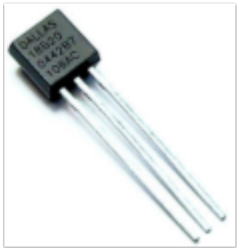
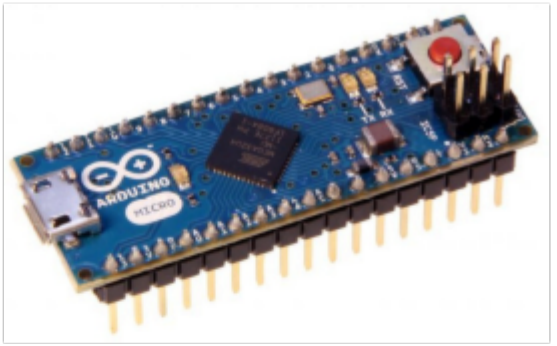
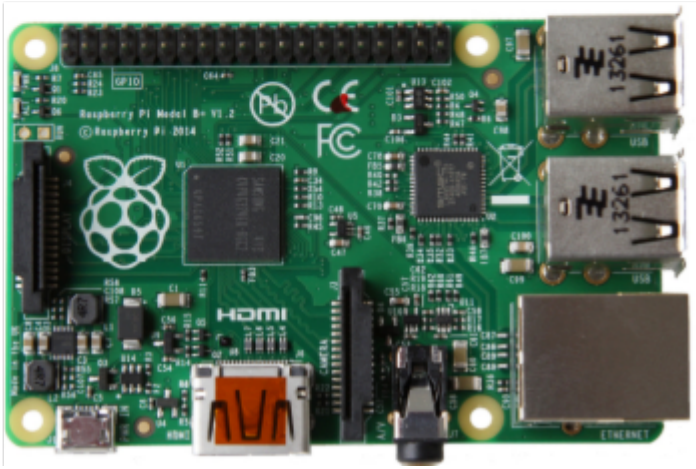
Construire une station météo en Node.js avec un Raspberry Pi et un Arduino

Vue d'ensemble





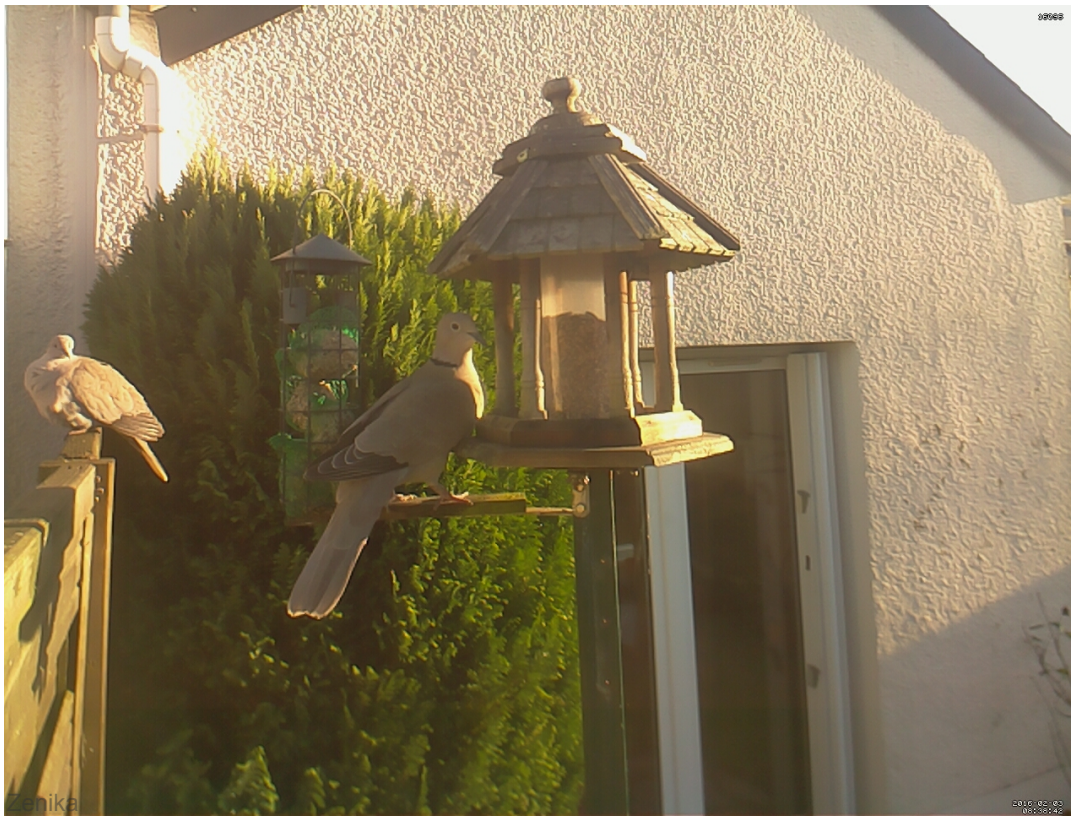
Le tout sans écrire une ligne de C !



Qui suis-je ?

- ***Guillaume Membré***
- 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/

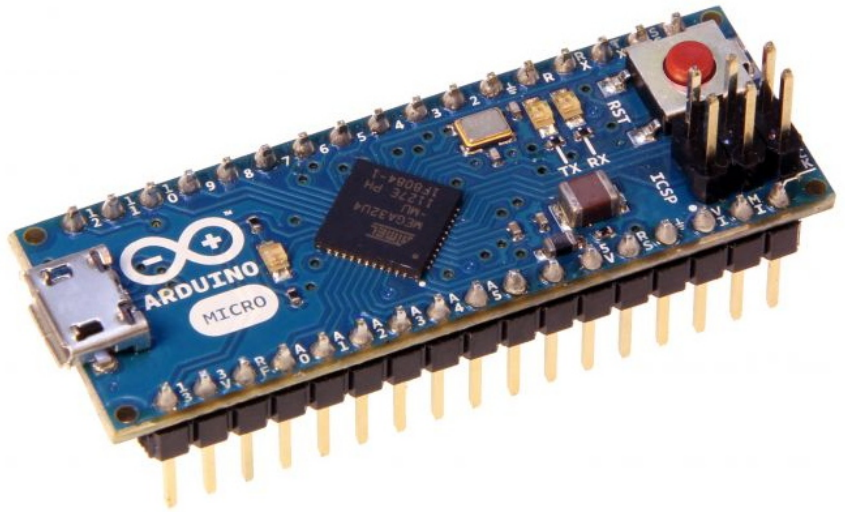


Arduino : qu'est ce ?

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
  - Microcontroleur ATMEL 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
  - Tarif : ≈ 5 → 30 €



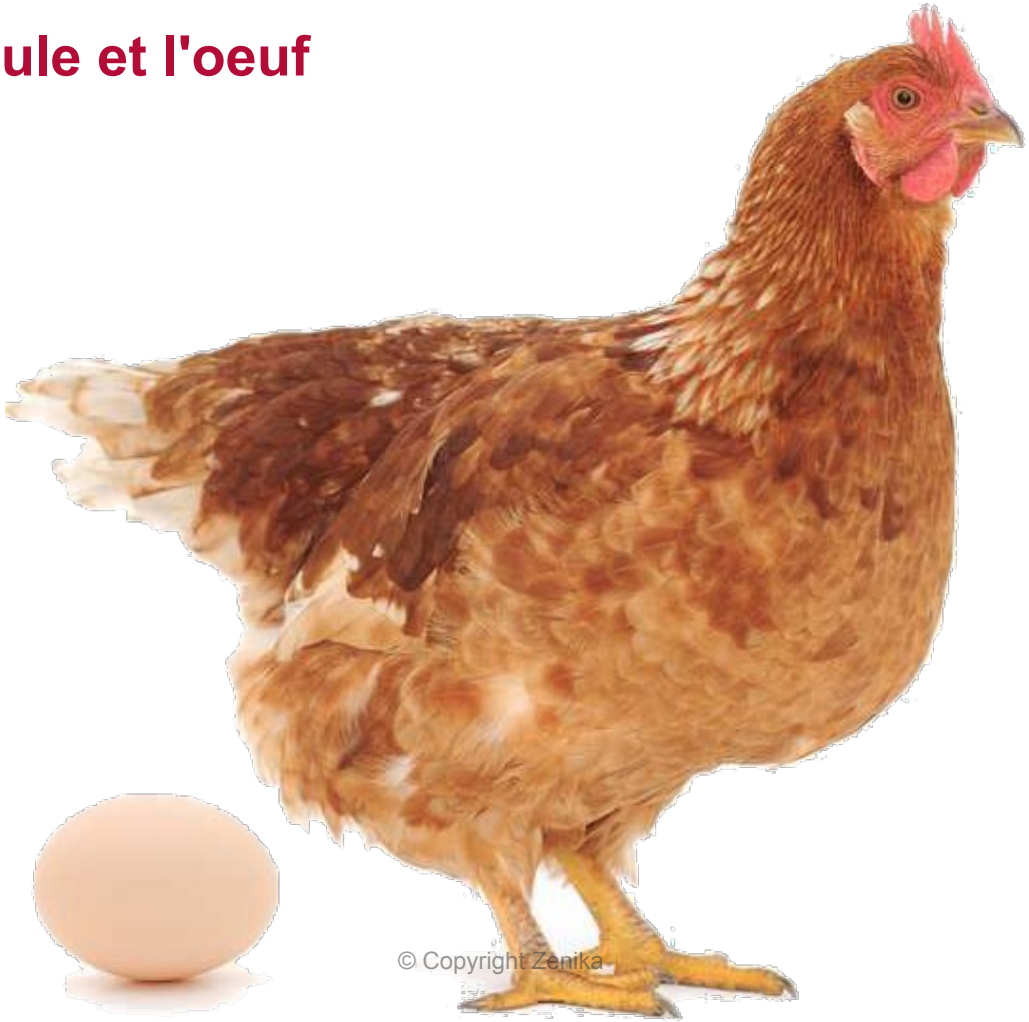
Arduino

- Philosophie "Open hardware"
  - Shéma electronique libre
  - Logiciels libres et gratuits :
    - bootloader (≈ "bios/UEFI d'un PC")
    - SDK + chaine de cross compilation + IDE simple
- Logiciels
  - Pas de système d'exploitation
  - Uniquement le programme s'exécute dessus.
  - Gestion des interruptions à coder si necessaire
  - 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



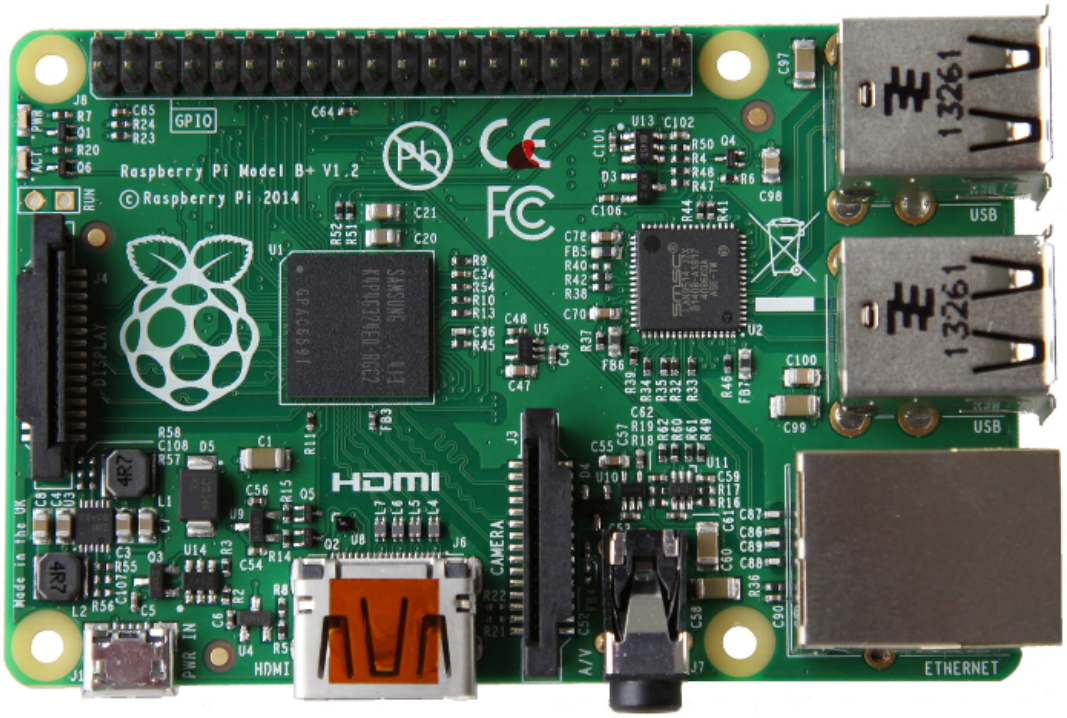
Connexion

- USB avec un PC
- Alimentation de la carte
- Port série émulé sur USB
- Le SDK fournit des primitives de communication série
- Coté hôte, il suffit d'ouvrir un terminal série
  - soit celui de l'IDE
  - `cu` sous linux
  - `screen` sous OS X

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...
  - pas de RTC :'(
- Tarif : ≈ 45 €



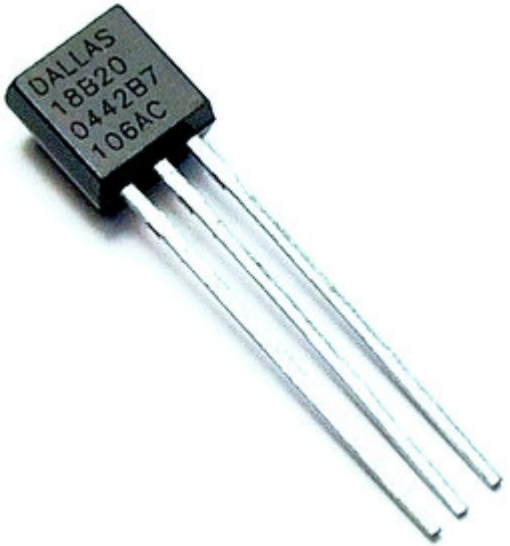
Usages

- ≈ 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 ...
- Sous linux :
  - Raspbian = port non officiel d'une debian sur **armhf**
    http://www.raspbian.org
  - Archlinux
- et Windows 10...

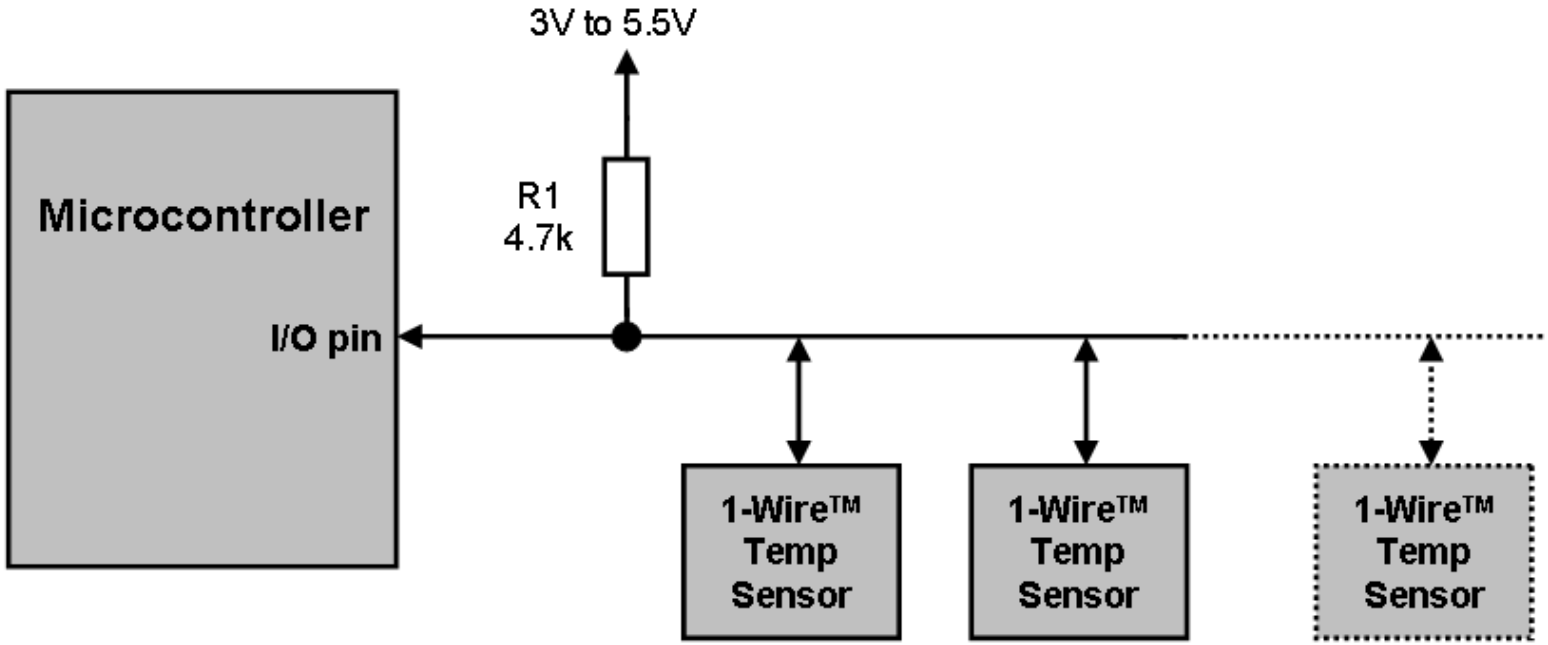
Comparaison avec du matériel connu

Plateforme	CPU	RAM	Flash
Arduino	Atmel@16 Mhz	2 **Ko**	32 **Ko**
Rpbi 2	Arm A7@900 Mhz Quad core (CPU + GPU)	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

Thermomètre



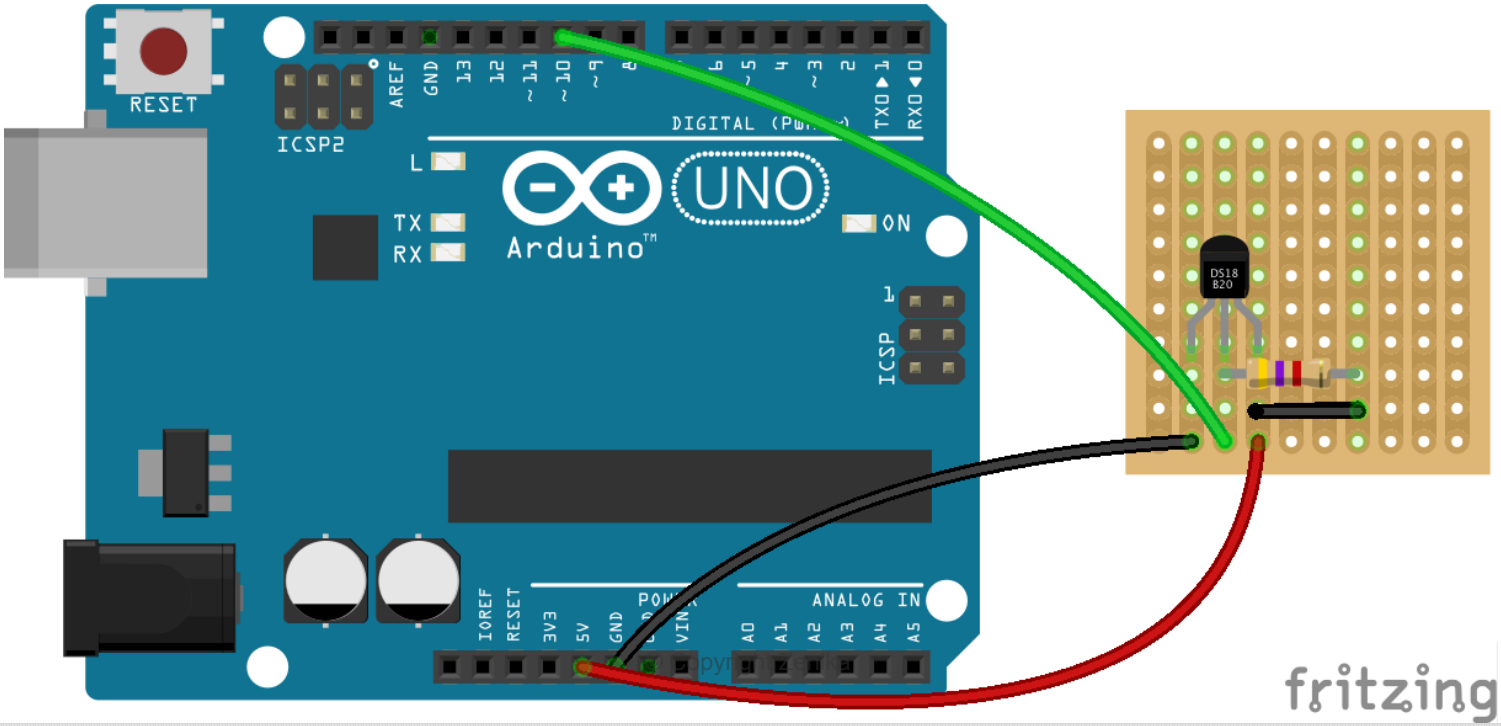
- Sonde de température **DS18B20**
- Caractéristiques : -55°C → +125°C, Résolution 9/12-bit
- Bus **1 Wire**, développé dans les années 90
  - 3 fils (+5V, masse, data), longueur max : 100 m
  - chaque composant à une adresse unique en 64 bit



Utilisation

- Montage très simple
- sketch : écrit en C, ~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 temperatures 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 simple)
- 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

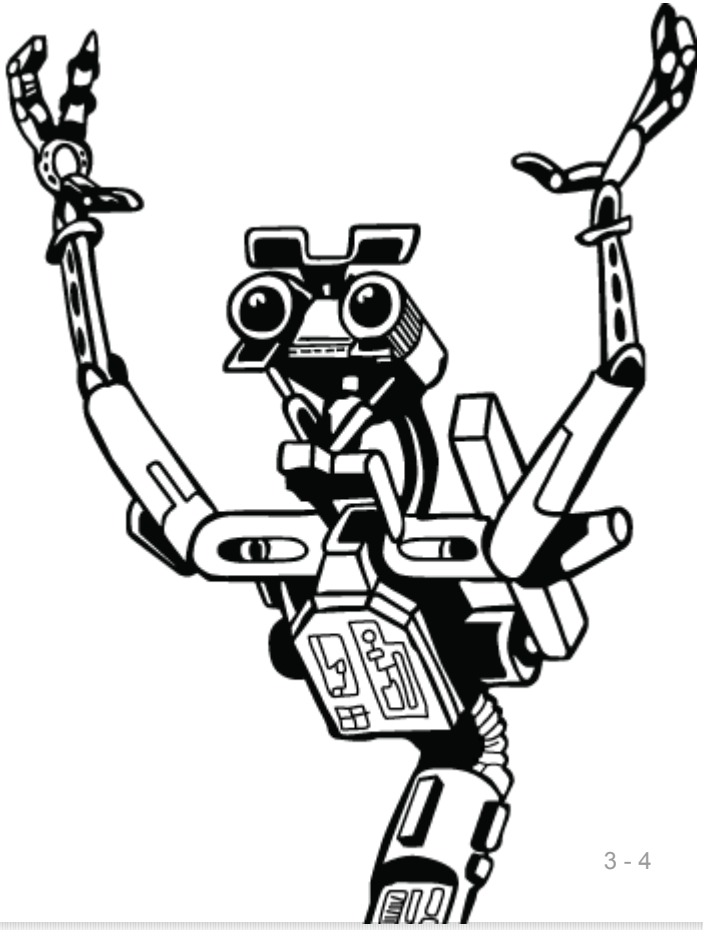
- 1ère solution : communiquer directement avec le port série en émettant des ordres en format texte
  - Simplicité : ★★☆
  - Documentation : ★★★
  - Evolutivité : ★☆☆ (maintient du code Arduino et hôte)
  - Exemple de sketch : *https://gitlab.com/coliss86/arduino-controller*

```
> h
    ___           _       _
   / _ \         | |     (_)
  / /_\ \_ __ __| |_   _ _ _ __   ___
  |  _  | '__/ _` | | | | | '_ \ / _ \
  | | | | | | (_| | |_| | | | | | (_) |
  \_| |_/_|  \__,_|\__,_|_|_| |_|\___/

Help
Command available :
      h - help
      t - temperature
```

Solution plus évoluée

- **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
- **Johnny-five** : *http://johnny-five.io/*
  - Lib Node.js s'interfacant avec ce protocole

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



Mise en oeuvre

- Documenté, illustré, libre et open source
- Utiliser un protocole binaire d'échange sur le port série
  - Transmission plus fiable
  - Sketch spécifique sur l'Arduino
  - Librairies clientes coté hôte

  - Simplicité : 
  - Documentation : 
  - Evolutivité : 

Johnny-five

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

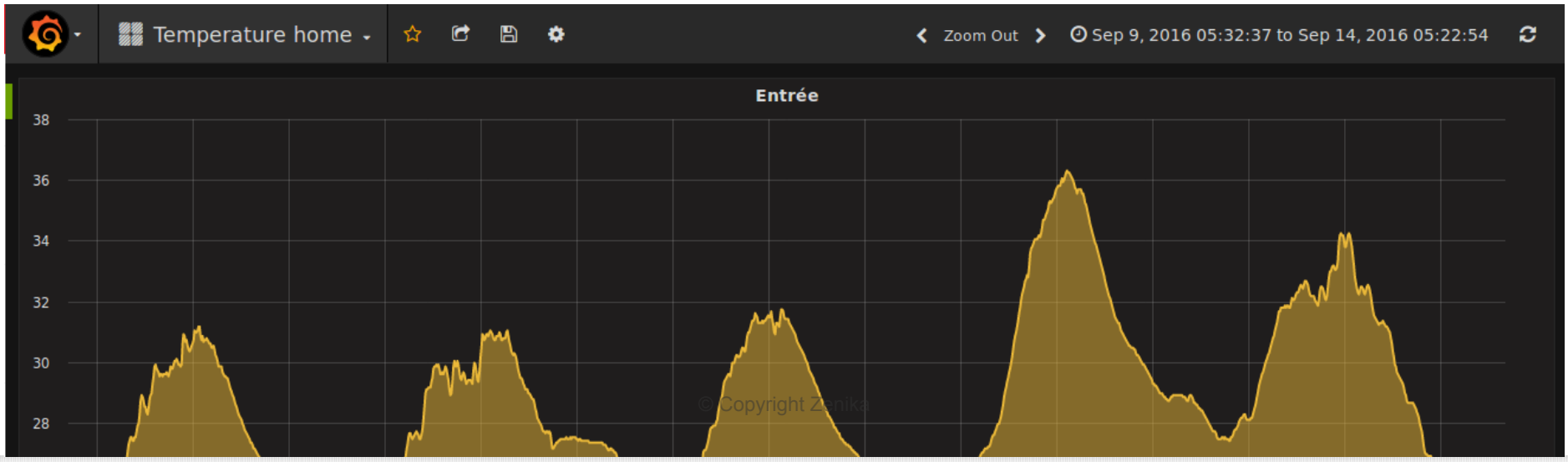
```
board.on("ready", function() {
  var thermometer = new five.Thermometer({
    controller: "DS18B20",
    pin: 10,
    freq: 1000
  });

  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 & 0xFFF0) + 12 - data[6];
    ...
}}
```

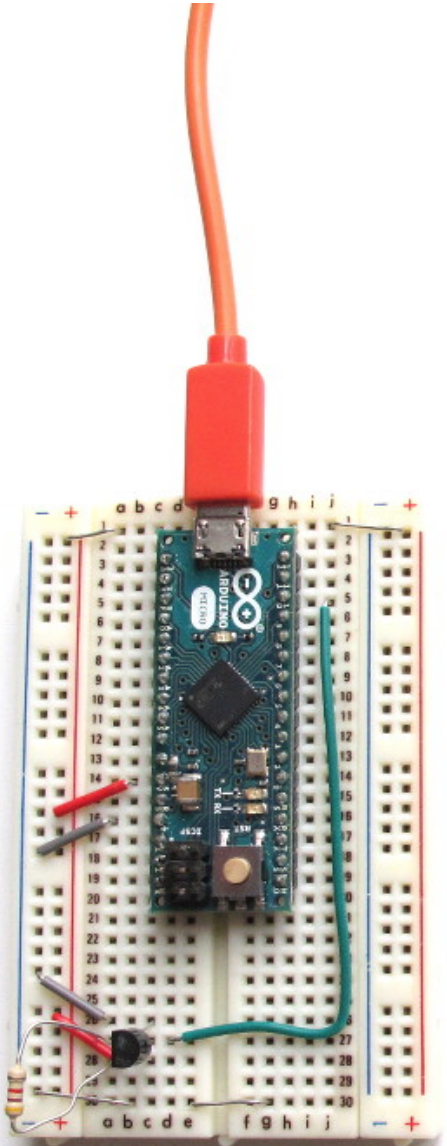
Graphiques

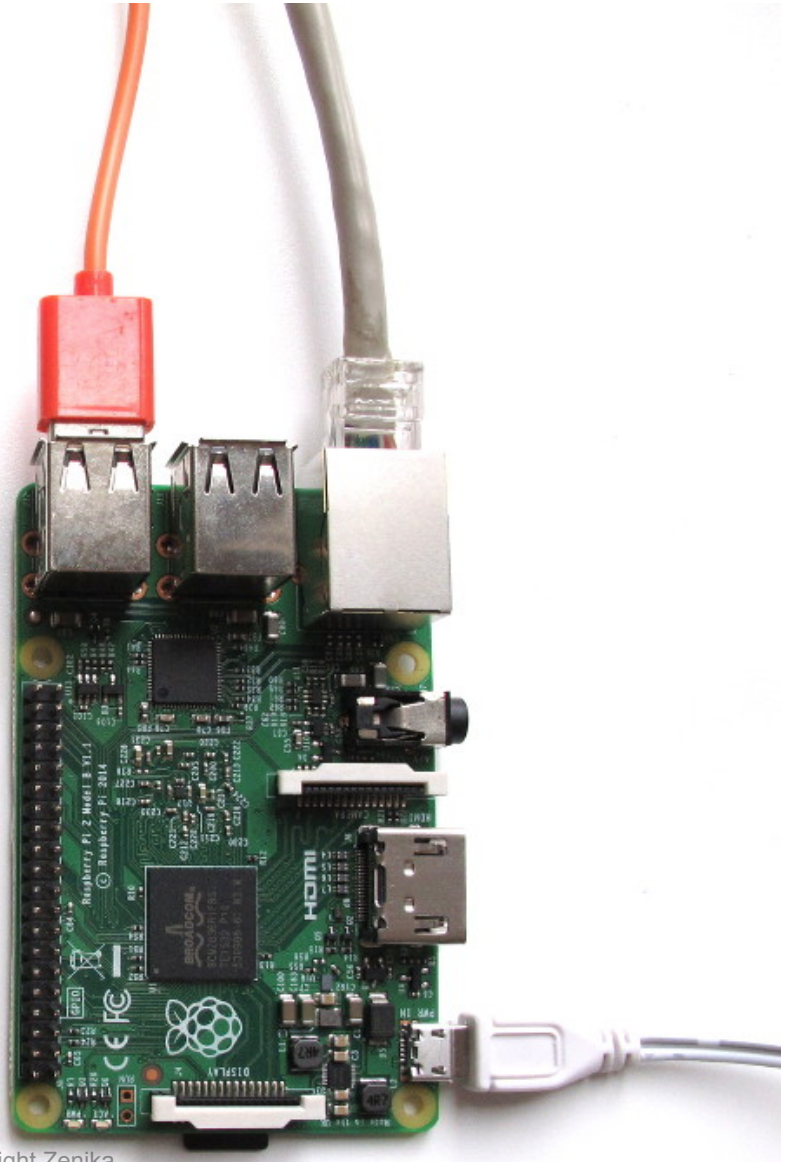
- L'objectif final est de tracer des courbes de température
- **Influxdb** : *https://www.influxdata.com/*
  - base nosql temporelle : les données sont indexées sur un timestamp précis la nanoseconde
- **Grafana** : *http://grafana.org/*
  - Tableau de bord compatible avec de nombreuses sources de métriques
  - Mise en forme des données sous forme de graphique



All together

- Assemblage électronique

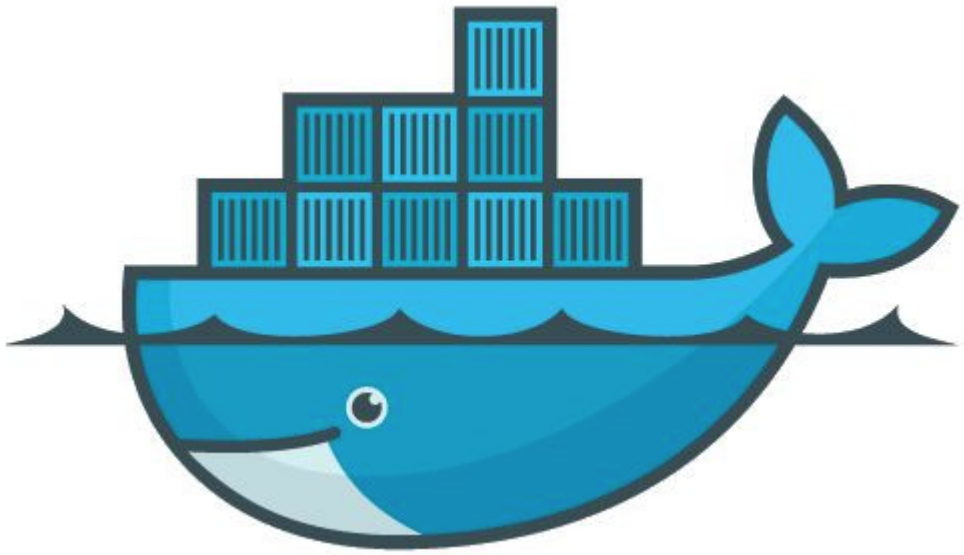




All together

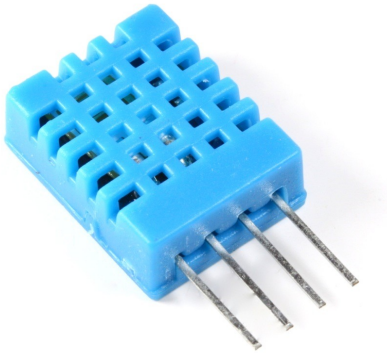
- **Grafana** et **Influxdb** sont déployés sous forme de conteneurs **Docker** sur le **Raspberry Pi**

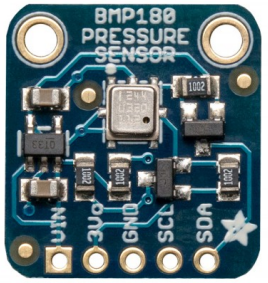
⚠ Obligation de reconstruire les images from scratch car les images disponible sur le Docker Hub sont en très grand majorité pour **x86 amd64** et non **armhf**



Et ensuite ?

- A ce niveau, il va devenir aisé d'ajouter des nouveaux capteurs :
  - **DHT22** : mesure d'humidité
  - **BMP180** : capteur de pression atmosphérique
  - Capteur d'humidité de sol
  - ...





Démo

Appelez-moi Johnny 5 (1988)