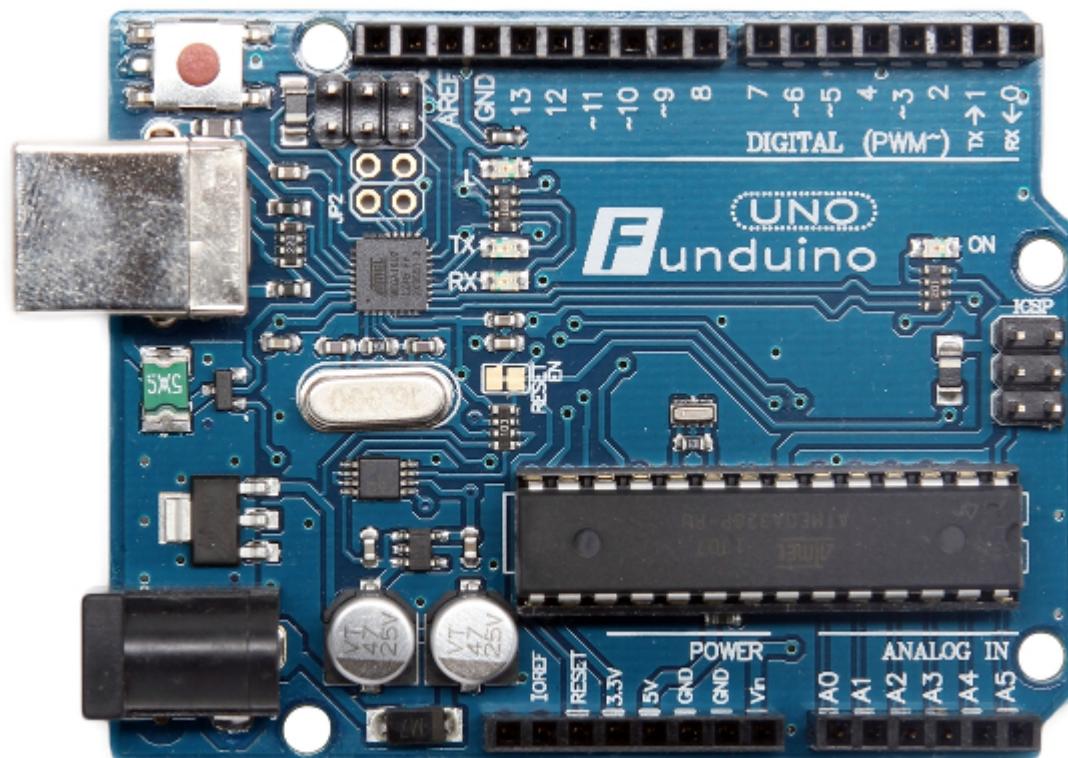
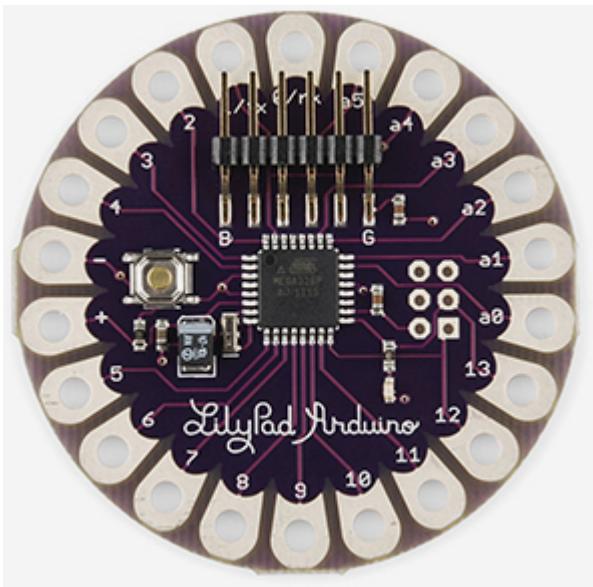
1 - 4



Autres modèles & clones

⇒ approche ***Open Hardware***

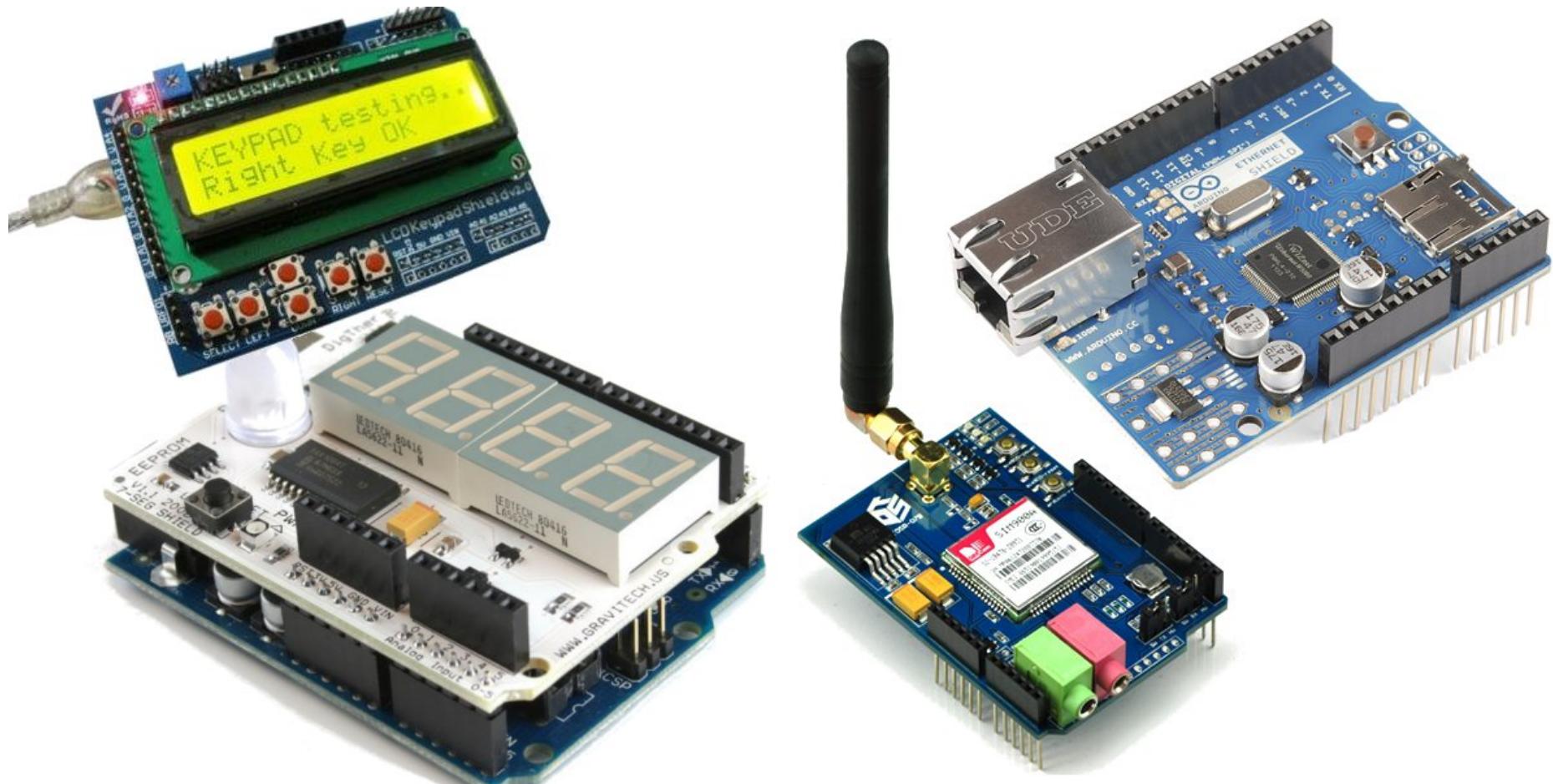
- Nombreuses variantes avec plus de I/O, plus de mémoire, un processeur Intel...
- La license libre de l'ensemble à donné naissance à de nombreux clone
 - Funduino, Nanode, Freeduino



Extensions



- Shield = carte d'extension se branchant sur les pins de la carte



Exemple de code



- Code écrit en C (**sketch**)
- **HelloWorld** = Clignotement d'une LED :

```
void setup() {  
    pinMode(13, OUTPUT); // initialisation de la pin 13 en sortie  
}  
  
// boucle infinie  
void loop() {  
    digitalWrite(13, HIGH); // allume la LED (=> niveau logique 1 = +5V)  
    delay(1000); // attente d'1 seconde  
    digitalWrite(13, LOW); // éteint la LED (=> niveau logique 0 = 0V)  
    delay(1000);  
}
```

L'API est riche malgré les contraintes de la plateforme :

- fonctions de lecture d'une tension, de génération de signaux basiques
- accès aux bus matériel les plus répandus (I2C, 1 Wire...)

Connexion



- USB avec un PC
- Alimentation de la carte + shield
- Port série émulé sur USB
 - Facile d'accès
 - Interopérable



Communication série



- Le SDK fournit des primitives pour écrire et lire sur le port série coté Arduino
- Coté hôte, il suffit d'ouvrir un terminal série
 - soit celui de l'IDE
 - un autre outil (**cu** sous linux, **screen** sous OS X)

```
Serial.begin(9600);  
  
Serial.print("Hello world.");  
Serial.print(78);  
Serial.print(1.23456);  
Serial.print('N');
```



```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.println("Hello world");  
    delay(1000);  
}
```

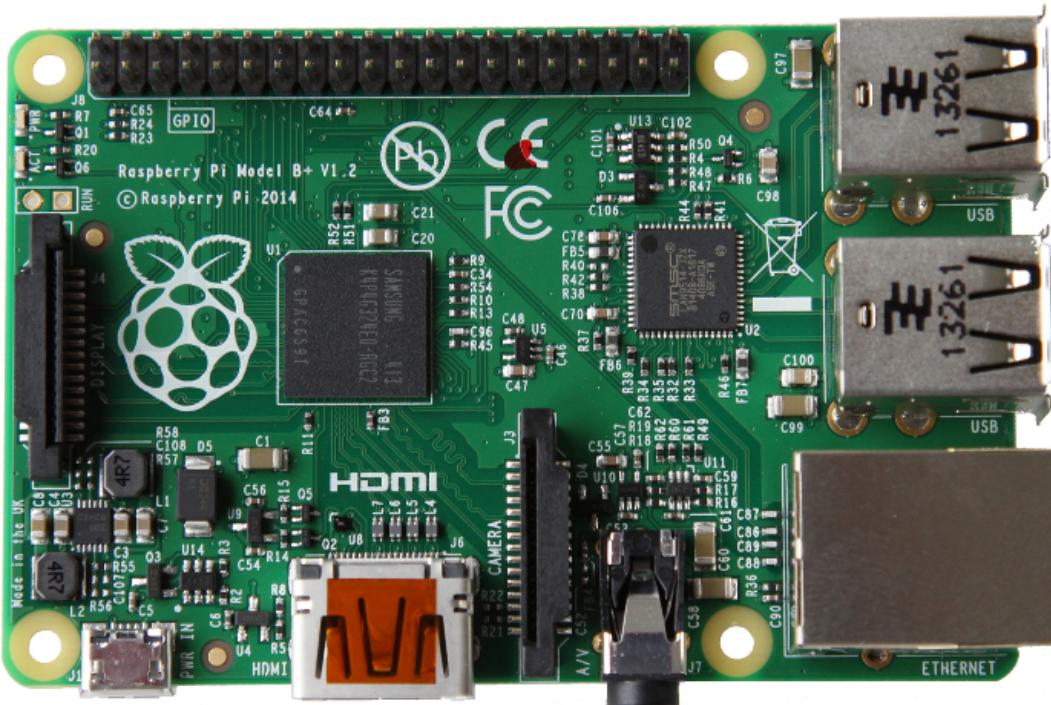
© Copyright Zenika

Raspberry Pi



<http://www.raspberrypi.org>

- 1er exemplaire distribué en février 2012, 10 millions d'unités vendu depuis
- Spécifications (modèle V2):
 - CPU Arm A7@900 Mhz Quad core
 - 1 Go de ram
 - 4xUSB 2.0
 - 1 eth 100 Mbit/s (bridge USB)
 - gpio, port caméra, HDMI...
- Tarif : ≈ 45 €



Usages



- ≈ Mini pc
- Compagnon idéal du bidouilleur en herbe
 - petite taille, faible consommation donc embarquable facilement
 - HTPC, robot, mangeoire d'oiseau, clusters Docker, sur laboratoire sur la station spatiale ...

OS et logiciels



- Sous linux :
 - Raspbian = port non officiel d'une debian sur **armhf**
<http://www.raspbian.org>
 - Archlinux
- et Windows 10...
- Communauté très active
 - Le projet Raspbian produit plus de paquets que le port officiel Debian





Comparaison avec du matériel connu

Plateforme	CPU	RAM	Flash
Arduino	Atmel@16 Mhz	2 Ko	32 Ko
Rpbi 2	Arm A7@900 Mhz Quad core	1 Go	-
Samsung S4	Arm A15@1.6 GHz Quad core + Arm A7@1.2 Ghz Quad core	2 Go	16 Go
MBP	Intel i5 x86_64@2.7 Ghz Quad core	8 Go	250 Go

Consommation en Idle

Plateforme	Idle (W)	Burn (W)
Arduino	0.170 (\rightarrow 0.011)	0.2
Rpbi 2	1.1	4.5
Tour de gamer	-	600-1000 W

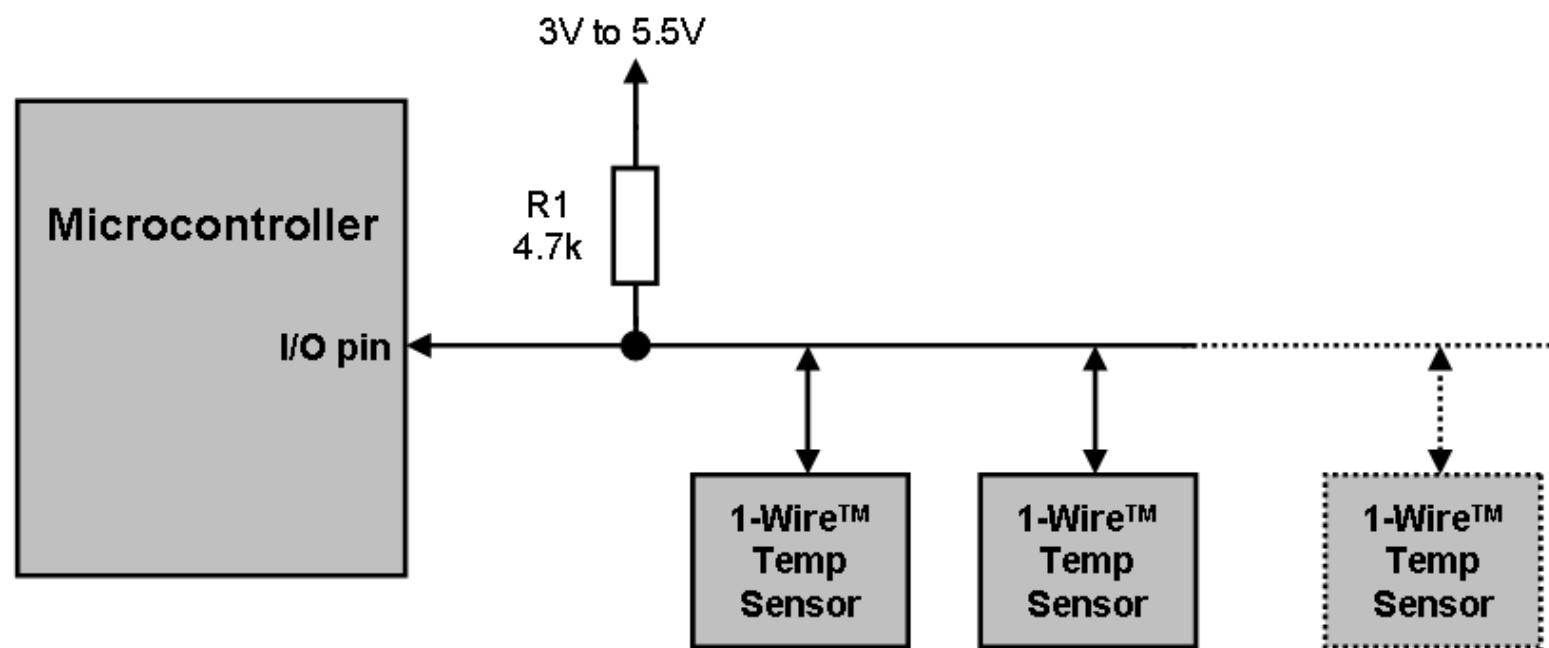
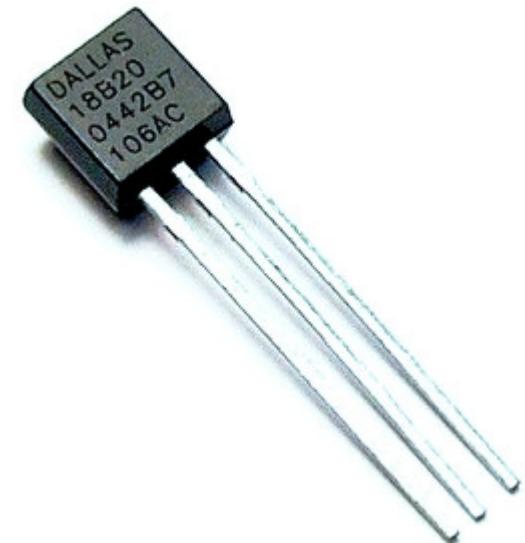
Idle : Pas d'activité CPU

Burn : Taux d'occupation CPU de 100%



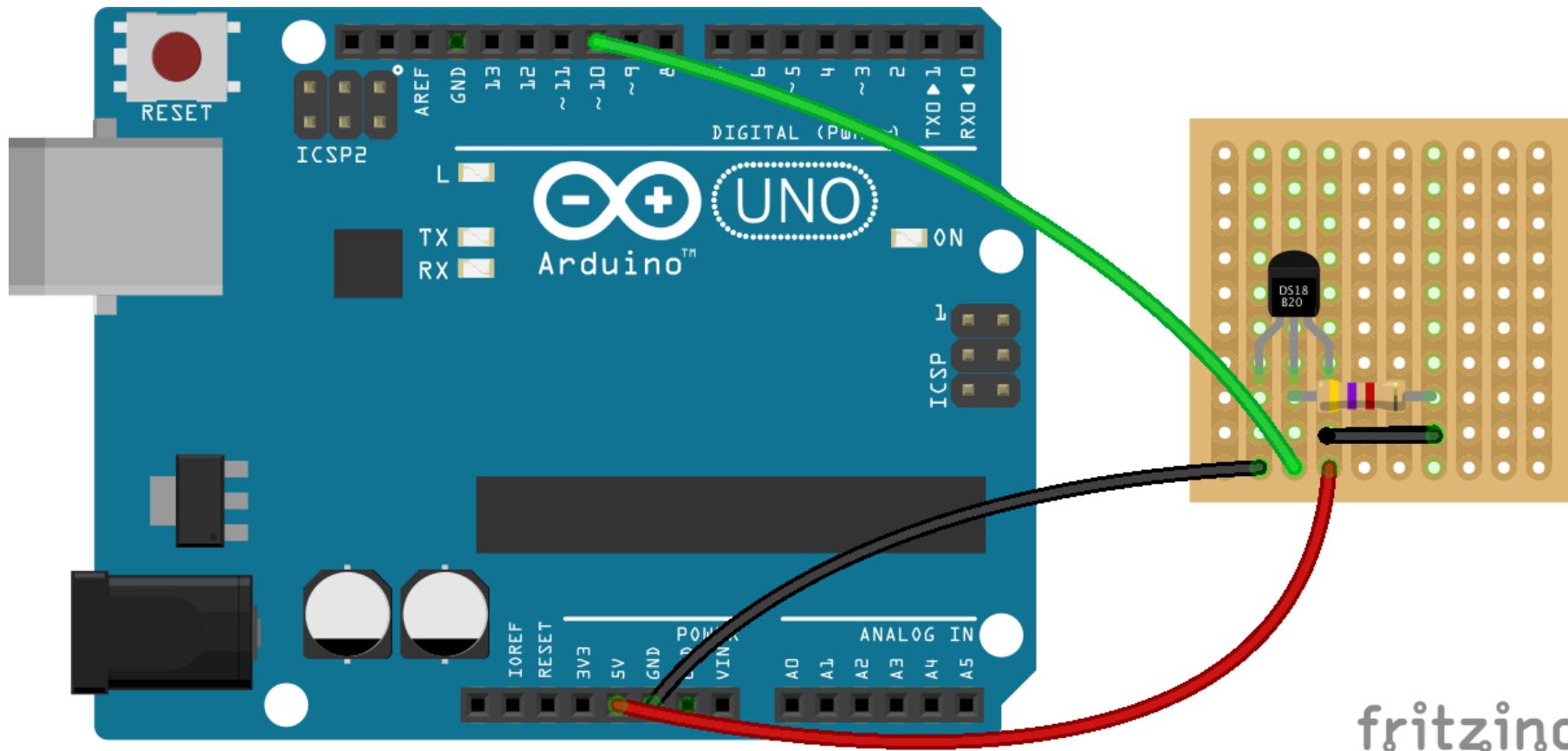
Thermomètre

- Sonde de température **DS18B20**
- Caractéristiques : $-55^{\circ}\text{C} \rightarrow +125^{\circ}\text{C}$, Résolution 9/12-bit
- Bus **1 Wire**, développé dans les années 90
 - 3 fils (+5V, masse, data), longeur max : 100 m
 - chaque composant à une adresse unique en 64 bit



Montage

- Montage très simple



fritzing



Sketch de lecture

- Sketch non trivial à cause du dialogue sur le bus **1 Wire**
- ~110 lignes → 8,5 ko de flash, ~360 octets de RAM
- exemple de sortie :

```
ROM = 28 65 DC 33 4 0 0 17
Chip = DS18B20
Data = 1 43 1 4B 46 7F FF D 10 BD CRC=BD
Temperature = 20.19 Celsius, 68.34 Fahrenheit
```

Limites de l'Arduino



- L'Arduino dispose de très peu de ressources
 - Contraintes : le programme doit tenir dans 32 Ko de flash et 2,5 Ko de RAM...
 - Exemples : relevé de sondes de températures et pilotage de radiateur, drône quadricoptère, machine enigma...
 - Connectivité avec le monde extérieur très limitée (pas d'eth ou wifi sur les modèles les plus simples)
- Dès qu'il s'agit de faire plus intelligent (robot autonome, reconnaissance de voix, graphique temps réel...) une CPU plus puissante va être nécessaire.

 Utiliser un hôte qui utilise l'Arduino comme un esclave en lui émettant des ordres via le port série

Communication série

- 1ère solution : communiquer directement avec le port série en émettant des ordres en format texte
 - Simplicité : 
 - Documentation : 
 - Evolutivité :  (maintient du code Arduino et hôte)
 - Exemple de sketch : <https://gitlab.com/coliss86/arduino-controller>

1

Help

Command available :

h - help

t - temperature



Solution plus évoluée

- Utiliser un protocole binaire d'échange sur le port série
 - Transmission plus fiable
 - Sketch spécifique sur l'Arduino
 - Librairies clientes côté hôte
 - Simplicité : 
 - Documentation : 
 - Evolutivité : 

Librairies



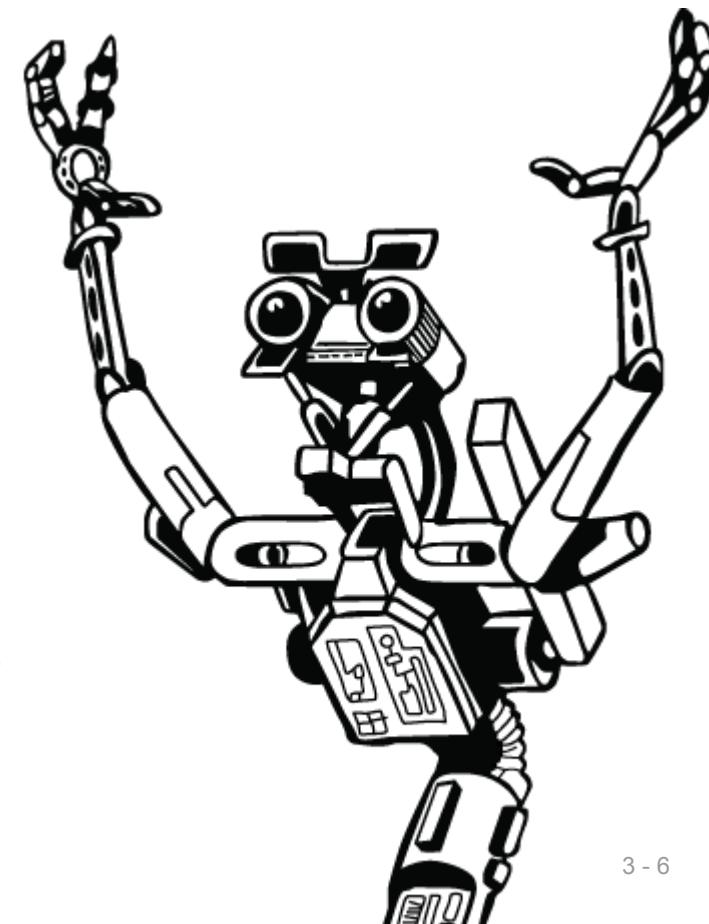
- **Firmata** : <https://github.com/firmata/arduino>

- Protocole très similaire au MIDI (commande de 8 bits, data : 7 bits)
- Implémentation pour plusieurs micro-contrôleurs
- Sketch rendant l'Arduino esclave d'un hôte
⇒ l'Arduino n'est plus autonome
- Documenté, libre et open source

- **Johnny-five** : <http://johnny-five.io/>

- Lib Node.js s'interfacent avec ce protocole
- Documentée, illustrée, libre et open source

⇒ Le Raspberry Pi exécutera le programme en Node.js pour piloter l'Arduino



Johnny-five



- **Johnny-five** intègre une API de très haut niveau s'interfaisant avec :
 - de nombreux composants du marché : servo, relais, moteur pas à pas, altimètre, LCD...
 - les bus les plus répandus : I2C et OneWire
- Hello world :

```
var five = require("johnny-five");
var board = new five.Board();

board.on("ready", function() {
  var led = new five.Led(13);

  // change l'état de la LED toutes les 500 ms
  led.blink(500);
});
```



Relevé de température



- Mise en oeuvre avec la sonde

```
board.on("ready", function() {
  var thermometer = new five.Thermometer({
    controller: "DS18B20",
    pin: 10,
    freq: 1000
  });

  thermometer.on("data", function() {
    console.log(this.celsius + "°C");
  });
});
```

Note : le code équivalent pour Arduino fait 100 lignes avec :

```
int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  raw = raw << 3;
  if (data[7] == 0x10) {
    raw = (raw & 0xFFFF) + 12 - data[6];
    ...
}}
```

Graphiques



- L'objectif final est de tracer des courbes de température



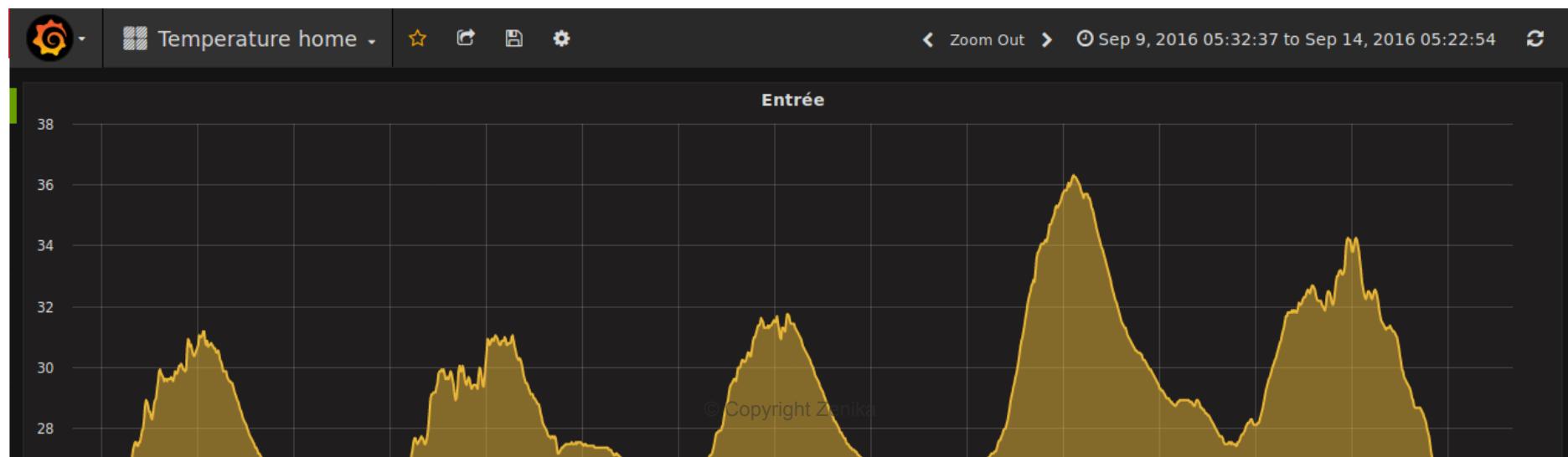
- **Influxdb** : <https://www.influxdata.com/>

- base nosql temporelle : les données sont indexées sur un timestamp précis la nanoseconde

- **Grafana** : <http://grafana.org/>



- Tableau de bord compatible avec de nombreuses sources de métriques
 - Mise en forme des données sous forme de graphique





Alimentation de la base

- Appel REST sur la base **Influxdb** pour ajouter périodiquement les relevés de température
 - Valeurs de la forme : <série> value=<valeur> [ts en ns]

```
var influx = require('influx');

var client = influx({host: 'raspberrypi', port: 8086, protocol: 'http',
database: 'temperature', username: '', password: ''});

client.writePoint('temp', {value: valeur}, {}, function(err) {
  if (err) console.log(err);
});
```



All together

Programme complet, lancé depuis le **Raspberry Pi** :

```
var influx = require('influx');
var five = require('johnny-five');

var client = influx({host: 'raspberrypi', port: 8086, protocol: 'http',
database: 'temperature', username: '', password: ''});
var board = new five.Board();

board.on('ready', function() {
  var thermometer = new five.Thermometer({
    controller: 'DS18B20',
    pin: 10,
    freq: 1000
  });

  thermometer.on('data', function() {
    console.log(this.celsius + '°C');
    client.writePoint('temp', {value: this.celsius}, {}, function(err) {
      if (err) console.log(err);
    });
  });
});

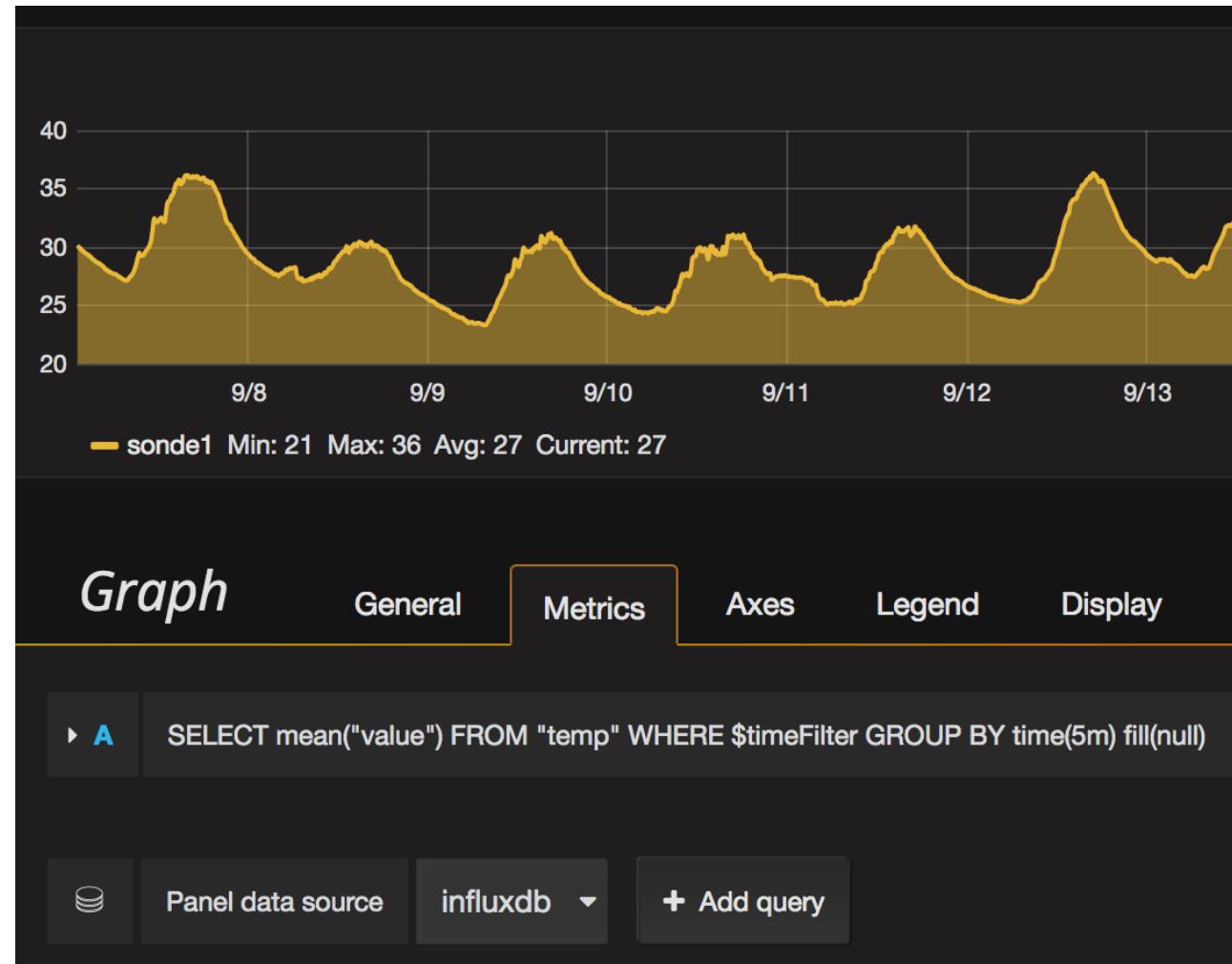
});
```

All together



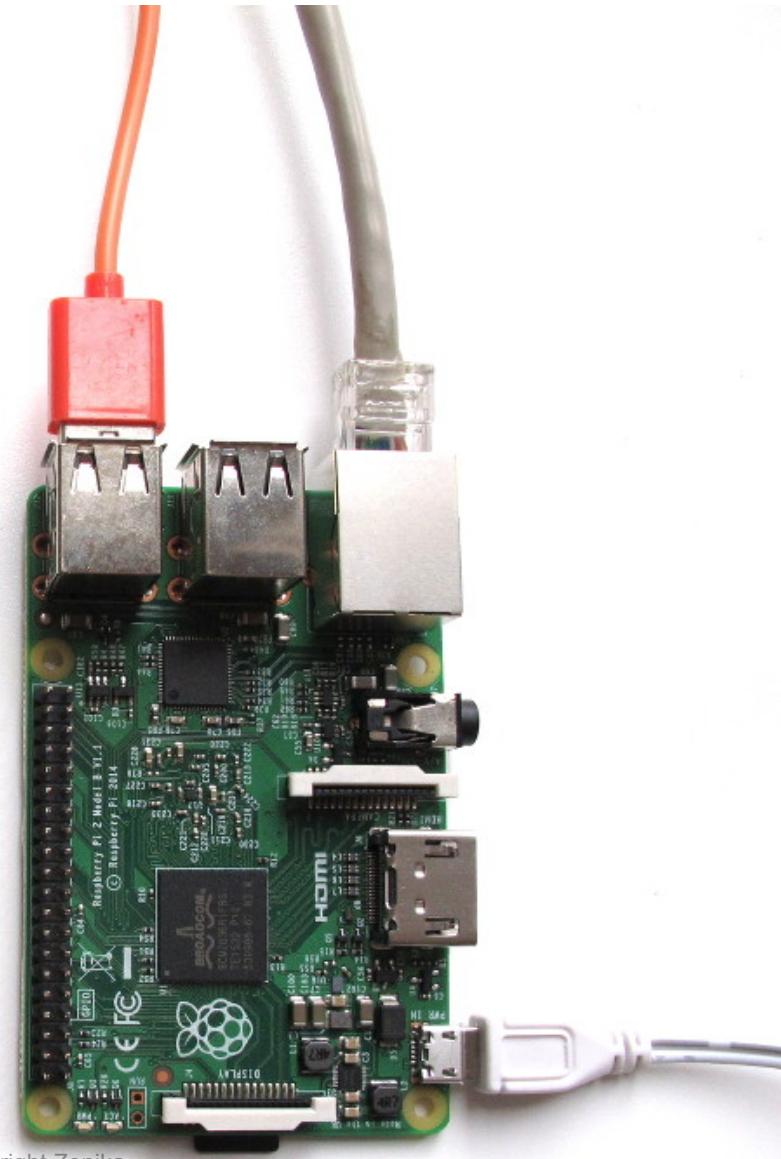
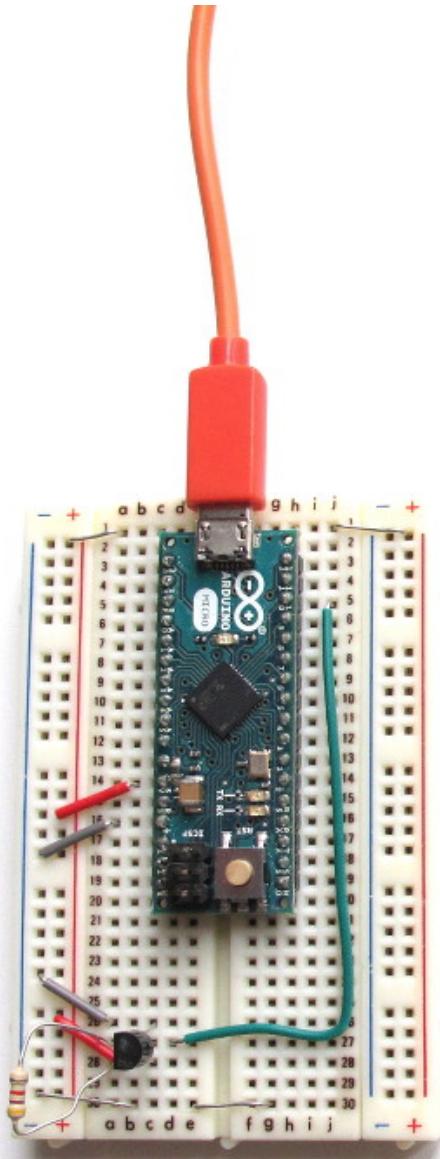
- Grafana

The screenshot shows the 'Edit data source' page in Grafana. At the top, there is a navigation bar with a gear icon and the text 'Data Sources'. Below it, a table lists a single data source named 'influxdb' of type 'InfluxDB'. The 'Default' checkbox is checked. Underneath the table, there is a section titled 'Http settings' containing fields for 'Url' (set to 'http://raspberrypi:8086'), 'Access' (set to 'direct'), and 'Http Auth' (with options for 'Basic Auth' and 'With Credentials').



All together

- Assemblage électronique

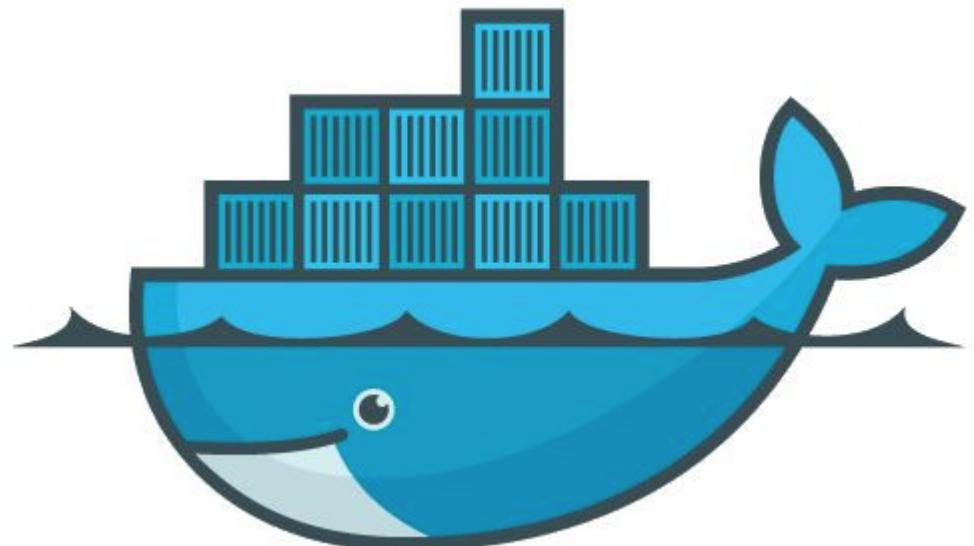


All together



- **Grafana** et **Influxdb** sont déployés sous forme de conteneurs **Docker** sur le **Raspberry Pi**

⚠ Obligation de reconstruire les images from scratch car les images disponibles sur le Docker Hub sont en très grande majorité pour **x86 amd64** et non **armhf**

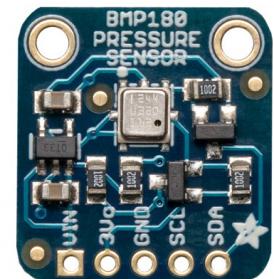
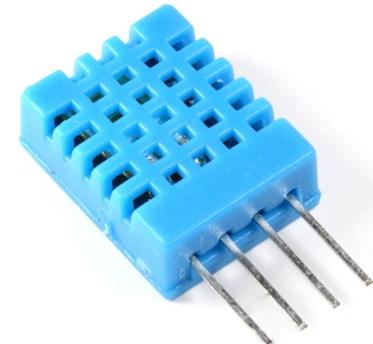


Et ensuite ?



- A ce niveau, il va devenir ais  d'ajouter des nouveaux capteurs :

- **DHT11** : mesure d'humidit 
- **BMP180** : capteur de pression atmosph rique
- Capteur d'humidit  de sol
- ...



Démo



Questions ?

Appelez-moi Johnny 5 (1988)

