Smile
ECS
Embedded & Connected Systems

IoT, état des lieux en 2017

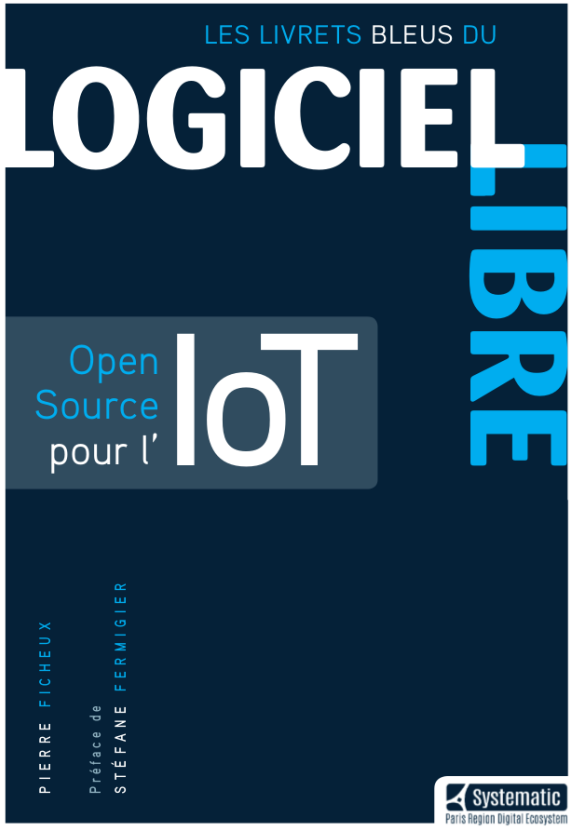
Pierre FICHEUX (pierre.ficheux@smile.fr)

07/2017

Smile ECS
Embedded & Connected Systems

$ whoami

- Développeur Linux embarqué, enseignant, auteur
- Co-fondateur Open Wide en 2001
- CTO Smile-ECS (Embedded & Connected Systems)



Smile ECS
Embedded & Connected Systems

Sujets évoqués

- Généralités
- Solutions globales libres ou propriétaires
- Protocoles
- OS
- Use cases

Smile ECS
Embedded & Connected Systems

Quelques citations :-)

« IoT is the new hip way of talking about Embedded Systems, something that's been around for ~50+ years »

K. Yaghmour

« Security is not a technical challenge, it's a business problem »

K. Yaghmour

« The manufacturer sells a camera and that's the end of the revenue stream, that cannot work a long term »

C. Simmonds

Smile ECS
Embedded & Connected Systems

Une définition de l'IoT

- IoT (Internet Of Things)
  - Extension d'Internet à des objets et à des lieux du monde physique
  - Échanges d'informations et de données provenant de dispositifs présents dans le monde réel vers/depuis le réseau Internet

« Cyber-physical systems (CPS) enable the physical world to merge with the virtual leading to an Internet of things, data, and services. »

Smile ECS
Embedded & Connected Systems

Nouveau concept ?

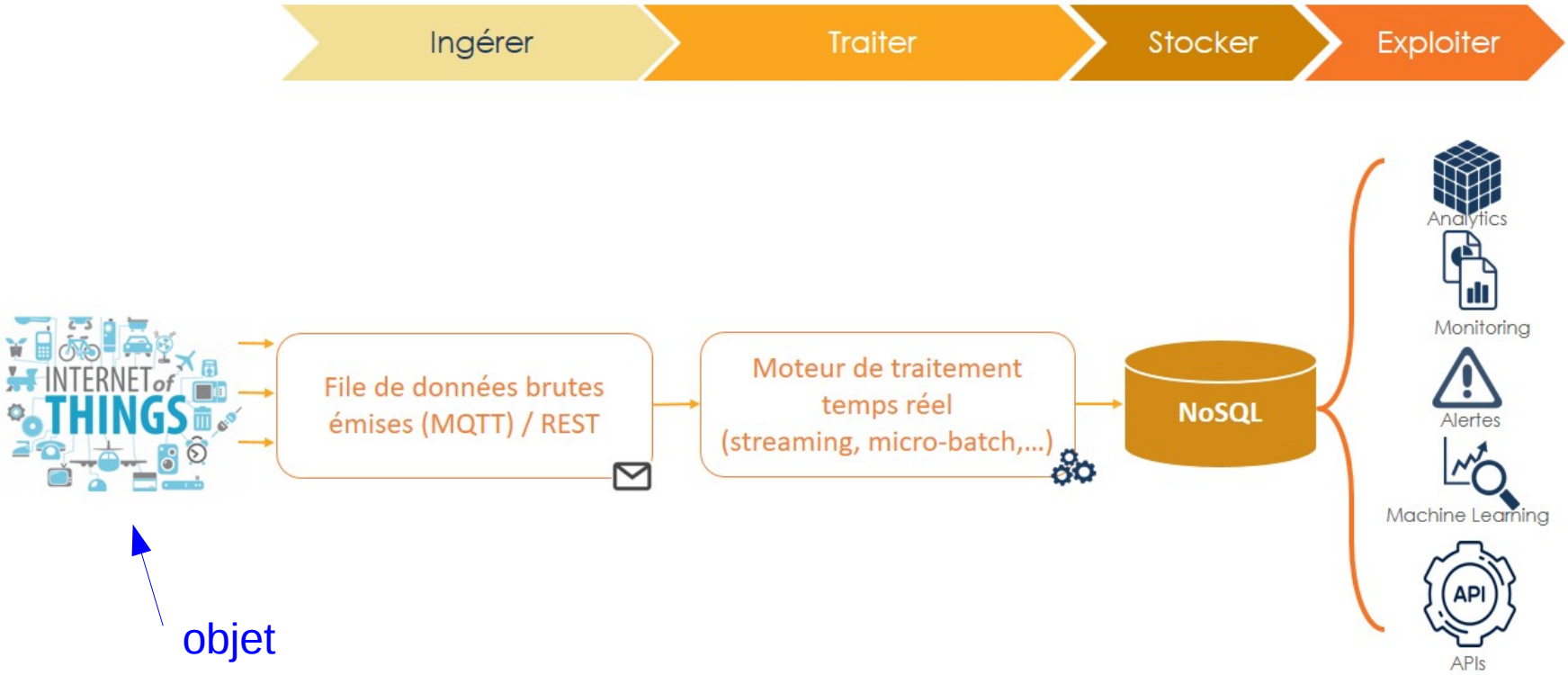
- IoT n'est pas un nouveau concept au sens technologique du terme
- Conséquence de la généralisation des systèmes embarqués
- Omniprésence de la technologie et de la communication auprès du grand public
- Utilise/nécessite les technologies d'intégration actuelle des réseaux (IPv6, « cloud », MoSQL)
- Adaptation aux architectures matérielles des objets
- Au niveau industriel, évolution du M2M (machine to machine) vers des protocoles standards → unification des réseaux

Déjà évoqué en 1995

- En 1995, le chercheur français Christian HUITEMA évoque l'évolution vers l'IoT

  « Il y a déjà des microprocesseurs, en fait de tout petits ordinateurs dans bien d'autres endroits [...]. D'ici quelques années, le développement et les progrès de l'électronique aidant, ces microprocesseurs deviendront sans doute de vrais ordinateurs élaborés et il sera tout à fait raisonnable de les connecter à Internet. »

Architecture IoT (5 niveaux)



Conséquences économiques / sociales

- Nouvelle « ruée vers l'or » pour les éditeurs de solutions propriétaires (du matériel vers le cloud)
  - Samsung ARTIK
  - AWS
  - Microsoft
  - IBM
  - Renesas
  - ThingWorx
  - MicroEJ
- A l'heure actuelle succès industriel mais pas grand public (le smartphone occupe la place !)
- Confidentialité / sécurité des données (en période de cyberattaques médiatisées !)
- Nouvelles opportunités pour le logiciel libre !

Smile ECS
Embedded & Connected Systems

Plate-formes libres

- Eclipse
- WSO2
- Kaa
- SiteWhere
- DeviceHive
- Zetta
- ThingSpeak
- Parse
- IoTDSA
- Quickstart IoT (Smile)
- Snootlab
- Etc.

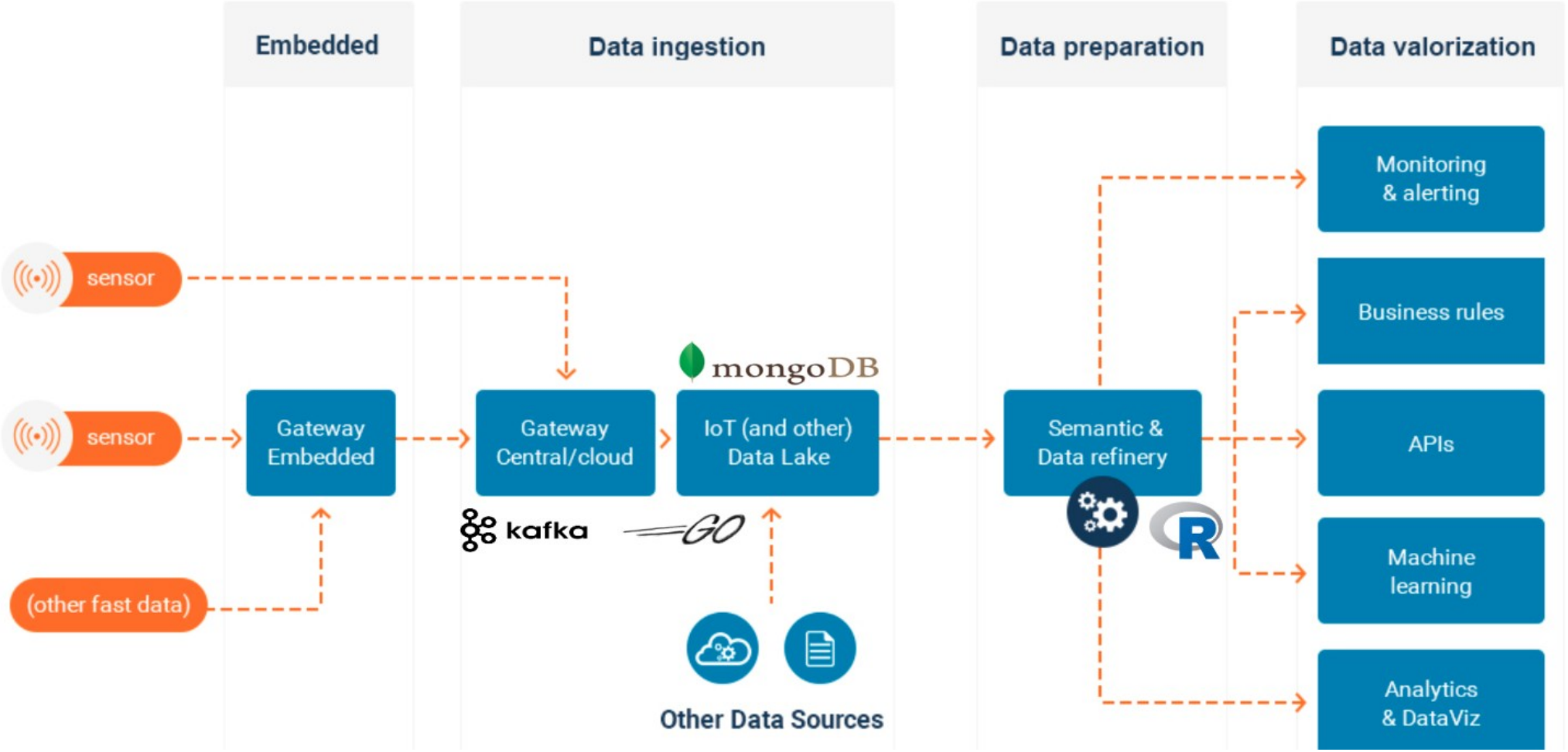
Smile ECS
Embedded & Connected Systems

Pourquoi Quickstart IoT ?

- Très simple à déployer (Quickstart!)
- Plug & play vis à vis des capteurs
- Ne nécessite pas de définition de structures de données ingérées à priori
- Fonctions de visualisation + monitoring génériques
- Modulaire pour s'adapter aux cas d'usages métiers
- Très utile pour un POC ou une expérimentation !

Smile ECS
Embedded & Connected Systems

Architecture Quickstart IoT



Smile ECS
Embedded & Connected Systems

Sécurité / confidentialité

- Explosion du nombre de cibles potentielles
- Faible niveau de sécurité des produits (ports ouverts, faible investissement R&D)
- Faible qualité de certains SDK (Android)
- Les acteurs de l'IoT ne sont pas des informaticiens
- Prise en compte de la sécurité au niveau de la conception !
- Adaptation à l'IoT des règles de sécurité habituelles
- Évolution vers RGPD en 2018 (Règlement Général Protection des données) → responsabilité des fabricants

Smile ECS
Embedded & Connected Systems

Bonnes pratiques en sécurité

- Outils de vérification automatique
  - Règles de codage
  - Test unitaire
  - Suivre les guides de bonne pratique
  - « Fuzzing » et autres outils spécialisés
- Les bonnes pratique système
  - Séparation des privilèges
  - Utilisation des fonctionnalités disponibles
  - Suivre les guide de bonne pratique
  - Toujours développer avec la dernière version des outils
- Certification à venir ? (mise à jour d'un logiciel drone vers DO-178)

Deux types d'objets

- Capteur basique
  - MCU/µC (pas de MMU)
  - Logiciel « bare metal » OS léger comme Contiki ou RIOT
- Capteur intelligent
  - CPU avec MMU (32 bits ou +)
  - OS de type Linux / Tizen / Android

Parrot flower power (µC)

"Tesla car is a connected computer on wheels !"

Eccelenza touch (Yocto)

Smile ECS
Embedded & Connected Systems

Protocoles réseau

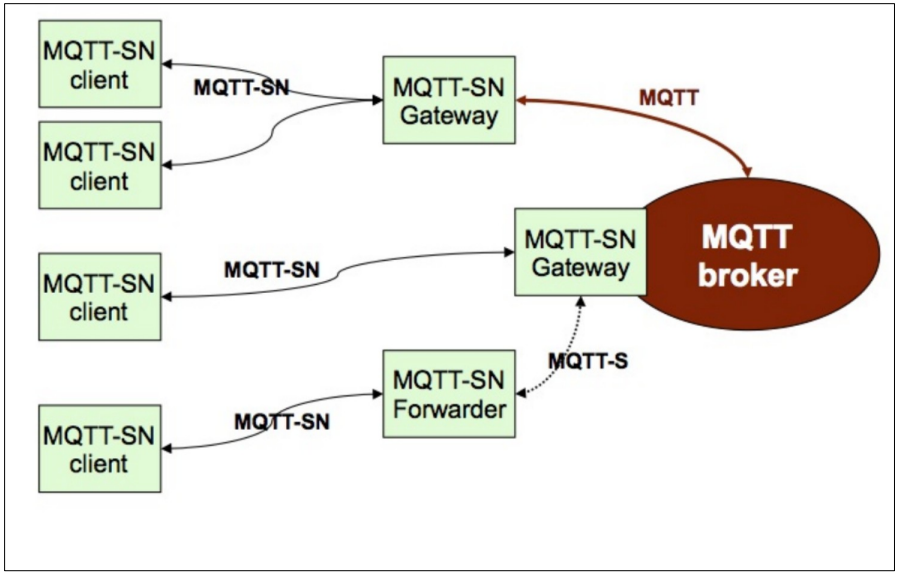
- IoT = embarqué + communication !
- Protocoles locaux générique (Wi-Fi, BLE, Zigbee)
- Protocoles locaux dédiés (6LoWPAN, Thread)
- Protocoles applicatifs (MQTT, CoAP)
- Protocoles ultra-narrow-band (LoRa / Sigfox) pour capteurs
  - LoRa est « ouvert »
  - Sigfox est propriétaire mais s'ouvre de plus en plus !
- Réseau libre, pas de licence (868 Mhz en France)
- Très basse consommation
- Faible bande passante (12 octets de données par trame Sigfox)
- Abonnement Sigfox à partir d' 1 € / an
- Matériel bon marché (test possible sur Raspberry Pi)

6LoWPAN

- Normet IETF (RFC-4944)
- Basé sur le 802.15.4
- IPv6 pour les objets connectés
- Compression des entêtes IPv6
- Trames courtes (128 octets)
- Supporté par Linux, Contiki, RIOT, FreeRTOS, etc.
- Une Adresse IPv6 pour chaque objet
- Routage de type « Mesh » possible

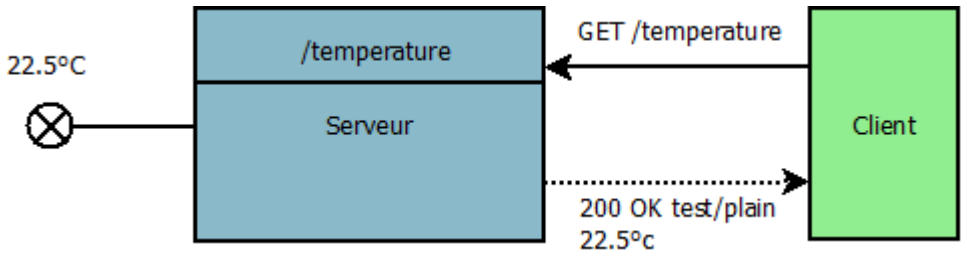
MQTT

- (M)essage (Q)ueue (T)elemetry (T)ransport
- Première version par IBM en 1999
- Architecture Publish / Subscribe
- MQTT-SN : version MQTT over UDP
- Piloté par la fondation Eclipse
- Implémentation libre = Mosquitto
- Serveur ouvert iot.eclipse.org, port 1883
- Sécurisation TLS/SSL



CoAP

- (Co)nstrained (A)pplication (P)rotocol
- HTTP sur UDP
- Compression de HTTP pour les protocoles contraints
- L'objet est serveur
- Le client consulte les informations
- Implémentation libre = libcoap (en C)
- Californium (Java - Eclipse)



Contiki

- Système d'exploitation développé par le Swedish Institute of Computer Science (SICS, 2002)
  - Open source, licence BSD
  - Ultra léger
  - Flexible
  - Plate-forme d'émulation et de simulation → Cooja
- Couche réseau uIP et uIPv6
- Optimisé pour la consommation
- Chargement dynamique de modules
- Bien adapté aux capteurs (quelque dizaines de Ko) → à partir des 8 bits (démo sur 8051, datant de 1980)
- Bonne documentation et nombreux exemples

RIOT

- Démarré en 2008 et maintenu par l'INRIA
- Licence GNU LGPL (et non GPL)
- Empreinte mémoire > 1 Ko
- Temps réel
- Micro-noyau
- POSIX
- Support C/C++ standard, programmation POSIX → réponse aux limitations de Contiki
- Réseau 6LoWPAN
- CPU 8 à 32 bits
- 80+ cartes supportées + émulation QEMU
- Présenté par ses concepteurs comme le « Linux de l'IOT »

Tizen

- Proche de Linux
- Popularisé par Samsung (TV, mobiles, etc.)
- Dispose de « profils » dont IoT
  - TV
  - Mobile
  - Micro-profile IoT
- https://fr.slideshare.net/badaindonesia/tizen-micro-profile-for-iot-device

Smile ECS
Embedded & Connected Systems

Android (Things)

- Dernier né de Google
- Remplaçant de l'infortuné Brillo
- Retour du « framework » Java Android (simplifié)
- Support matériel I²C, SPIO, GPIO
- Pour l'instant en « preview » sur 4 cibles dont la Pi 3
- Empreinte mémoire = 250 Mo …

Linux

- N'est pas l'OS universel pour l'IoT mais...
- Selon « IoT developer Survey 2016 »
  - 73 % Linux
  - 23 % « bare metal » (no OS)
  - 12 % FreeRTOS
  - 6 % Contiki
- Chez Linux il y a  et 
  - Distribution (Debian, Ubuntu, etc.)
  - « Build system » (Yocto, Buildroot, etc.)
- Aujourd'hui la plupart des objets sont des capteurs intelligents !

Distribution Linux

- La plupart des utilisateurs connaissent les distributions
  - Ubuntu, Debian, Fedora, etc.
- Environnement connu, simple à aborder, idéal pour débuter mais :
  - Empreinte mémoire importante
  - Temps de démarrage
  - Reste un environnement de développement avant tout
  - Faible traçabilité
  - Multi-plateforme limité (ARM, x86)
  - Bref, peu adapté à l'IoT
- Solution alternative → le « build system » !

Smile ECS
Embedded & Connected Systems

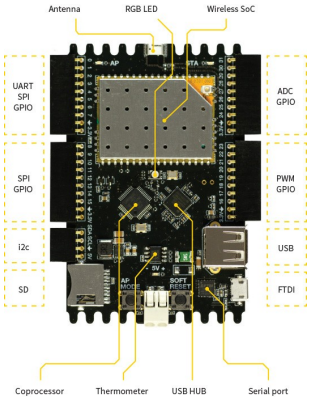
Qu'est-ce qu'un « build system » ?

- Construction d'une distribution à partir des sources
- Ne fournit pas les sources mais des « recettes »
- Produit les composants binaires à installer
  - Bootloader
  - Noyau Linux + DT
  - Root-filesystem image + applications
- Ajout d'informations
  - Licences
  - Graphes de dépendances, empreinte mémoire
- Meilleure empreinte mémoire, temps de démarrage
- Android utilise un « build system » spécifique

Smile ECS
Embedded & Connected Systems

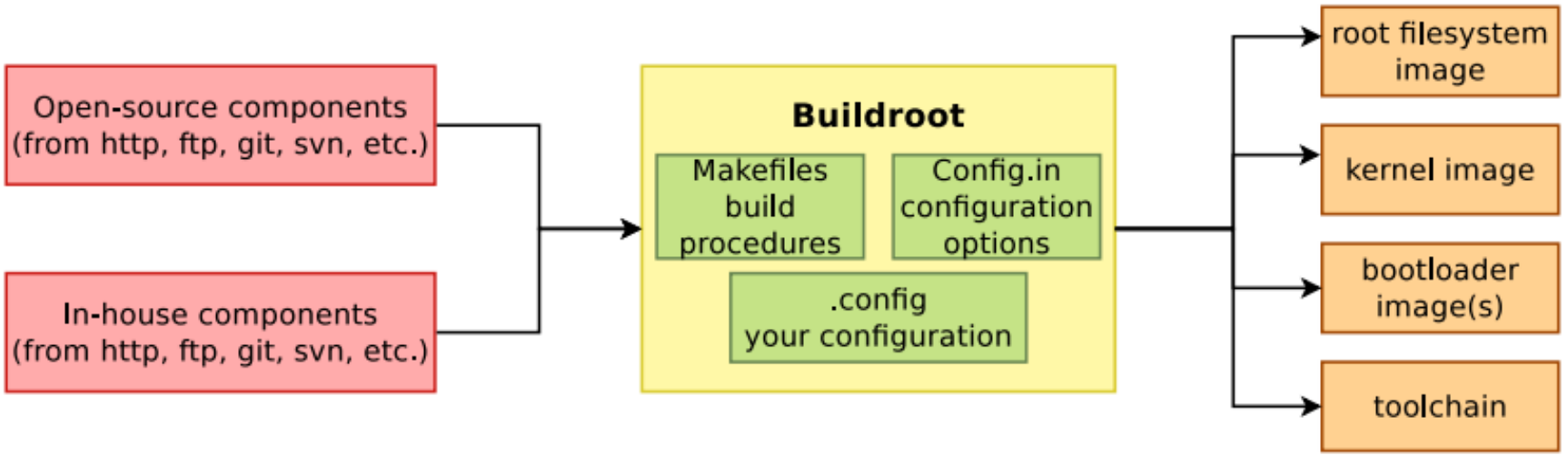
Les principaux outils disponibles

- Yocto/OpenEmbedded
  - Moteur écrit en Python
  - Très puissant mais lourd
  - Basé sur des fichiers de configuration
- Buildroot
  - Basé sur la commande « make »
  - Au départ un démonstrateur pour uClibc
- OpenWrt
  - Dérivé de BR
  - Gère les paquets binaires
  - Utilisé sur WeIO (IoT)



Buildroot

- Initialement un démonstrateur de uClibc (Micro-C-libC)
- Une version officielle tous les 3 mois depuis 2009.02
- Outil de configuration graphique identique à celui du noyau
- Léger, rapide, basé sur des fichiers `Makefile`
- Pas de système de paquets → « firmware Linux »
- Largement européen, voire français !

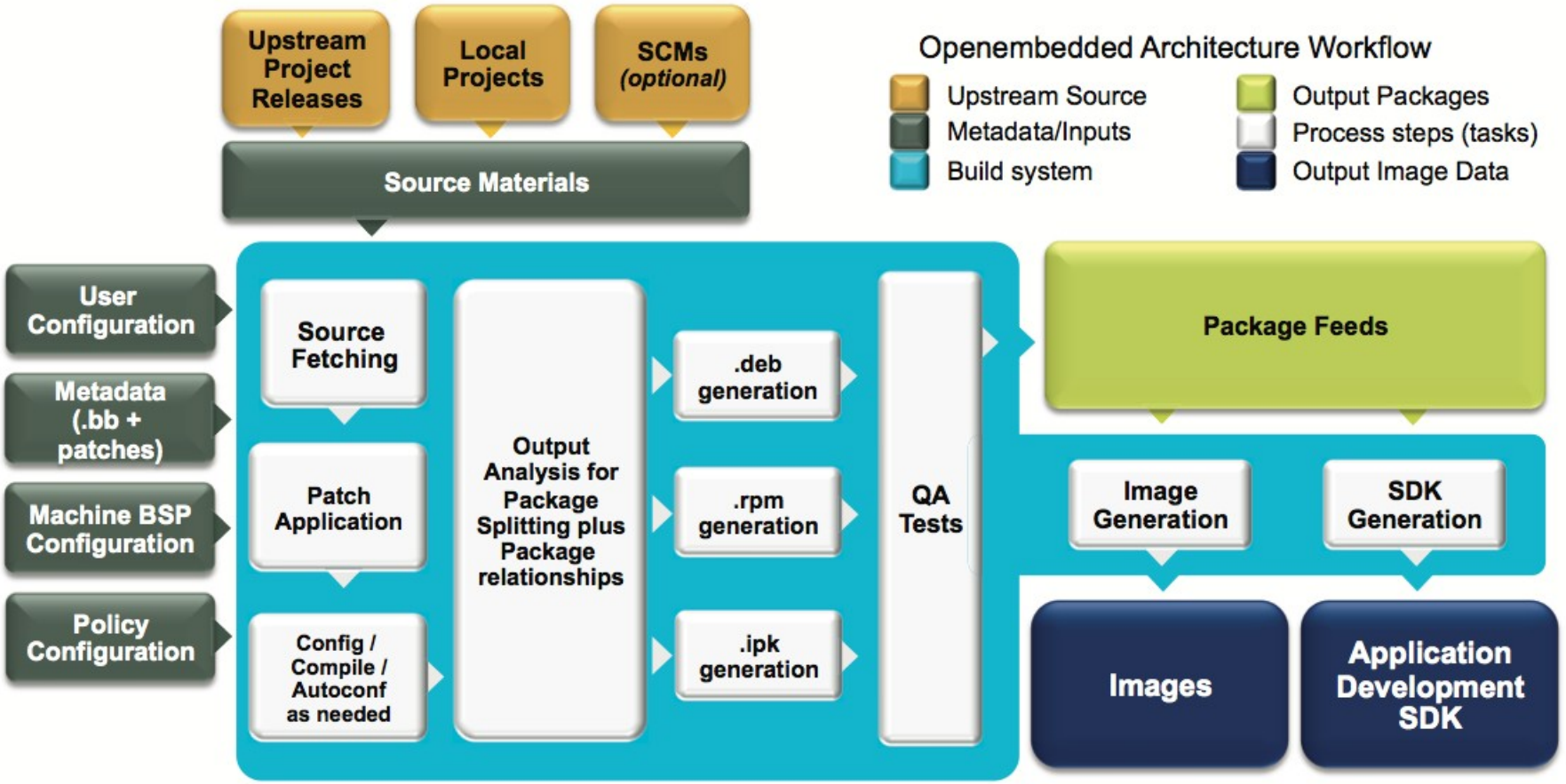


Yocto (OpenEmbedded)

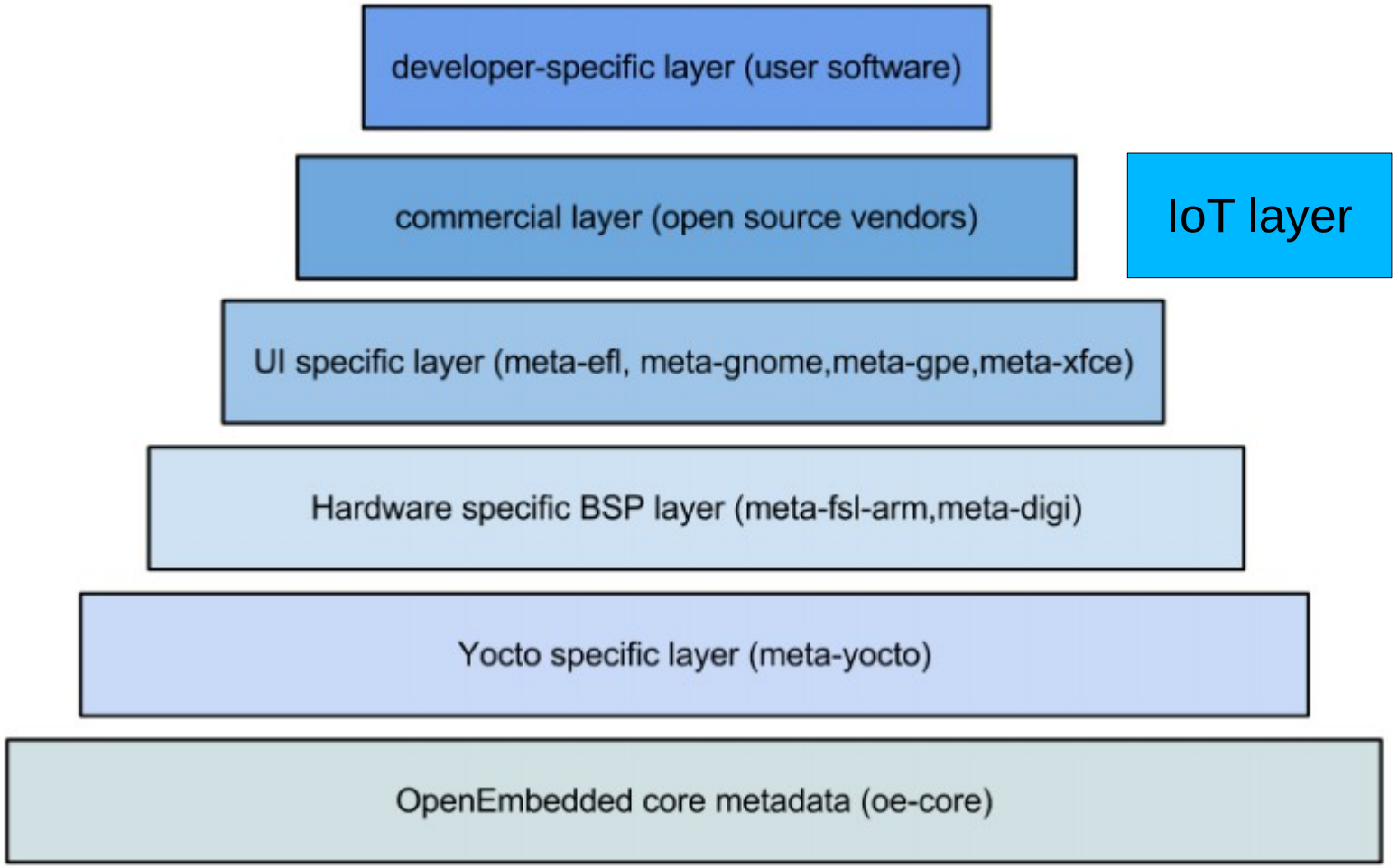
- Un « framework » de compilation croisée
- Démarré pae Chris Larson, Michael Lauer et Holger Schuring pour « OpenZaurus » (2002)
- Zaurus (SHARP) = premier PDA sous Linux/Qt
- Structure en couches (layers)

Smile ECS
Embedded & Connected Systems

Yocto / OE workflow

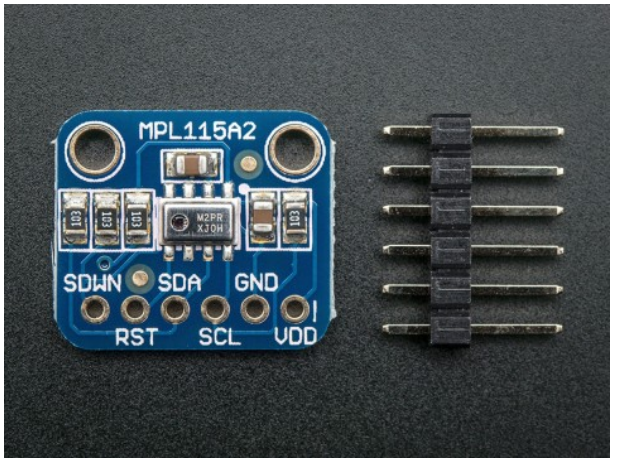


Yocto / OE layers

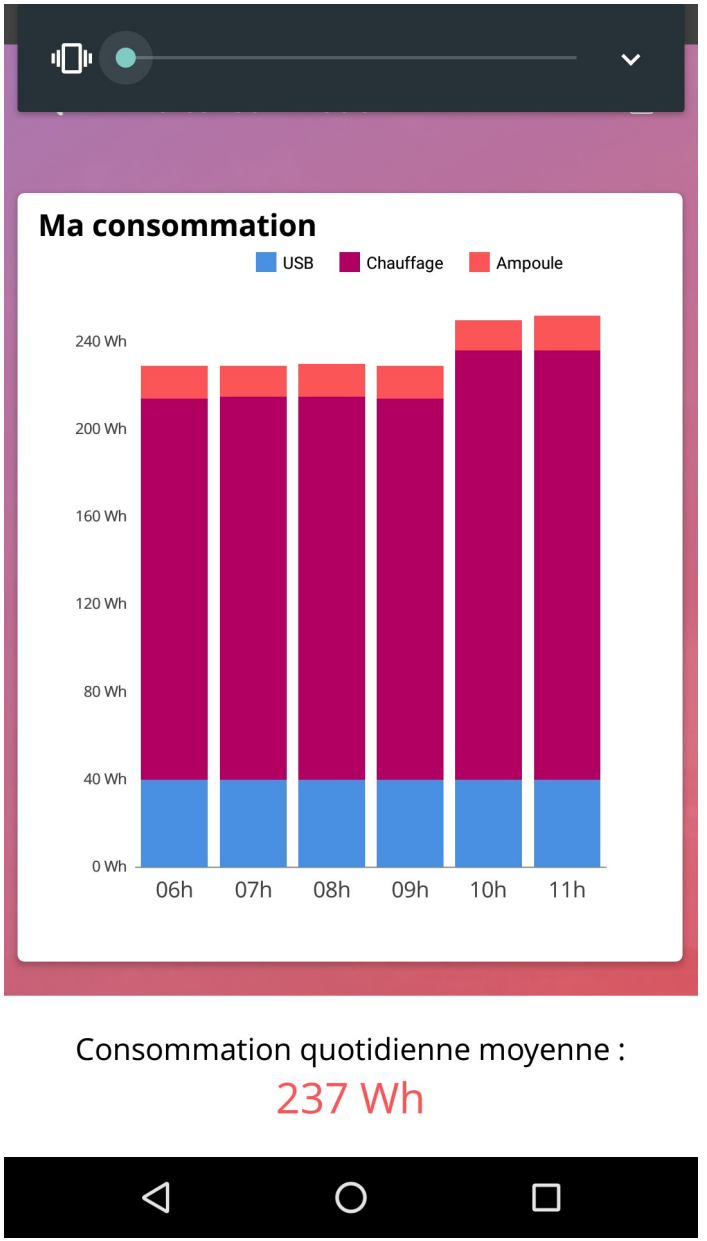


Use case 1 = capteur IoT Yocto

- Démonstrateur pour Smile (Quickstart IoT)
  - Raspberry Pi (zero Wi-Fi)
  - Capteur I²C température/pression (MPL115A2)
  - Protocole MQTT
  - Yocto
  - Affichage sur application mobile Android
  - Voir l'atelier cet après-midi !

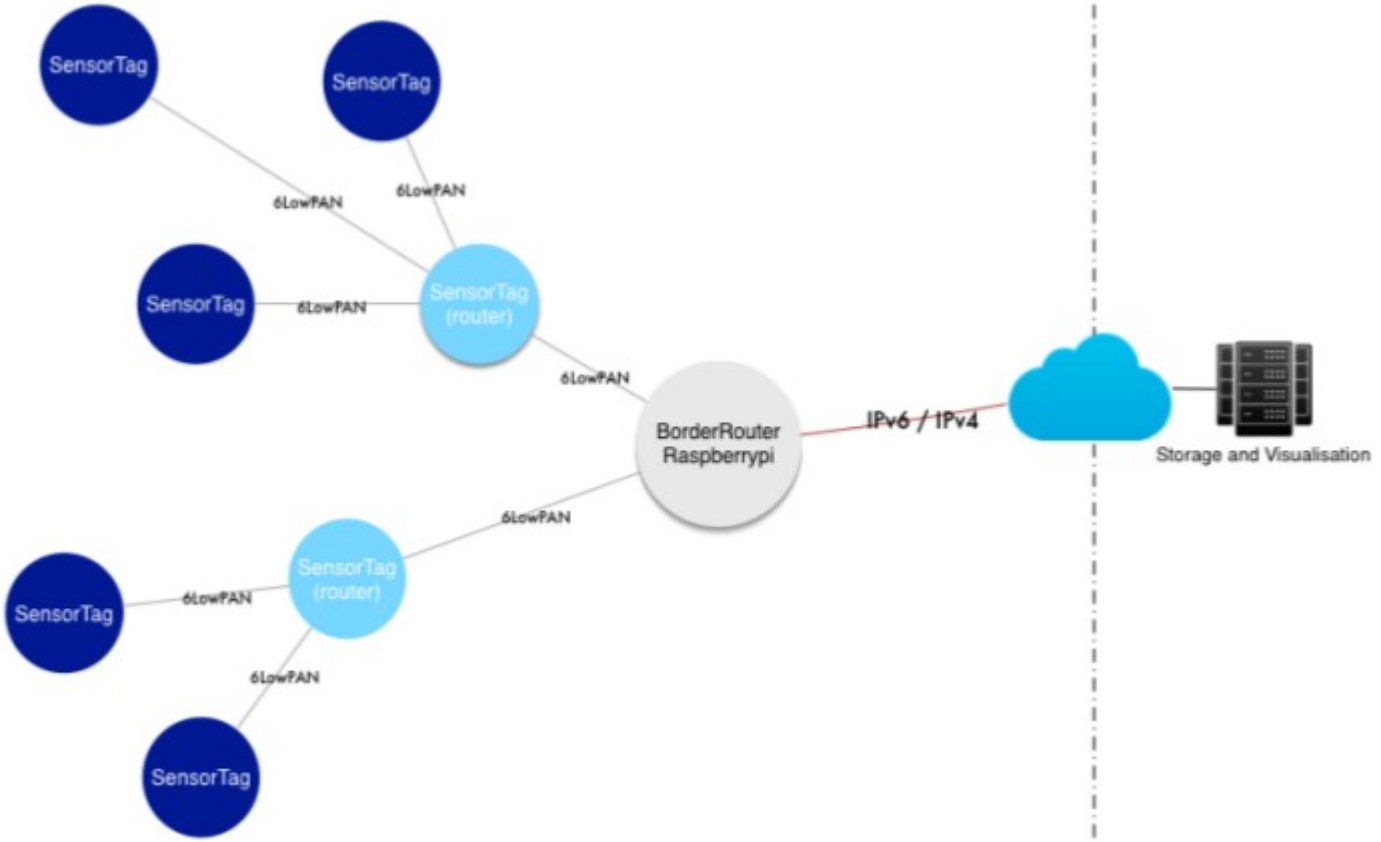


Application Android (NeoPixl)



Use case 2 = Border router

- Démonstration plus complète basée sur SensorTag (TI)
- Raspberry Pi (Yocto 2.1) comme « border router »

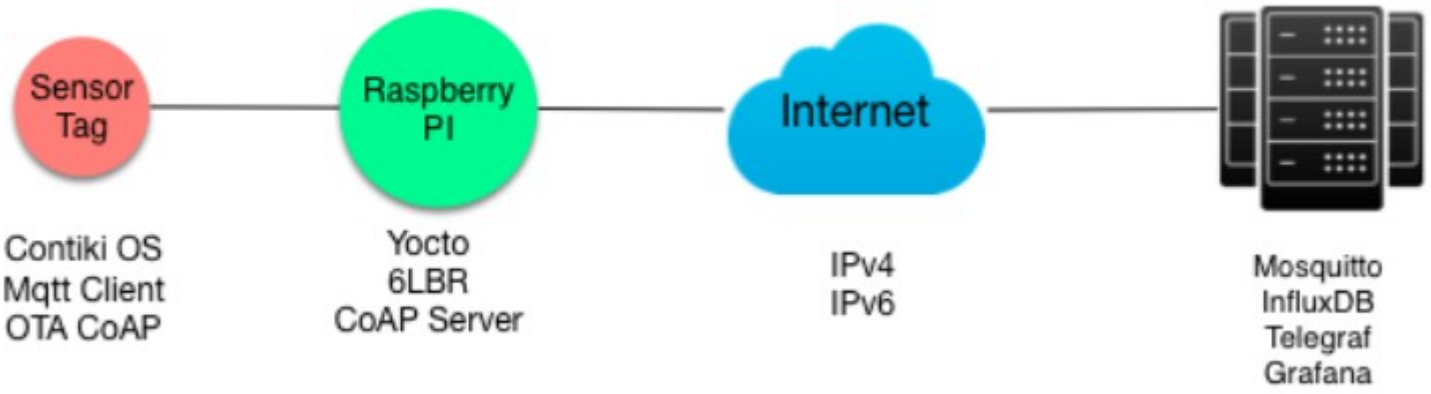


SensorTag

- Cortex M3 (48MHz, 128KB flash, 8KB RAM)
- 512KB external flash, OTA et/ou stockage
- Faible consommation (10 mA active, 100 uA veille)
- Radio 802.15.4 + Bluetooth Low Energy (BLE)
- $ 30 sur site TI

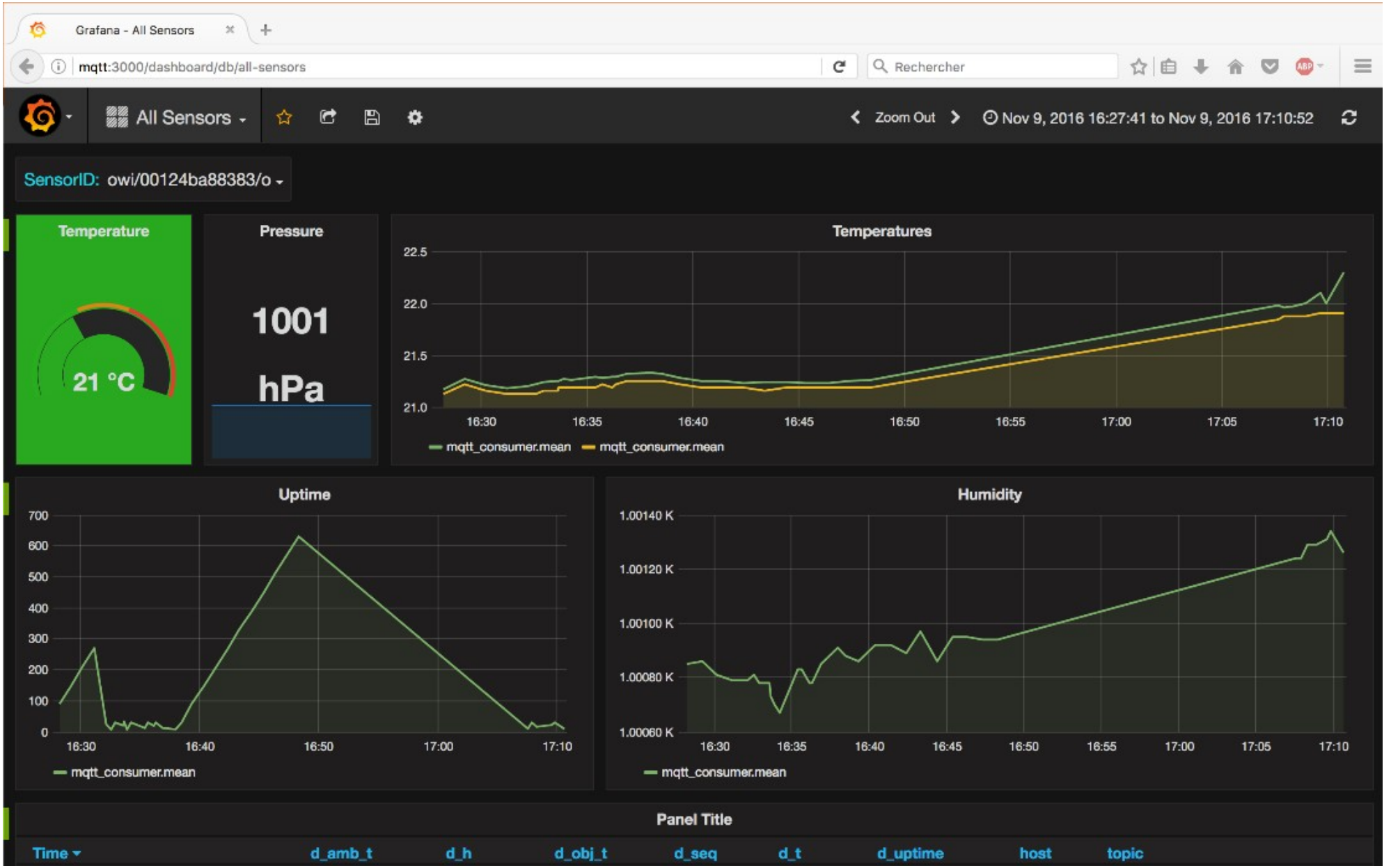
Raspberry Pi + 6LBR

- 6LBR = border router (lien entre capteur et Internet)
- Obtention des données auprès des modules SensorTag (6LoWPAN)
- Envoi des données au « cloud »
- Broker MQTT
- Base de données Influxdb
- Affichage web (Grafana)



Smile ECS
Embedded & Connected Systems

Grafana display



Smile ECS
Embedded & Connected Systems

Merci !