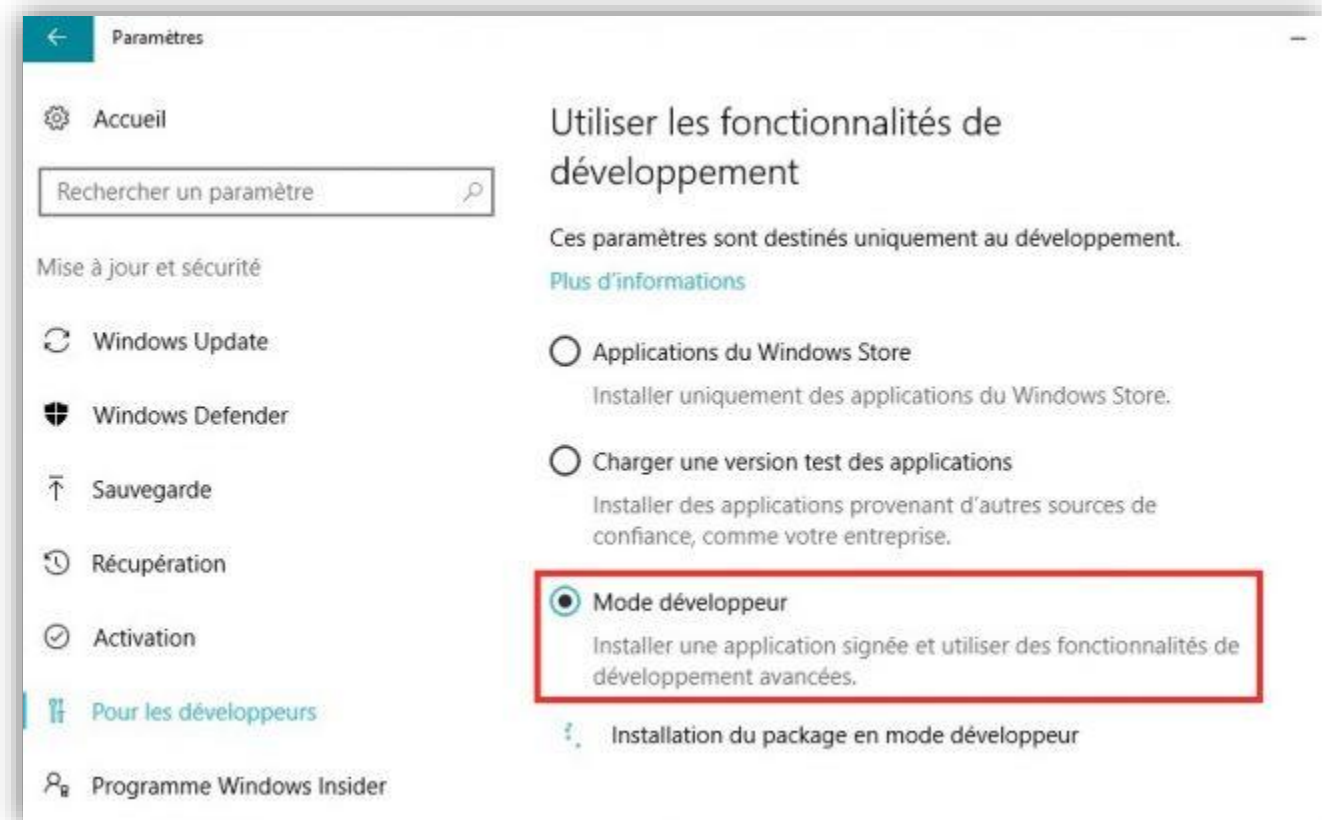


Software technologies

Internet of Things

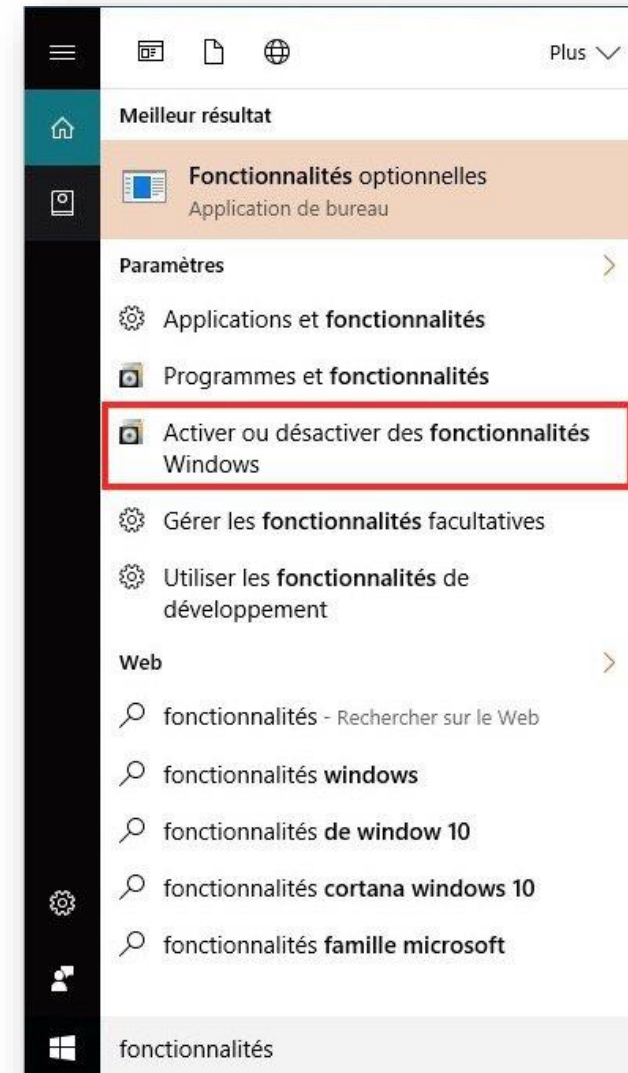
Installer Bash sous Windows 10

- Rendez vous dans les **Paramètres** -> **Mise à jour et sécurité** et dans le menu "**Pour les développeurs**", cochez le bouton "**Mode développeur**".



Installer Bash sous Windows 10

- Activation du sous-système Linux de Windows: Tapez "***fonctionnalités***" dans la barre de recherche et cliquez sur "***Activer ou désactiver des fonctionnalités Windows***".



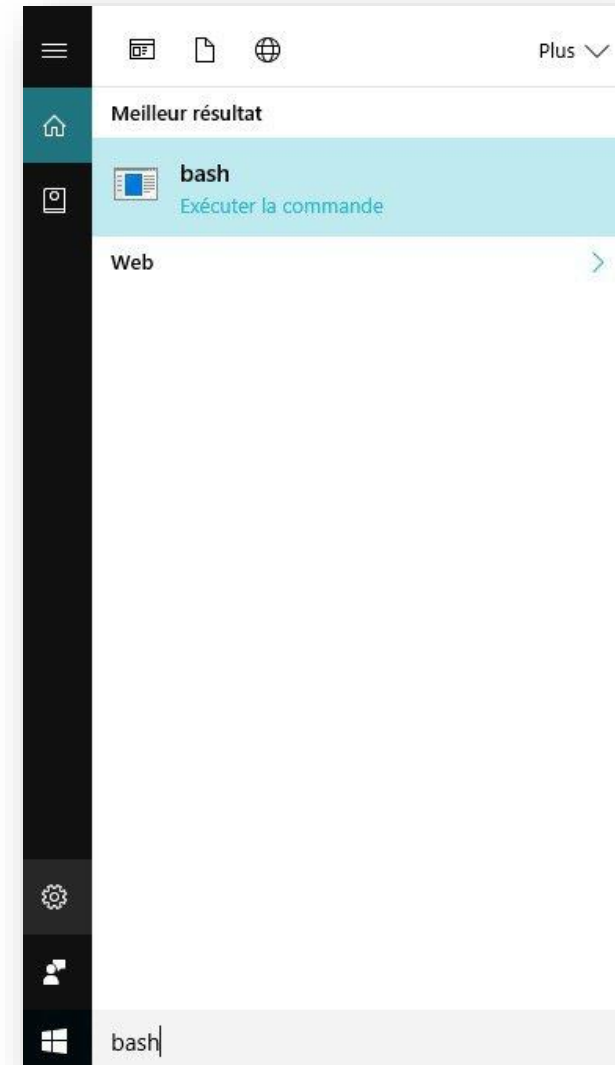
Installer Bash sous Windows 10

- Cochez la case "***Sous-système Windows pour Linux***" et faites OK. Votre ordinateur devra ensuite redémarrer.

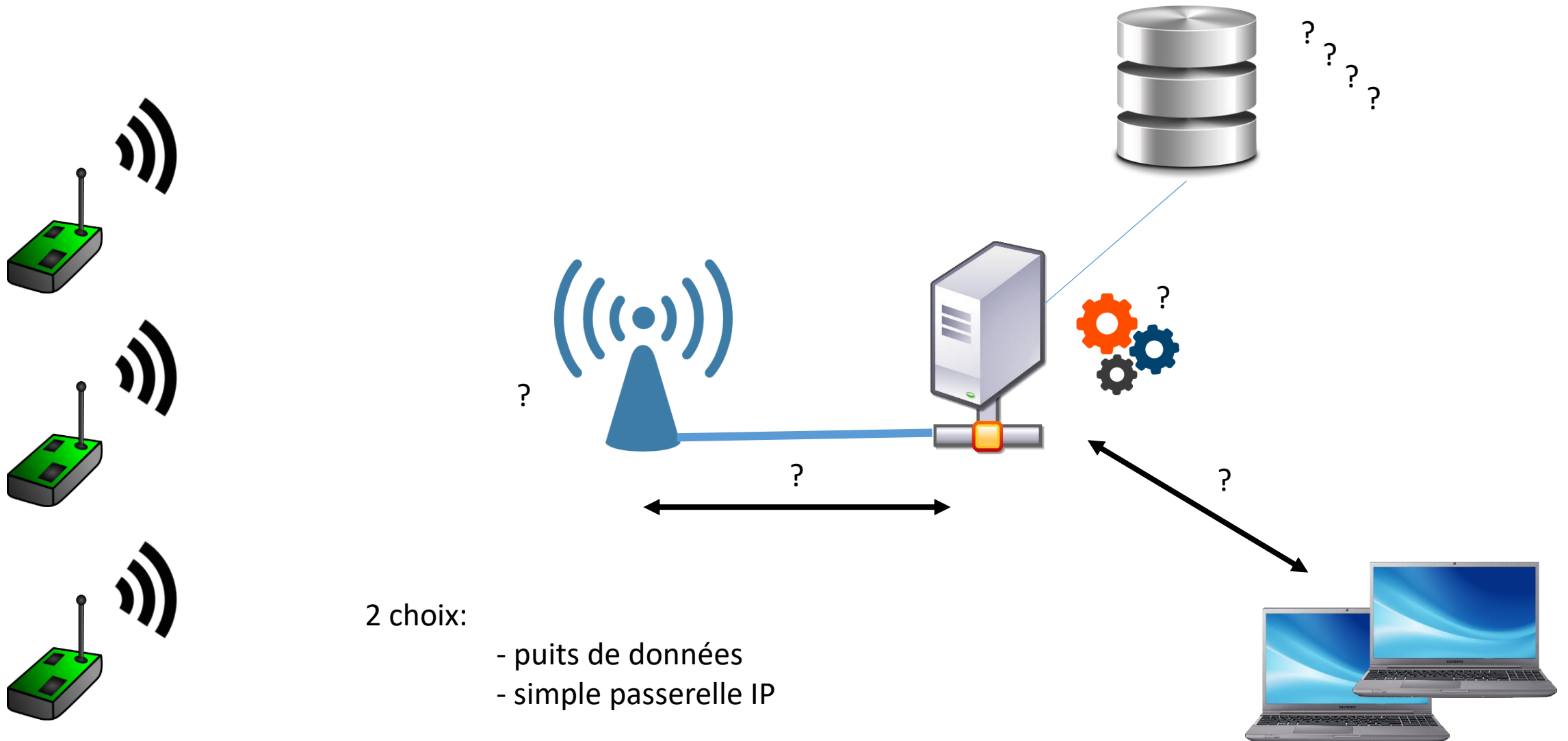


Installer Bash sous Windows 10

- Pour lancer Bash: tapez bash dans la recherche
 - Finir l'installation
 - `sudo apt-get update`
 - `sudo apt-get upgrade`



Software – Cloud Computing



Du capteur à l'IP

- Il faut transmettre la donnée du capteur sur de l'IP quelque soit le protocole utilisé
- 2 choix:
 - Puits de données
 - Capteur n'est pas IP
 - Traitement
 - Simple passerelle IP
 - Capteur est déjà IP (IPv4, IPv6, 6lowPan, ...)
 - Attention car les capteurs sont directement exposé.



Du capteur à l'IP: la base

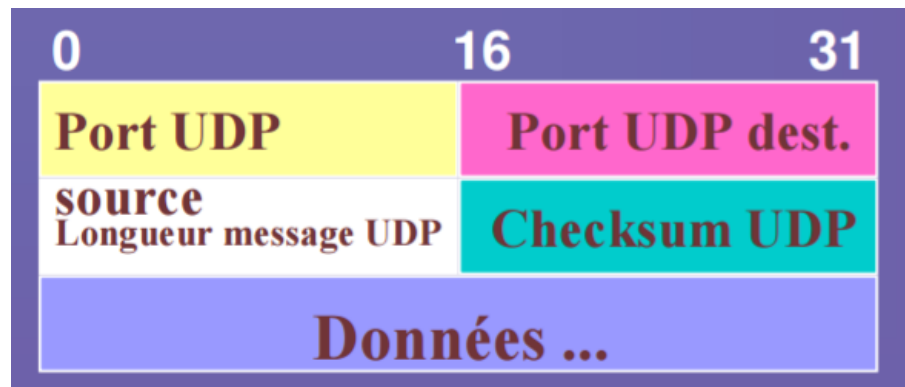
- UDP: User Datagram Protocol
 - Protocole de transport sans connexion
 - Au service des applicatifs
 - Emission de messages applicatifs
 - Sans établissement de connexion au préalable
 - Avec adressage (IP + transport)
 - Arrivée des messages non garantie – information non connue
 - Ordonnancement des messages non garanti

Du capteur à l'IP: la base

- UDP: User Datagram Protocol
 - L'émission d'un message se fait sur la base d'un port source et un port destinataire
 - Les processus disposent d'une interface système leur permettant de spécifier un port ou d'y accéder
 - RESUME : communication UDP = (@IP, Port #) <-> (@IP, Port #)

Du capteur à l'IP: la base

- UDP: User Datagram Protocol
 - Les messages UDP sont également appelés des datagrammes UDP. Ils contiennent deux parties : un en-tête UDP et les données UDP.



Format des messages UDP

Du capteur à l'IP: la base

- TCP: Transmission Control Protocol
 - transport fiable de la technologie TCP/IP.
 - Echange en mode connecté
 - garantie de non perte de messages
 - Garantie de l'ordonnancement
 - Full Duplex

Du capteur à l'IP: la base

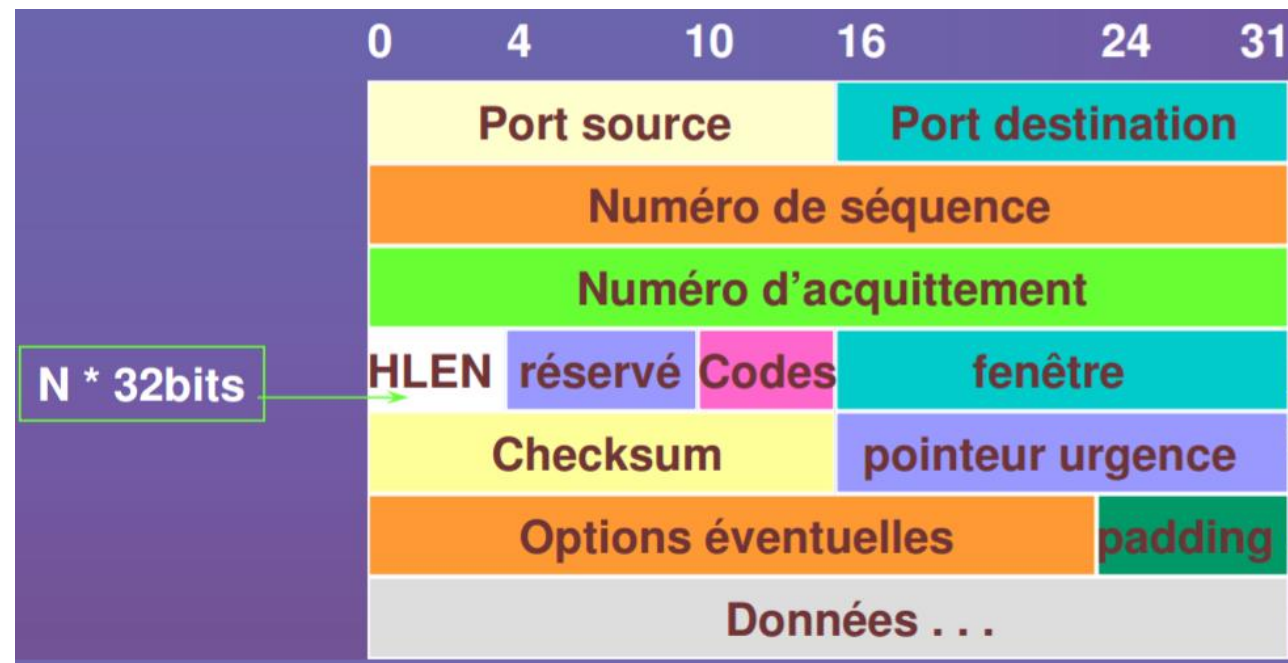
- TCP: Transmission Control Protocol
 - Machine A et B avec un couple (adresse IP, port)
 - Contrairement à UDP, TCP garantit l'arrivée des messages, c'est à dire qu'en cas de perte, les deux extrémités sont prévenues
 - Ce concept repose sur les techniques d'acquittement de message
 - lorsqu'une source A émet un message M1 vers une destination B, A attend un acquittement Ac1 de B avant d'émettre le message suivant M2
 - Si l'acquittement Ac1 ne parvient pas à A, A considère au bout d'un certain temps que le message est perdu et re-émet M1

Du capteur à l'IP: la base

- TCP: Transmission Control Protocol
 - La technique acquittement simple pénalise les performances puisqu'il faut attendre un acquittement avant d'émettre un nouveau message.
 - Le fenêtrage améliore le rendement des réseaux.

Du capteur à l'IP: la base

- TCP: Transmission Control Protocol
 - Messages TCP



Du capteur à l'IP: la base

- TCP: Transmission Control Protocol

Port	Service
21	FTP
22	SSH
25	SMTP
80	HTTP
110	POP3
443	HTTPS

Du capteur à l'IP: la base

- UDP et TCP
 - Utile
 - Efficace
 - Voir le besoin
 - Quantité de donnée
 - Transmission
 - Etc....
 - Interopérabilité nulle
 - Besoin d'un service au dessus si besoin

Réalisation d'un serveur et d'un client UDP en python

Serveur UDP

- Fonctionnement du python:
 - Pas besoin de déclarer le type des variables
 - `a=10` , `a=« toto »`
 - Pour afficher: `print()`
 - Pas de « ; »
 - Jamais d'accolades en python (pour les if, boucle for, fonction, etc...) mais uniquement de l'indentation avec des tabulations ou 4 espaces
- Pour plus d'infos: <https://openclassrooms.com/courses/apprenez-a-programmer-en-python>

Serveur UDP

- Exemple de base de python:
 - nano hello.py (ctrl + x pour quitter)

```
a = 10
print (a)
a = "hello"
print(a)
```
 - python hello.py
- Exemple de fonction: ici 2 fonctions
 - nano hellofct.py (ctrl +x pour quitter)

```
def afficherHello():
    print("hello")

def ajouter(a,b):
    return a+b

afficherHello()
print( ajouter( 1,2 ) )
```
 - python hellofct.py

Serveur UDP

- nano serverUDP.py (CTRL + x pour quitter)

```
import socket
```

```
UDP_IP = "127.0.0.1"
```

```
UDP_PORT = 5005
```

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP  
sock.bind((UDP_IP, UDP_PORT))
```

```
while True:
```

```
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
```

```
    print "received message from " + str(addr) + " : " + str(data)
```

- python serverUDP.py (CTRL + c pour quitter)

Client UDP

- nano clientUDP.py (CTRL + x pour quitter)

```
import socket
```

```
UDP_IP = "127.0.0.1"
```

```
UDP_PORT = 5005
```

```
MESSAGE = 2
```

```
print "UDP target IP:" + UDP_IP
```

```
print "UDP target port:" + str(UDP_PORT)
```

```
print "message:" + str(MESSAGE)
```

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
sock.sendto( str(MESSAGE) , (UDP_IP, UDP_PORT))
```

- python clientUDP.py

Serveur et client UDP

- Essayer de lancer le client avec serveur allumé ou éteint
 - On observe que le client ne peut savoir si le serveur est allumé

Un simulateur de température

- nano simulateur.py (CTRL + X pour quitter)

```
import random
import time
```

```
def getTemp():
    return float( random.randint(1850,2100) )/100
```

```
while True:
    time.sleep(1.5)
    print (getTemp())
```

- python simulateur.py (CTRL + c pour quitter)
- Mixer les deux programmes pour faire un simulateur de température qui transmet ses données en UDP

Réalisation d'un serveur et d'un client TCP en python

Serveur TCP

- nano serverTCP.py

```
import socket
```

```
TCP_IP = '127.0.0.1'
```

```
TCP_PORT = 5005
```

```
BUFFER_SIZE = 20 # Normally 1024, but we want fast response
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #TCP
```

```
s.bind((TCP_IP, TCP_PORT))
```

```
s.listen(1)
```

```
conn, addr = s.accept()
```

```
print 'Connection address:' + str(addr)
```

```
while 1:
```

```
    data = conn.recv(BUFFER_SIZE)
```

```
    if not data:
```

```
        break
```

```
    print "received data:" + str( data )
```

```
    conn.send(data + " world")
```

```
conn.close()
```

- python serverTCP.py

Client TCP

- nano clientTCP.py

```
import socket
```

```
TCP_IP = '127.0.0.1'
```

```
TCP_PORT = 5005
```

```
BUFFER_SIZE = 1024
```

```
MESSAGE = "Hello"
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #TCP
```

```
s.connect((TCP_IP, TCP_PORT))
```

```
s.send(MESSAGE)
```

```
data = s.recv(BUFFER_SIZE)
```

```
s.close()
```

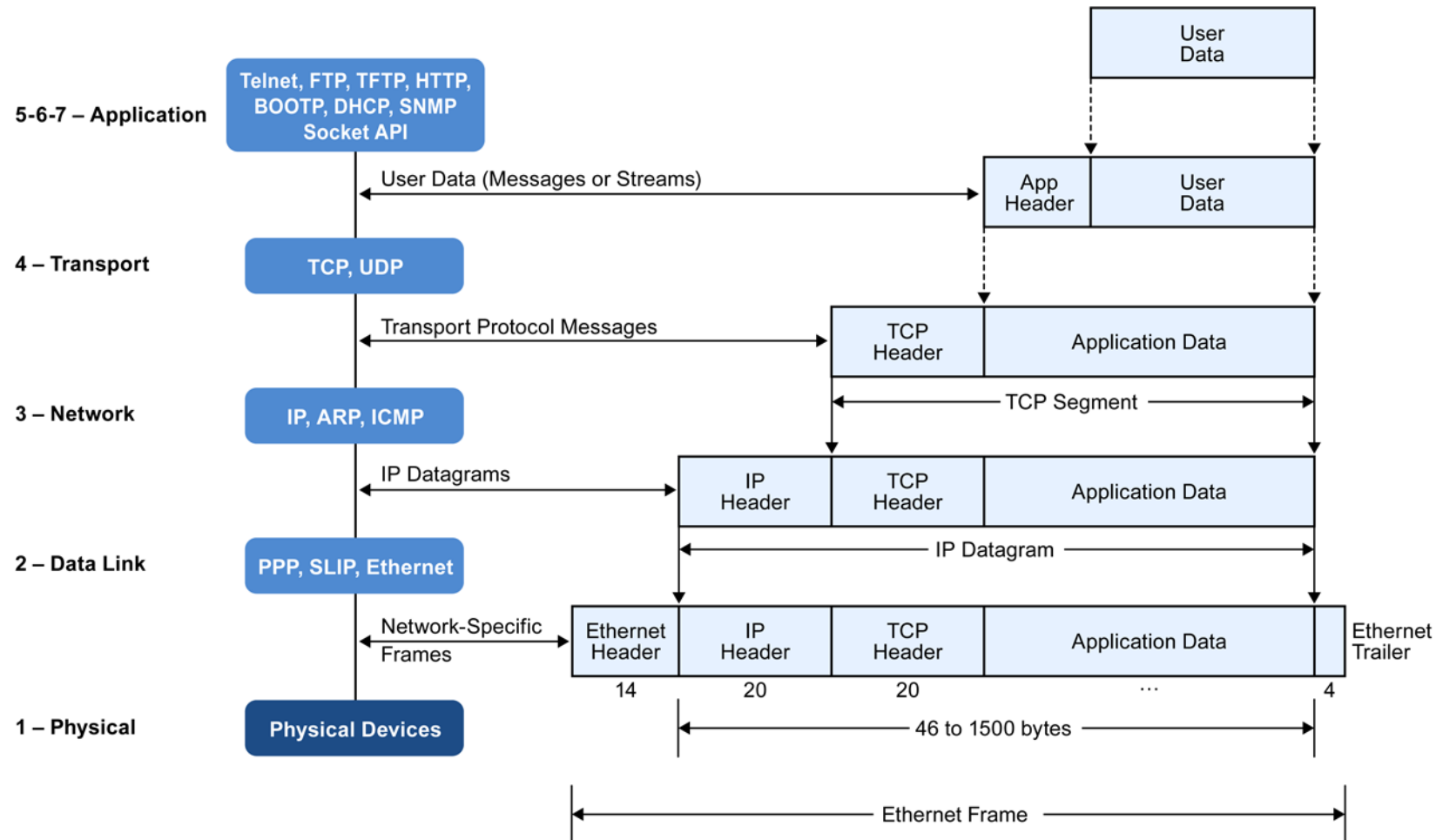
```
print "received data:" + str(data)
```

- python clientTCP.py

Serveur et client TCP

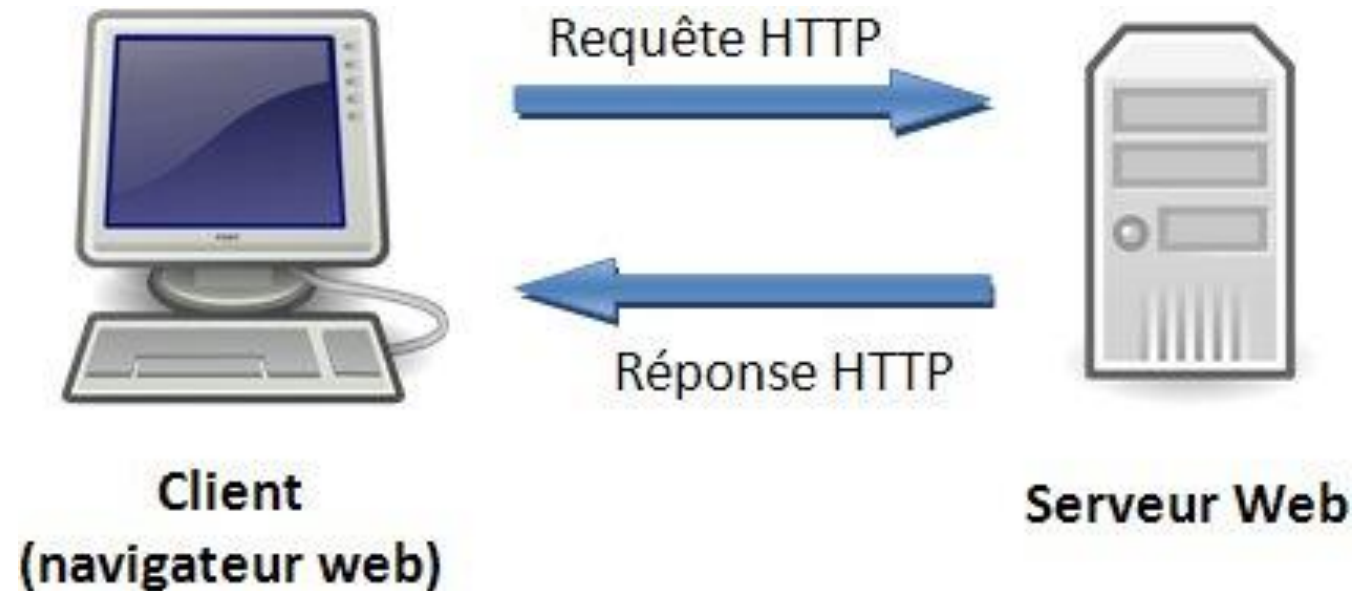
- Essayer de lancer le client avec serveur allumé ou éteint
 - On observe que le client sait si le serveur est allumé
 - Une fois le canal ouvert, nous pouvons envoyer de la donnée bi-directionnelle

Modèle OSI

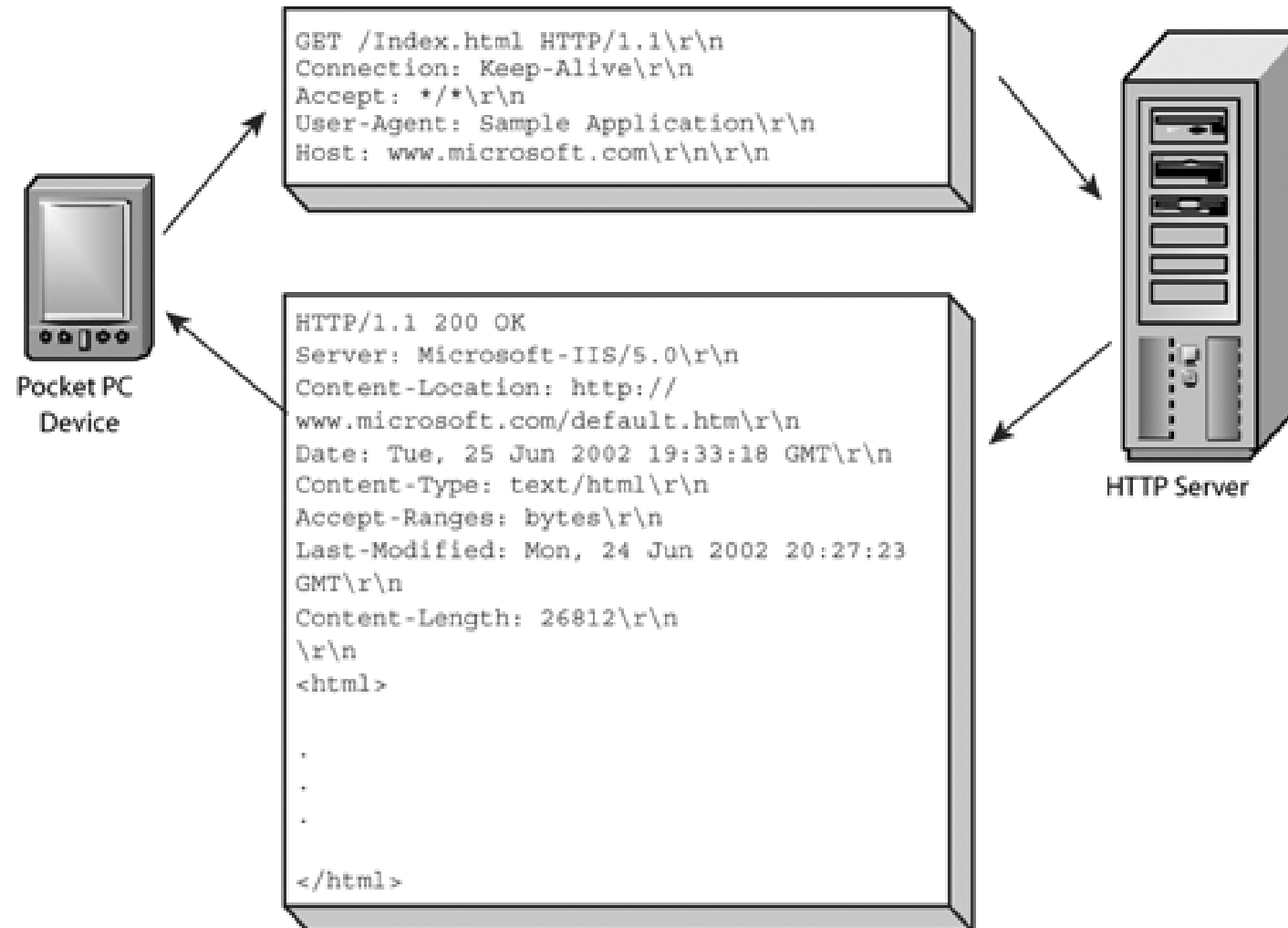


HTTP

- Utilisation de vieilles technologies
 - Repose sur le TCP
- Requêtes HTTP
 - GET
 - POST
 - PUT
 - DELETE
 - ...



HTTP



Requêtes HTTP

- WIKIPEDIA

- **GET:** *C'est la méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource.*
- **POST:** *Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML)*
- **PUT:** *Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur.*
- **DELETE:** *Cette méthode permet de supprimer une ressource du serveur.*

HTTP

- REST ou RESTful

Verb	Path	action	used for
GET	/photos	index	display a list of all photos
GET	/photos/new	new	return an HTML form for creating a new photo
POST	/photos	create	create a new photo
GET	/photos/:id	show	display a specific photo
GET	/photos/:id/edit	edit	return an HTML form for editing a photo
PUT	/photos/:id	update	update a specific photo
DELETE	/photos/:id	destroy	delete a specific photo

Exemple: développer des API

- Installation de python-pip – outils d'installation de bibliothèque
 - `sudo apt install python-pip`
- Installation de la bibliothèque
 - `sudo pip install Flask`
- Installation de POSTMAN – tester nos APIs
- Télécharger NGROK – créer un tunnel entre internet et localhost

Exemple: développer des API

- Installation de python-pip – outils d'installation de bibliothèque
 - `sudo apt install python-pip`
- Installation de la bibliothèque
 - `sudo pip install Flask`
- Installation de POSTMAN – tester nos APIs
- Télécharger NGROK – créer un tunnel entre internet et localhost

Exemple: développer des API

- nano serverFlask.py

```
from flask import Flask
app = Flask(__name__)

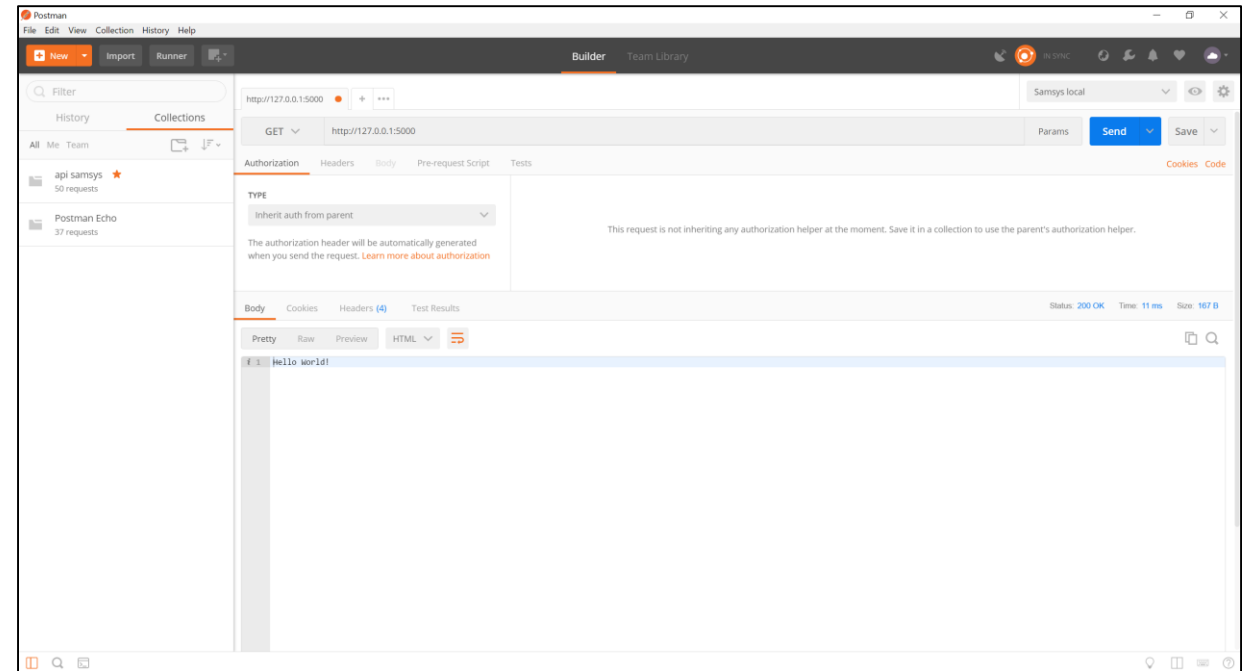
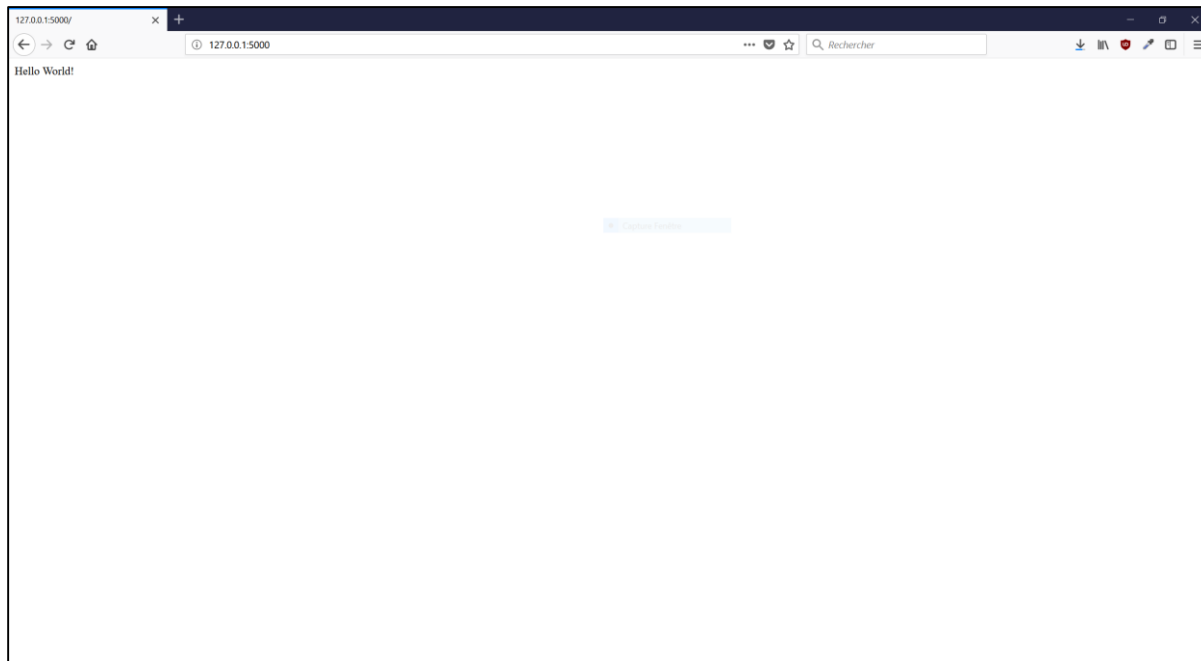
@app.route("/")
def hello():
    return "Hello World!"

app.run(port=5000, debug=True)
```

- Python serverFlask.py

Exemple: développer des API

- Test de notre API: `http://127.0.0.1:5000/`



- Changer le `@app.route` et tester

Exemple: développer des API

API: GET / POST

```
from flask import request

app = Flask(__name__)

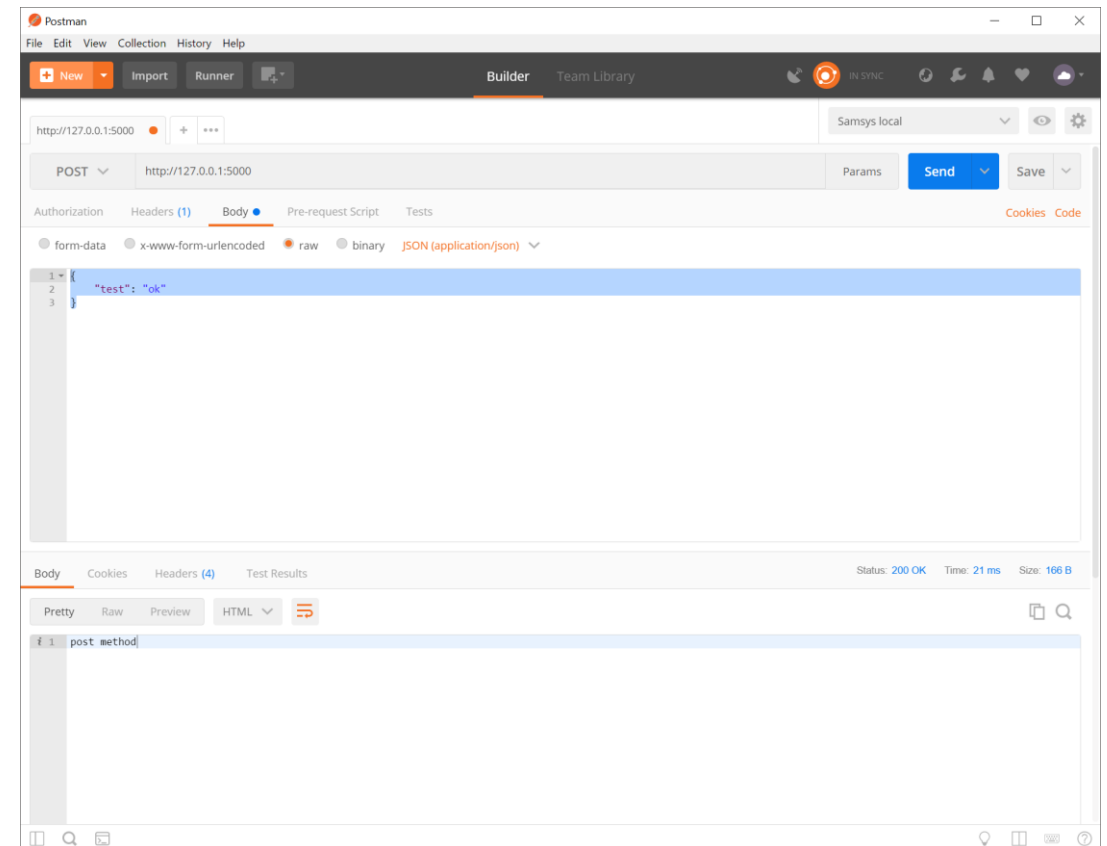
@app.route("/", methods=['GET', 'POST'])
def hello():
    if request.method == 'POST':
        if request.json:
            print(request.json)
            return 'post method'
        else:
            return 'get method'

app.run(port=5000, debug=True)
```

- python serverFlask.py

Exemple: développer des API

- POSTMAN - Test de notre API: <http://127.0.0.1:5000/>
 - Changer le GET en POST
 - Ajouter un body
 - Choisir 'raw' – JSON (application/json)
 - { "test" : "ok" }
 - Regarder le résultat en console



HTTP

- REST ou RESTful

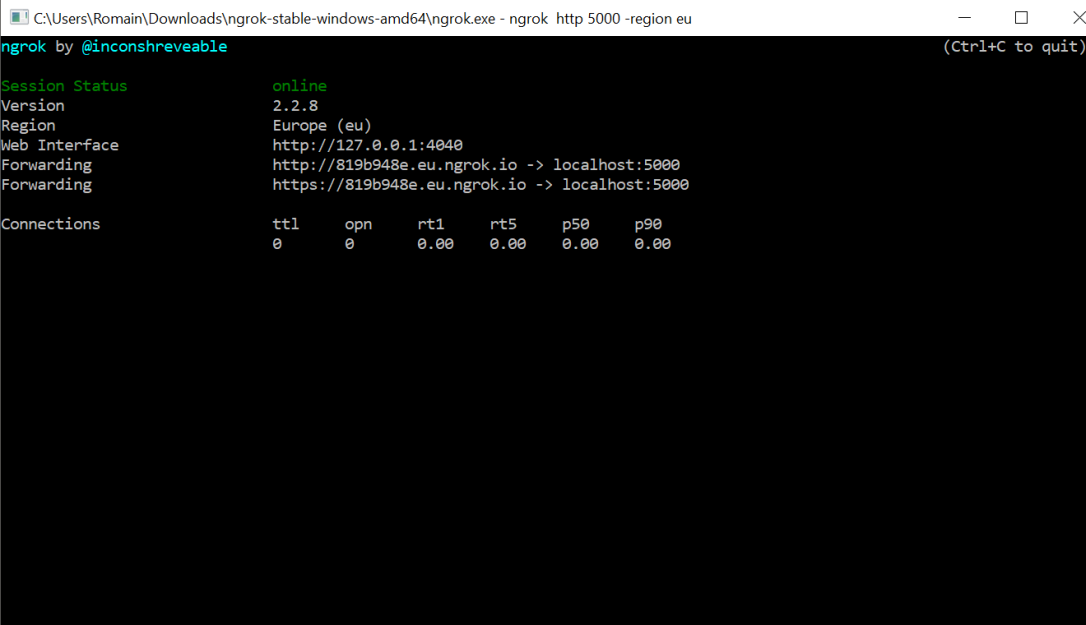
Verb	Path	action	used for
GET	/photos	index	display a list of all photos
GET	/photos/new	new	return an HTML form for creating a new photo
POST	/photos	create	create a new photo
GET	/photos/:id	show	display a specific photo
GET	/photos/:id/edit	edit	return an HTML form for editing a photo
PUT	/photos/:id	update	update a specific photo
DELETE	/photos/:id	destroy	delete a specific photo

Exemple: développer des API

- Utilisation de NGROK
 - Utiliser le CMD ou terminal
 - Aller dans le dossier de ngrok
 - Lancer ngrok
 - Linux `./ngrok http 5000 --region eu`
 - Windows `ngrok.exe http 5000 --region eu`
 - ngrok crée un tunnel de l'extérieur (internet) vers votre machine

Exemple: développer des API

- Utilisation de NGROK



```
C:\Users\Romain\Downloads\ngrok-stable-windows-amd64\ngrok.exe - ngrok http 5000 -region eu
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Version             2.2.8
Region              Europe (eu)
Web Interface       http://127.0.0.1:4040
Forwarding           http://819b948e.eu.ngrok.io -> localhost:5000
Forwarding           https://819b948e.eu.ngrok.io -> localhost:5000

Connections         ttl    opn    rt1    rt5    p50    p90
                   0      0      0.00   0.00   0.00   0.00
```

- Test avec postman
 - Exemple: <https://819b948e.eu.ngrok.io>

Protocoles applicatifs pour l'IoT

- Utilisation de vieilles technologies
 - Requêtes HTTP
 - 10 fois plus consommateur de data
 - Lourd
 - Que dans un sens, c'est le client qui initie la conversation
 - Repose sur le TCP
 - A fait ses preuves

Protocoles plus léger et plus
adapté pour l'lot

CoAP

- CoAP: Constrained Application Protocol
 - **CoAP** est un protocole conçu par l'IETF (Internet Engineering Task Force)

L'Internet Engineering Task Force, abrégée IETF, littéralement traduit de l'anglais en « Détachement d'ingénierie d'Internet » est un groupe informel, international, ouvert à tout individu, qui participe à l'élaboration des standards Internet. L'IETF produit la plupart des nouveaux standards d'Internet

Wikipedia

CoAP

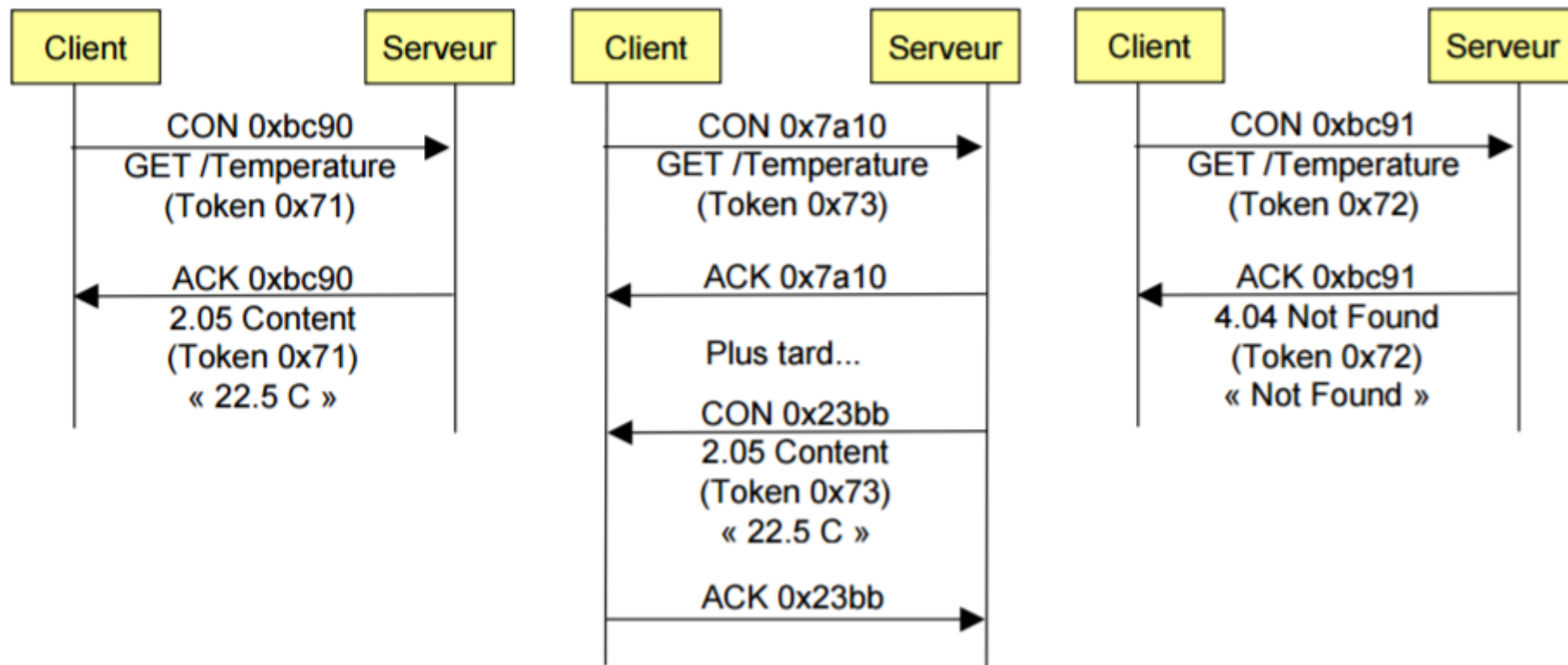
- CoAP: Constrained Application Protocol
 - Utilise l'UDP
 - Mais peut être aussi utilisé sur TCP
 - Sécurité: DTLS
 - Possibilité de faire des passerelle CoAP <-> HTTP facilement
 - Protocole RESTfull
 - Adapté aux faibles ressources des équipements de l'IoT
 - Un token par requête car asynchrone
 - Service Discovery

CoAP

- CoAP: Constrained Application Protocol
- Dans l'en-tête de chaque paquet:
 - 2 bits pour:
 - GET
 - POST
 - DELETE
 - PUT
 - 2 bits pour le type de message
 - Confirmable: Message envoyé avec une demande d'acquittement.
 - Non-Confirmable: Message envoyé sans demande d'acquittement.
 - Acknowledgment: Confirmation de la réception d'un message de type « confirmable ».
 - Reset: Confirmation de la réception d'un message qui n'est pas exploitable.

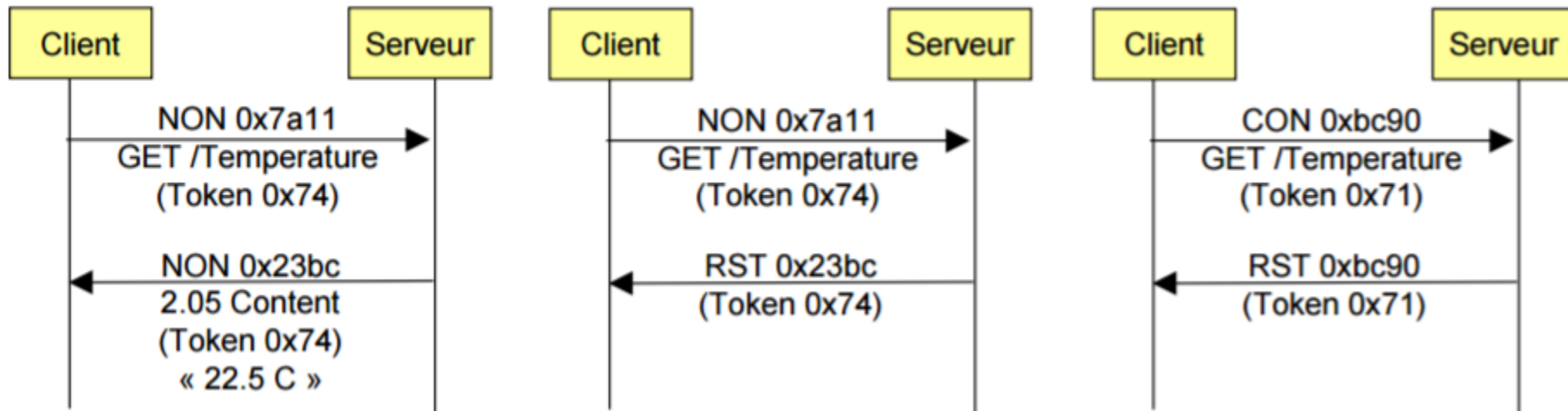
CoAP

- CoAP: Constrained Application Protocol



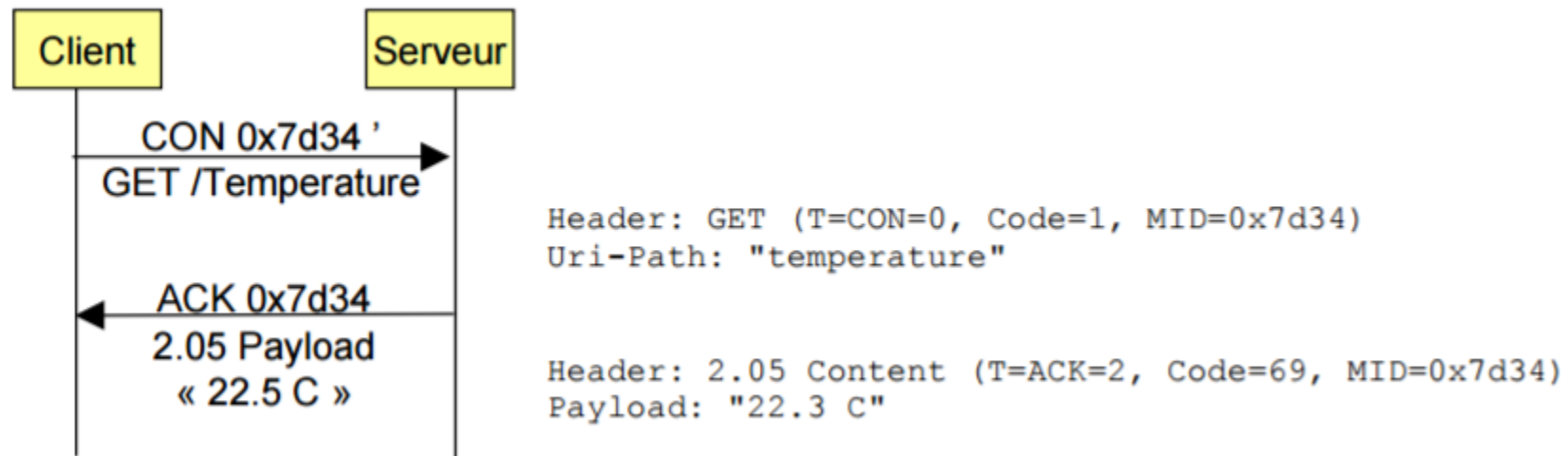
CoAP

- CoAP: Constrained Application Protocol



CoAP

- Message CoAP



HTTP / CoAP

- HTTP et CoAP
 - Architecture Request/Response
 - le client (votre navigateur) émet des requêtes à destination du serveur, lequel lui répond le contenu demandé. Le protocole HTTP, largement utilisé pour l'Internet, repose sur cette architecture
- HTTP vs CoAP

	Bytes per transaction	Power	Lifetime
CoAP	154	0.744 mW	151 jours
HTTP	1451	1,333 mW	84 jours

Publish/Subscribe

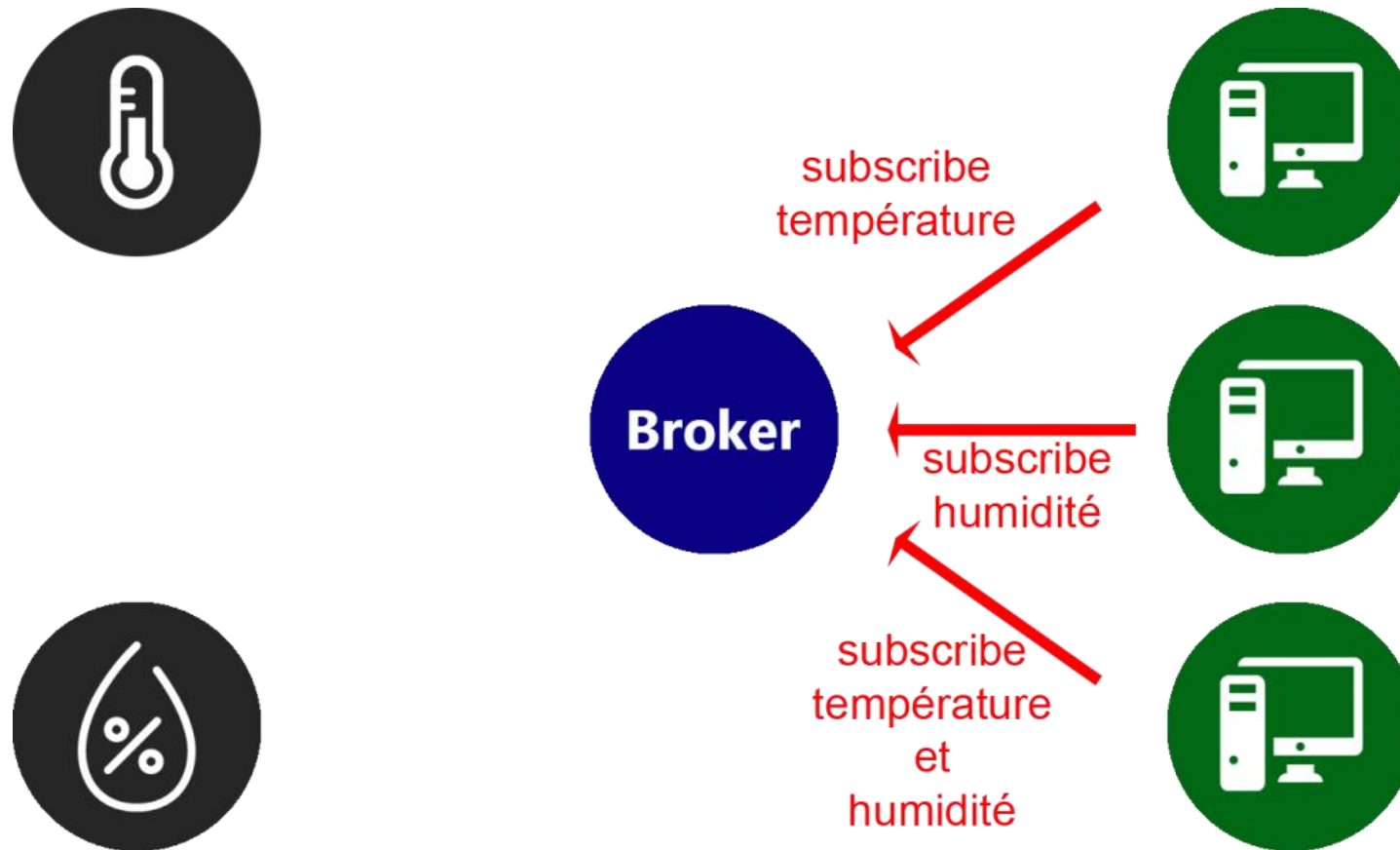
Publish/Subscribe

- Les protocoles Pub/Sub
 - Différent du classique client-serveur (comme en HTTP)
 - le modèle « pub/sub » différencie
 - le client qui envoie de la donnée (publisher)
 - d'un client qui reçoit de la donnée (subscriber)
- Les clients ne s'échangent pas directement la donnée mais passent par un serveur (nommé broker) qui reçoit, filtre et distribue les messages entre les clients.

Pub/Sub

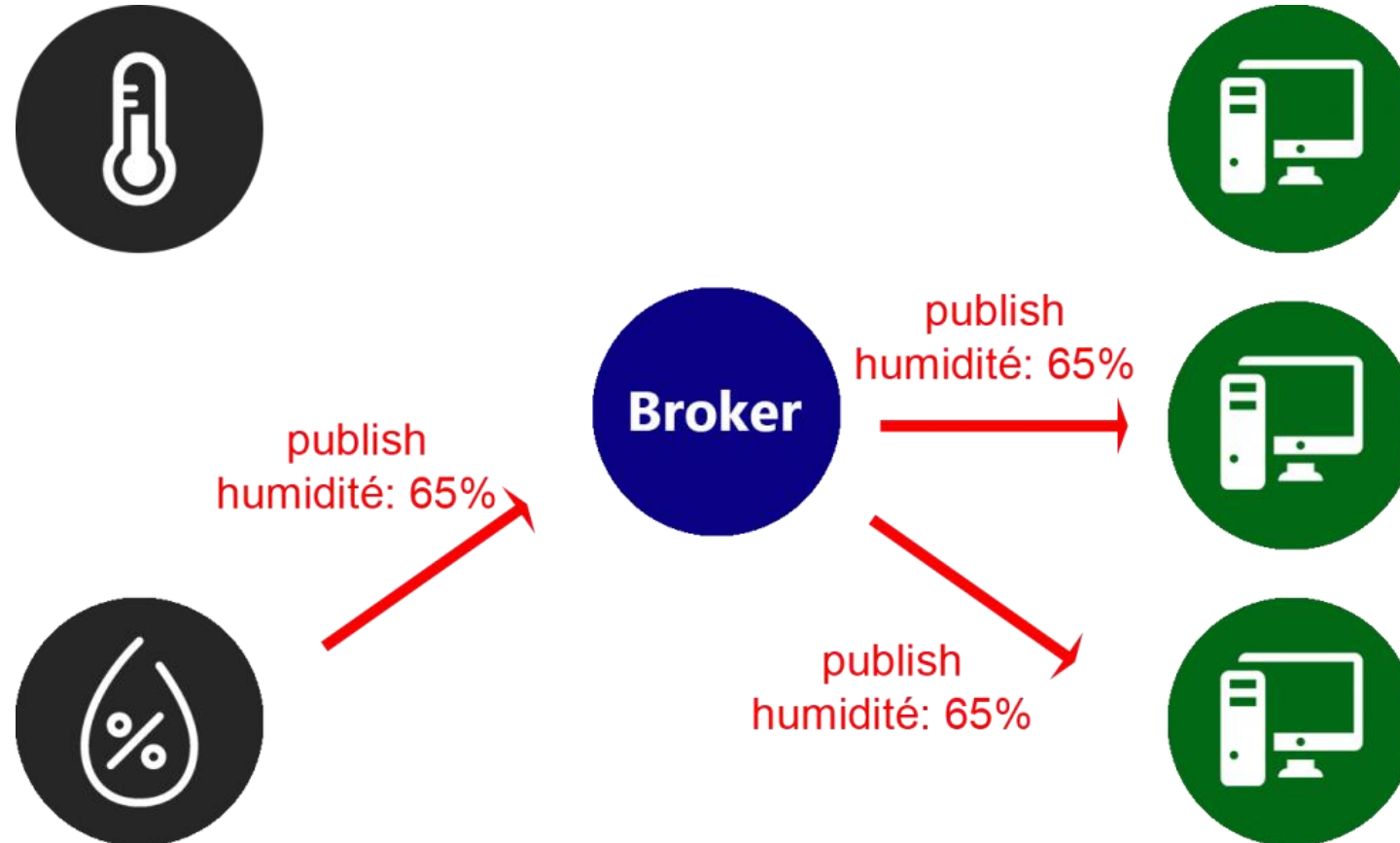
- Broker (Serveur)
 - les clients n'ont pas besoin:
 - de se connecter entre eux
 - De se connaître (par leur adresses IP par exemple)
 - De fonctionner et d'être actifs en même temps
 - d'être synchronisés
- Pour résumer, les publisher et subscriber ne sont pas directement liés entre eux, ils passent par un broker qui reçoit, filtre et transmet les messages entre les clients

Pub/Sub



Des clients se connectent au broker (serveur) et s'abonnent aux sujets qu'ils souhaitent recevoir

Pub/Sub



D'autres clients se connectent et publient de la donnée au broker, qui la distribue ensuite aux clients concernés sans que ces clients ne se connaissent forcément. Le broker fait le lien entre l'offre et la demande.

MQTT

- Repose sur le concept de "pub/sub" était un outil interne chez IBM dès 1999
- Il s'adresse en premier lieu aux objets avec peu de puissance de calcul n'étant pas actif 100 % du temps et ayant une activité occasionnelle sur le réseau
- Fonctionne sur la couche TCP
- Repose sur 3 spécifications:
 - Topics
 - Message
 - Client/Broker

MQTT

- Topics
 - chaînes de caractères (strings)
 - but de filtrer, trier et répartir les messages (les données) entre les clients
- créés à la volée par les clients auprès du broker
- représentation en texte d'un arbre hiérarchique
 - chacun des étages de cet arbre est séparé par le caractère slash ("/")

MQTT

- Topics
 - Exemple
 - un capteur mesure l'humidité et la température de la chambre_1
 - un autre capteur mesure l'humidité et la température de la chambre_2
 - Les topics peuvent être
 - **maison/chambre_1/temperature**
 - **maison/chambre_2/temperature**
 - **maison/chambre_1/humidite**
 - **maison/chambre_2/humidite**

MQTT

- Topics
 - Rappel
 - Avantage, les topics sont directement créés à la volée par les clients (que ce soit publisher ou subscriber).
 - Attention:
 - Ces topics sont sensibles aux majuscules/minuscules,
 - Il faut éviter au maximum l'utilisation les caractères spéciaux ("é", "@", "ç", etc.) et des espaces.

MQTT

- Topics
 - Souscrire à un topic
 - Un client peut recevoir des messages en souscrivant (subscriber) à un topic
 - 2 possibilités
 - Il souscrit explicitement à un/des topic(s)
 - exemple, mon client souscrit à "**maison/chambre_1/temperature** «
 - Il souscrit à plusieurs topics du même type,
 - il peut utiliser ce que l'on appelle en MQTT les wildcards, il y en existe 2 ("+" ou "#")

MQTT

- Topics - wildcards
 - "+" (Single Level) permet de se substituer à un niveau de hiérarchie dans les topics
 - Exemple 1, mon client souscrit à **"maison/+/temperature"**, il va recevoir les températures des deux chambres: **"maison/chambre1/temperature"** et **"maison/chambre2/temperature"**
 - Exemple 2, mon client souscrit à **"maison/chambre1/+"**, il va recevoir toutes les données relatives à la chambre1: **"maison/chambre1/temperature"** et **"maison/chambre1/humidite"**.
 - "#" (Multi Level) permet de se substituer à plusieurs niveaux de hiérarchie dans les topics. Il se met forcément à la fin.
 - Exemple, mon client souscrit à **"maison/#"**, il va recevoir les températures et l'humidité des chambre1 et chambre2: **"maison/chambre1/temperature"**, **"maison/chambre2/temperature"**, **"maison/chambre1/humidite"** et **"maison/chambre2/humidite"**

MQTT

- Messages
 - Chaque message est publié dans un topic avec une qualité de service (QoS).
 - Il existe trois niveaux de services:
 - **QoS = 0**, le message est envoyé une fois au broker et on ne vérifie pas s'il a bien été reçu. Il n'y a pas d'accusés de réception. Ce mode est utilisé en cas d'excellente connexion (câblée ou en local), et permet les meilleures performances.
 - **QoS = 1**, le message a été reçu au moins une fois par le broker par un système d'accusé de réception. Il est possible que plusieurs messages soient transmis. Les performances sont moins bonnes que le QoS 0.
 - **QoS = 2**, le message a été reçu une et une seule fois par le broker par un système de double accusés de réception. Cette double transaction assure une bonne transmission mais ralentit la transmission par rapport à QoS 1 et QoS 0.

MQTT

- Messages
 - Message retenu
 - Il est possible de demander au serveur de garder en mémoire le dernier message publié dans chaque topic
- Clients/Brokers
 - Keep Alive
 - Régulièrement le client doit dire au serveur qu'il est en vie
 - Testament
 - Au moment de la connexion au serveur, mon client peut envoyer un testament au broker. Le Broker enverra ce testament si et seulement si mon client se déconnecte mal (s'il n'a pas fait l'action de se déconnecter)

Exemple MQTT

- Exemple MQTT
 - Serveur
 - Client
 - python
 - javascript

Installer un serveur MQTT: mosquitto

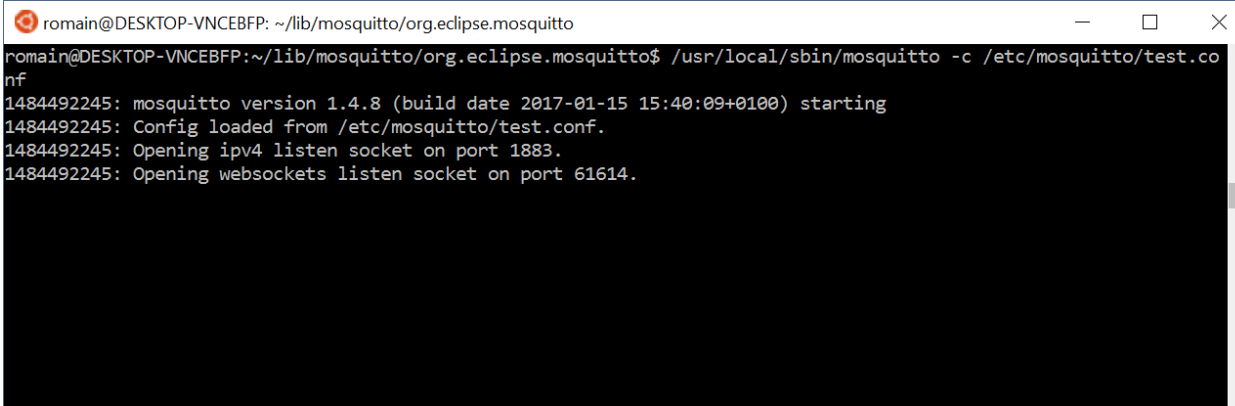
- `sudo apt-get install cmake libssl-dev uuid-dev xsltproc docbook-xsl libwebsockets-dev git build-essential`
- `mkdir ~/lib/mosquitto/`
- `cd ~/lib/mosquitto/`
- `git clone https://git.eclipse.org/r/mosquitto/org.eclipse.mosquitto`
- `cd org.eclipse.mosquitto/`
- `nano config.mk`
 - `WITH_SRV:=no`
 - `WITH_WEBSOCKETS:=yes`

Installer un serveur MQTT: mosquitto

- Compilation et installation du serveur
 - make
 - sudo make install
 - sudo ldconfig
- Configuration du serveur:
 - *sudo nano /etc/mosquitto/test.conf*
 - listener 1883 0.0.0.0
- Lancement du serveur:
 - *sudo useradd -r -m -d /var/lib/mosquitto -s /usr/sbin/nologin -g nogroup mosquitto*
 - *sudo /usr/local/sbin/mosquitto -c /etc/mosquitto/test.conf*

Installer un serveur MQTT: mosquitto

- Si tout est ok, vous devriez avoir un équivalent à ça, laisser la fenêtre ouverte:



```
romain@DESKTOP-VNCEBFP: ~/lib/mosquitto/org.eclipse.mosquitto
romain@DESKTOP-VNCEBFP:~/lib/mosquitto/org.eclipse.mosquitto$ /usr/local/sbin/mosquitto -c /etc/mosquitto/test.conf
1484492245: mosquitto version 1.4.8 (build date 2017-01-15 15:40:09+0100) starting
1484492245: Config loaded from /etc/mosquitto/test.conf.
1484492245: Opening ipv4 listen socket on port 1883.
1484492245: Opening websockets listen socket on port 61614.
```

- Pour info, sous ubuntu, c'est un peu plus simple:
 - *sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa*
 - *sudo apt-get update*
 - *sudo apt-get install mosquitto mosquitto-clients*

Nos premiers clients MQTT

- Utilisation du client mqtt en consommateur:
 - Ouvrez un nouveau bash:
 - `mosquitto_sub -h "localhost" -t "nom_du_topic" -v`
 - Laissez le processus tourner
- Utilisation du client mqtt en producteur:
 - Ouvrez un nouveau bash:
 - `mosquitto_pub -h "localhost" -t "nom_du_topic" -m "message"`
- A ce stade, vous devriez recevoir les messages publiés sur le topic dans le premier bash

Nos premiers clients MQTT

- Essayer les différents concepts de topics:
 - Ecoute sur: (un bash par topic)
 - *maison/+/temperature*
 - *maison/#*
 - *+/+/temperature*
 - Publier sur:
 - *maison/chambre/temperature*
 - *maison/cuisine/temperature*
 - *maison/garage/piece1/temperature*
 - *maison/chambre/humidite*
 - *maison2/chambre/temperature*

Simulateur de capteur de température en python et MQTT

Notre premier client MQTT en python

- Notre simulateur de température:
 - Nous avons besoins d'une bibliothèque python pour faire du MQTT, nous allons utiliser un gestionnaire de bibliothèque python pour l'installer: pip
 - `sudo apt-get install python-pip python-dev`
 - Une fois le gestionnaire installé, installons la bibliothèque
 - `sudo pip install paho-mqtt`

Notre premier client MQTT en python

- Publiions maintenant en MQTT les valeurs plutôt qu'en UDP ou TCP
 - Il faut importer une partie de la bibliothèque mqtt dans le projet:
`import paho.mqtt.publish as mqtt`
 - Et pour publier:
`mqtt.single("topic", "message")`
 - Vous pouvez ajouter les paramètres suivant dans `mqtt.single`:
 - hostname: l'ip du broker
 - qos: qualité de service du message vers le broker (0, 1, 2)
 - retain : garde en mémoire le dernier message (True, ou False)
- A tester en écoutant avec *mosquitto_sub*
- Simuler ensuite l'envoi de donnée de plusieurs capteurs de température sur des topics différents

Récepteur MQTT en python

- nano receiverMQTT.py

```
import paho.mqtt.client as mqtt
```

```
def on_connect(client, userdata, flags, rc):  
    print("Connected with result code "+str(rc))  
    client.subscribe("#")
```

```
def on_message(client, userdata, msg):  
    print(msg.topic+" "+str(msg.payload))
```

```
client = mqtt.Client()  
client.on_connect = on_connect  
client.on_message = on_message  
client.connect("127.0.0.1", 1883, 60)  
client.loop_forever()
```

- python receiver.py

Javascript MQTT

- Modification du serveur MQTT

- Couper le serveur (CTRL + c)

- Configuration du serveur:

- *nano /etc/mosquitto/test.conf*

- listener 1883 0.0.0.0

- listener 61614

- protocol websockets

- Relancer le serveur:

- *sudo /usr/local/sbin/mosquitto -c /etc/mosquitto/test.conf*

Javascript MQTT

- Utilisation des websockets

Emetteur/récepteur MQTT en python

- Lancer plusieurs récepteurs MQTT, ils reçoivent tous la même donnée au même moment
- Test entre vos machines

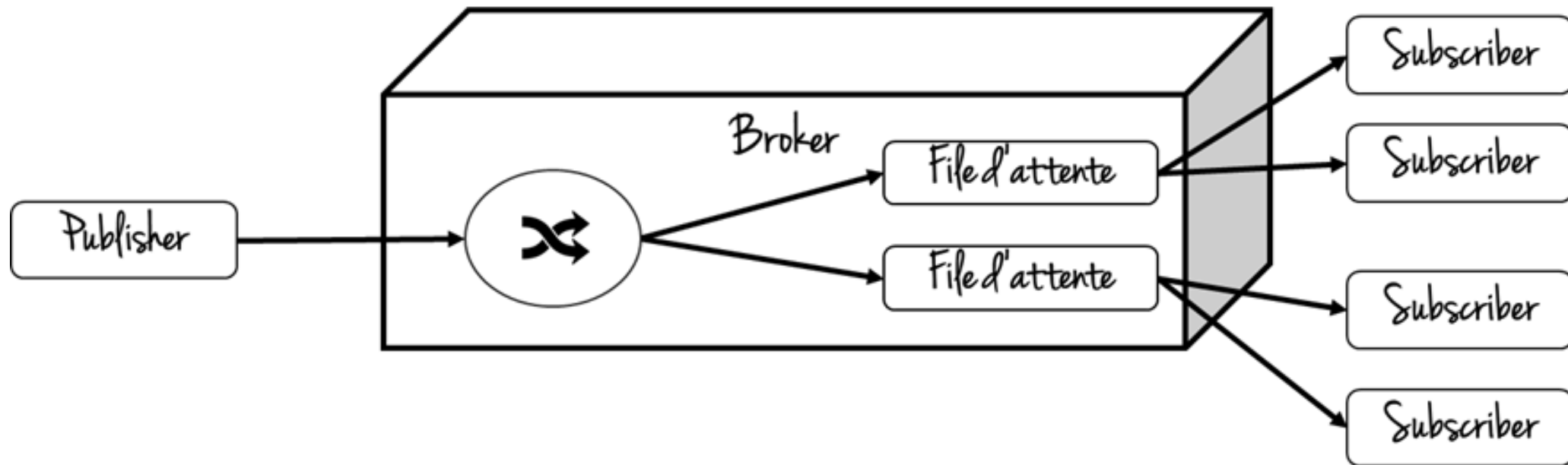
Conclusion MQTT

- MQTT est très léger
- Il permet aussi de faciliter les échanges de données
- Il existe un MQTT encore plus économe en ressource
 - MQTT-SN

AMQP

- Fonctionnement proche du MQTT
- Utilisé dans le monde de la finance
- AMQP apporte en plus un mécanisme de « files d'attente »
- Soit le message est reçu en live par le consommateur du message, soit il est placé dans une file d'attente. C'est lorsqu'un consommateur se connecte qu'il reçoit la file d'attente

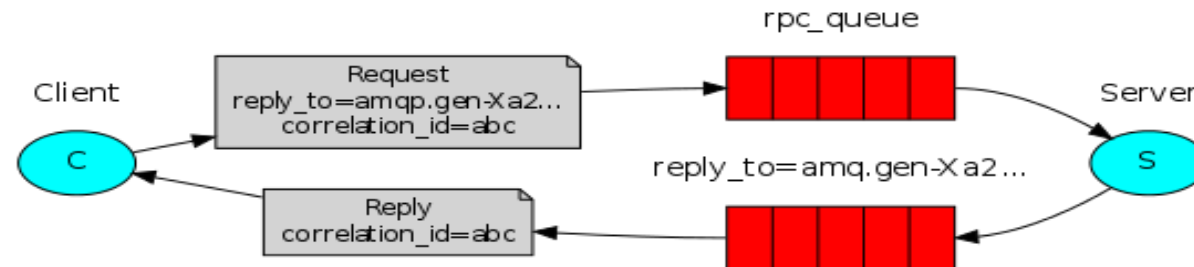
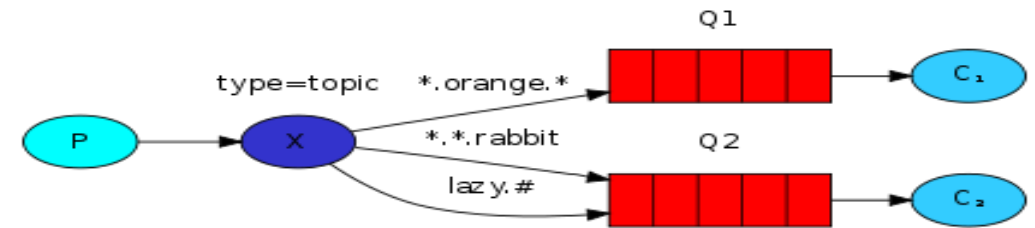
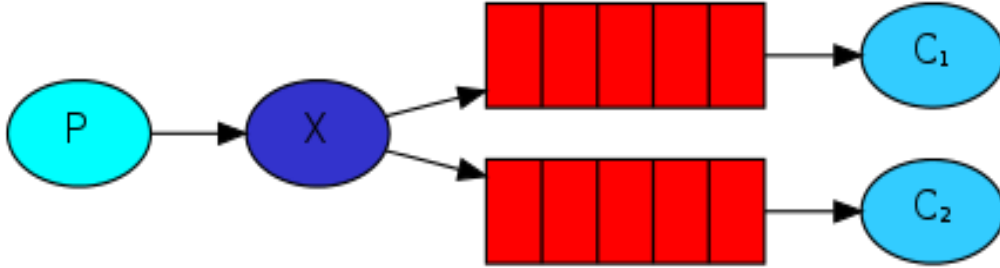
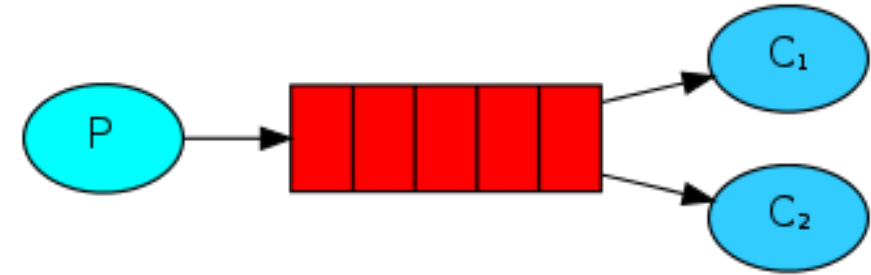
AMQP



AMQP

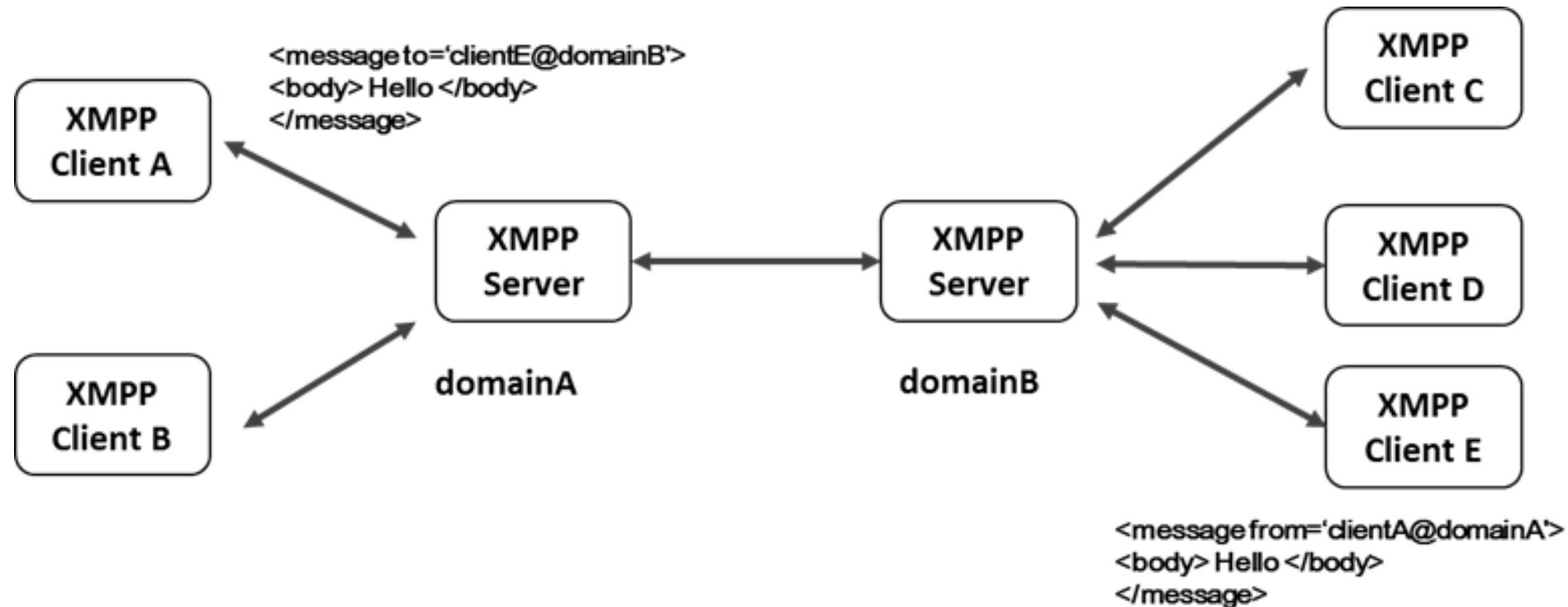
- AMQP
 - Beaucoup plus lourd que le MQTT
 - Adapté aux situations exigeant:
 - Fiabilité
 - Sécurité
 - A plusieurs schéma de fonctionnement

AMQP



XMPP

- Le **XMPP** est un protocole conçu à l'origine pour la messagerie instantanée (Google Talk, Jabber, iChat).



Conclusion du concept pub/sub

- Facilite l'échange de données entre objets/processus
- Il est utilisé dans beaucoup d'endroits
 - Robotique
 - SMS payants de vote
 - Architecture scalable
 - Logs
 - Etc...

Représentation de la donnée

Représentation de la donnée

- XML

```
<?xml version="1.0" encoding="utf-8"?>
<station>
  <adress>17 RUE SOLFERINO </adress>
  <status>0</status>
  <bikes>11</bikes>
  <attachs>28</attachs>
  <paiement>AVEC_TPE</paiement>
  <lastupd>18 secondes</lastupd>
</station>
```

Représentation de la donnée

- JSON

```
{  
  "station": {  
    "adress": "17 RUE SOLFERINO ",  
    "status": "0",  
    "bikes": "11",  
    "attachs": "28",  
    "paiement": "AVEC_TPE",  
    "lastupd": "18 secondes"  
  }  
}
```

Représentation de la donnée avec python

- Utilisation des dictionnaires en python (différent des tableaux)

- Générer un json

```
import json
```

```
message = {}
```

```
message["value"]=10
```

```
print json.dumps(message)
```

- Charger un JSON depuis un string

```
import json
```

```
jsonstring = "{\"value\": 10}"
```

```
test = json.loads(jsonstring)
```

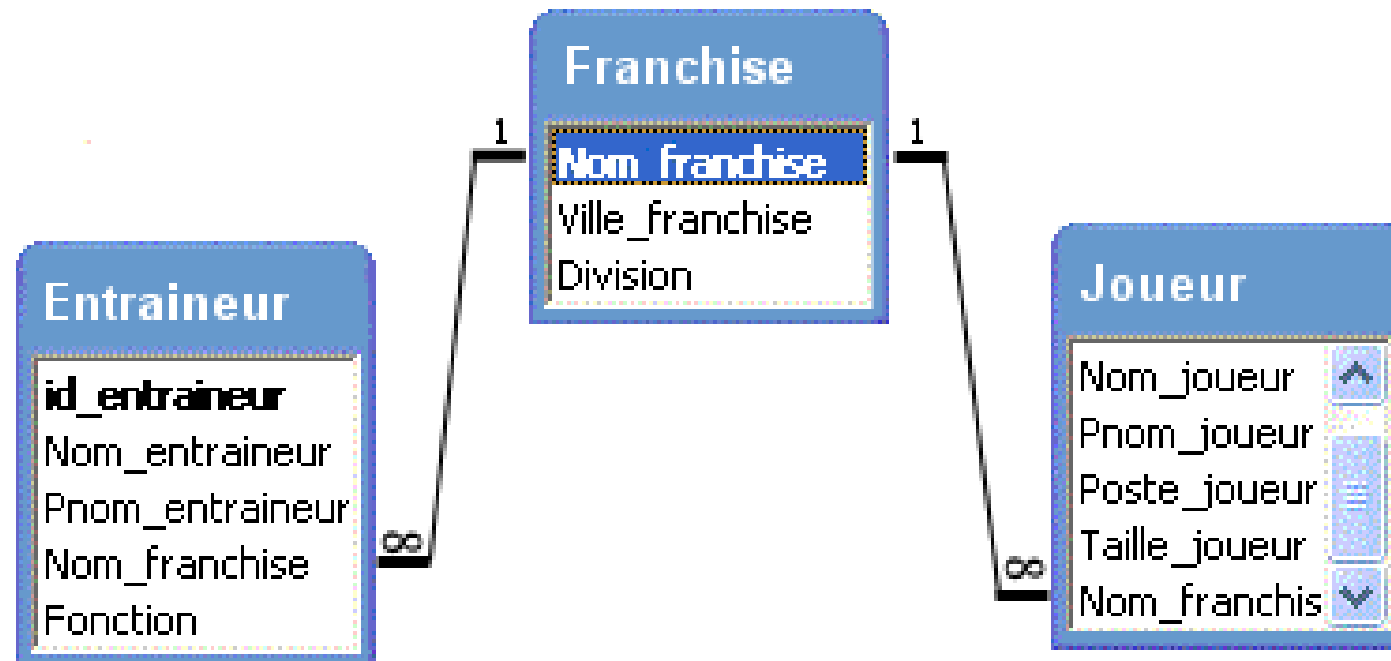
```
print test["value"]
```

Exemple avec le python et javascript simulateur de température

- Réutiliser le simulateur MQTT
- En javascript pour passer d'un string à un json
`JSON.parse(xxxxxxxxxxxx)`
- Attention, il y a aucune sécurité dans le code que nous avons écrit
 - Pas de test si json ou pas par exemple
 - Pas de test sur le type de la donnée, etc...

Stockage de la donnée

- SQL
 - Base de donnée relationnelle



Stockage de la donnée

- SQL
 - Plusieurs moteur
 - MySQL
 - PostgreSQL
 - Oracle
 - Sqlite

Stockage de la donnée

- SQL

- Ecriture: exemple en python avec sqlite

```
import sqlite3  
import datetime
```

```
conn = sqlite3.connect('example.db')
```

```
c = conn.cursor()  
c.execute("""CREATE TABLE data (date text, sensor text, value real)""")
```

```
temperature = 25.3  
sensor = "temperature_chambre"  
date = datetime.datetime.now().isoformat()
```

```
c.execute("INSERT INTO data VALUES (?, ?, ?)", (date, sensor, temperature) )  
conn.commit()  
conn.close()
```

Stockage de la donnée SQL

- Lecture: exemple en python avec sqlite

```
import sqlite3
import json

def dict_factory(cursor, row):
    d = {}
    for idx, col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d

connection = sqlite3.connect("example.db")
connection.row_factory = dict_factory
cursor = connection.cursor()
cursor.execute("select * from data")

# fetch all or one we'll go for all.
results = cursor.fetchall()

print json.dumps(results)

connection.close()
```

Réaliser un log de températures

- Premier programme: OK
 - Notre simulateur de température qui publie ses données en MQTT
 - {« value »: 22.9 } dans un topic donné
- Second programme: A faire
 - Ecoute en MQTT et stockage en sqlite
- Vérifiez que tout est ok avec le troisième programme (lecture de la base de donnée sqlite)

Stockage de la donnée

- SQL
 - Avantage
 - Relationnelle
 - Bien structurée
 - Puissant
 - Ayant fait ses preuves
 - Désavantage
 - Structure fixe

Stockage de la donnée

- NoSQL
 - Les bases **clef/valeur**, permettent de stocker des informations sous forme d'un couple clef/valeur où la valeur peut être une chaîne de caractère, un entier ou un objet sérialisé.
 - Les bases orientées **colonnes**, ressemble aux bases de données relationnelles, car les données sont sauvegardées sous forme de ligne avec des colonnes, mais se distingue par le fait que le nombre de colonnes peut varier d'une ligne à l'autre.
 - Les bases orientées **document**, représente les informations sous forme d'objet XML ou JSON. L'avantage est de pouvoir récupérer simplement des informations structurées de manière hiérarchique.
 - Les bases orientées **graphe**, présentent les données sous forme de noeud et de relation. Cette structure permet de récupérer simplement des relations complexes.

Stockage de la donnée

- Exemple mongodb et python
 - Installez mongo db
 - `sudo apt-get install mongodb`
 - Lancez mongodb
 - `cd`
 - `mongod -dbpath .`
 - Installez la bibliothèque pymongo
 - `sudo pip install pymongo`

Stockage de la donnée

- Exemple mongodb et python
 - Installez mongo db
 - `sudo apt-get install mongodb`
 - Lancez mongodb
 - `cd`
 - `mongod -dbpath .`
 - Installez la bibliothèque pymongo
 - `sudo pip install pymongo`

Stockage de la donnée

- Exemple mongodb et python

```
from pymongo import MongoClient
import datetime
```

```
client = MongoClient()
client = MongoClient('localhost', 27017)
db = client.exemple
data = db.data
```

```
sensor = {}
sensor["value"]=21.5
sensor["date"]=datetime.datetime.now().isoformat()
sensor["sensor"]="temperature_cuisine«
```

```
data.insert_one(sensor)
```

Stockage de la donnée

- Exemple mongodb et python

```
from pymongo import MongoClient
from bson.json_util import dumps
```

```
client = MongoClient()
client = MongoClient('localhost', 27017)
db = client.exemple
data = db.data
```

```
print dumps(list(data.find()))
```

- Pour une recherche spécifique: print
 - `data.find({"sensor": "temperature_cuisine"})`

Récupérer les données et les afficher sur une page Web

- Le but est de faire travailler les clients et non le serveur
- Utilisation d'API qui retourne uniquement des json (la donnée utile)
- Exemple de framework
 - Nodejs
 - Python – Flask

Récupérer les données et les afficher sur une page Web

- Exemple avec python flask
 - Installation de la bibliothèque Flask: `sudo pip install Flask`

- nano serverHTTP.py

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route("/")  
def hello():  
    return "Hello World!"
```

```
if __name__ == "__main__":  
    app.run()
```

- `python serverHTTP.py`
- Ouvrez ensuite votre navigateur: `http://127.0.0.1:5000/`

Récupérer les données et les afficher sur une page Web

- Exemple avec python flask – récupération d'un paramètre en URL

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route("/")  
def hello():  
    return "Hello World!"
```

```
@app.route('/sensors/<sensorName>')  
def show_sensor_value(sensorName):  
    # show the user profile for that user  
    return 'Sensor Name: %s' % sensorName
```

```
if __name__ == "__main__":  
    app.run()
```

- Testez avec: <http://127.0.0.1:5000/sensors/nimportequoi>

Récupérer les données et les afficher sur une page Web

- Exemple avec python flask – récupération d'un paramètre en URL

```
from flask import Flask
import sqlite3
import json

def dict_factory(cursor, row):
    d = {}
    for idx, col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d

app = Flask(__name__)
connection = sqlite3.connect("example.db")
connection.row_factory = dict_factory
cursor = connection.cursor()

@app.route('/sensors/<sensorName>')
def show_sensor_value(sensorName):
    t=(sensorName,)
    cursor.execute("select * from data where sensor=?",t)
    results = cursor.fetchall()
    return json.dumps(results)

if __name__ == "__main__":
    app.run()
```

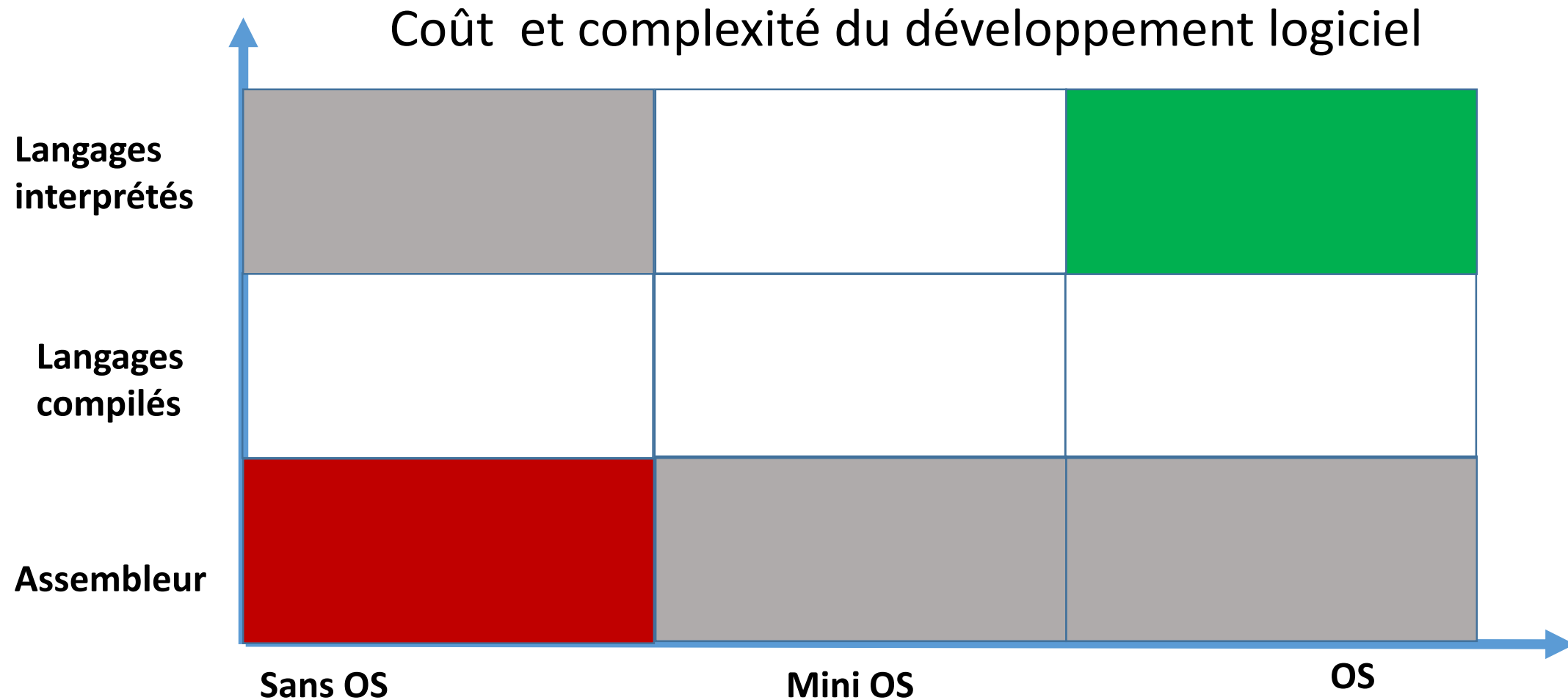
Software - Systèmes embarqués

- C'est l'usage qui fait le système embarqué
- Un système dédié à un usage clairement identifié

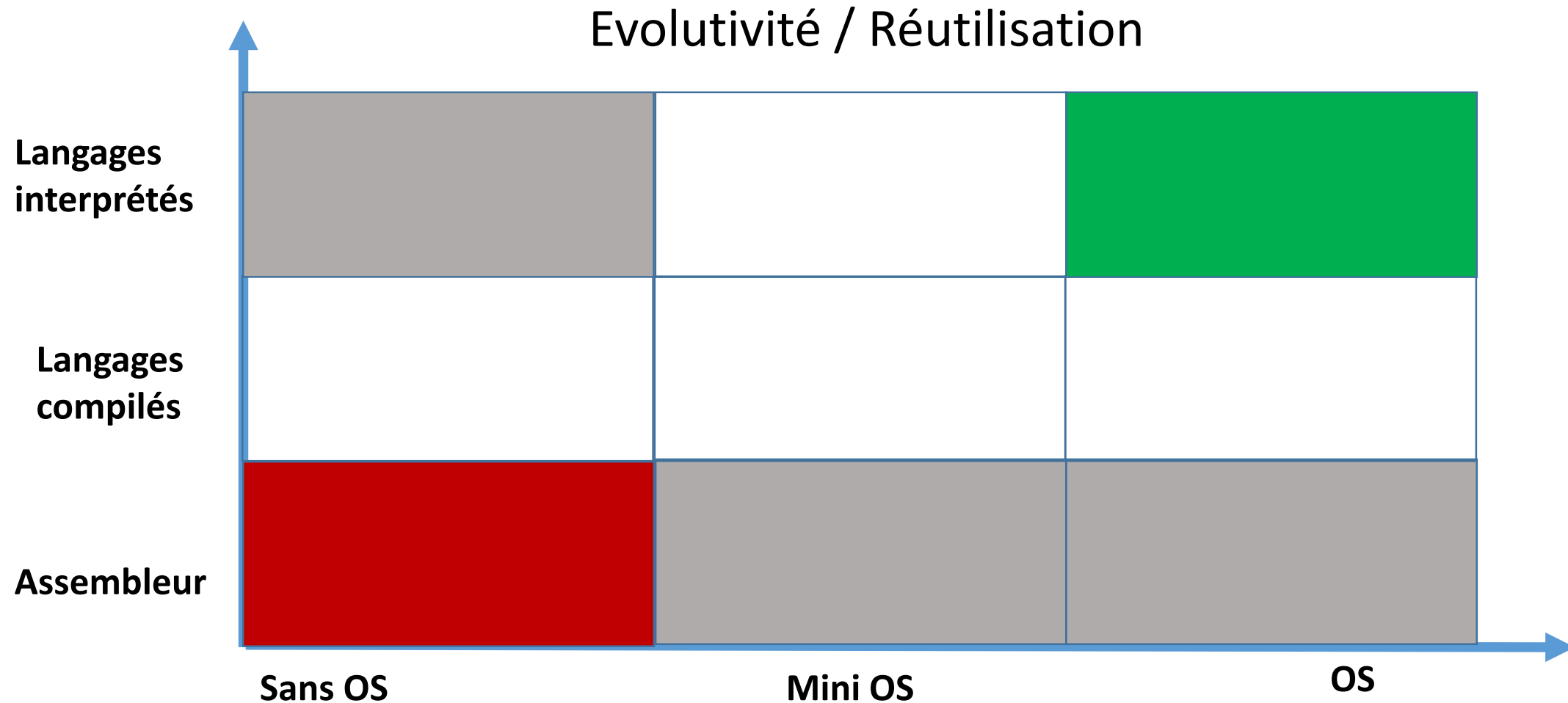
PERSPECTIVE DÉVELOPPEMENT

Langages interprétés		Open AT + Lua	openWRT + Python openWRT + Javascript Linux + Java
Langages compilés	C Ada	Contiki + C Ravenscar + Ada	Linux + C++ Windows Emb. + C# Android Wear + Java
Assembleur	Thumb-2 x86		
	Sans OS	Mini OS	OS

PERSPECTIVE DÉVELOPPEMENT



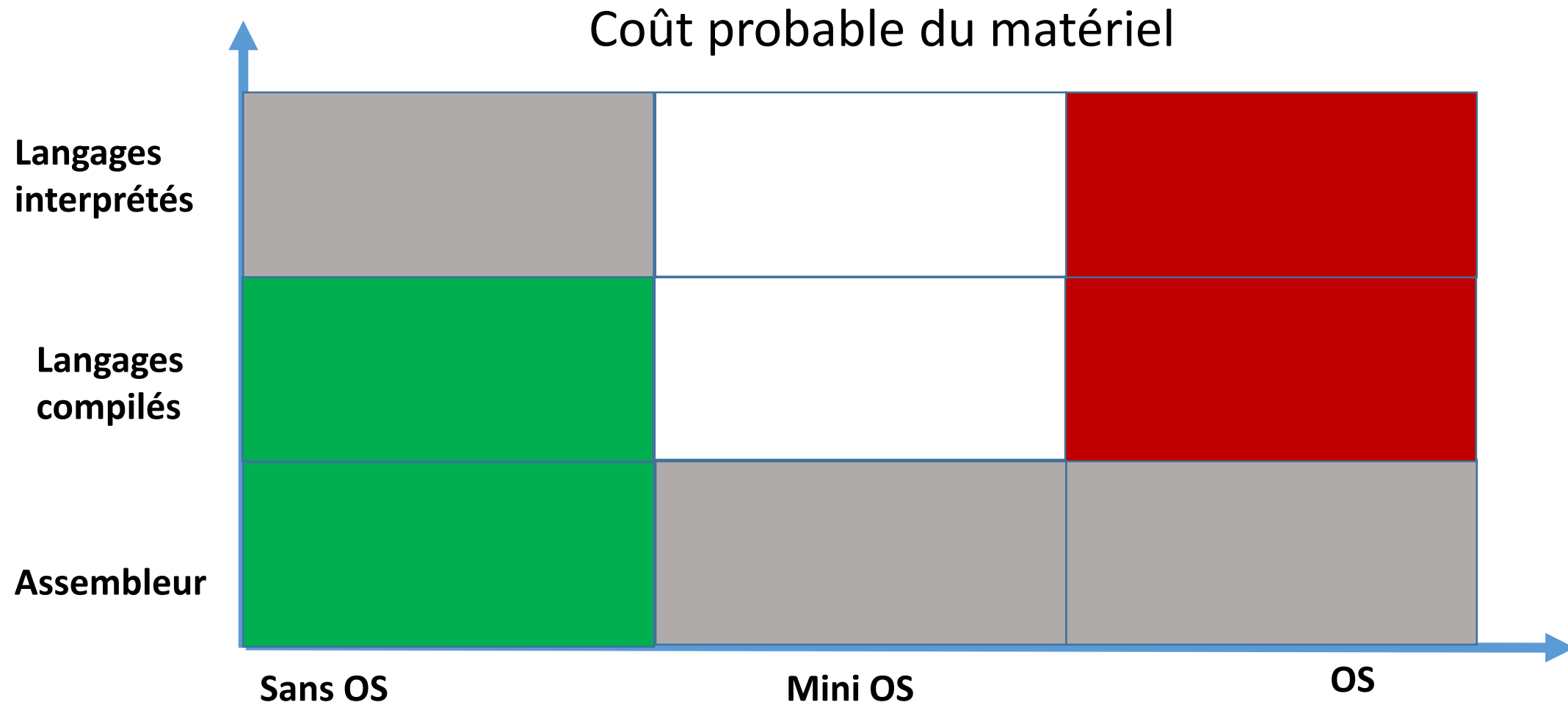
PERSPECTIVE DÉVELOPPEMENT



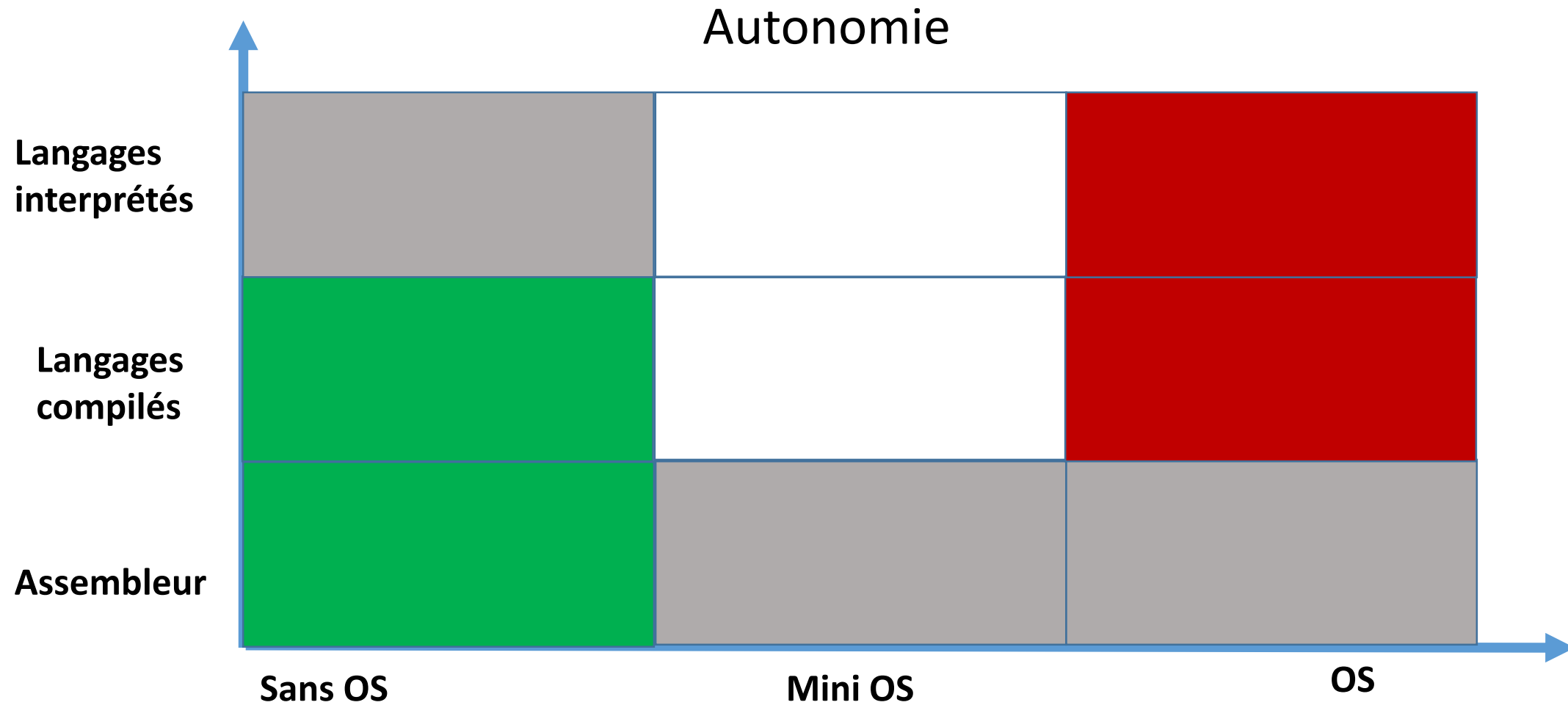
PERSPECTIVE DÉVELOPPEMENT

So why all the pain ?

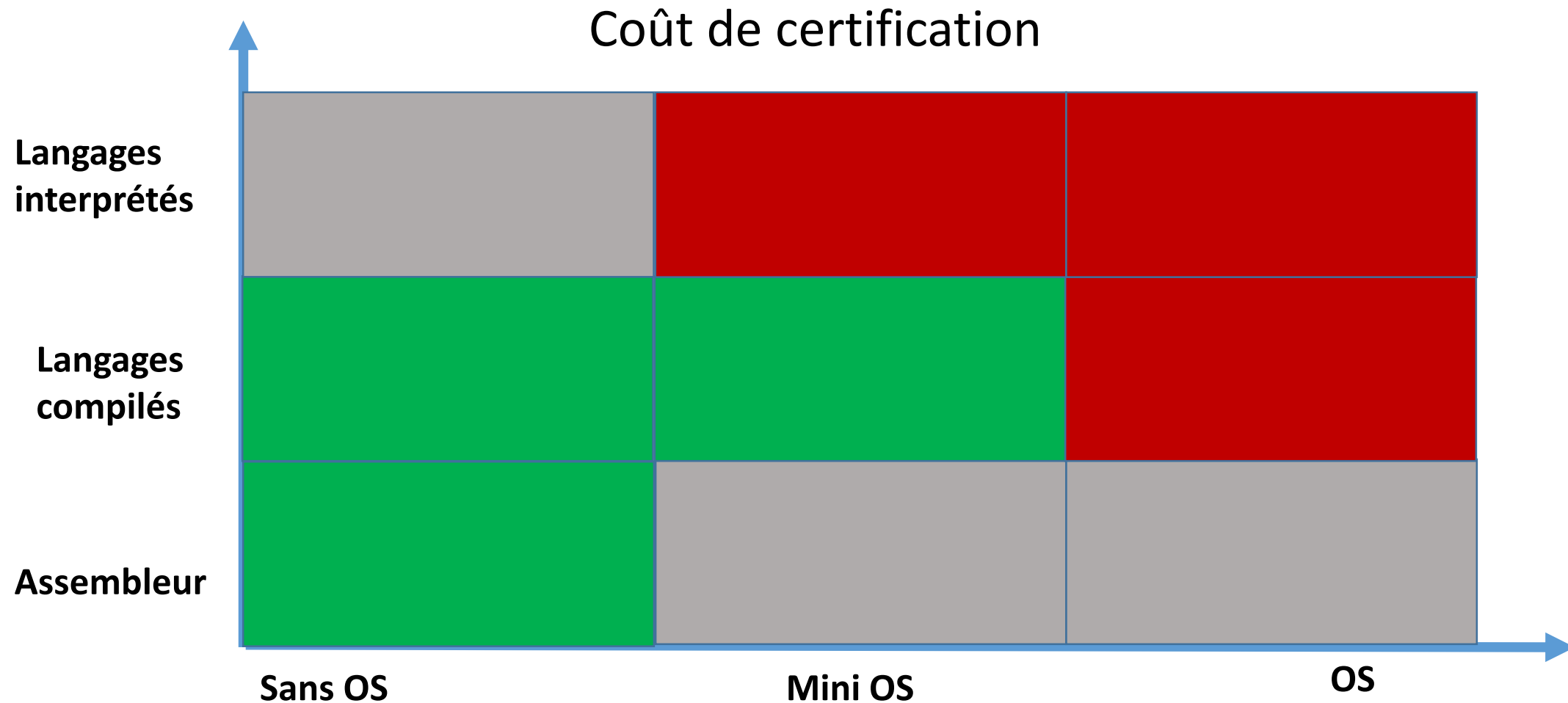
PERSPECTIVE DÉVELOPPEMENT



PERSPECTIVE DÉVELOPPEMENT

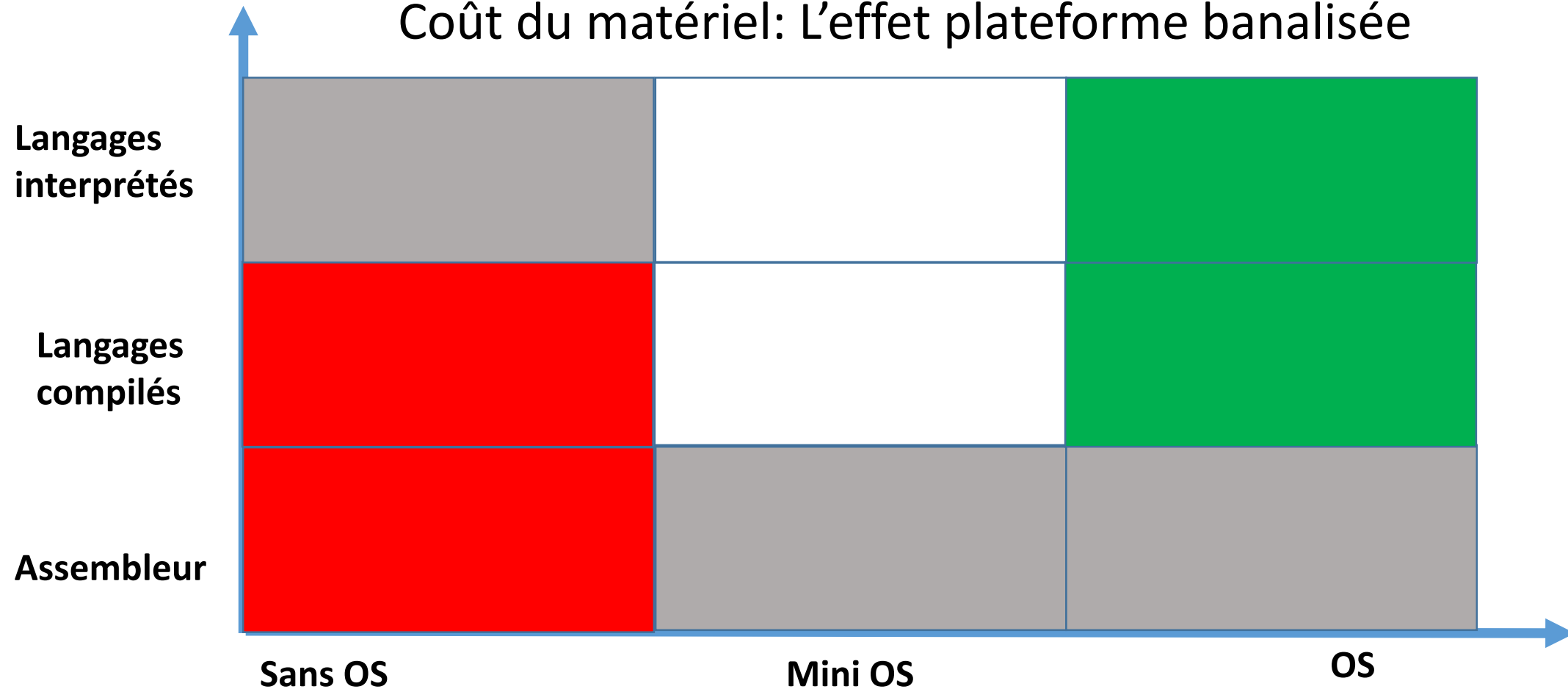


PERSPECTIVE DÉVELOPPEMENT



PERSPECTIVE DÉVELOPPEMENT

Coût du matériel: L'effet plateforme banalisée



CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

- Système électronique
- Système informatique
- Système autonome
- Système spécialisé dans une tâche / dédié à un usage
- Limité en espace et en énergie

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Une plateforme dimensionnée pour:

- Respecter les contraintes temporelles
 - *Système temps-réel*
- Respecter les contraintes de coût
- Respecter les contraintes de consommation énergétique

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Systèmes embarqués critiques

- Sûreté et sécurité
 - Certain système embarqué peuvent mettre des vies humaines en danger. Ce sont des système critique et ne doivent jamais faillir
- Résultats juste
- Délais juste
- Sécurisé la donnée
- Sécurité dans la transmissions

Exemple: DO178B

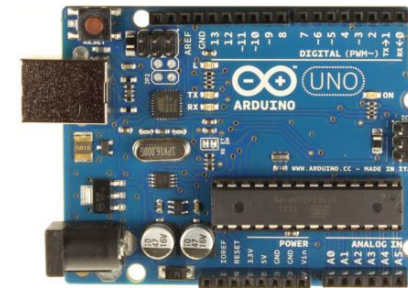
CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Lequel est temps-réel ?



Quad-Cœur 2 Ghz
8 GB de RAM

VS



16 Mhz
2 kB de RAM

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

- Terme souvent mal utilisé
- Définition: « lors d'un développement logiciel sur une plateforme embarquée, la prise en compte du respect des contraintes temporelles est aussi important que l'exactitude des résultats »
- Le système doit donc livrer des résultats exacts et dans les temps
- La bonne information, au bon moment
- Hard real-time / soft real-time

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

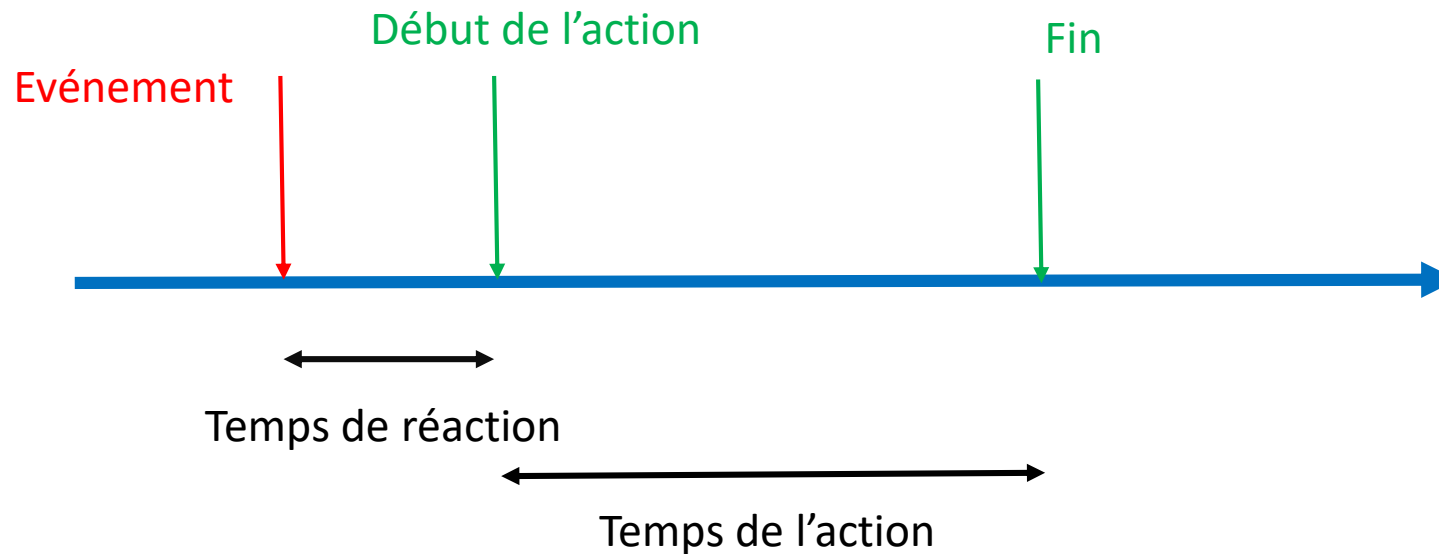
Temps-réel

- Apporter une réponse aux événements en temps borné
 - Freins dans une voiture
 - Événements
 - Frein
 - Desserrer les freins
 - Instrument de musique numérique
 - Événements
 - Frappe de la touche

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

Temps de réponse à un événement, temps déterministe (borné)



CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

- Priorité dans les événements
 - Certains événements sont plus importants que d'autres
 - Un traitement en cours peut être interrompu, pour traiter un événement plus important, et ainsi de suite
- Éviter les inversions de priorités

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

- Pour avoir un système temps réel complet:
 - Toute les tâches d'un système doivent être temps réel
 - Tout les éléments de la chaine doivent être temps réel
- Pour un système complexe, parfois impossible
 - Partitionnement physique ou virtuel des ressources
 - Exemple ARINC 653
 - Arduino YUN / Intel Edison

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

Domaines:

Aéronautique

Transport

Production

Fusion de données

...

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

- Dans le cas ou un système a plusieurs tâches en parallèle:
 - Utilisation d'un OS temps réel
 - Ordonnancer les tâches en fonction de leur durées
 - Ordonnancer les tâches en fonction de leur priorités
 - Ordonnancer les tâches suivant leurs fréquences
 - Gestion des interruptions et événements

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

- OS non temps-réel
 - Windows
 - MACOSX
 - Linux de base
- OS temps réel
 - Windows CE -> payant
 - VxWorks -> payant
 - FreeRTOS
 - RTLinux
 - Contiki OS
 - Etc...

CARACTÉRISTIQUES DU SYSTÈME EMBARQUÉ

Temps-réel

Pour résumer:

- « temps réel » ne veut pas dire rapide
- « temps réel » implique une forme de déterminisme
 - être sûr à 100% que le système réponde à un évènement dans un temps borné, mais qui peut-être long.
- En cas de présence d'un système d'exploitation
 - C'est une propriété de l'OS, en particulier de l'ordonnanceur de tâche, et n'a rien à voir avec le langage utilisé.

PLAN

1. Objet connecté / Internet des Objets
2. **Systèmes embarqués pour l'IdO**
 - Embarqué ou pas ?
 - Développement embarqué
 - Caractéristiques du système embarqué
 - **Les plateformes embarquées**
3. Réseaux et capteurs pour l'IdO

PLATEFORMES EMBARQUÉES

Choix de la plateforme en fonction des besoins:

- Consommation électrique
- Besoin de calcul
- Besoin de temps réel
- Complexité
- Ecosystème

Eviter de sous-dimensionner et de sur-dimensionner

PLATEFORMES EMBARQUÉES

Principaux types de plateformes matérielles:

- Microcontrôleurs
- DSP
- FPGA
- Microprocesseurs
- Hybrides

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Différence par rapport à un microprocesseur
 - Consommation
 - Coût
 - Dimensions
 - Vitesse de fonctionnement
 - Unités périphériques
 - Interfaces d'entrées-sorties
 - Pas toujours d'OS

PLATEFORMES EMBARQUÉES

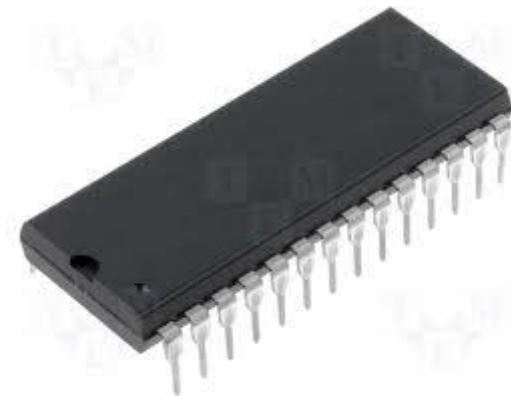
Microcontrôleurs

- Exemples de microprocesseurs
 - TI MSP430
 - ATMEL AVR
 - Microchip PIC
 - Intel 8051
 - ARM Cortex M0
 - ARM Cortex M3

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Quelques MHz
- 4 bits à 64 bits (voir jusqu'à 128 bits)
- Moins cher
- Peu consommateurs d'énergie
- Whatchdog
- Interruptions



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Le processeur (CPU)
 - Mission: rechercher instruction en mémoire, les décoder, et les exécuter
 - Eléments:
 - Unité arithmétique et logique (UAL): calcul simple de base (+,-,*/, ET OU NOT)
 - Unité de contrôle (UC): décodage des instructions en instructions simples, récupération des données en mémoires, mémoriser l'adresse de la prochaine instruction, mémoriser les résultats des calculs en mémoire
 - Registre CPU: PC (compteur instructions), registre instruction, registre de données, registres d'adresses

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- La mémoire
 - Stockage des instructions
 - Stockage des données
 - Stockage de la pile (pour sauvegarder le contexte d'exécution)
 - Mémoire est constitué de module de mémorisation binaires: mots de 8, 16, 32, ou 64 bits
 - La mémoire est adressée par mots de 8 ou 16 bits.

PLATEFORMES EMBARQUÉES

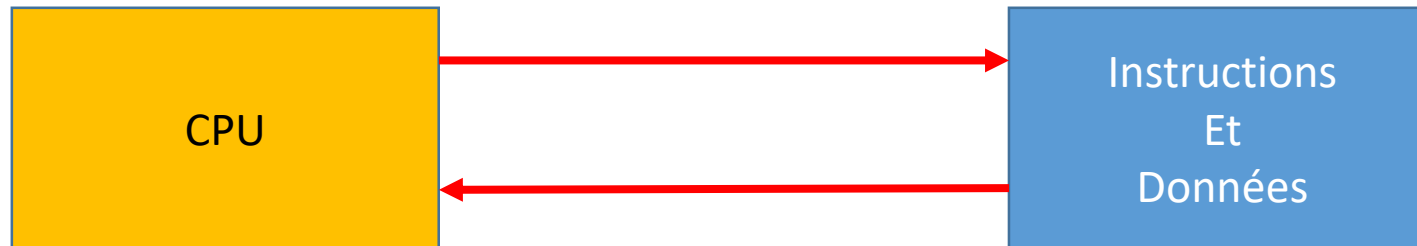
Microcontrôleurs

Adresses	Données
0001h	1010 1010 1010 1010
0002h	0000 0000 0000 0000
0003h	1111 0000 1111 0000
...
xxxxh	0101 0101 0101 0101

PLATEFORMES EMBARQUÉES

Microcontrôleurs

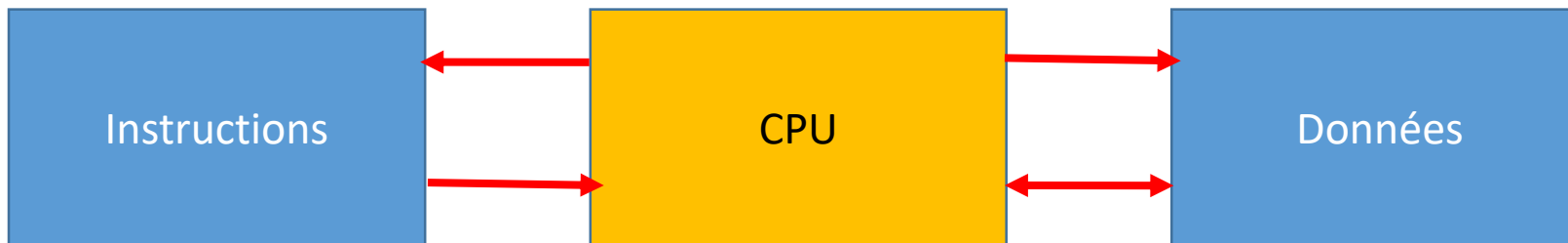
- Espace d'adressage: Architecture Von Neuman



PLATEFORMES EMBARQUÉES

Microcontrôleurs

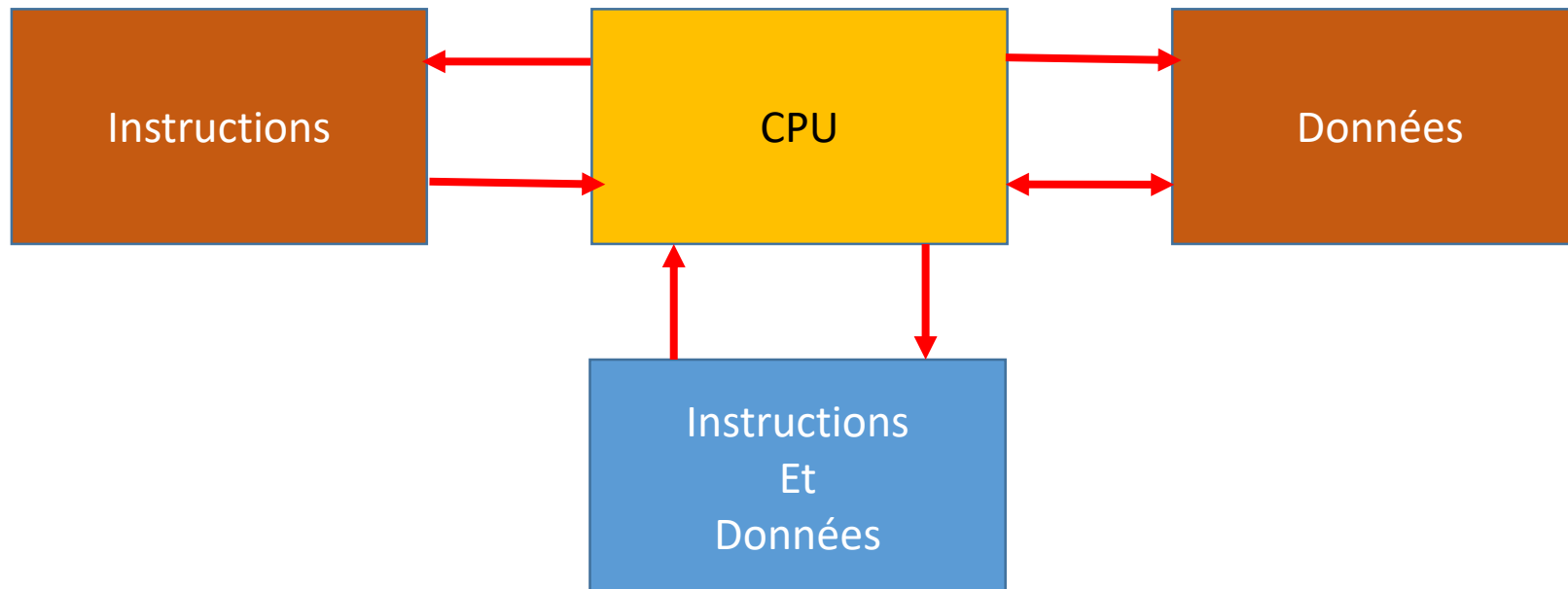
- Espace d'adressage: Architecture Harvard



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Espace d'adressage: Architecture Harvard modifiée



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Les différents types de mémoires:
 - Mémoire de données
 - Mémoire vive (RAM) volatile donc efface hors tension
 - Mémoire morte (EEPROM) non-volatile
 - Mémoires programmes
 - ROM
 - PROM (Programmable read only memory)
 - EPROM ou UVPROM (Erasable Programmable read only memory)
 - EEPROM (Electrically erasable programmable read only memory)
 - Flash

PLATEFORMES EMBARQUÉES

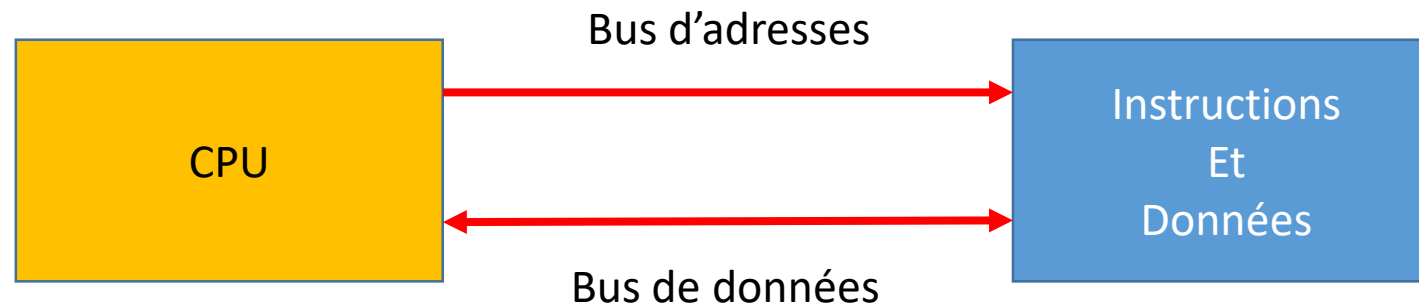
Microcontrôleurs

- Le bus système
 - « Lignes » permettant de relier le processeur, la mémoire, les entrées sorties
 - Les différentes lignes:
 - Lignes d'adresses – unidirectionnel (venant du processeur)
 - Lignes de données – bidirectionnelles

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Le bus systèmes dans Architecture Von Neuman



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Les horloges
 - Circuit oscillateur délivrant des impulsions à une certaine fréquence
 - Plus la fréquence est grande plus les opérations se feront rapidement
 - Permet la synchronisation des opérations
 - Base de temps

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Entrées / Sorties
 - Analogique
 - Convertisseur analogique / numérique (ADC)

Convertir une Tension en un nombre

Résolution par rapport a un nombre de bits

Exemple: Arduino A/N sur 10 bits pour 5 volts, résolution de 0,0045 volts

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Entrées / Sorties
 - Analogique
 - Plus de précision sur convertisseur analogique / numérique (ADC)
 - 3 méthodes
 - Variation simple ou double rampes
 - Variation par palier
 - Variation par approximation

PLATEFORMES EMBARQUÉES

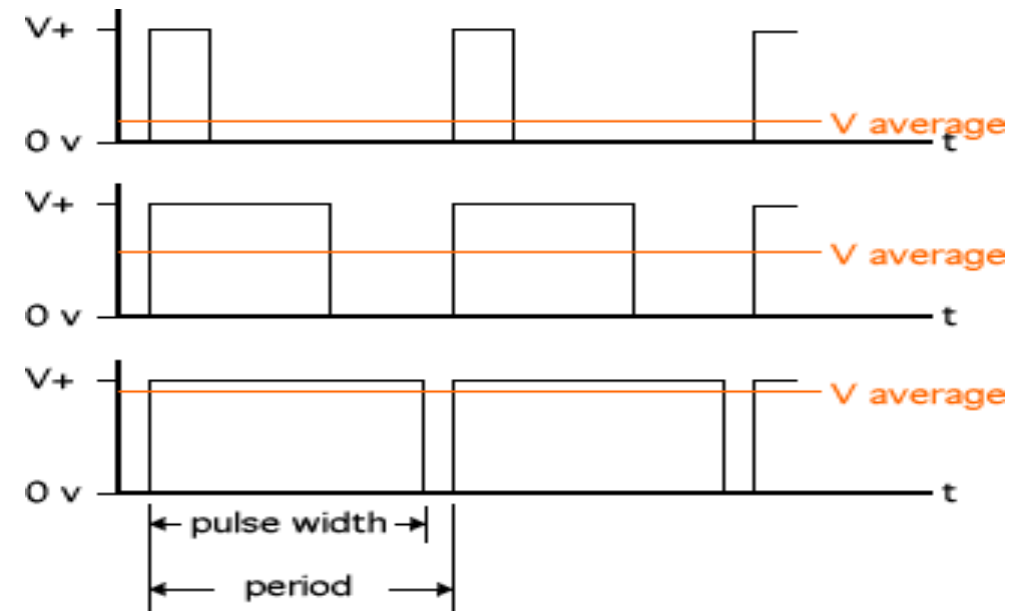
Microcontrôleurs

- Entrées / Sorties
 - Numérique
 - Générateur d'état haut/bas (par exemple 0 volt et 5 volt)
 - Niveaux logiques (1 ou 0)
 - Détection d'états

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Entrées / Sorties
 - Numérique
 - Générateur de PWM (Pulse Width Modulation)
 - Influence sur le duty cycle
- On génère grâce à cela des tensions analogique
- Exemple Pour obtenir 4,2 volts avec une tension de 5 volts: $(4,2/5) * 100 = 84\%$



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Schéma bloc MSP430
- Schéma bloc ARM Cortex M3

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Les Timers / Compteurs
 - Mesurer des durées ou des fréquences
 - Compteurs interne incrémenté par un signal externe ou interne
 - On peut modifier la fréquence grâce à des pré-diviseurs
 - Plusieurs compteurs dans un microcontrôleur (8 bits / 16 bits)
 - Exemple Timer sur 8 bits: compte de 0 à 255 en boucle

PLATEFORMES EMBARQUÉES

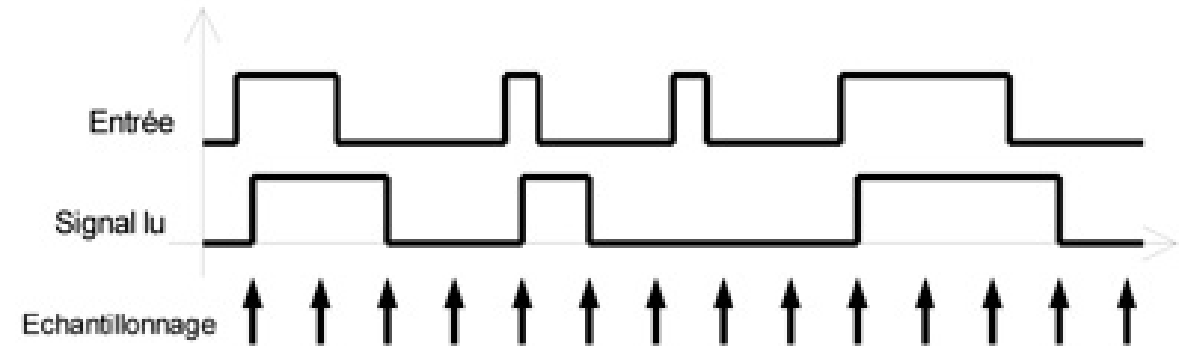
Microcontrôleurs

- Les interruptions
 - Un microcontrôleur n'est pas multitâches
 - Événement externe asynchrone
 - GPIO
 - Timer / compteur
 - Arrivée d'un événement inopiné
 - Capturer cet événement
 - Erreur a capturer (division par zéro par exemple)

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Sans Interruption
 - Mauvais échantillonnage
 - Manquer un événement
 - Scrutation périodique
 - Utilisation de ressources
 - Monopolise du temps processeur
 - Temps de réaction long



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Fonctionnement d'une interruption
 - Arrêt de l'exécution de ce que le microcontrôleur fait: interruption du programme en cours
 - Sauvegarde de l'état complet du programme en mémoire
 - Exécution d'une méthode, routine ou fonction liée a cette interruption
 - Restauration du contexte
 - Exécution du programme

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Les interruptions, quelques événements:
 - Appuie d'un bouton
 - Pin qui change d'état
 - Erreur
 - Un compteur qui déborde
 - Routine régulière (mix avec les timers)
 - Etc...

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Les interruptions, quelques exemples:
 - Un compteur et son débordement
 - Mode sleep et réveil, très important pour la consommation énergétique
 - Un compteur de temps avec un seuil fixe
 - Lancer une application toute les minutes

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Rôle du Watchdog (même principe que les interruptions)
 - Assurer le bon fonctionnement
 - S'assure que le programme n'est pas bloqué
 - Redémarrage si jamais il y a un soucis
 - Peut être software ou hardware



PLATEFORMES EMBARQUÉES

Microcontrôleurs

Différents schémas de communication

- Full duplex
- Half duplex
- Simplex

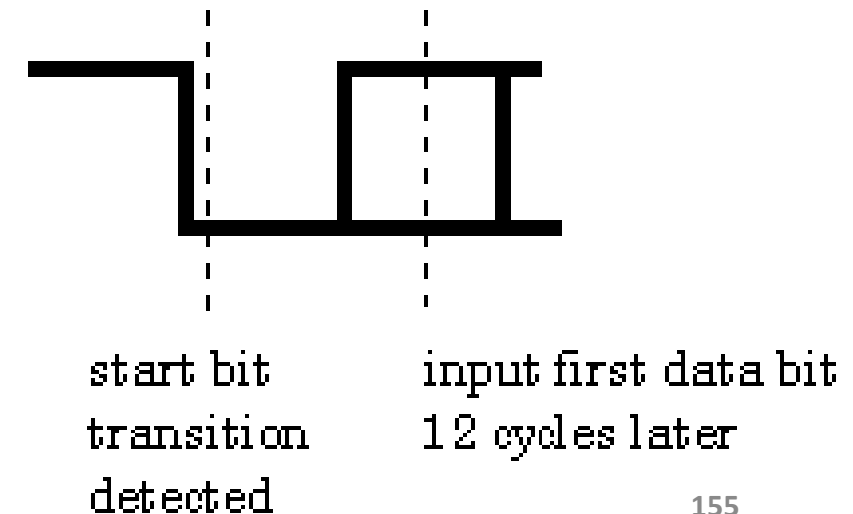
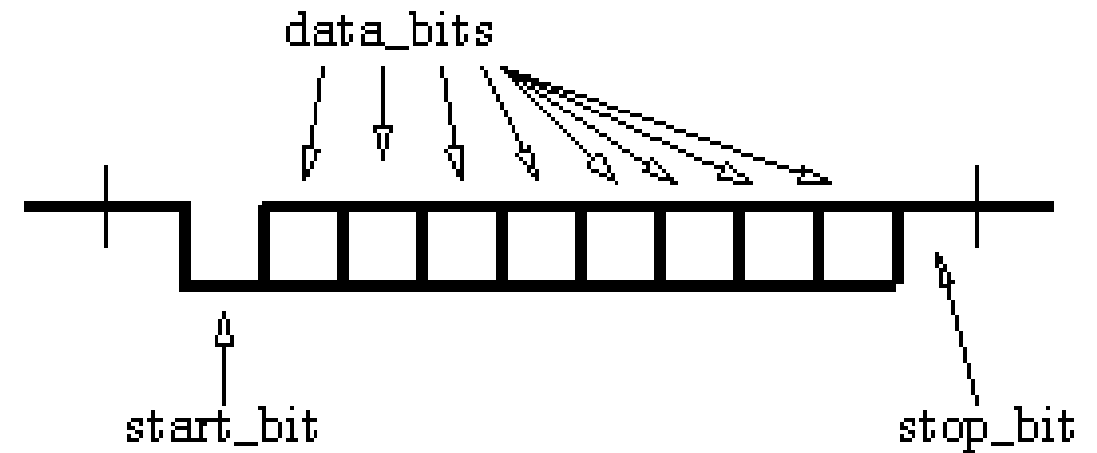
PLATEFORMES EMBARQUÉES

Microcontrôleurs

Communication: UART

- Convertir une liaison parallèle en série
- émetteur-récepteur asynchrone universel
- Masse en commun
- Bit de démarrage 0
- Un émetteur
- Un récepteur

Exemple: liaison entre un modem et un microcontrôleur



PLATEFORMES EMBARQUÉES

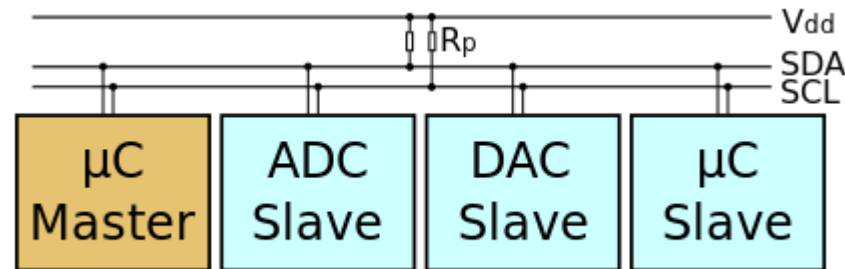
Microcontrôleurs

- Communication: série
 - Simple à mettre en place
 - Obligé de connaître la vitesse de transmission côté maître esclave
 - Réseau complexe si beaucoup d'objets.
- Utilisation d'un bus

PLATEFORMES EMBARQUÉES

Microcontrôleurs

Les bus de données: I2C



SDA (Serial Data Line) : ligne de données bidirectionnelle,

SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle

Masse commune

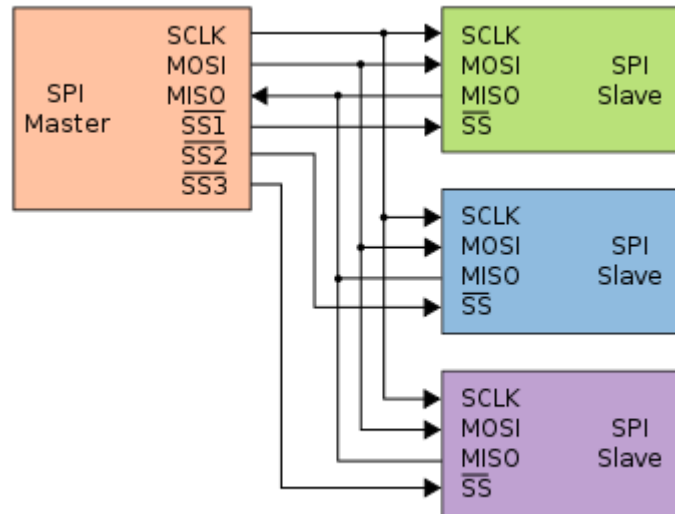
I2C -> Standard mode » débits ≤ 100 kbit/s

Communication: Half duplex

PLATEFORMES EMBARQUÉES

Microcontrôleurs

Les bus de données: SPI



SCLK — Serial Clock (Maître)

MOSI — Master Output Slave Input

MISO — Master Input, Slave Output

SS — Slave Select (Actif à l'état bas / par le Maître)

Masse commune

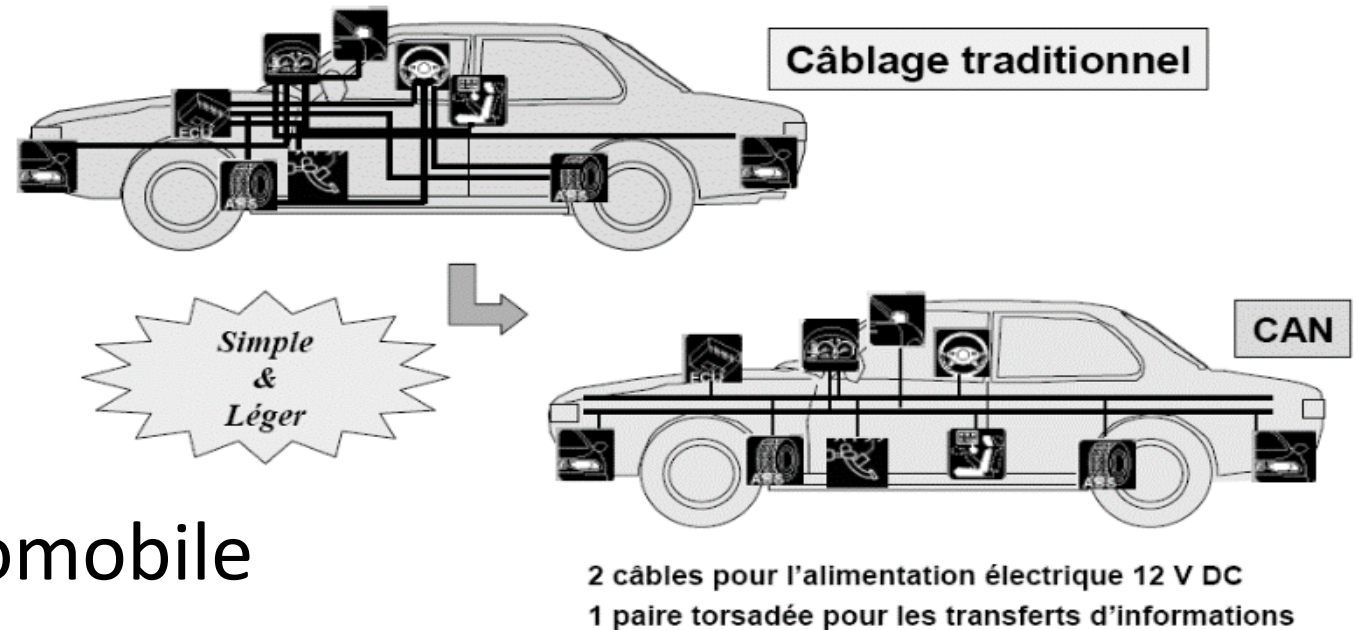
Débit du SPI: 48 Mbits

Communication: Full Duplex

Besoin de plus de fils

PLATEFORMES EMBARQUÉES

Microcontrôleurs

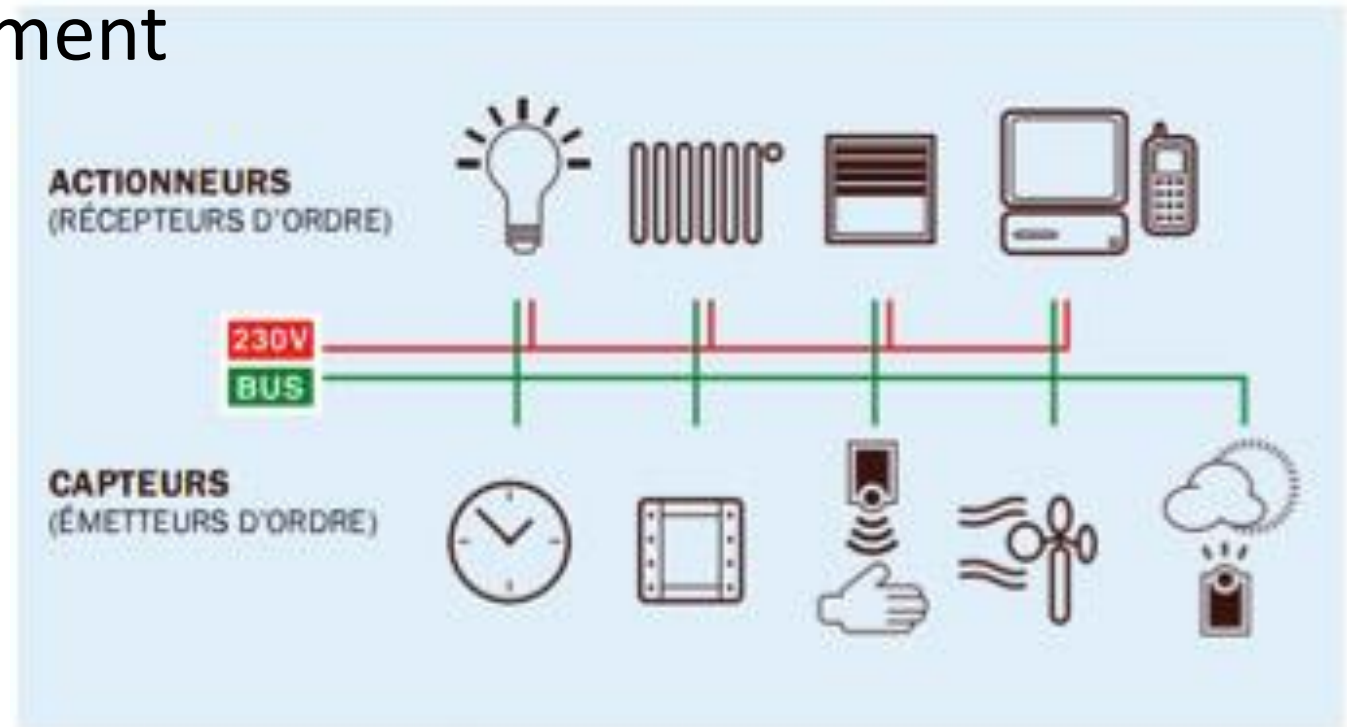


- Les autres bus: pour l'automobile
 - Bus CAN (Controler Area Network, ISO 11898)
 - Bus Flexray (ISO 17458-1 à 17458-5)

PLATEFORMES EMBARQUÉES

Microcontrôleurs

- Les autres bus: pour le bâtiment
 - KNX
 - Modbus



PLATEFORMES EMBARQUÉES

Microcontrôleurs

- USB: Universal Serial Bus
 - Composé de quatre fils
 - La masse
 - Alimentation
 - Deux fils de données appelés D- et D+
 - D+ et D- forment une paire torsadée, transmission différentielle

PLATEFORMES EMBARQUÉES

DSP

- DSP
 - « Digital Signal Processor »
 - Structure Identique à un microcontrôleur
 - Plus haut en fréquence
 - Optimiser pour le traitement du signal
 - Réaliser des filtres numériques
 - Coprocesseurs FFT



PLATEFORMES EMBARQUÉES

FPGA

- FPGA
 - Circuit numérique programmable
 - Composé de portes logiques
 - Programmation en VHDL
 - Plus performant qu'un microcontrôleur
 - Moins cher qu'un ASIC



PLATEFORMES EMBARQUÉES

Microprocesseurs

- Single-Board computer
 - Raspberry PI
- SoC
 - Intel Edison (Atom 500 Mhz + Quark 100 MHz)
 - Atheros AR9331

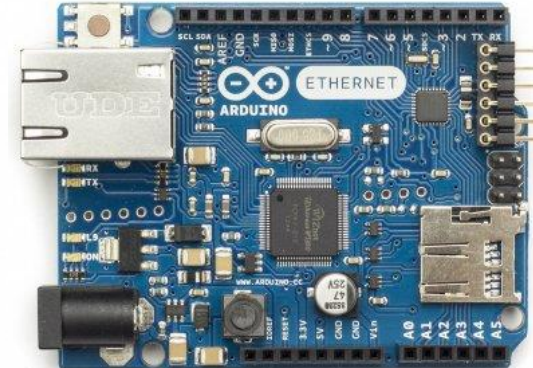
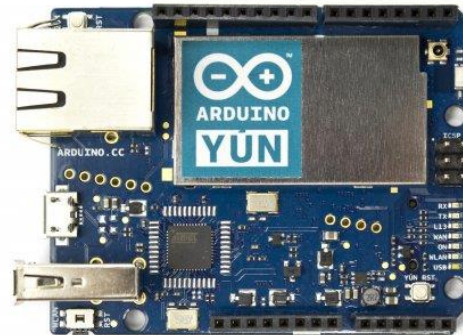
La famille Arduino

- Une plateforme matérielle et logicielle libre
 - www.arduino.cc
- Carte électronique avec
 - Microcontrôleur ATMEGA AVR, 16 MHz, 8 bits
 - Régulateur 5V
 - Entrées / sorties (numériques / analogiques)
 - Bootloader programmation série
 - Pas d'OS

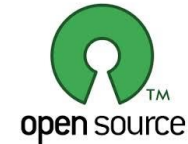


La famille Arduino

- Uno / Mega / Leonardo / Mini / Nano / Yun / Ethernet / ...



De 12€ à 52€



La famille Arduino

AR9331 + microcontrôleur

OS sur AR9331:

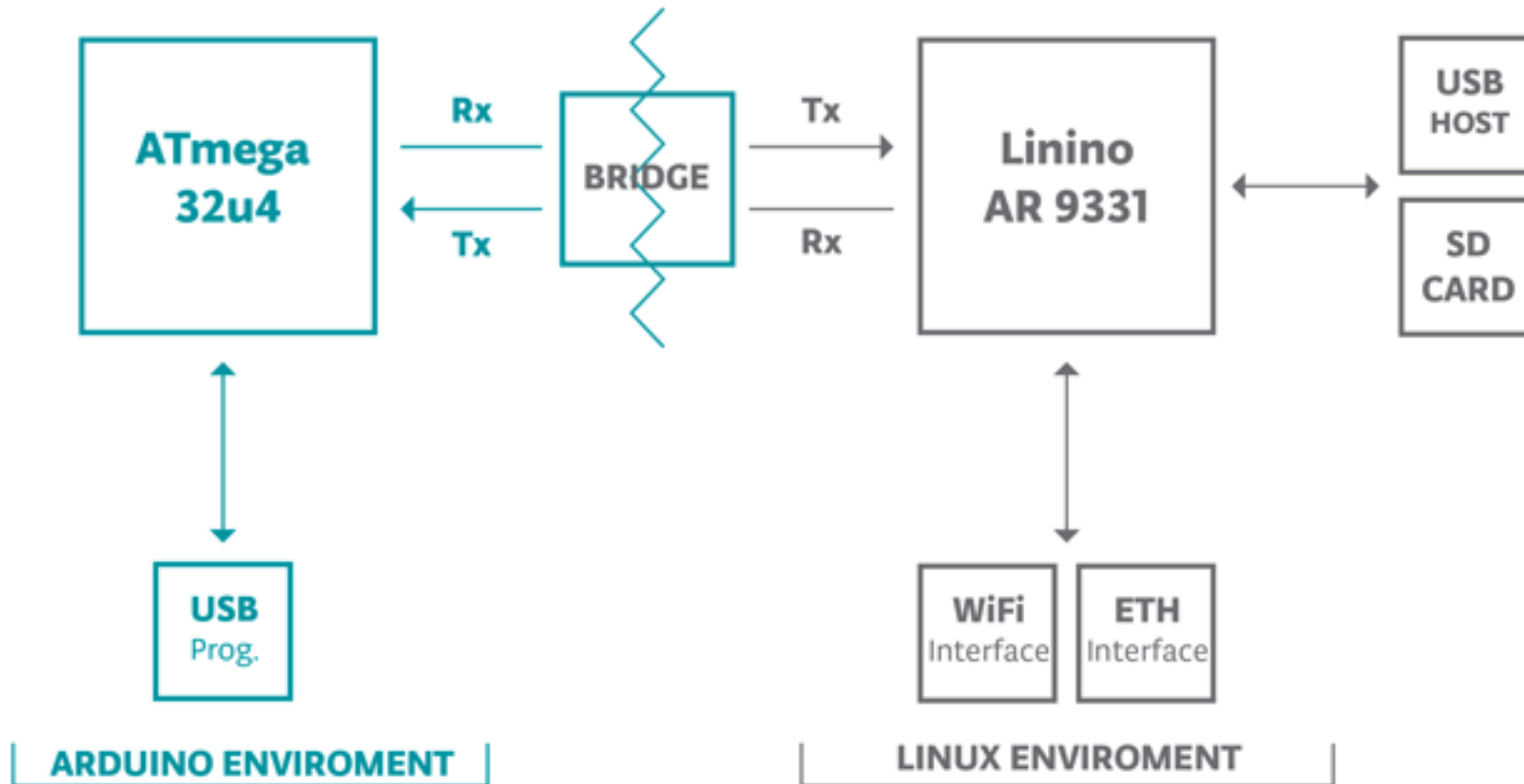
- OpenWRT
- SoC pour routeur
- Excellente gestion Wi-Fi, Ethernet

Microcontrôleur Atmega32U4:

- Arduino classique
- Passerelle avec microprocesseur

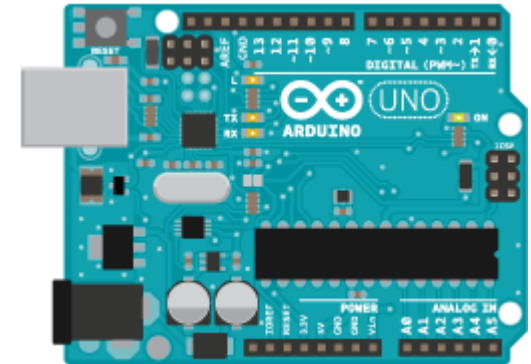


La famille Arduino



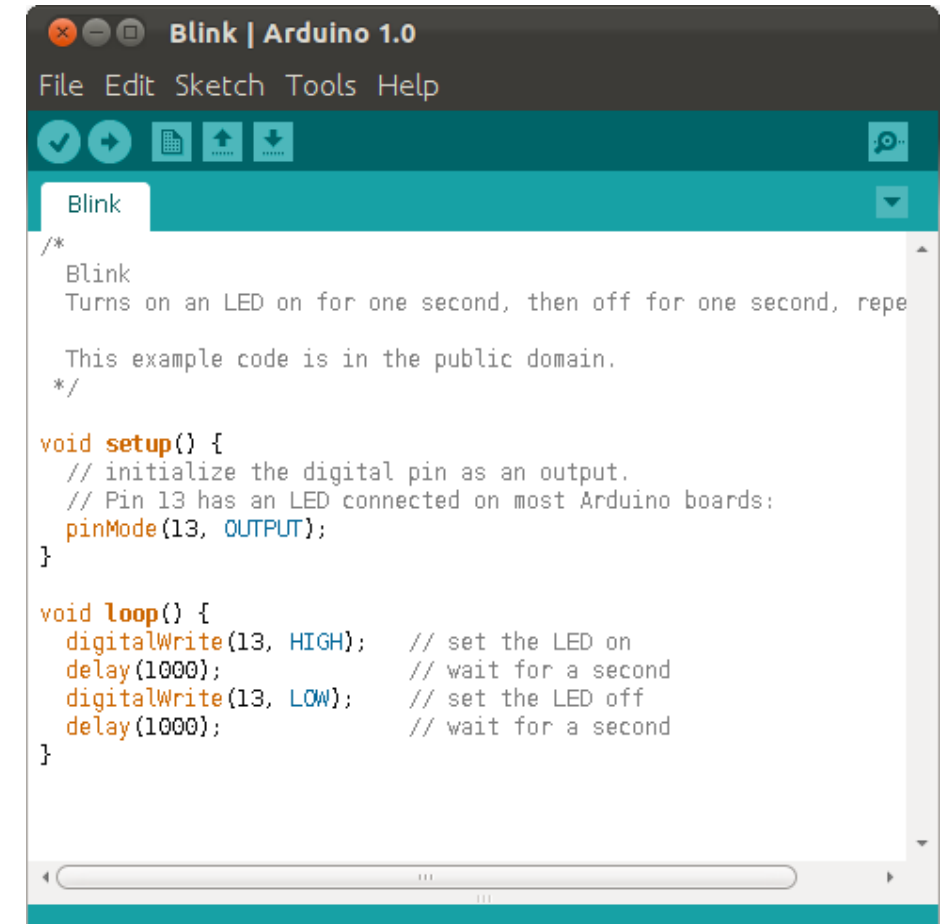
La famille Arduino

- Les extensions (shields)
 - WiFi / Ethernet / Bluetooth / NFC / GSM
 - LCD / ePaper
 - Contrôle moteurs
- Accessoires nombreux et variés



La famille Arduino

- Les outils de développement
 - Une plateforme complète libre
 - IDE (C /C++)
 - Compilateur
 - Bibliothèques

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening files, saving, and uploading. The main text area shows the code for the "Blink" sketch. The code is as follows:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repe  
  This example code is in the public domain.  
  */  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH);   // set the LED on  
  delay(1000);              // wait for a second  
  digitalWrite(13, LOW);    // set the LED off  
  delay(1000);              // wait for a second  
}
```

La famille Arduino

- Une famille de plateformes dérivée compatibles

TinyDuino

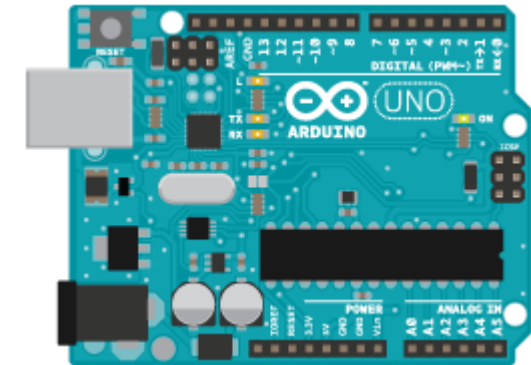


FLORA

netduino

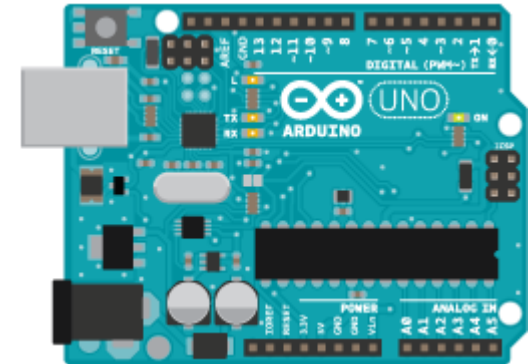
- Un réseau de distributeurs à valeur ajoutée

- Arduino.cc
- Seeed Studio
- Cooking Hacks
- AdaFruit



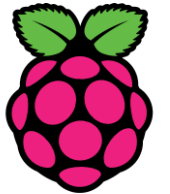
Apprendre Arduino

- Apprendre
 - Les tutoriels (Arduino / AdaFruit / Cooking Hacks)
 - Les exemples
 - Les forums
 - Les kits
- Arduino c'est simple

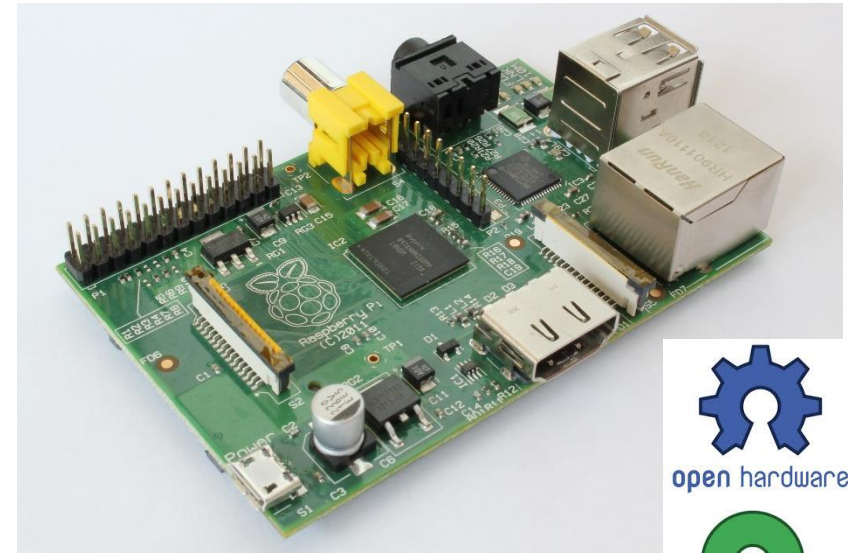


Raspberry Pi

- Une plateforme pour encourager les vocations
 - Fondation pour l'éducation
 - www.raspberrypi.org
- Un nano-ordinateur de la taille d'une carte de crédit
 - Microcontrôleur ARM
 - Entrées / sorties digitales
 - USB / Ethernet / HDMI / RCA / Jack
 - Carte SD
 - OS famille Linux



RaspberryPi



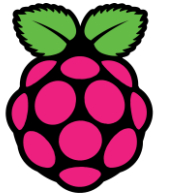
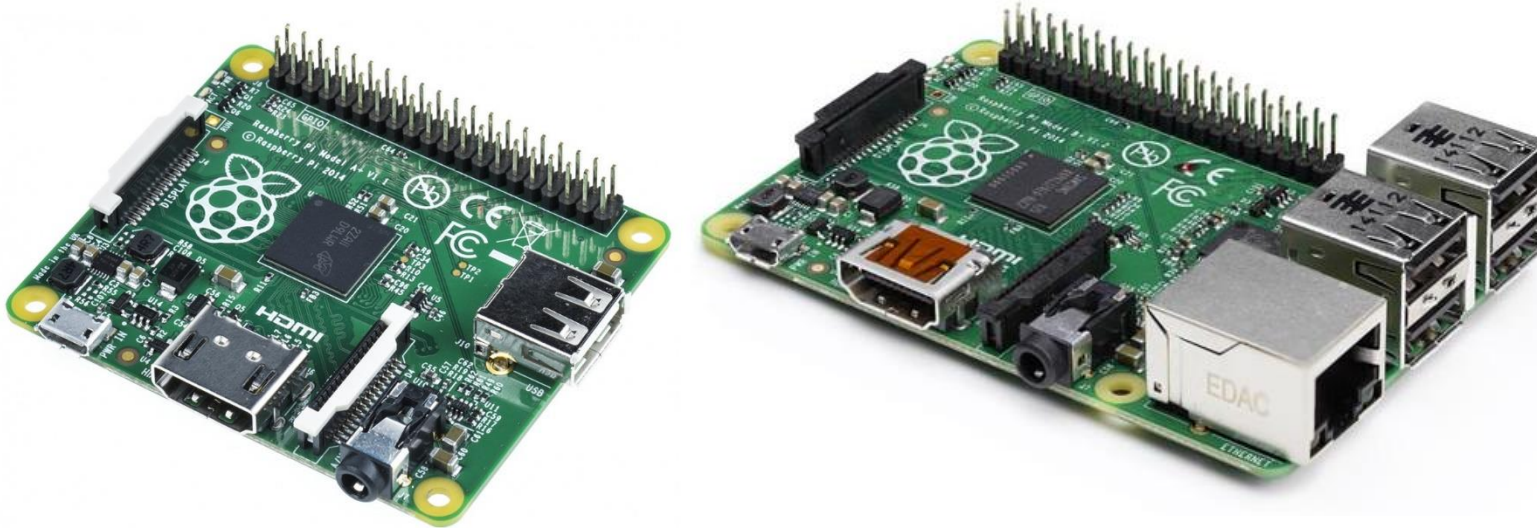
open hardware



open source

Raspberry Pi

- Modèles A+ / B+ / Compute Module



RaspberryPi



De 18€ à 34€



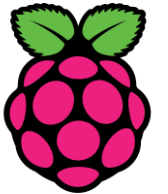
open hardware



open source

Raspberry Pi

- Les accessoires
 - PC
 - Caméra
 - Ecrans tactiles



RaspberryPi



open hardware



open source

Raspberry Pi

- Les outils de développement

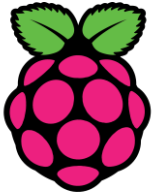
-



et [scratch](#)



- Et aussi tous les outils de développement informatique PC (C / C++ / Python / Java / C# / F# / Ada / ...)



RaspberryPi



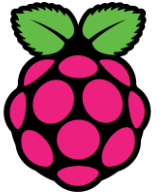
open hardware



open source

Raspberry Pi

- Apprendre
 - Les tutoriels ([Raspberry.org](https://www.raspberrypi.org/) / AdaFruit / ...)
 - Les exemples
 - Les forums
 - [The MagPI](#)



RaspberryPi



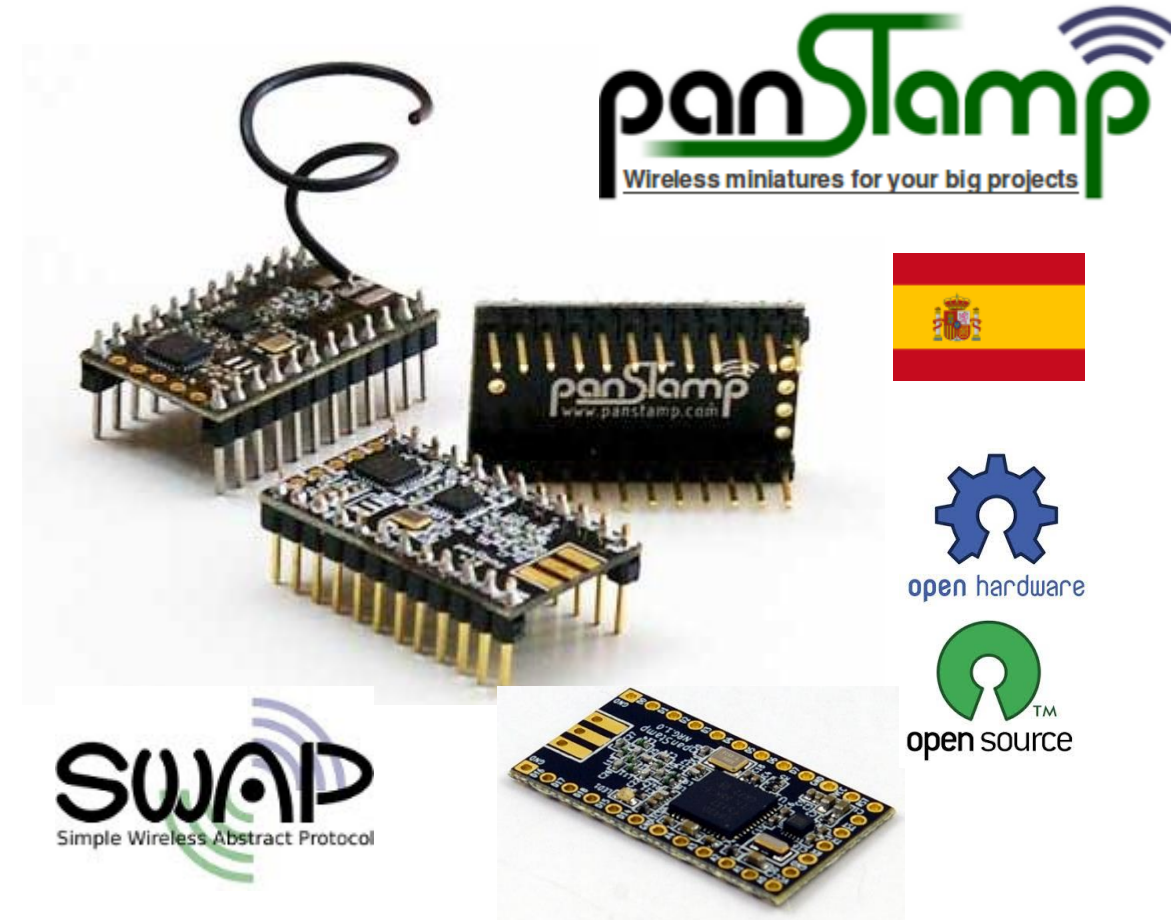
open hardware



open source

SAMSYS® PanStamp

- Origine
 - Encore une plateforme libre
- Architecture
 - Microcontrôleur ATMEL + Interface radio TI CC1101
 - Ou microcontrôleur TI CC430
 - Entrées sorties numériques / analogiques
 - Communication radio 868 MHz



- Extensions
 - Cartes capteurs (température / humidité / pression / ..)
 - Carte interface Raspberry PI
 - Carte relais



- Les outils de développement
 - Compatible Arduino IDE
 - Protocole SWAP
- Apprendre
 - Tutoriels
 - Forum
 - Compatible Arduino



LaunchPad MSP430

TI LaunchPad

- Origine
 - Promouvoir l'apprentissage de l'architecture MSP430
- Architecture
 - MSP430, 8 MHz 16 bits
 - programmeur / debugueur
 - Pas de bootloader
 - ES numériques / analogiques
 - Très faible consommation
 - Très faible coût



open hardware



open source

LaunchPad MSP430

- Les extensions (BoosterPack)
 - Bluetooth / NFC
 - DSP
 - Touch
- Ecosystème
 - Olimex
 - Anaren
 - DLP Design
 - ...

TI LaunchPad




LaunchPad MSP430

- Les outils de développement

- Code Composer Studio

-

- IA  **Energia**

- Apprendre

- [Wiki TI](#)



TI LaunchPad



- Un microprocesseur Atom 32 bits, 500 MHz
 - Linux / Windows 10 (prochainement)
- Un microcontrôleur Quark 32 bits, 100 MHz
 - RTOS

