Table des matières

1 - Support d'étude

Basé sur le système EWTS, le nouveau système EWTS-CORONIS a été conçu autour d'exemples des dernières technologies en matière de télémétrie pour montrer les principes de base et les concepts associés de la mesure d'énergie :

- Capteurs intelligents
- Technologies réseau et transmetteurs sans fil
- Intelligence distribuée et architecture de stockage
- Système de télémétrie basé sur micro-serveur web embarqué
- Gestion de base de données
- Développement C/C++
- SysML/UML

2 - Avancement actuel des travaux

Rappel des équipements et codes sources fournis par la société Technext :

• Un capteur de température sans fil Wavetherm de la société Coronis	
• Un capteur/actionneur tout ou rien sans fil Wavelog de la société Coronis	
• Une passerelle USB/Wavenis Waveport de la société Coronis	
• Ensemble de classes C++ pour la supervision d'un réseau de capteurs de la famille Wavenis (sans la couche de communication USB/RS232)	

Plusieurs phases ont été nécessaires pour la mise en œuvre du matériel fournit et le développement d'une première application :

- Appropriation des documentations techniques des modules Wavetherm, Wavelog et Waveport ;
- Appropriation du protocole Wavenis ;
- Appropriation des codes sources et des classes métiers fournis par la société Technext ;
- Conception de l'architecture logicielle embarquée :
  - Choix de la nouvelle couche de communication sur le réseau Wavenis ;
  - Conception du module d'acquisition ;
  - Choix du framework de persistance et supervision Web (Backoffice et Frontoffice) pour la partie Télémétrie Web ;
  - Choix d'une carte embarquée pour les tests ;
- Réalisations et tests :
  - Réalisation et test de la nouvelle couche de communication sur le réseau Wavenis ;
  - Intégration de la nouvelle couche de communication aux classes métiers existantes pour réaliser le module d'acquisition ;
  - Réalisation de la couche de persistance, du Backoffice et Frontoffice Web pour un modèle de capteur pour valider le prototype (capteur choisi : Wavetherm) ;
  - Test du module d'acquisition sur une plateforme Linux embarqué ;
  - Test du module de Télémétrie Web déportée sur un serveur web sur Internet ;
  - Intégration du module de Télémétrie Web sur la carte embarquée (reste à faire).

Fig. 1: La plateforme Linux embarqué Gumstix utilisée pour le test

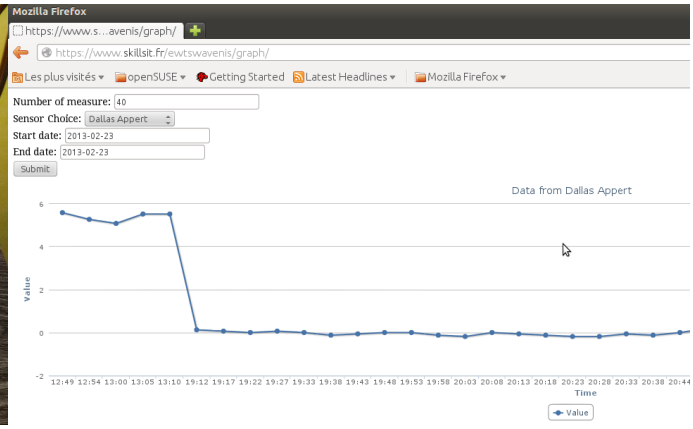


Fig. 2: Vue du système de Télémétrie Web (FrontOffice)

3 - Architecture mise en place pour l'étude

L'architecture mise en place pour l'étude se base sur une configuration où l'on retrouve les capteurs à tester, une plateforme Linux embarqué et une infrastructure réseau filaires/non-filaires classique.

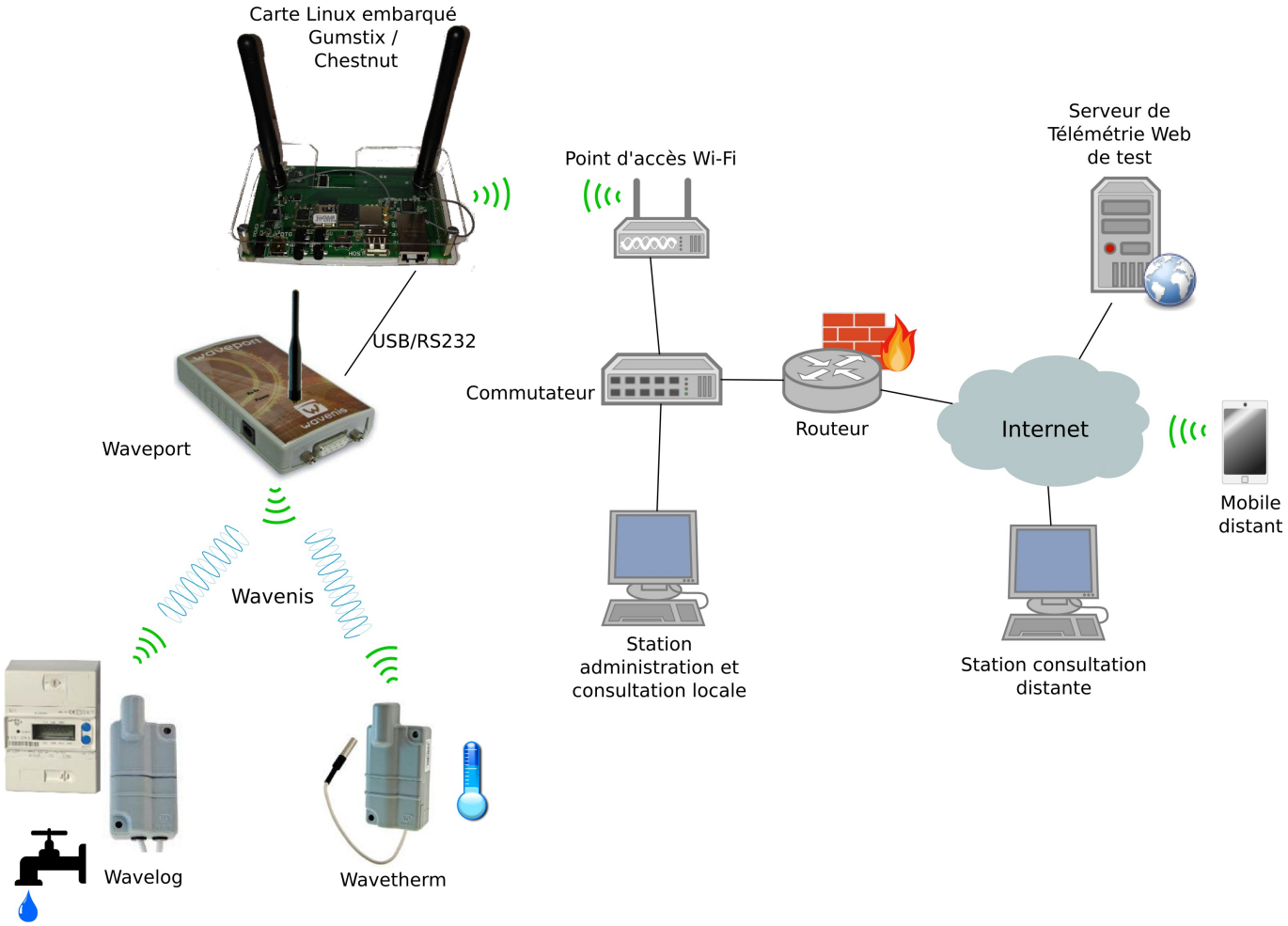


Fig. 3: Architecture visée

4 - Caractéristiques des composants

4.1 - Capteur de température Wavetherm

4.1.1 - Caractéristiques

Capteur de température	• Dallas 18S20 (modèle PT100/PT1000 existant) • Possibilité de deux capteurs par Wavetherm
Plage	• -55 / +125° C
Précision	• ±0.5° C
Résolution	• ±0.1° C
Fonctionnalités générales	• Autonomie de plusieurs années • Interface sans fil • Data logging programmable • Transmission de données sur requête immédiate ou retardée (48 mesures) • Transmission d'alertes en cas de batterie faible, câble coupé ou conditions de seuil
Caractéristiques sans fil	• Protocole sans fil Wavenis • Portée radio jusqu'à 1km en extérieur et 200m en intérieur • Résistance aux interférences par des mécanismes sophistiqué (FHSS, BCH, etc.) • Réseau maillé sans limite de taille • Lien bi-directionnel entre le réseau et le point de contrôle du réseau • Point-à-point, point-à- multipoint (diffusion, polling), mode répéteur • Topologies réseau arbre, étoile et maillée. • Bande de fréquence libre ISM 433/868/915MHz
Installation	• Facile

4.1.2 - Mise en œuvre

- Matériel testé et conforme aux attentes pour les activités envisagées.

4.2 - Monitor sans fil Wavelog

4.2.1 - Caractéristiques

Le Wavelog est un équipement sans fil bi-directionnel permettant de monitorer des capteurs tout-ou-rien et de contrôler des actionneurs comme des valves, moteurs et alarme d'intrusion.

Entrées/Sorties	• 4 entrées (contact sec), 4 sorties • Impulsions de sortie programmable
Fonctionnalités générales	• Autonomie de plusieurs années • Interface sans fil • Transmission programmable • Stockage jusqu'à 500 lectures • Transmission d'alertes en cas de batterie faible ou de câble coupé
Caractéristiques sans fil	• Protocole sans fil Wavenis • Portée radio jusqu'à 1km en extérieur et 200m en intérieur • Résistance aux interférences par des mécanismes sophistiqué (FHSS, BCH, etc.) • Réseau maillé sans limite de taille • Lien bi-directionnel entre le réseau et le point de contrôle du réseau • Point-à-point, point-à- multipoint (diffusion, polling), mode répéteur • Topologies réseau arbre, étoile et maillée.
Installation	• Facile

4.2.2 - Mise en œuvre

- Matériel pas encore testé.

4.3 - Module Waveport

4.3.1 - Caractéristiques

Point d'accès au réseau Wavenis, Waveport permet la lecture, le contrôle et la configuration des capteurs sans fil basés sur la technologie Wavenis.

Caractéristiques générales	• Plateforme Wavenis et pile de protocole intégrée • Classe 25mW ou 500 mW (longue portée - sur port série uniquement) • Interfaces possibles ◦ USB ◦ Série (modèle testé) ◦ CF II ◦ Bluetooth

4.3.2 - Mise en œuvre

- Matériel testé et conforme aux attentes pour les activités envisagées.

4.4 - Carte Gumstix et carte d'extension Chestnut

4.4.1 - Caractéristiques matérielles

La série Linux Overo® computer-on-module est disponible dans plusieurs configurations afin accélérer le prototypage et la mise en production.

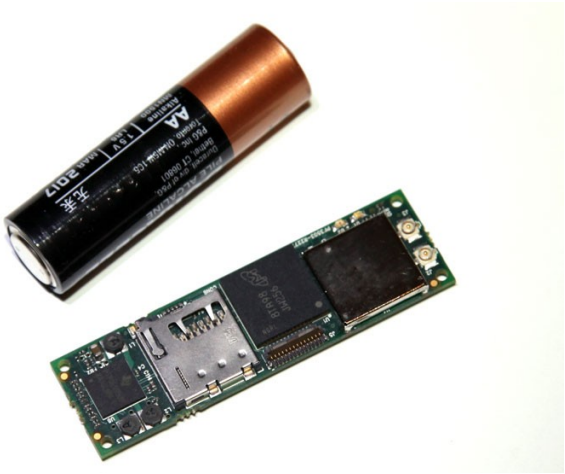


Fig. 4: Carte Linux embarqué Gumstix Overo FireSTORM COM

Fig. 5: Carte d'extension pour Gumstix Chestnut

Caractéristiques générales	• Architecture : ARM Cortex-A8 • Processeur : Texas Instruments OMAP3730 application processor • Fréquence processeur : jusqu'à 1 GHz • DSP : C64x Fixed Point DSP 660,800 Mhz (Max.) • RAM/NAND : 512Mo / 512Mo • Graphique : OpenGL POWER SGX™ for 2D and 3D graphics acceleration • Emplacement pour carte SD (bootloader et OS)
Connectivité	• Bluetooth • Wi-Fi 802.11 b/g • Connecteur 27-pin pour signaux camera • Ethernet sur carte d'extension • USB HOST/OTG/Console sur carte d'extension • Connecteur LCD sur carte d'extension
Site constructeur	• https://www.gumstix.com • Carte Gumstix : https://www.gumstix.com/store/product_info.php?products_id=267 • Carte d'extension : https://www.gumstix.com/store/product_info.php?products_id=237
Licence	• Open Source : ○ Ensemble des schémas disponibles

4.4.2 - Caractéristiques logicielles

4.4.2.1 - Yocto

- https://www.yoctoproject.org/
- Le projet Yocto est un projet collaboratif open-source qui fournit les modèles, les outils et les méthodes afin d'aider les développeurs à créer des systèmes Linux personnalisés quelque soit l'architecture matérielle choisie. Le projet Yocto s'appuie sur OpenEmbbeded
- Plugin pour Eclipse

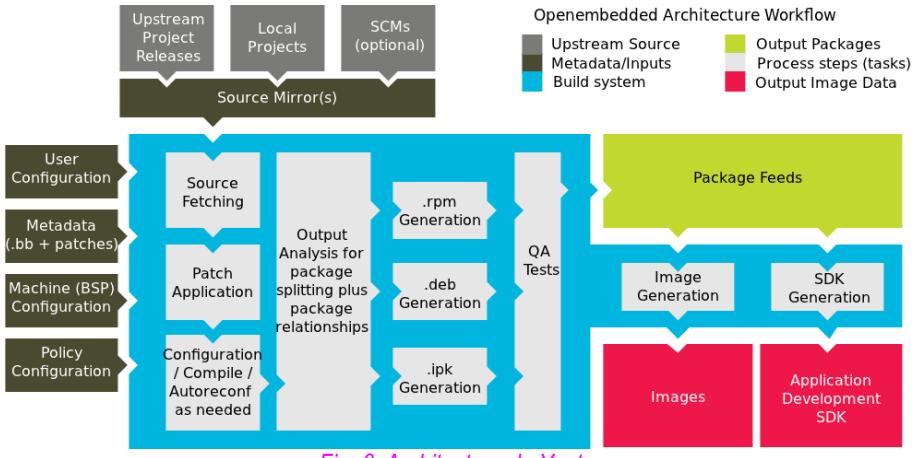


Fig. 6: Architecture de Yocto

4.4.2.2 - Caractéristiques du système Linux embarqué

- Bootloader : UBoot
- Noyau Linux 3.5.0 et système de fichiers sur carte SD jusqu'à 64 Mo
- Accès console / ssh
- Communications
  - Ethernet / Wi-Fi / Bluetooth
  - USB (HOST/OTG)
- Gestionnaire de paquets embarqués
- Agent de débogage distant pour Eclipse

4.4.3 - Mise en œuvre

- Suite logicielle Yocot et système Linux embarqué testés et conforme aux attentes pour les activités envisagées.
- Tutoriels : https://www.skillsit.fr/mediawiki/index.php/Yoctogumstix060113

4.5 - Matériel complémentaire

- Hub USB 4 port alimenté pour recevoir l'adaptateur USB/Sérial connecté au Waveport.

5 - Propositions pour le développement logiciel de la partie embarquée

Le diagramme de déploiement plus loin décrit l'architecture actuelle de la réalisation.

5.1 - Sous-Système Acquisition

Nous avons choisi de découpler le module d'Acquisition de la partie persistance et Télémétrie Web afin de faciliter la mise au point et d'utiliser des techniques de communication entre les processus modernes et basées sur des standards :

- Protocole HTTP basé sur des requêtes POST pour la transmission entre le module d'acquisition et la persistance (voir Protocole REST)
- Cela permet de placer la partie persistance et Web du système sur une plateforme de test sur Internet, plus confortable et utile en phase de développement distribué entre plusieurs développeurs.

Le module d'Acquisition est écrit en C++ et intègre la couche de communication OpenSource (C / C++/ Java) fournit par Coronis depuis Décembre 2012.

5.2 - Sous-Système de Télémétrie Web

Pour le sous-système de Télémétrie Web et la persistance (Postgresql, MySQL ou Sqlite), nous avons choisi le framework Django écrit en Python.

Il est chargé :

- de la réception et du contrôle des données en provenance du module d'Acquisition via le protocole HTTP
- de la gestion du BackOffice (contrôle et agrégation des mesures, ajout de capteur, etc.)
- de la gestion du FrontOffice (tracé des courbes, etc.)

L'intérêt de ce framework réside dans sa forte présence dans le monde du Web et de ses caractéristiques

- OpenSource ;
- Intégration de la persistance ;
- Intégration du BackOffice ;
- De nombreux développeurs Django/Python qui proposent de multiples extensions ;
- Langage Objet Python ;
- Développement rapide et adapté à l'embarqué ;
- Bonne intégration aux serveurs HTTP pour l'embarqué (Ngnix et Lighttpd) ;
- Bonne performance en embarqué sur carte Gumstix

La partie FrontOffice est basée sur des technologies standards et OpenSource (Javascript, HTML5, JQuery, Highchart, etc.)

Cette architecture est opérationnelle sur un serveur de test sur Internet et sera intégrée sur la carte Gumstix dans les prochains sprints.

Le serveur HTTP embarqué sera choisi entre Lighttpd et Ngnix. La plateforme Web de test actuelle est basée sur Apache.

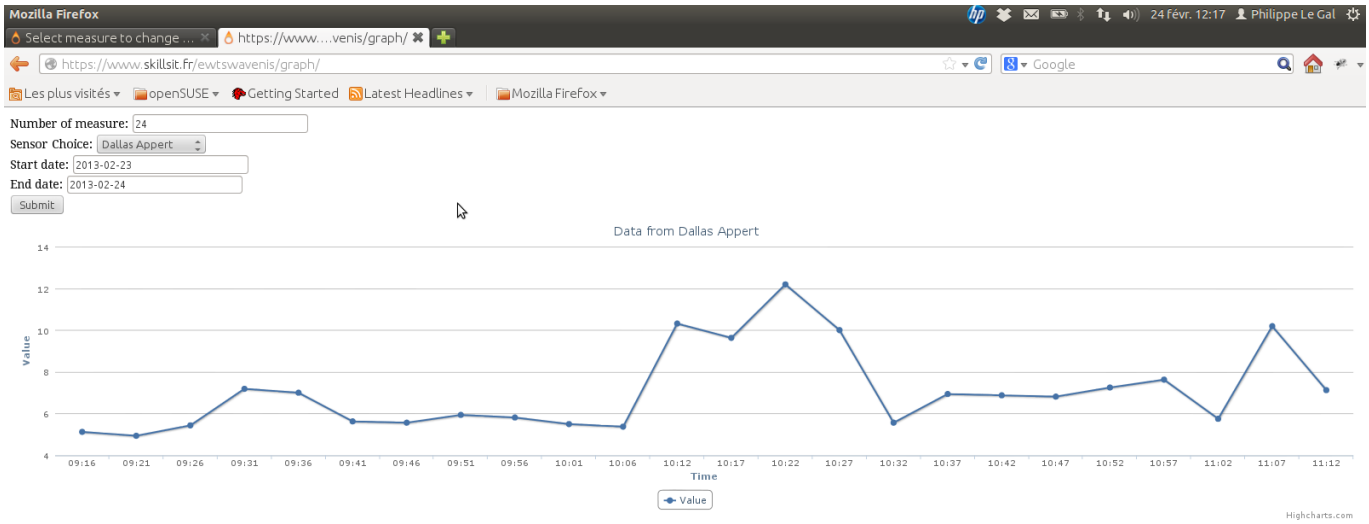


Fig. 7: Vue FrontOffice du système de Télémétrie

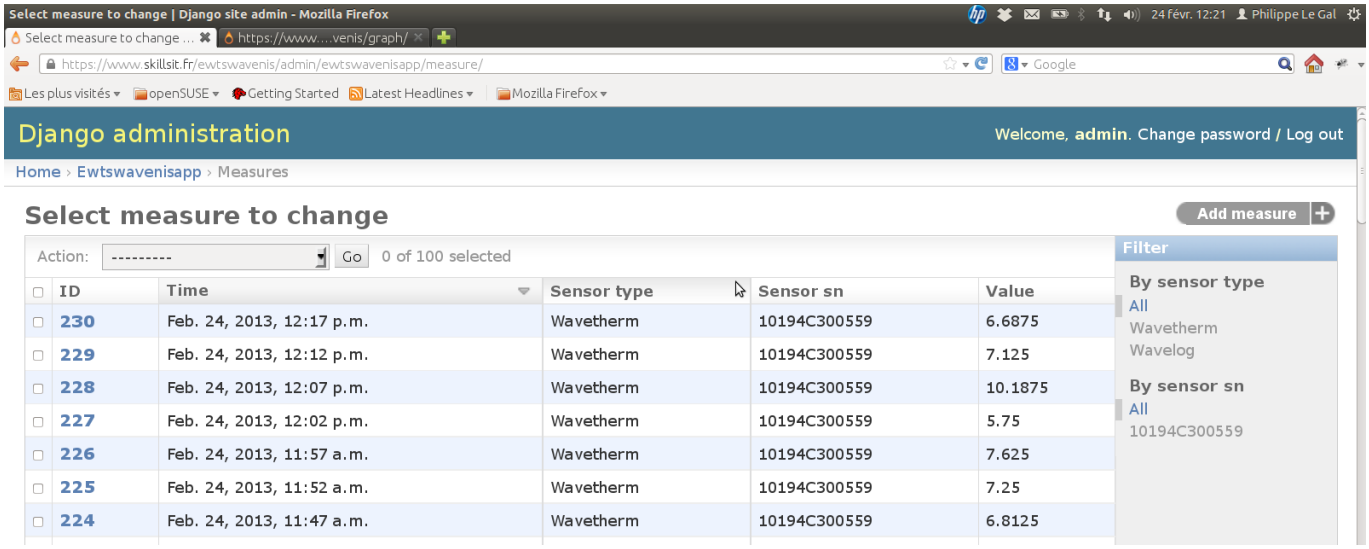


Fig. 8: Vue BackOffice du système de Télémétrie : Consultation des mesures

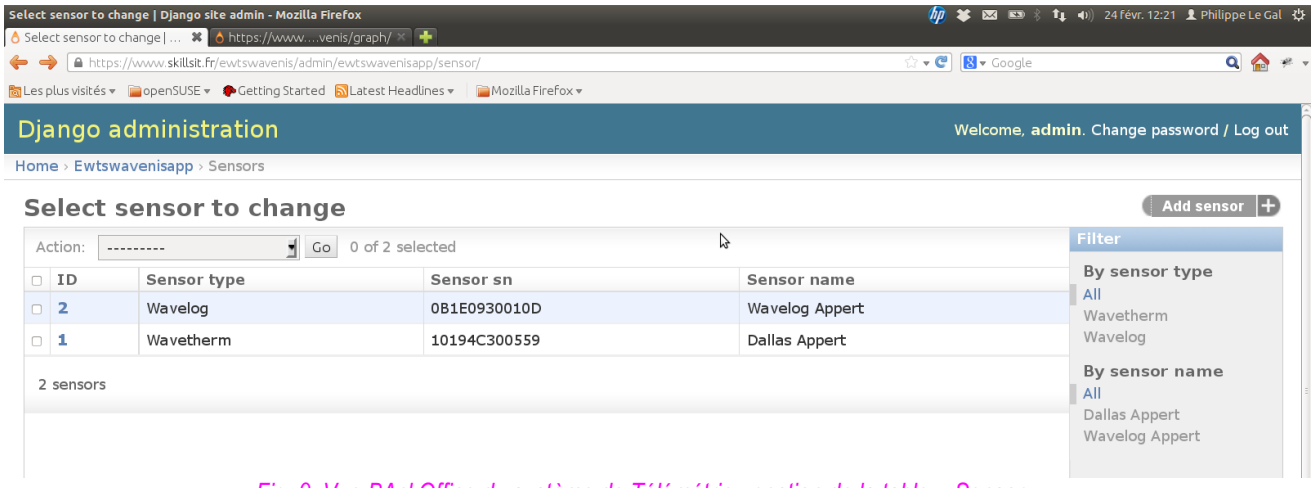


Fig. 9: Vue BAckOffice du système de Télémétrie : gestion de la table « Sensor »

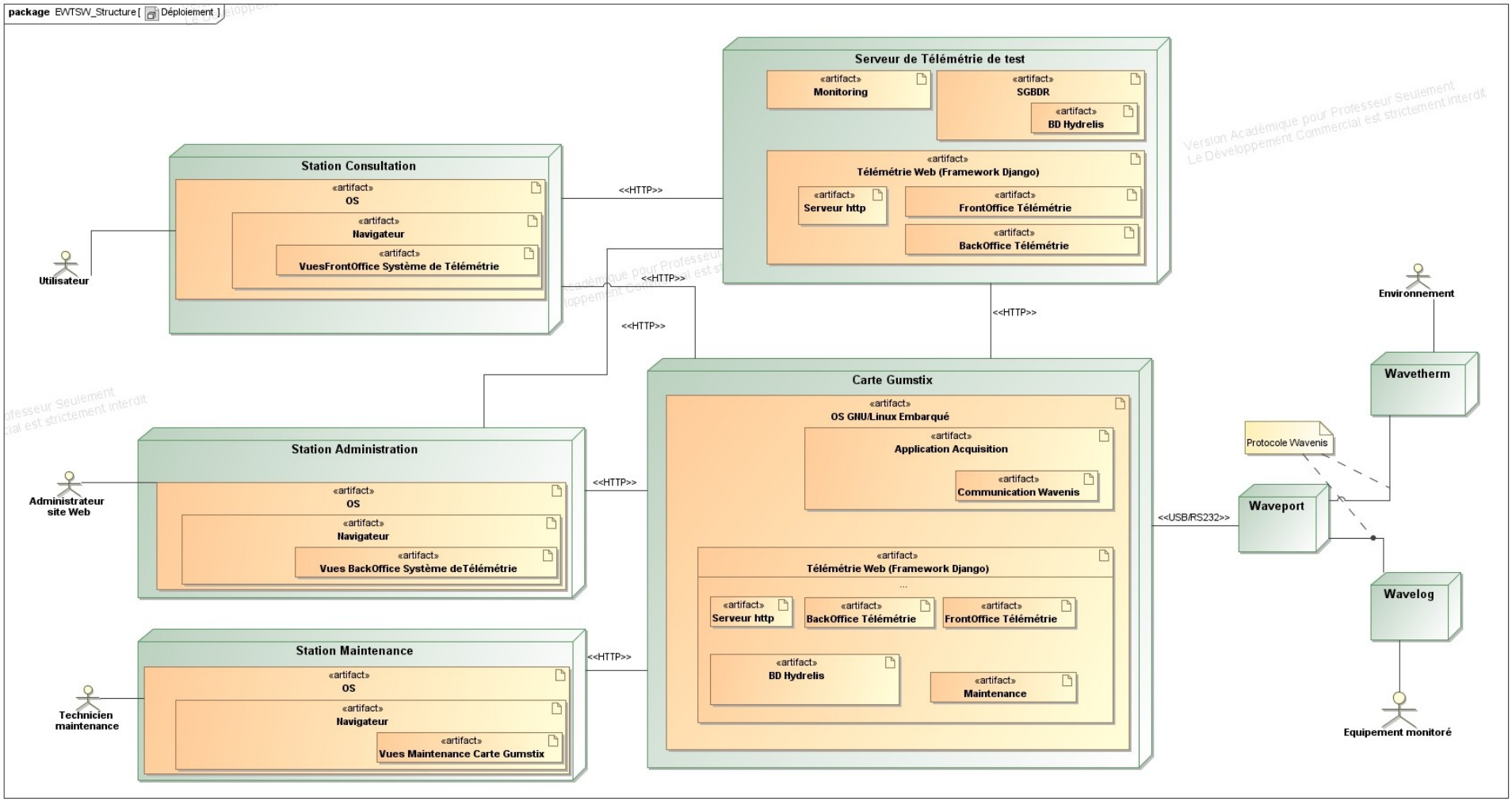


Fig. 10: Diagramme de déploiement

6 - Réalisation actuelle

6.1 - Réalisation matérielle

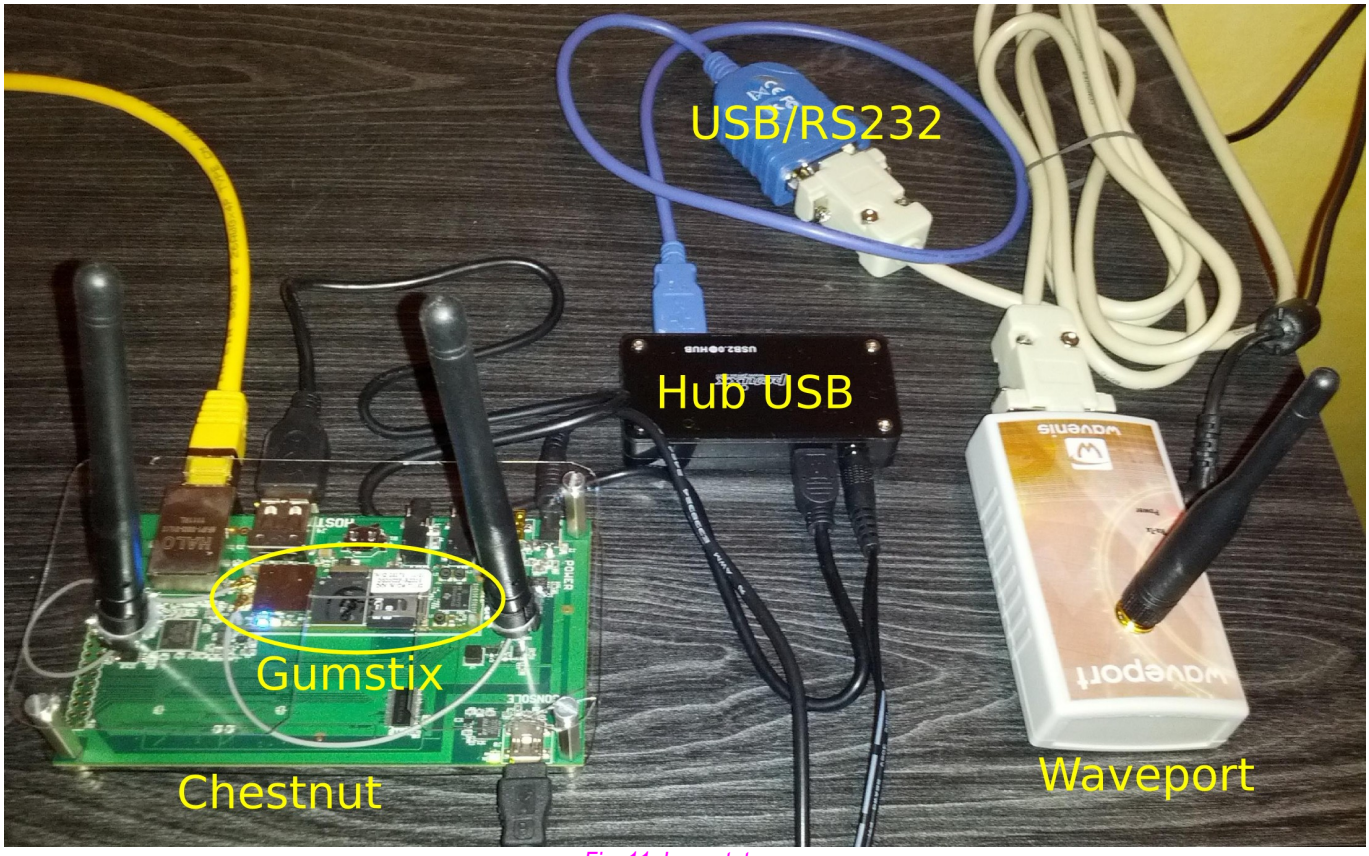


Fig. 11: Le prototype

Capture d'écran de l'application embarquée :

```
 -> Online
 -> Online
job1...
stm::process_measure
value call
CmdParamCtrl
CmdRadioMd :  1
CmdRadio
[17:32:04:701] process send request status: OK_PROTOCOL_SUCCESS
[17:32:04:704] process send request response: 10194C300559;810A80004D4FFF
temperature brute :   4d
temperature : 4.812500
2013-02-24%2017:32:4
Response returned with status code 201
 -> Online
 -> Online
```

La carte Gumstix envoie en permanence (toutes les 5mn) une donnée de température à la plateforme Télémétrie Web. La courbe de température est visible à l'adresse suivante :

- https://www.skillsit.fr/ewtswavenis/graph/

6.2 - Diagramme de classe du sous-système d'Acquisition

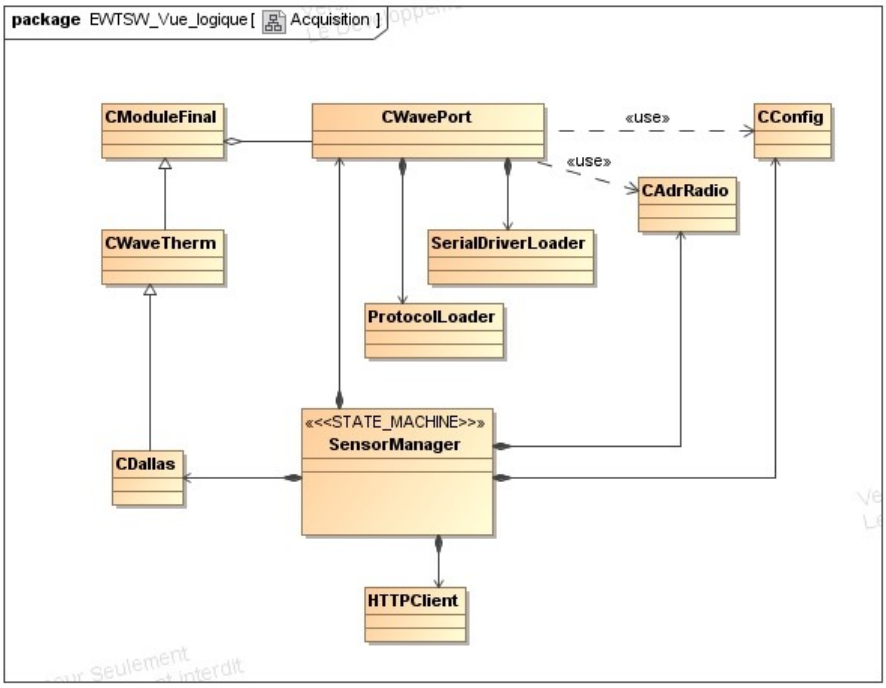


Fig. 12: Diagramme de classe du module d'Acquisition

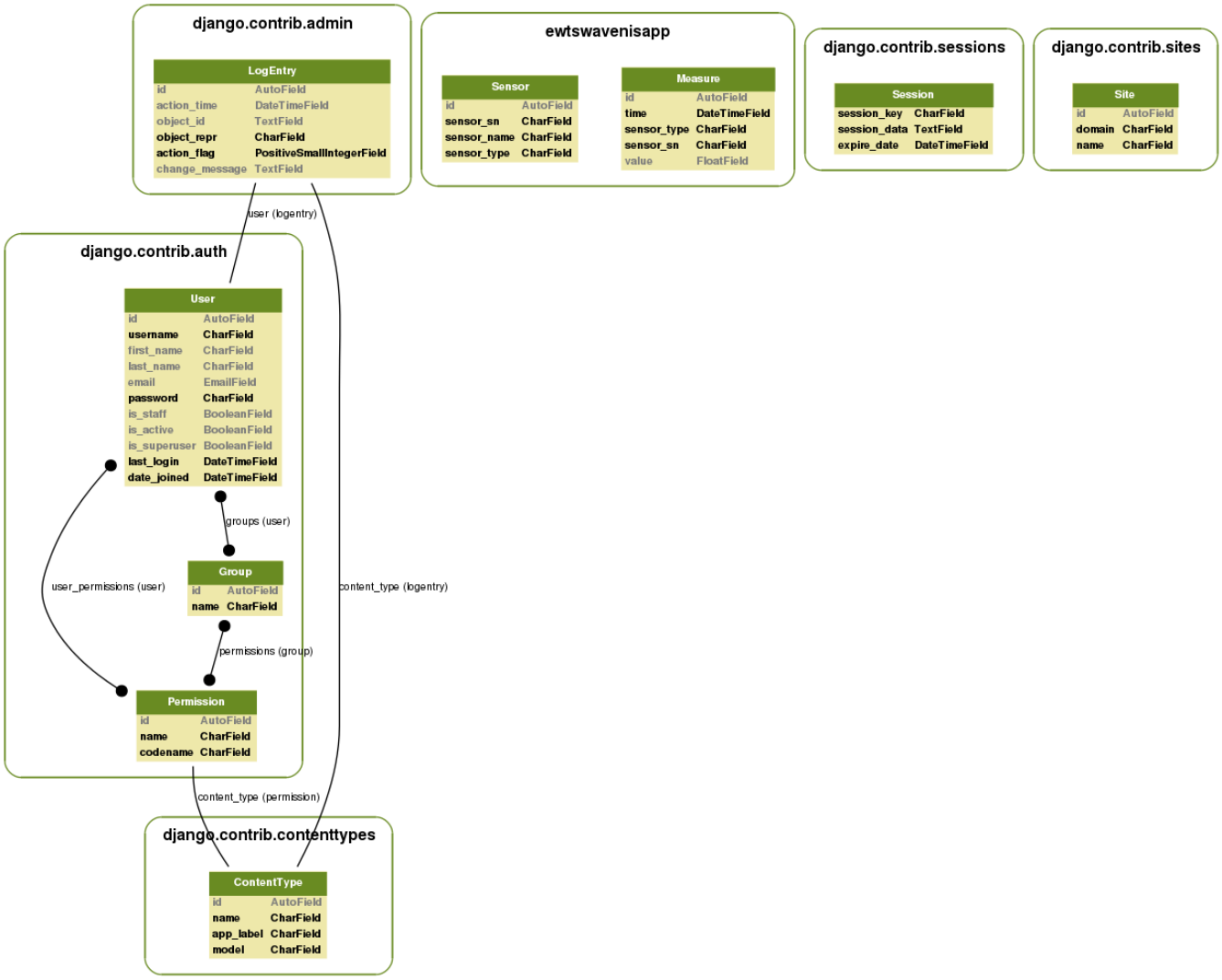
La classe CWavePort a été adaptée pour pouvoir utiliser la couche de communication sur le réseau Wavenis fournie par Coronis (http://sourceforge.net/p/coronissdk/wiki/Home/). La couche de communication est livrée sous la forme de code source C/C++ et fichier de construction (Eclipse et Makefile) permettant de générer trois bibliothèques chargées dynamiquement au lancement de l'application.

La classe SensorManager est la classe principale de l'application et est basée sur la machine à état MSM de la bibliothèque C++ boost (http://www.boost.org/doc/libs/1_49_0/libs/msm/doc/HTML/index.html).

La classe HTTPClient permet le dialogue avec le sous-système Télémétrie Web et persistance réalisé par le framework Django. La classe HTTPClient est basée sur la bibliothèque asynchrone asio de boost.

Le capteur de température Wavetherm est représenté par la classe CDallas. Pour l'instant seule la méthode de lecture de la température a été testé avec succès.

6.3 - Base de données du sous-système de Télémétrie Web



Pour l'instant le diagramme entité/relation représentant les tables de l'application de Télémétrie est encore incomplet.

Les tables métiers configurées sont :

- La table Sensor listant les capteurs d'une installation ;
- La table Measure dans laquelle sont stockées les mesures en provenance du module d'Acquisition ;
- Les autres tables sont générées et gérées paar Django.

Pour l'instant la persistance est gérée par un serveur Postgresql. Les autres choix possibles sont :

- Mysql
- Sqlite3
- Oracle

Sqlite3 et Mysql seront testées sur la plateforme embarquée.

6.4 - Diagramme de classe du sous-système de Télémétrie Web

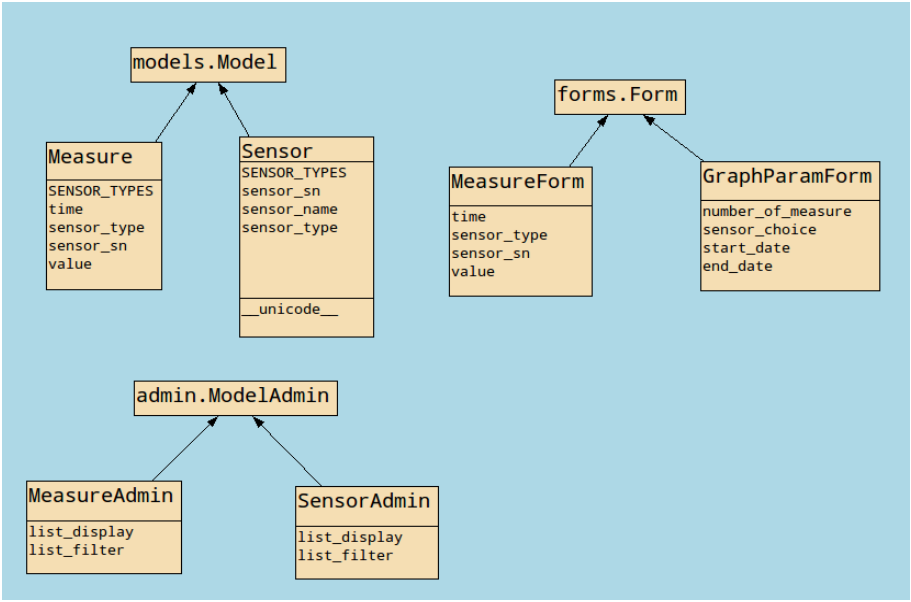
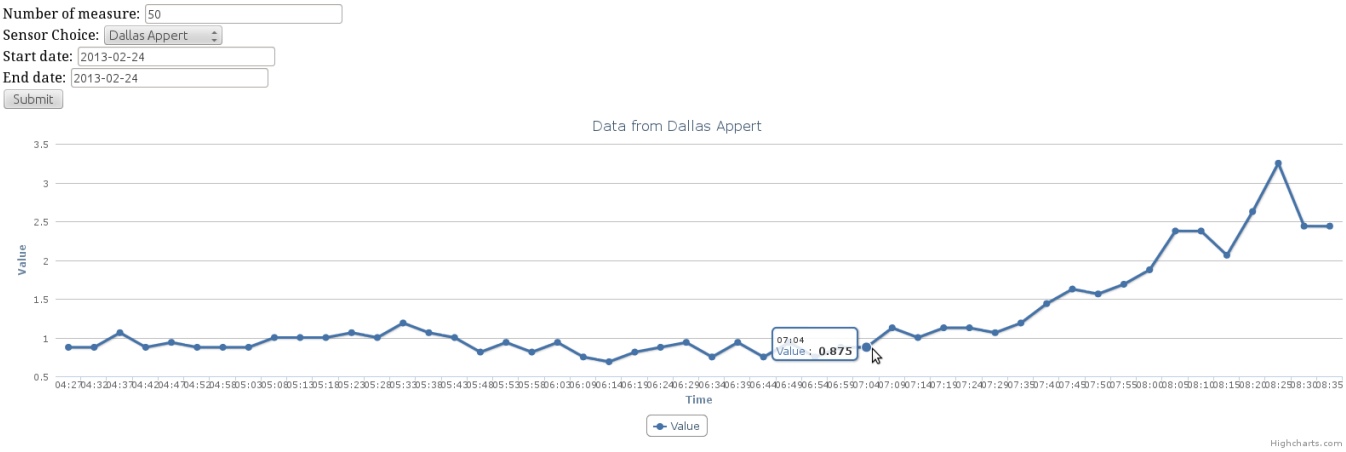


Fig. 13: Diagramme de classe de la partie Django

- Django est un framework basé sur le modèle MVC (Modèle - Vue - Contrôleur).
- Le Modèle intègre les classes Measure et Sensor qui mappent directement les tables correspondantes de la base de données.
- Ici, la Vue et le Contrôleur du modèle MVC sont répartis entre les « Template » et les « View » de Django.
- La classe MeasureForm représente ce qui est reçu du module Acquisition par le biais de requête HTTP POST
- La classe GraphParamForm représente le formulaire de la vue FrontOffice concernant le graphique.
- Plusieurs fonctions seront exécutées en fonction des URL des requêtes HTTP arrivant sur le framework Django :
  - requête POST sur l'URL /measure/ : on attend une mesure correctement formatée dans la partie data de la requête depuis le module d'Acquisition ;
  - requête POST sur l'URL /graph/ : on retourne une page HTML contenant le graphique et son formulaire de contrôle ;
  - requête GET sur l'URL /admin/ : on arrive sur la partie BackOffice du site.



7 - Infrastructure de développement

L'infrastructure mise en place pour le développement est la suivante :

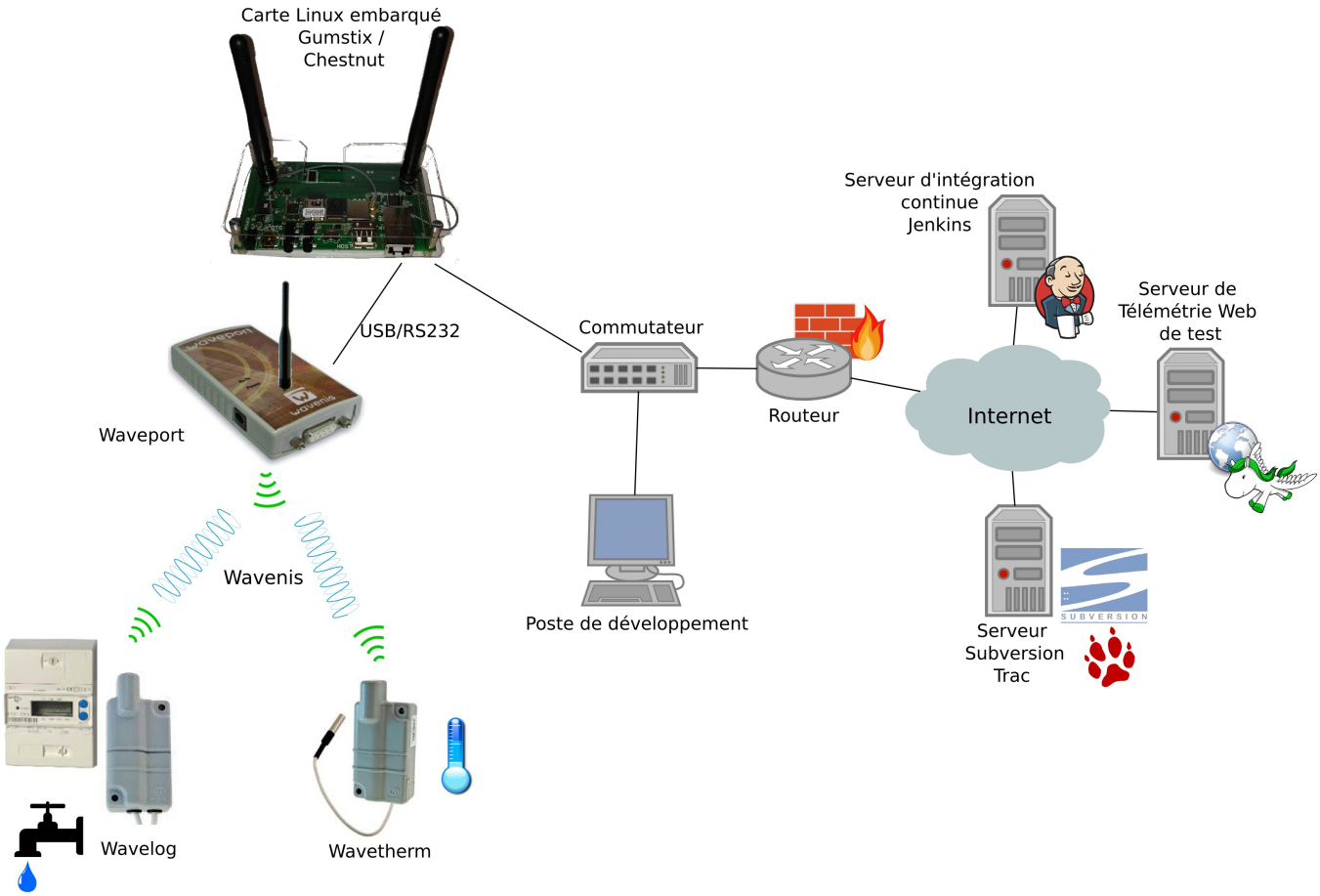


Fig. 14: Infrastructure de développement

- Poste de développement :
  - Ubuntu 12.04 ;
  - Plateforme de cross-compilation Yocto pour Gumstix - Overo ;
  - Environnement de développement Eclipse ;
  - Framework Django ;
- Une carte Gumstix connecté au réseau filaire ;
- Gestion de version : le code et la documentation sont versionnés par Subversion ; (https://www.skillsit.fr/tsvn/ewts-coronis/)
- Intégration Continue : le code est compilé et testé à chaque commit sur le serveur Subversion par Jenkins (http://iris2.appert44.org:21880/job/Ewts-Coronis/). Un rapport de test et de compilation est envoyé au développeur en cas de problème.
- Gestion de projet : serveur Trac (https://www.skillsit.fr/svn/ewts-coronis/)
- Serveur de test sur Internet : https://www.skillsit.fr/ewtswavenis/graph/ et https://www.skillsit.fr/ewtswavenis/admin/

8 - Propositions d'activités pédagogiques

8.1 - Présentation de la thèmatique

8.1.1 - Problématiques

- TODO

8.1.2 - Conditions générales de réalisation

- Ressources documentaires :
- Ressources logicielles :

8.1.3 - Compétences et Connaissances SN - IR

8.1.4 - Liste des activités

8.2 - Scénario 1 : TODO

8.2.1 - Problématiques

- TODO

8.2.2 - Compétences et Connaissances SN - IR

8.2.3 - Conditions générales de réalisation

- Ressources documentaires :
- Matériel à disposition :
- Logiciel à disposition :
- Matériel de test à disposition

8.2.4 - Liste des activités

8.2.4.1 - Modélisation et Simulation

- TODO

8.2.4.2 - Réalisation pratique et tests

-