ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Machine-to-Machine Communication: A heterogeneous Internet-of-Things testbed

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

(MASTER THESIS)

της

ΓΡΗΓΟΡΙΟΥ ΕΛΙΣΑΒΕΤ

(GRIGORIOU ELISAVET)

Thesis Supervisor:	Dr. Jesus Alonso Zarate, Head of M2M Department, CTTC, Barcelona, Spain
Thesis Advisor:	Franscisco Vazquez Gallego, Senior Researcher, CTTC, Barcelona, Spain
Academic Supervisor:	Dr. Periklis Chatzimisios, Associate Professor, Alexander TEI of Thessaloniki, Greece

Θεσσαλονίκη, Απρίλιος 2015

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ

ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Machine-to-Machine Communications: A heterogeneous Internet-of-Things testbed

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

(MASTER THESIS)

της

ΓΡΗΓΟΡΙΟΥ ΕΛΙΣΑΒΕΤ

(GRIGORIOU ELISAVET)

Thesis Supervisor:	Dr. Jesus Alonso Zarate, Head of M2M Department, CTTC, Barcelona, Spain
Thesis Advisor:	Franscisco Vazquez Gallego, Senior Researcher, CTTC, Barcelona, Spain
Academic Supervisor:	Dr. Periklis Chatzimisios, Associate Professor, ATEITHE, Thessaloniki, Greece

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 22/04/2015.

Χατζημίσιος	Ιωσηφίδης	Σαρηγιαννίδης
Περικλής	Αθανάσιος	Παναγιώτης
Αναπληρωτής Καθηγητής, Αλεξάνδρειο Τ.Ε.Ι.Θ.	*Επίκουρος Καθηγητής Αλεξάνδρειο Τ.Ε.Ι.Θ.*	*Λέκτορας, Π. Δυτικής Μακεδονίας*

Θεσσαλονίκη, 22/4/2015

....

Γρηγορίου Ελισάβετ

Περίληψη

Η Machine-to-Machine επικοινωνία χρησιμοποιείται με σκοπό την επιτυχή σύνδεση ανάμεσα στον κόσμο του Διαδικτύου και στον Φυσικό κόσμο. Η εποχή της Μ2Μ επικοινωνίας έχει έρθει και αναφέρεται σε συσκευές οι οποίες επικοινωνούν με πλήρη απουσία ή ελάχιστη ανθρώπινη παρέμβαση. Ο ρόλος της Μ2Μ επικοινωνίας είναι να δημιουργεί τέτοιες συνθήκες ώστε να είναι δυνατή η ανταλλαγή πληροφοριών μεταξύ μιας συσκευής και μιας εφαρμογής μέσω ενός δικτύου επικοινωνίας. Η παρούσα Διπλωματική Εργασία είναι μέρος του "Smartworld" Project του CTTC στην Βαρκελώνη. Το "Smartworld" είναι μια end-to-end πλατφόρμα επίδειξης για το Διαδίκτυο των Πραγμάτων (Internet-of-Things - IoT). Το Smartworld είναι μια ετερογενής πλατφόρμα δοκιμών που αποτελείται από διαφορετικούς τύπους Μ2Μ συσκευών και από τον Οκτώβρη του 2014 περιλαμβάνει 2 τεχνολογίες αισθητήρων, τους PanStamp και Zolertia Z1. Η πλατφόρμα υποστηρίζει διαφορετικές τεχνολογίες πρόσβασης, όπως 3G/4G, IEEE 802.15.4, WiFi και SIGFOX.

Τα δεδομένα των αισθητήρων μεταδίδονται μέσω διαφορετικών τεχνολογιών ασύρματης επικοινωνίας σε μια Cloud υπηρεσία στο Διαδίκτυο, όπου οι πληροφορίες αναλύονται και αποθηκεύονται. Τέλος, όλα τα δεδομένα που συλλέγονται μπορούν να ανακτηθούν από μια εφαρμογή Android που έχει εγκατασταθεί σε ένα Tablet, ώστε να υλοποιηθεί μια εφαρμογή για τον τελικό χρήστη, όπως είναι το Έξυπνο Παρκινγκ.

Λέξεις Κλειδιά:<<M2M, IoT, sensors, heterogeneous, testbed, cloud , android >>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Machine-to-machine (M2M) communications are used to realize successful connections between Internet and our physical world. The era of M2M communications has come and refers to communication devices that operate under the complete absence or with minimal human intervention. The role of M2M is to create the conditions to enable a device to exchange information with an application via a communication network. The current Master Thesis is part of the “Smartworld” project in CTTC in Barcelona. SmartWorld is an End-to-End (E2E) demonstration platform for the Internet of Things (IoT). SmartWorld heterogeneous testbed is formed by different types of M2M devices and sincethe beginning of October 2014, two sensor technologies are integrated, PanStamp and Zolertia Z1. Multiple radio access technologies such as 3G/4G, IEEE 802.15.4, WiFi and SIGFOX are supported. Sensor data is transmitted via different radio communication technologies to the Internet cloud , where information is stored and analyzed. Finally, all collected data can be retrieved from an Android application running in a tablet, in order to implement end-user applications like smart parking.

Keywords:<< M2M, IoT, sensor, heterogeneous, testbed, cloud , android >>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Acknowledgements

This thesis was conducted at the Centre Tecnològic Telecommunications Catalunya of Barcelona. The current M.Sc. thesis is part of the SmartWorld project developed by the Communication Technologies Division of CTTC.

Firstly, I would like to express my deepest acknowledgment to my Erasmus placement Supervisors Dr. Jesus Alonso Zarate (Head of the M2M Department) and Francisco Vazquez Gallego Senior Researcher at CTTC Spain for welcoming me into the world of research. I would like to also thank Selva Via and Giacomo Genovese (researchers at CTTC Spain). Without their guidance this dissertation would not have been possible. Special thanks goes to my University Supervisor Dr. Periklis Chatzimisios (Associate Professor at the Department of Informatics of Alexander of Thessaloniki, Greece) for consulting and aiding me in writing the current Thesis.

I would also like to thank several open source contributors who have replied to me in online forums and have guided me during various stages of this thesis work. Moreover, I appreciate the help from my friends and fellow students of the M.Sc. Program in ATEITHE. A large part of what I managed owe to them. Last but not least, I would like to express my gratitude to family for their endless support during my study years.

Thanks You!

Elisavet Grigoriou

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analog-to-Digital Converter
CTTC	Centre Tecnològic de Telecommunications de Catalunya
DSP	Digital Signal Processing
E2E	End – to – End
ETSI	European Telecommunications Standards Institute
EVB	Evaluation Board
FDDI	Fiber Distributed Data Interface
GPS	Global Positioning System
GPIO	General Purpose I/O
GW	Gateway
I2C	Inter-Integrated Circuit
ICT	Information and Communication Technology
IoT	Internet of Things
IPv6	Internet Protocol Version 6
ITU	International Telecommunication Union
LOS	Line of Sight
LPWA	Low Power Wide Area
LTE	Long Term Evolution
H2H	Human – to – Human
M2M	Machine-to-Machine
MAC	Medium Access Control
OS	Operating System
PLC	Power Line Communication

PWM	Pulse-Width Modulation
RPi	Raspberry Pi
RFID	Radio-frequency identification
QoS	Quality of Service
QoE	Quality of Experience
SPI	Serial Peripheral Interface
SWAP	Simple Wireless Abstract Protocol
UART	Universal Asynchronous Receiver/Transmitter
UICC	Universal Integrated Circuit Card
UNB	Ultra Narrow Band
USN	Ubiquitous Sensor Network
WSN	Wireless Sensor Network
WLAN	Wireless Local Area Networks
WPAN	Wireless Personal Area Networks
WWAN	Wireless Wide Area Networks
XSM	Exreme Scale Motes
XSS	Extreme Scale Stargates

1. Introduction

1.1 Overview

The Internet of Things (IoT) is the future of Information and Communication Technology (ICT), which will add to Internet the "*connectivity for anything*" feature. During the last years, renovation in technologies that support Machine-to-Machine (M2M) has allowed the development of the IoT vision. However, the M2M world is characterized by a wide heterogeneity of compatible technologies that are integrated into a larger system. The aim of the work developed so far is to overcome the problem of heterogeneity by creating a heterogeneous M2M platform.

Over the years, mobile networks have been utilized to serve the needs of Human - to - Human communications (H2H). However, machines utilize the same mobile networks to serve their communication needs. The developed applications behind these machines can be used for various purposes and in particular to serve the different needs of various industries, as well as to satisfy the needs of end-customers by improving their quality of life.

In the near future, most of the mobile network users will utilize M2M type communications. This ongoing rise of M2M usage also brings in new business opportunities and sources of gain. Enhanced services and their good availability enable more and more devices and complicated applications to be connected. For the telecommunications industry, it would be beneficial to serve a high number of connected devices and sources of revenue growing and also be able to address the needs of the different verticals with a harmonized horizontal solution.

Recent developments predict that we will have more than 16 billion connected devices by the year 2020 [1], which will average out to six devices per person. Devices like smart-phones and M2M or thing-to-thing communications will be the main drivers for further development.

Wireless connected sensors in everything we own already form a new web. However, it will only be of value if the big data that are generated can be collected and analyzed. The first direct effect is the generation of enormous quantities of data from physical or virtual connected objects. As a result, consumer-device-related messaging volume could easily reach between 1,000 and 10,000 per person per day [2].

In the current Thesis, M2M communication refers to machine-to-machine, machine-to-man, or man-to-machine communications. These contexts all have a common element which is machine interaction. Some examples for the above-mentioned categories are:

- Machine-to-machine without any human interaction, e.g. remote sensors sending measurements to a shared data collection server.
- Man-to-machine, e.g. an operator that remotely configures, supervises and operates equipment.
- Machine-to-man, e.g. a vehicle initiating a call with an emergency service when an accident occurs.

M2M key features

- **Device Heterogeneity:** The wide heterogeneity of devices that take part to the network has to be managed at architectural and protocol levels.
- **Scalability:** In amount of connected objects.
- **Ubiquitous data exchange through proximity wireless technologies**: Wireless play an important role inside IoT, hence it will important to optimize the access to frequency spectrum.
- **Energy optimized solution**: Energy efficiency means longer autonomous life for all objects and, therefore, it is important to research the most efficient way to provide energy to devices for long periods.
- **Localization and tracking capabilities**.
- **Semantic interoperability and data management**: The usage of data independently from the source in order to build valid meta-data is very important to ensure service provision.
- **Embedded security and privacy-preserving mechanisms**. It's obvious that these systems will be present in our everyday life and it should make users feel comfortable with this technology.

1.2 Thesis Scope

The main objective of the current Thesis is to deploy an IoT testbed comprising the following elements:

1. A **wireless M2M area network** to connect sensors and actuators. In particular, sub-GHz transmissions have been chosen due to their suitability for IoT.
2. A **gateway** providing the wireless M2M area network with connectivity to the Internet.

3. A **cloud platform** to store data gathered from the real world and properly formed to be used by end-users promoting the so-called smart applications.

4. An **application** on a tablet to get access to the cloud and retrieve information that is gathered from the wireless network and interact with the actual sensors and actuators.

This Thesis is part of “Smartworld” project in CTTC in Barcelona. **SmartWorld** is an End-to-End (E2E) demonstration platform for the IoT developed by the Machine-to-Machine Communications Department at CTTC (M2M@CTTC) in Barcelona, Spain [3].

SmartWorld is an innovative and cloud-based solution which integrates various wireless communication technologies into a single platform. It integrates 2G/3G cellular communications, Long Term Evolution (LTE), WiFi, ZigBee, low-power long-range transmissions (at sub-GHz bands) that all operate in a seamless manner. Data travels from the sensors, through wireless networks, to reach the cloud in the Internet. Consequently, remote access from all kind of user devices such as tablets, smartphones, web-clients, or wearable devices is possible to create an unprecedented number of innovative applications.

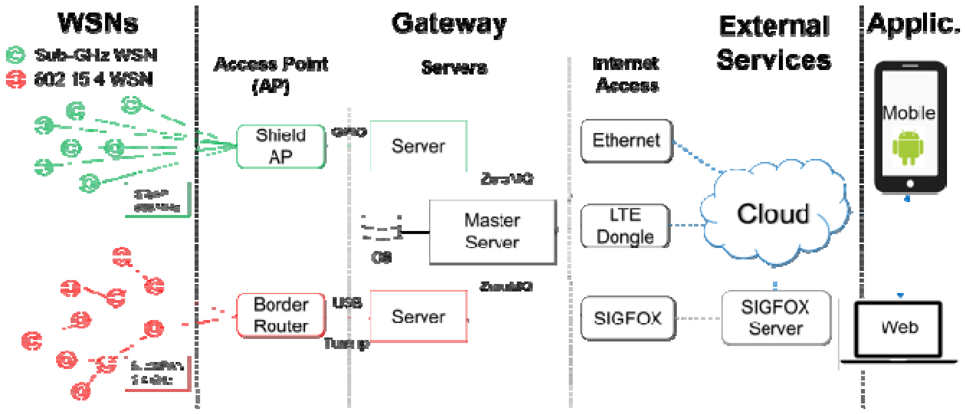


Figure 1 Smartworld detailed architecture [3]

Figure 2 Smartworld technologies [3]

1.3 Thesis Organization

The current M.Sc. Thesis is organized in the following chapters:

Chapter 2 presents a review of some relevant existing IoT platforms. The presented platforms are based on two basic features: heterogeneity and scalability.

Chapter 3 provides general information about the Internet-of-Things, Machine-to-Machine communications and a detailed description of the IoT components that were utilized to develop the IoT testbed.

Chapter 4 describes in detail the implementation of the testbed that was deployed during this Thesis.

Chapter 5 concludes the Thesis by providing overall conclusions and stating future challenges and issues for further research in the area.

2. Relevant Research

2.1 Introduction to performance evaluation tools for WSNs

Wireless Sensor Networks (WSNs) are made up of cooperative cheap sensors with limited resources, such as computing power, communication, storage and energy that form a network. Sensor nodes monitor physical or environment conditions like temperature, humidity, light, sound, pressure, velocity, acceleration acoustic and magnetic field. These sensors send the sensed raw data through the network to the main location of interest or to the sink node.

WSNs have been considered as the core technology of the emerging IoT, in which objects with intelligence connect with each other through heterogeneous communication networks and work collaboratively. WSNs are captivating significant attention from research community because of their huge range of applications. Therefore, most applications of WSNs charge significant challenges due to hardware availability, resource constraints, unreachable terrain, cost, effort etc. Thus, performance evaluation of algorithms and protocols becomes challenging at all different stages of design, development and implementation.

There are three main performance evaluation tools for WSNs, simulators, emulators and physical testbeds. Each tool is appropriate at different stages of implementation under different scenarios and this means that each one of them is appropriate for specific situations [4].

1) A simulator is a good choice at the earlier phase of design and development since it provides higher level of abstraction. It involves low cost and can simulate network with thousands of nodes. The time, effort and resources required for simulation is the minimum. On the other hand, very few specialized simulators for WSNs are currently available. There are object-oriented based simulators that simplify implementation and extensibility due to modular approach but they lack scalability. Component-based simulators are more efficient, offer higher extensibility, reusability and scalability. Java-based simulators are platform independent but relatively slower in execution time. Commercial simulators offer good graphical interface can help and support modeling sensor-specific hardware but a significant cost is incurred. Most of the simulators cannot guarantee the accuracy of derived results due to many considered assumptions.

However, simulation has many limitations, among which low efficiency and inconsistency between the simulated scenario and the real world being the most critical ones.

2) Emulators are more effective and useful exclusively when physical testbeds are not available or it is not feasible to be deployed for sensor network applications. Emulators execute same code on real nodes, thus, they reduce implementation effort and results obtained from emulators are more accurate than simulators. Most of the emulators are based on component-based architecture in order to offer higher scalability. The problem with emulators is that they can simulate code for specific platforms.

3) Testbeds constitute an environment that provides support to measure number of physical parameters in a controlled and reliable environment. This environment contains hardware, instrumentation, simulators, various software and other supporting elements that are needed to conduct a test. Testbeds bridge the gap between simulation and the deployment of real devices. Testbeds can actually improve the speed of innovation and productive research and in general the derived results are considered as more accurate than software-based tools. Physical testbeds are the best choice when it is not feasible to test the performance in real environment due to cost, time and inaccessible terrain restrictions. However, testbeds are limited in scope and may not provide practical environment.

Testbeds are typically assigned to one or more of the following classes:

• General-purpose or application- specific.

• Indoor or outdoor.

  o As indoor testbed deployments typically feature a set of nodes with debugging and deployment support interconnections, the node behavior can be easily traced, and software development and modifications performed quickly.

  o Outdoor experiments are well suited to conduct long-term experiments to gain information on the performance and efficiency of existing algorithms that have passed their verification on an indoor testbed. However, debugging support is often limited in outdoor deployments

• Mobile or stationary.

  o The mobility can be provided by mounting the sensor nodes on small sized robots. In order to provide a 24x7 operation of the testbed, the mobile nodes should have auto-recharging capability.

• Additional subclasses: Integration with IP networks or support for heterogeneous nodes.

A typical testbed includes a network of real sensor nodes that are controlled and accessed through a software layer. The usage of real sensor nodes preserves the impacts of radio communication and mote limitations whereas the software layer improves the usability of the testing environment and allows the automation of tasks such as data collection and monitoring of nodes. A testbed should support the implementation of protocols, algorithms, an easy and changeable deployment and testing. It should have interfaces for researchers and developers to simply develop their code, load and test. Further, the experimental results should be collected so that an analysis can be performed. Moreover, a user has to update the experimental code due to bugs, errors or simple enhancements to the already available code. On a WSN, all these requirements will indeed make it difficult for the experimentalists to run their code and get accurate and useful results.

Existing testbeds normally require a fixed support system to control individual nodes/devices, here sensor motes, through gateways. These gateways have an USB or Ethernet interface through which they are wired into a network. A user issues new experimental code or commands to sensor motes by first sending them to the gateways, which then relay them to the sensor motes. In a similar way, experimental results are passed to gateways and then they are collected by a data server. In these testbeds, sensor motes only take care of experiments. The control and management are all carried out by the backbone, which consists of servers and gateways wired together. Although this kind of testbeds can reliably carry out experiments, they are not flexible because of the wired backbone that implies restrictions.

The main features of WSN testbeds can be found as follows [5]:

1. Scalability and re-configurability: The main feature of a WSN testbed is its ability to be expanded and adapted to meet requirements caused by the growth in the research domain. Increasing the number of nodes in a scalable and reconfigurable testbed should require very limited hardware adjustments with minimal impact on the software components.
2. Heterogeneity: Plenty of different platforms developed in the related research

require that WSN testbeds support heterogeneity, which allows the production of autonomous non-uniform networks with the restriction that the motes within each network are homogeneous. Network-level heterogeneity and its application-level variant discard this constraint, allowing mixed mote types within the same network. An extra level of heterogeneity can be accomplished by adding programmable robots to support mobility. The existence of mobile nodes increases the complexity of the node communication schemes and often requires node localization services. Many testbeds provide support for computational heterogeneity through two-tier architecture with small sensing and more powerful computing nodes. Other architectures support heterogeneity by allowing to attach different sensor platforms that use the same radio channels and protocols.

3. Robustness and controllability: Mote programming and management are made possible by exchanging control commands between a management station and motes. A robust and controllable WSN testbed is capable of establishing a bi-directional communication link for the transfer of control messages between nodes and a management station at any time instance.
4. Debugging, data logging and monitoring: Debugging is the process of tracking the execution of sensor programs at selected nodes in order to identify erroneous fragments of code or to preview the values of key variables. Debugging mechanisms are necessary for the implementation of data logging tools capable of capturing, time stamping, and storing sequences of readings obtained by sensor nodes. Data logging is often accompanied by a monitoring.
5. Ease of access: A common practice in WSN testbeds is the implementation of a web interface that provides the user with a set of graphical interfaces for accessing the functionality of the testbed. An advantage of web interfaces is that they centralize access to the testbed’s resources for both programming and monitoring motes.
6. Real time interaction: This characteristic implies that the application developer has the opportunity to communicate directly with the mote that has gained access to. This element is essential since applications that follow the request/reply model are very common in wireless sensor networks. It is often attractive to observe the results as sampling happens in real time.

2.2 Overview of WSN testbeds

We present Testbeds based on the Scalability and the Heterogeneity.

	Scalability	Heterogeneity	Mobile	Large Scale	Indoor / Outdoor
Kansei	✓	✓	✓	✓	Indoor
Sensei-UU	✓	✓	✗	✓	Outdoor
WSNTB	✓	✓	✗	✗	—
Easitest	✗	✓	✗	✗	—
TWiNS.KOM	✗	✓	✓	✗	—
FIT IoT-Lab	✗	✓	✓	✓	Indoor/Outdoor
GNOMES	✗	✓	✗	✗	—
Scorpion	✗	✓	✓	✗	Indoor / Outdoor
Versatile Testbed	✗	✓	✗	✗	—
Twonet	✗	✓	✗	✓	Indoor / Outdoor
Emulab	✓	✗	✓	✓	Indoor / Outdoor
SenseNET	✓	✗	✗	✗	Indoor
MoteLab	✗	✗	✗	✗	Indoor
CitySense	✗	✗	✗	✓	Outdoor
Mint-m	✗	✗	✓	✗	Indoor
Pharos	✗	✗	✓	✗	Outdoor
Secure M2M	✗	✗	✗	✗	—

Table 1 Characteristics of the testbeds - A comparison

We found that existing testbeds go through some problems and limitations, some of them are the following:

1. The sensor node structure is inflexible. The nodes are incapable of supporting diverse kinds of applications.
2. The communication interfaces are simple and limited, and do not support complex heterogeneous network.
3. Capabilities for debugging, programming, and testing are limited.
4. Low-speed USB and serial ports are used for monitoring and control, which limits the scalability of the testbed.
5. None of the existing testbeds supports multiple radios. Novel multi-channel protocols and mechanisms cannot be evaluated.

2.2.1 Support Heterogeneity

2.2.1.1 Support Scalability

Kansei [6] from the Ohio State University is a large-scale testbed that includes 210 Extreme Scale Motes (XSM) and Extreme Scale Stargates (XSS). The topology utilizes Ethernet and IEEE 802.11b to control the testbed. Kansei also provides a web interface for users to upload programs, scheduled jobs and retrieve results with EmStar software framework. Kansei focuses on sense and scale. In particular, heterogeneous hardware infrastructure supports complex experimentation, time accurate hybrid simulation engine, high fidelity and real-time sensor data generation.

Figure 3 Kansei testbed demonstration [6]

Sensei-UU [7] supports different types of sensor hardware and sensor Operating Systems (OSs). Moreover, Sensei-UU supports scalability and employs low-cost hardware and open-source software. The ability to relocate a WSN testbed makes it possible to evaluate application behavior in different environments. The key for a relocatable testbed is to have a wireless control channel that allows Sensei-UU to be independent of existing communication infrastructure. Furthermore, it supports the several combinations of motes and in particular a variety of sensor hardware from motes to smart phones, running different sensor OSs like TinyOS, Contiki, Linux and Symbian. In fact, Sensei-UU does not consider sensor nodes to be static, but mobile by default with a static location as a special case of mobility. This means that the testbed supports mobility. The aim of this testbed is to be a WSN testbed that is not tightly coupled to location, sensor hardware or sensor OS.

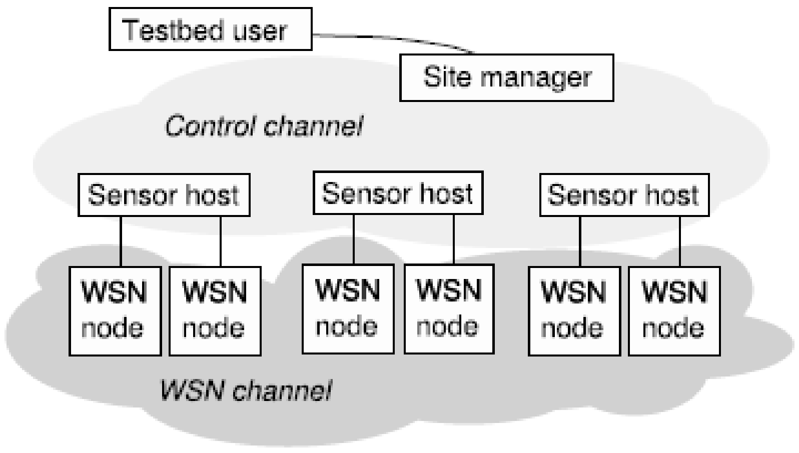


Figure 4 High level design of Sensei-UU [7]

WSNTB [8] is a reconfigurable heterogeneous WSNs testbed and its architecture is divided into the following three layers:

- The first layer is the servers. There are two kinds of self-designed sensor, namely Octopus I and Octopus II. The former is based on the Atmel AVR architecture and the latter is based on the Texas Instruments MSP430 architecture. The testbed consists of 34 sensor nodes (Octopus I and Octopus II). These nodes are distributed over two laboratory rooms. Since it incures low cost, expending the testbed scale is easy.
- The second layer is the gateways with converters. The converters help sensor nodes to convert serial data into TCP/IP packets. The gateways are bridges between Internet and WSNs and they have their own physical IP address to communicate with sensor nodes.

The gateways and converters are controlled by servers. The result is the WSNTB with gateways supports multiple WSNs over the Internet.

- The third layer is the sensor nodes. The testbed supports the famous sensor operating system TinyOS and implement LOS Operating System. In order to help users to process their individual experiments within different sensor nodes and heterogeneous WSNs, the testbed`s team has developed a middleware. This software framework allows users to run their programs and collect experiment results. Using the testbed website, users can book a period of time and choose nodes to experiment. Consequently, users can upload their own programs and reprogramming them. By using this testbed, researchers can save lots of time on building a huge experiment environment, in order to reduce hardware cost enhance the devices utility rate and in the same time fast verify the experiment results.

2.2.1.2 Do not support Scalability

Easitest [9] is a multi -radio testbed for heterogeneous wireless sensor networks. It is more powerful, more flexible, more user-friendly, and more scalable. Moreover, it provides a powerful tool not only for the study of heterogeneous WSNs, but also for quick-prototyping of novel WSN applications. In particular, EasiTest is separated into the following two domains:

- The WSN domain, which is composed of high-speed multi-radio sensor nodes EZ271 and low-speed sensor nodes EZ521, and supporting devices (e.g. wired or wireless gateways, switches, etc). WSN domain hosts the WSN applications.
- The administration domain, which is composed of the system management, the database and the web server. The administration domain is connected with the WSN domain through Ethernet in order to realize monitoring and control functions.

The testbed can be connected to the Internet to support remote access and can support multi-radio wireless communications, so researchers can develop protocols on each layer of network and the nodes in the testbed are equipped with rich interfaces to fulfill the requirements of various applications. The testbed also includes GUI, background supporting system, management software, and database system which provide a complete and effective solution for every research task. The management software is developed in JAVA and based on the Struts architecture with high extension capability. EasiTest management software is divided into three parts, namely application, testbed and server management. The database server is responsible for storing data and responding to search queries. On the other hand, the web server provides full-functional web interface for the end users to access the testbed.

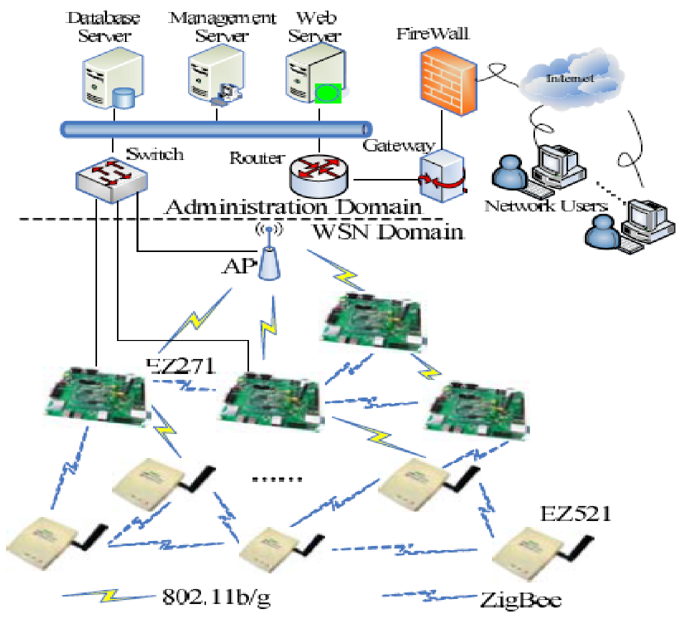


Figure 5 Structure of EasiTest [9]

In [10], authors present the tubicle, an integrated sensor node tube comprising three embedded systems with different characteristics, hence building a heterogeneous sensing platform. TWiNS.KOM testbed targets to deploy 20 tubicles, controlled and administered by testbed's management application. The tubicles have been designed to give extremely support to application developers, in terms of node control and debugging capabilities. The access is possible to be either wireless or wired. TWiNS.KOM, also supports mobility since the sensors are attached to moving objects. Hence, mobile and moving sensors are fundamental entities in object and person tracking scenarios, although not widely included in existing indoor sensor network testbeds.

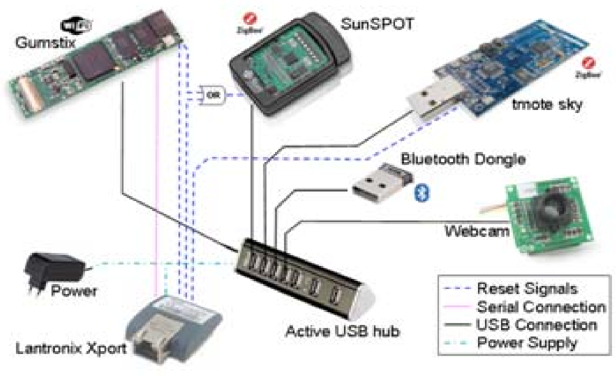


Figure 6 Interconnections between the employed platforms [10]

FIT IoT-LAB [11] is the logical evolution of the SensLAB. It provides a very large scale testbed infrastructure designed to experiment with heterogeneous embedded communicating objects, with small wireless sensor devices. At full capacity, the platform will give direct access to 2728 wireless sensors. The deployment is designed in phases: global infrastructure, tools, legacy WSN 430 nodes and the first set of M3 nodes are already available for experiments. Other parts, of this, are under construction and full capacity is planned very soon [11].

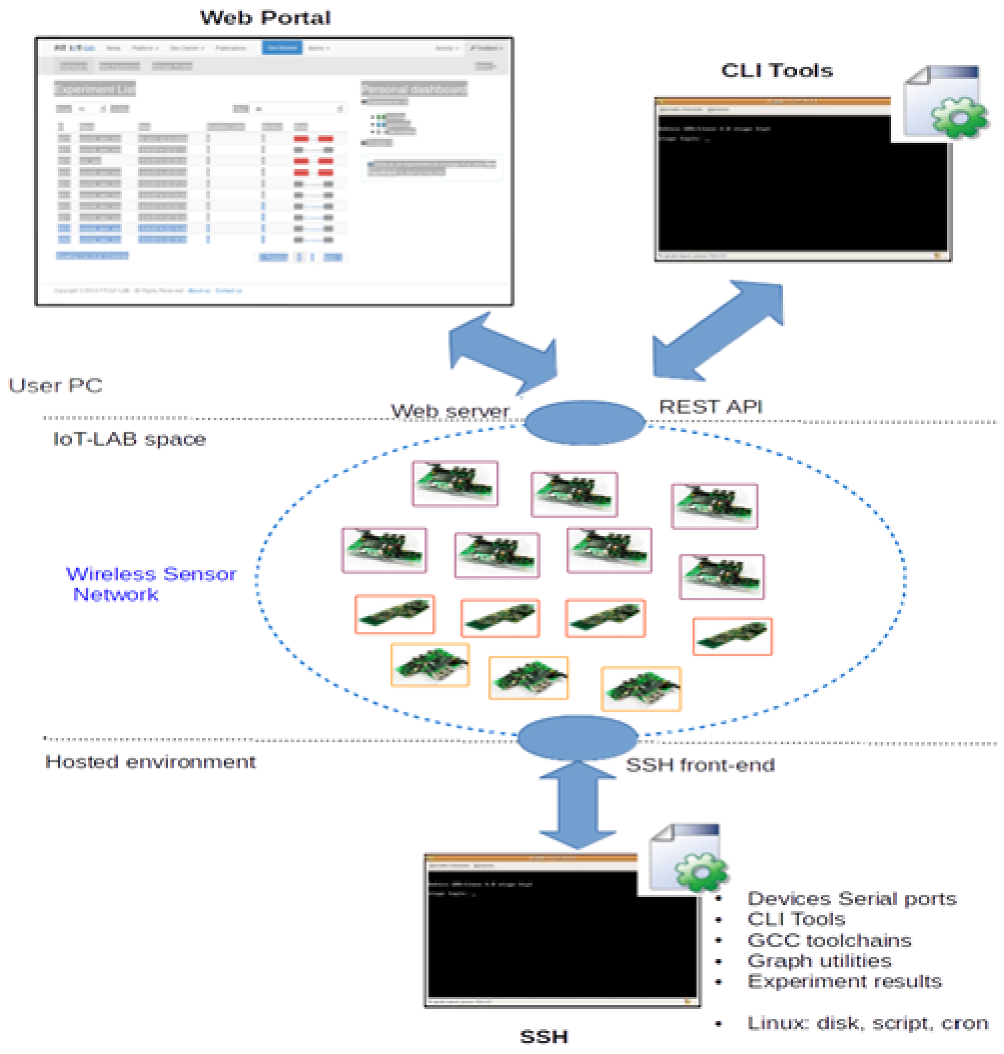


Figure 7 FIT IoT Platform access [11]

GNOMES [12] is a low-power and inexpensive testbed for WSNs that is implemented in practical application scenarios. Its aim was to create a system of low-cost sensor nodes that are self-organizing and can work in a collaborative manner to solve a given problem. The overriding design constrains of this development effort were cost and power. These constraints were based on traditional macro sensor devices that are massive and expensive but totally accurate. Data collected from multiple sensors can be combined to improve the accuracy through the use of advanced DSP algorithms. It is crucial to note that the system is heterogeneous. Each node in the network may sense different things. This has as a benefit keeping down the cost of each node since each one will not need a full complement of sensors.

The low cost and long life time of GNOMES make it ideal for inspecting individual approaches to traditional sensing applications, such as Seismic monitoring of civil structures. As this data is collected, the use of on-board computing power permits pre-processing of data, through decreasing the mandatory bandwidth for the centralized collection of data. This testbed is actively developed by creating new sensor and communication modules and by manufacturing a large quantity of the GNOMES nodes.

SCORPION [13] (Santa Cruz Mobile Radio Platform for Indoor and Outdoor Networks), is a heterogeneous wireless network testbed. This project is proposed by the Inter-Networking Research Group (i-NRG) at Santa Cruz University of California. The testbed implements a heterogeneous mobility platform both in the form of on-ground robots and airborne drones. The small remote controlled autonomous aircrafts are using Paparazzi autopilot control system to stay in. Paparazzi is a free and open source project under the supervision of ENAC University. The project is designed to provide highly reliable autopilot control systems. The main components include Airplane Node, Bus Node, Briefcase Node, and i-Robot Node that are detailed below:

a) Airplane Node: Four autonomous airplanes and four self-stabilizing helicopters provide aerial coverage. The aircrafts circle between different GPS waypoints. The nodes are mounted with IEEE 802.11 radios, can communicate with any other node in the testbed as well as act as bridge to the disconnected region of testbed.

b) Bus Node: Forty (40) nodes are equipped with wireless radios and deployed on campus busses in order to blanket the area effectively. Nodes have mini-ITX computer running Linux, three 802.11 a/b/g radios, a GPS tracking device and a 900 MHz radio to form network between buses and base stations deployed in campus.

c) Briefcase Node: Twenty (20) nodes are carried by students and change locations either by foot or bicycles while constantly transmit data to other nodes with which they come into contact with. These nodes include GPS receivers, a main and a laptop battery.

d) I-Robot Node: Twenty (20) terrestrial nodes roam the ground in an unpredictable way, making the testbed's behavior unspecified. These ground nodes are having mini-ITX computer running Linux Debian with three 802.11a/b/g radios. This hardware is carried by iRobot Create robots with different customizations.

The flying nodes include small sized self-stabilizing autonomous helicopters. The on-ground robots include an autonomous iRobot that is preassembled, programmable and ready to use robot. There are two non-autonomous robots, one being installed in a briefcase carried by a person in order to mime the human mobility patterns and the other is installed in the campus bus, with GPS navigator to illustrate vehicle's mobility pattern. The Bus nodes are installed with GPS tracking, and they transmit their GPS location to various base stations throughout the campus. This information is received by the central server where the current position of bus is indicated on Google Maps and then publishes information on the Internet. The briefcase nodes are carried by people that move according to ordinary routines, while the nodes are continuously communicating with other nodes in the area and can be tracked using GPS.

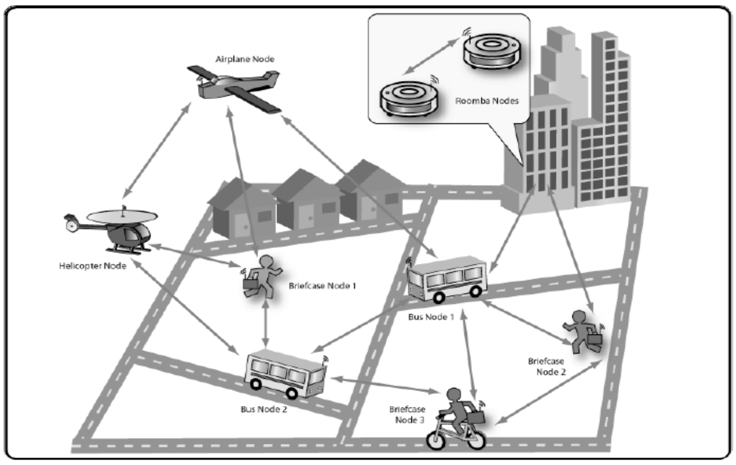


Figure 8 Its diverse set of nodes makes SCORPION suitable for experiments and evaluations under the most vast set of mobility applications [13]

The testbed should be capable of supporting heterogeneous networks, which means various wireless nodes should be supported, as much as the wired network. The supervision platform [14] is a main building part of the testbed. It can support more than 100 nodes, and measure the throughput, delay, packet loss rate, topology, etc. The heterogeneous networks are composed of high-speed nodes, medium-speed nodes, USRP2 software defined radio device, user computers, servers, WLAN access point, switches, and etc. USRP2 can be used as a standalone system without a host computer, so it is convenient for testbed work. EZ270-2 is a key building part of the testbed and it has a flexible role. EZ270-2 contains embedded WLAN node module and IEEE802.15.4 node module, so it can act as high speed node or medium speed node. In order to support a high data rate, G2M5477 (WiFi node module from G2

Microsystems) is selected as the radio interface. The module gain high reliability, low power consumption and supports the customization of the Medium Access Control (MAC) layer protocol with open source codes. The Ethernet port is connected to the wired network and used to transmit measurement data and control command with the supervision platform. The collected data and events log are stored in a database. As the program and daemon are optimized, they can support hundreds of connections simultaneously. Moreover, the utilized Web Server is Apache with PHP that provides an interface to testbed users. The administrator can create, modify and delete a project, and manage the users. The users can define what kind of data to be collected. The collected data is visualized with the customized format. Both components are deployed on Linux platform. In the implementation, is selected as Web Server.

Twonet [15] is an available large-scale sensor network testbed with dual-radio sensor nodes based on the modern Cortex-M3 architecture. Twonet's aim is to increase the interest in research on multi-channel wireless networking in sensor networks. Twonet contains 100 Opal nodes with 2.4 GHz and 900 MHz radios deployed across four floors of an academic building. Furthermore, Twonet is robust, efficient and cost effective for deployment. The debug board allows advanced debugging functionality, typically only available in a small-scale testbed.

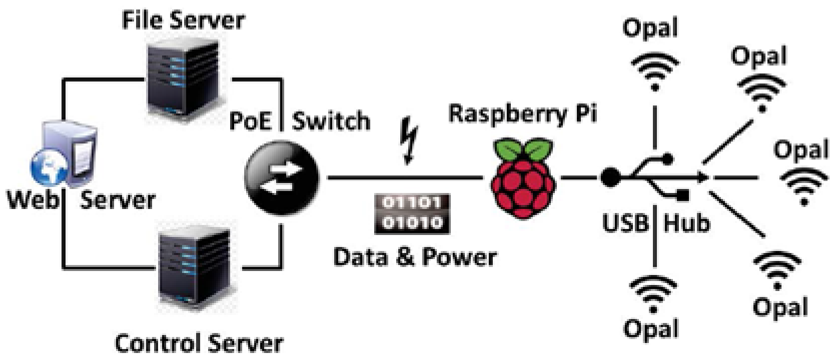


Figure 9 System architecture of the Twonet testbed: Up to five Opal sensor nodes are connected to each proxy via USB. Multiple proxies are connected to the testbed controller through a power-over-Ethernet backbone [15]

2.2.2 Do not support heterogeneity

2.2.2.1 Support Scalability

Emulab [16] is a network emulation testbed that offers a wide range of experimental environments in which we can develop, debug and evaluate our systems. In addition to fixed wireless nodes, Emulab also features wireless nodes attached to robots that can move around a small area. These robots consist of a small body with an Intel Stargate that hosts a mote with a wireless network interface. Emulab targets to be a “mobile wireless testbed” and offer to users the opportunity to conduct experiments with wireless nodes that are truly mobile.

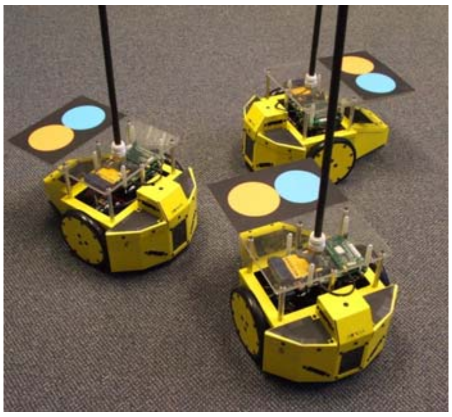


Figure 10 Emulab Robots [16]

SenseNET [17] is a scalable and cost efficient testbed architecture that allows the existence of multiple users to interact with different portions of the testbed without relying on the existence of a wired infrastructure or nodes with more capabilities other than simple MICA2 motes. The asset of this testbed`s implementation depends only on software that utilizes the wireless channel in order to achieve the necessary functionality. The final goal is to support scalability. More specifically, this deployment does not demand cables, special hardware and specific drivers. The only requirement is the involvement of a software component in the motes, a process which is easy and straightforward. This approach enhances the testbed with self-organizing and self-healing capabilities since when a node is gradually depleted from energy or when it suddenly dies another one may take its role.

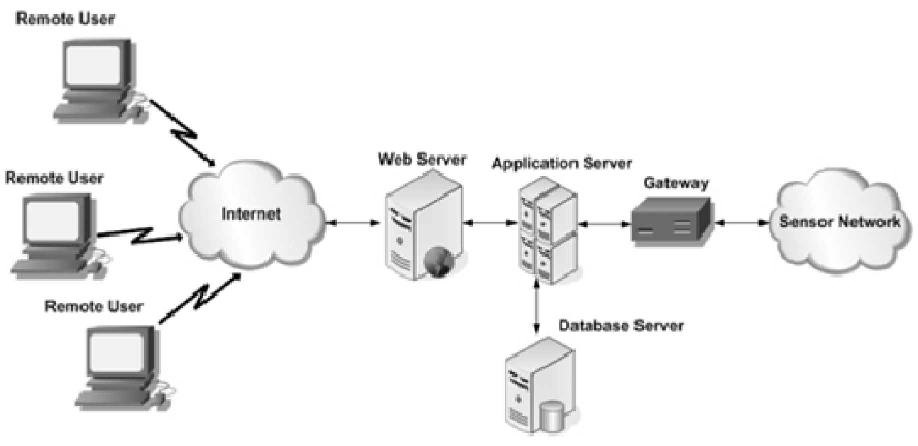


Figure 11 Overall Architecture [17]

2.2.2.2 Do not support Scalability

MoteLab [18] is an experimental WSN deployed in Maxwell Dworkin Laboratory, at Harvard University. In MoteLab, the sensor nods are connected to a central server that is capable to reprogram and data log and provides a web interface, in order to create and schedule events on the testbed. MoteLab prompts application deployment by consolidating access to a large and fixed network of sensor network devices. Moreover, by providing a web interface, MoteLab allows local and remote users to have access to the testbed. Furthermore, it is freely available and easy to install, so other organizations can easily set up their own MoteLab testbeds.

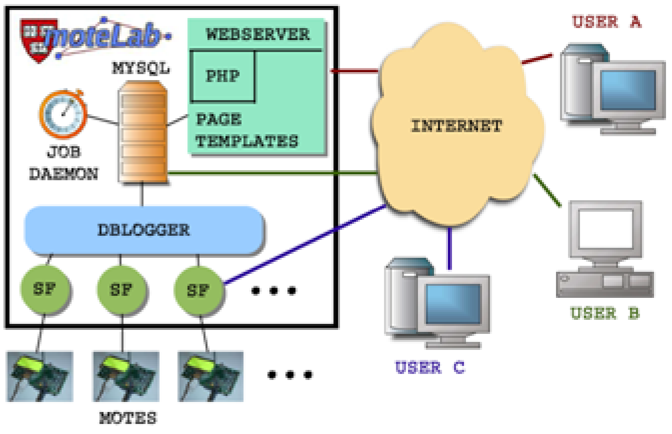


Figure 12 Component model showing the different software pieces that combine to form MoteLab [18]

MoteLab software components: MySQL Database Backend, Web Interface, DBLogger and Job Daemon. With these, MoteLab software can manage any lab of nodes providing remote reprogramming and data logging capabilities.

MoteLab hardware components: 26 Mica2 motes and 26 Ethernet interface boards. The Mica2 "mote" consists of a 7.3 MHz ATmega128L processor, 128KB of code memory, 4KB of data memory, and a Chipcon CC1000 radio operating at 433 Mhz with a data rate of approximately 34 Kbps. There are 6 EPRB's developed at Intel research by Phil Buonadonna and 20 Crossbow MIB-600 "emotes". Both provide one TCP port for reprogramming and another for data logging.

CitySense [19] is a scale sensor network testbed that is developed by Harvard University and BBN Technologies. CitySense consists of 100 wireless sensors will cover the city of Cambridge. The sensors are embedded to telephone poles measuring the specific locations and times of day when pollution peaks. Each node consists of an embedded PC, IEEE 802.11

a/b/g interfaces, and sensors for monitoring weather conditions and air pollutants. CitySense is an open testbed that researchers can use to evaluate wireless networking and sensor network applications in a large-scale urban environment.

Miniaturized Network Testbed for Mobile Wireless Research (Mint-m) [20] is an indoor testbed designed by researchers of Stony Brook University. Mint-m uses Roomba robots which have embed a wireless device called Router BOARD 230. The Router BOARD has four wireless interfaces each provided by a mini-PCI IEEE 802.11 a/b/g wireless card. With these four cards, the nodes can be used in multi-radio experiments. In order to keep the testbed area smaller, the radio signal attenuators are used between a wireless interface and its antenna to decrease the signal powers and ultimately decreasing the physical space requirement. Also, it supports mobility. In order to do that, it uses a low cost robotic vacuum cleaner called Roomba. Roomba is an externally controlled and self-operating robotic vacuum cleaner developed by iRobot. Roomba robot comes with auto-recharging features. Nevertheless, the electronics are modified to provide power to the wireless router mounted on the robot. The testbed uses commercial off the shelf available webcams to monitor the position of nodes with accuracy. The webcams are installed with the ceiling of the indoor site. Each wireless device has a board attached having a different color combination. The vision-based positioning system uses these color combinations in estimating and planning the trajectories of the robots to make a collision free movement of nodes.

The **Pharos** [21] project was developed by The University of Texas at Austin and it is an outdoor testbed that uses robots to provide mobility. Pharos focuses on the evaluation of mobile applications at all levels (from the mobile device and its physical radio up through the network stack to tailored application functionality) in live networks. More specifically, it uses Proteus1, as mobile node (a robot). The basic components include x86 Linux motherboard attached with Freescale microcontroller. This hardware supports a variety of devices that can be plugged in, such as different sensors. For wireless communication an IEEE 802.11 b/g wireless card is used. For the mobility commands the user applications developed in Player/Stage API are used. For the localization of nodes in Pharos testbed different devices and sensors are used like range finding sensors, digital compasses, GPS and cameras. The Pharos testbed achieves accuracy of measurements by requiring applications to be executed on real hardware that physically moves in real space.

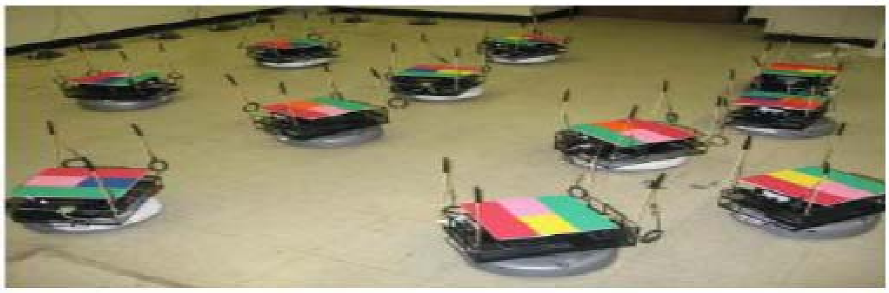


Figure 13Roomba robots with mounted wireless routers and the docking stations on top left corner [21]

Pharos also achieves reproducibility to a limited extent by making testbed experiments as routine as possible. Comparability of testbed measurements is driven by the need to understand the differences between competing hardware and software options. Finally, Pharos accomplishes swift validation of simulation results by enabling the semi-automatic adaptation of simulation behaviors to the testbed and the ability to rapidly swap components. In the end of implementation, Pharos should be available to third parties for validations of their mobile computing software implementations.

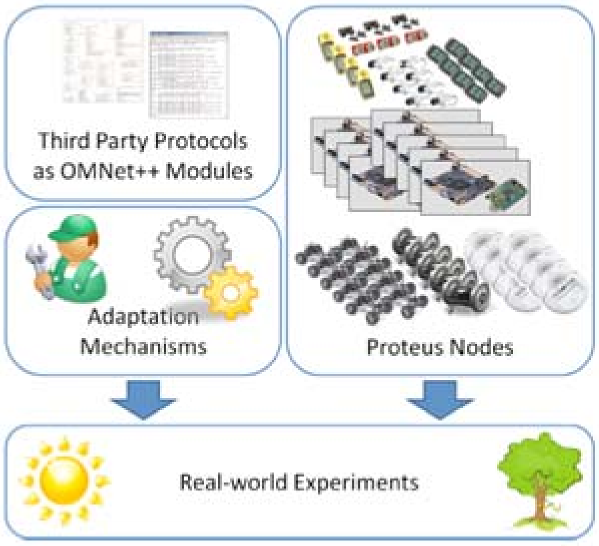


Figure 14 Components of Pharos Project [21]

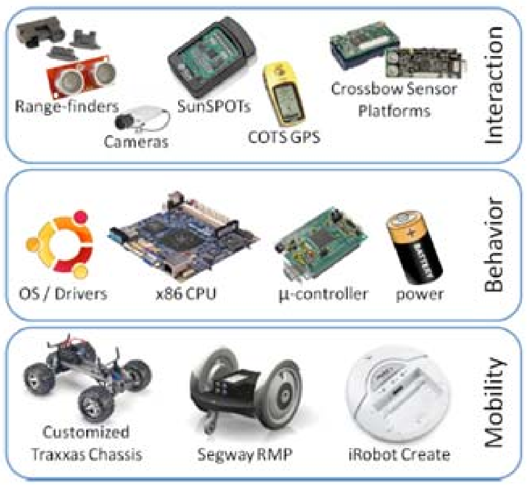


Figure 15 Hardware architecture of Pharos Project [21]

Secure M2M Cloud Testbed [22] is a prototype platform that provides resource-constrained smart objects with secure connectivity to a cloud environment over the Internet. It uses a cloud service allowing easy and dynamic deployment of services, as well as, facilitates network operators and enterprises with a platform where services can be managed centrally on a large scale. The smart objects are devices that utilize Arduino2 development boards and have sensors that report physical data (such as temperature) as well as actuators (i.e. LEDs) that respond to actuation commands. It relies on energy-efficient IEEE 802.15.4 short-range radio boards (Digi XBee3) for communication. In order to enable smooth interaction between users and devices across heterogeneous networks, it is important that these devices are accessible over the Internet. We therefore use a Java OSGi4 based M2M Gateway (GW) device running on a Raspberry Pi (RPi) development board to provide Internet connectivity to the smart object network.

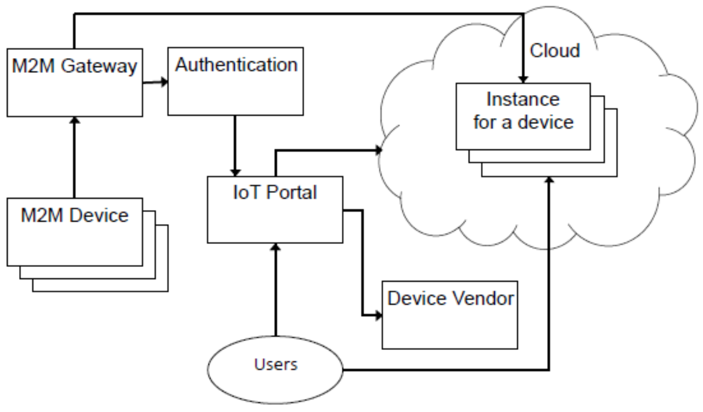


Figure 16 Secure M2M Cloud Testbed [22]

2.3 Conclusion

IoT applications put certain requirements on the supporting architecture and infrastructure. In our analysis, we derived a list of requirements that must be satisfied in order to provide adequate Quality of Service (QoS) and Quality of Experience (QoE) for various types of IoT applications. The major functional requirements of a testbed for WSN are listed below [23]:

- Support various kinds of networks for the evaluation of heterogeneous networks.
- Support multiple radios on a single node for the evaluation of multi-channel protocols.
- Support Scalability; logarithmic or better scaling of communication load in end-points.
- No central point of failure; fully distributed and several ways to connect to the platform.
- Bidirectional links; capable of communicating with both sensors and actuators.
- Fast; capable of signaling in real-time between end points.
- Current; all data retrieved should be the most current values.
- Lightweight; ability to run on mobile devices with limited resources.
- Stable; reliable handle transient nodes joining and leaving with high churn rates, while making sure that all queries into the platform should return an answer.
- Extensible; capable of adding new features and modules without complete redistribution, such as persistence, authentication, and reasoning.

3. Theoretical Background

3.1 Internet-of-Things: An Overview

The term *IoT* was conceived and first used by Kevin Ashton over a decade ago. The “things” are also known as “objects” “devices” “end nodes” “remotes” or “remote sensors” [24].

In 2005, the ITU (International Telecommunication Union) made a report on the IoT from technical, economical and ethical view that introduced a new axis in the ubiquitous networking path to complete the existing “*anywhere*” and “*anytime*” connectivity. It is the “*anything*” connectivity axis where the thing-to-thing or M2M interaction is added to complete the existing person-to-person and person-to-machine interaction in the possible connectivity framework (Figure 17).

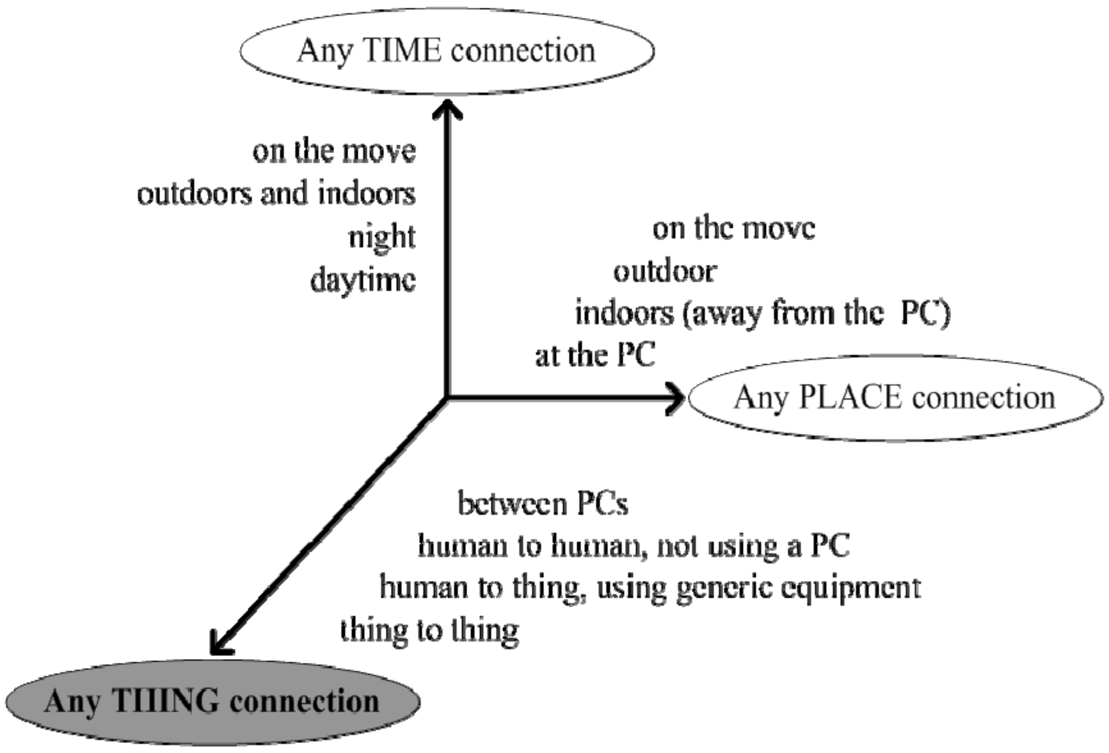


Figure 17 ITU any place, any time and any vision [24]

IoT generally uses low cost information gathering and propagation devices that facilitate fast-moving interactions in *anyplace* and at *anytime*, between the objects themselves, as well as between objects and people. Thus, IoT can be described as a new-generation information network that enables seamless and continuous M2M and/or H2M communication.

M2M services aim at automating decision and communication processes and support consistent, cost-effective interaction for ubiquitous applications. *M2M communication* is the communication between two or more entities that do not necessarily need direct human intervention: *it is the communication between remotely deployed devices with specific roles and requiring little or no human intervention*. [24]

Connecting objects might be *wireless*, as with the Radio Frequency IDentification (RFID), or sensor radio technologies that offer identification of items and sensing of the environment. Connection may be *wired*, as with Power Line Communication (PLC). PLC offers data transport over electrical media and has pioneered the in-home networking connectivity of electronic consumer devices that we also name "objects" like smart TVs, etc [25].

Microcontroller	8-, 16-, or 32-bit working memory and storage.
Power Source	Fixed, battery, energy harvesting, or hybrid
Sensors and Actuators	Onboard sensors and actuators, or circuitry that allows them to be connected, sampled, conditioned, and controlled
Communication	Cellular, wireless, or wired for LAN and WAN communication.
Operating System	Main-loop, event-based, real-time, or full featured OS.
Applications	Simple sensor sampling or more advanced applications
User Interface	Display, buttons, or other functions for user interaction
Device Management	Provisioning, firmware, bootstrapping, and monitoring.
Execution Environment	Application lifecycle management and Application Programming Interface (API).

Table 2 Properties of a device [25]

3.1.1 Basic Devices: Hardware Interfaces

The devices are usually designed for one purpose, such as measuring air. However, in some cases certain functions are deployed on the same device, such as monitoring humidity, temperature, etc. The hardware requirements are low, both in terms of processing power and memory. The main target is to keep the cost as low as possible by using inexpensive microcontrollers with built-in memory and storage, often on an integrated circuit with all main components on one single chip.

The microcontroller includes a number of ports that allow integration with sensors and actuators [26]:

- The *General Purpose I/O (GPIO)* interface is needed for controlling external peripherals and receiving digital input.
- An *Analog-to-Digital Converter* (ADC) for supporting analog input.
- *Pulse-Width Modulation* (PWM) provides a pseudo-analog output for M2M modules and is needed mainly for controlling lights, buzzers, and analog power output, but is very versatile.
- As *low-power* operation is supreme to battery-powered devices.

In order to interact with peripherals such as storage or display, it's ordinary to use a serial interface as it is described below:

- The *Serial Peripheral Interface (SPI)* bus is a synchronous serial data link that operates in full duplex mode where the devices are configured as either a master or a slave. This interface can support clock rates of up to 70 MHz. This is a general-purpose interface that can be used to control numerous peripheral devices but is commonly used for controlling external display equipment.
- *Inter-Integrated Circuit* (I2C) is a synchronous serial data link with a multi master bus. The transfer rate of the I2C bus can be 400 Kbps. This is a general-purpose interface that can be used to control numerous peripheral devices but is commonly used for controlling external displays, reading sensors or storage peripherals.
- *UART* is an extremely ordinary host interface and it is mainly used for communications or debugging functions. The baud rate supported by the module is 1200–115,200.

Other Hardware Interfaces:

- *Universal Integrated Circuit Card* (UICC) interface is a M2M module that will be used on GSM-evolved.
- Every wireless M2M module must have an *antenna interface* that may be an antenna pad or an antenna connector.

3.1.2 Basic Devices: Software Interfaces:

AT Commands: The AT command interface carries serializing seven-bit ASCII character string commands. The interface was designed to be sent over a serial link but can also be commonly used over USB. The AT command interface is the oldest and the most commonly used by M2M applications. Generally, it is very well-suited for use with smaller microcontrollers but can be also utilized for more complex implementation as well. Almost all module vendors support the AT command interface. One of its disadvantages is that it only allows one outstanding AT command at a time since it is synchronous. Moreover, raw binary data cannot be sent natively and requires hex-encoding.

SDK Interface: A SDK interface provides many function calls, methods, classes, and objects. The use of SDKs corresponds well with more powerful OS-based environments and can often be found in handheld mobile computing devices. Typically, the SDK runs on an external processor except for the case where the module supports an internal execution environment.

3.1.3 Smart objects characteristics & constraints

The emerging application area for smart objects requires scalable and interoperable communication mechanisms. IP is a stable and highly scalable communication technology that supports a wide range of applications, devices, and basic communication technologies. The IP stack is open, lightweight, versatile, ubiquitous, scalable, manageable, stable, and end-to-end. Moreover, it can run on tiny, battery-operated embedded devices. IP has the ability to make IoT a reality, connecting billions of communicating devices. A smart object is defined by IPSO [27] that includes the following:

- An intelligent (RFID) *tag*.
- A *sensor*: A device that measures a physical quantity and converts it to an analog or digital signal.
- An *actuator*: A device that controls a set of equipment.
- An *embedded device*: A purpose-built connected device that performs a specific function.
- Any *combination of the above* features to form a more complex entity.

Most IoT/M2M nodes have notable design constraints, such as the following [24]:

- Low power.
- Low cost (total device cost in single-digit dollars).
- Requirement for simple wireless communication technology. In particular, IEEE 802.15.4 standard is very promising for the lower Layers.

In close-to-market IoT applications, RFID tags and sensors are connecting inanimate objects and are building the actual things enabling the first IoT services. [24]

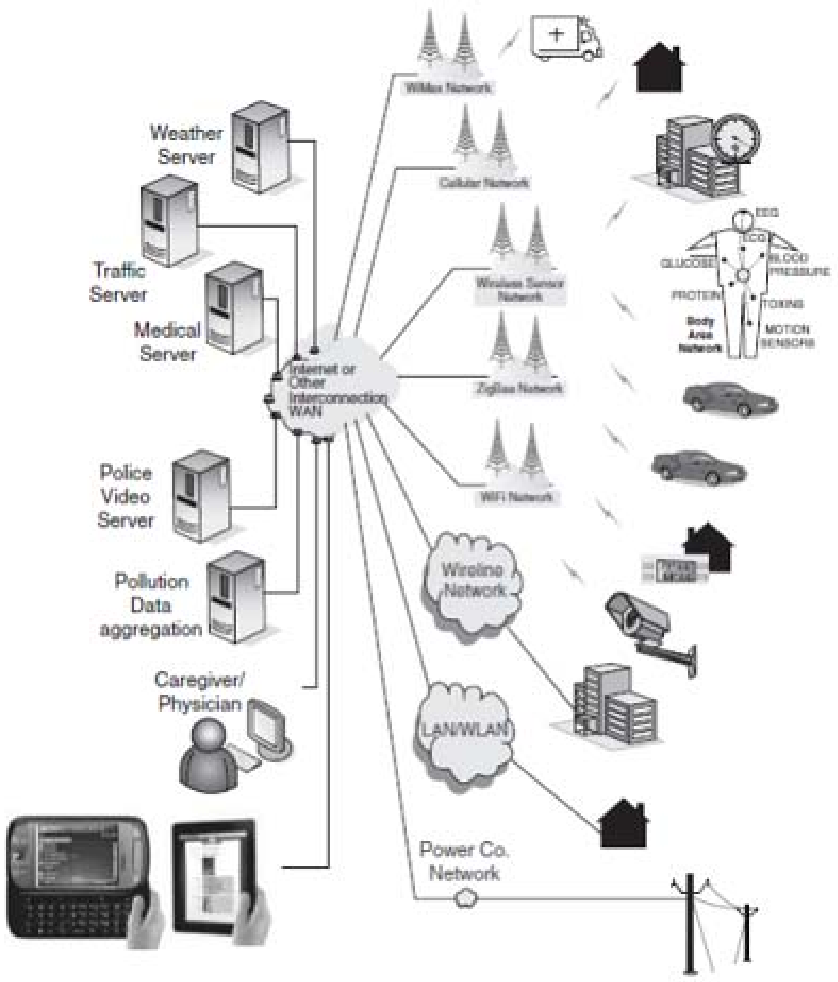


Figure 18 Illustrative example of the IoT [24]

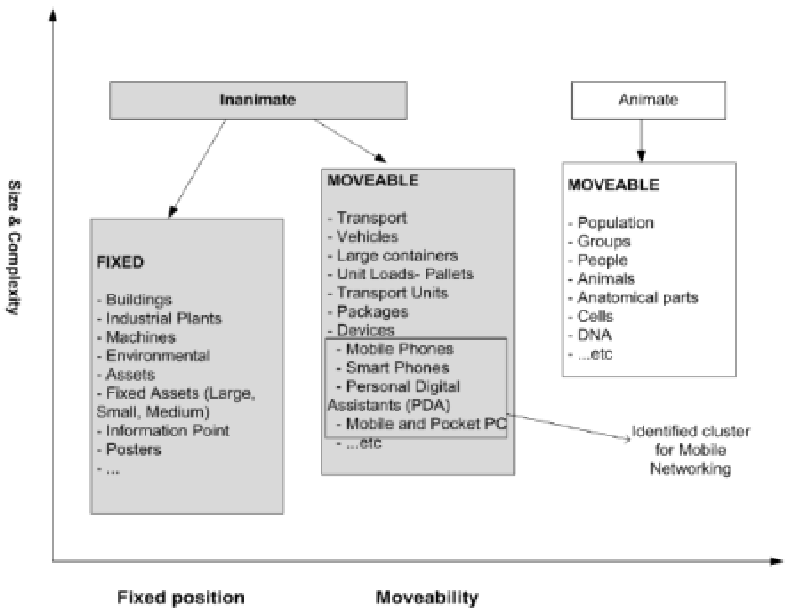


Figure 19 Objects classification [24]

Classic devices such as PCs and mobile phones are already connected objects using wired or wireless communications. IoT will extend the connectivity and interworking of these objects with new objects connected through radio sensing or identifying technologies. In particular, only small devices (such as sensors, actuators and RFID added to objects) are considered as connected things or objects. [25]

3.1.4 Technologies for Smart Objects

An expanding number of technologies will be connected to the existing and future network toward to interact with the real world to allow different applications around the user of IoT services. Other applications will affect more object-to-object communication for different types of IoT:

- Electronic *identification* technology.
- *Communication* technologies from object-to-object and from the network of objects to the existing networks.

Identification technology

Identification technology was accomplished with simple barcodes that uniquely identify items for tracking. RFID technology will identify, track the location and provide a specific IoT application to the object. It mainly answers the question *"What, which, where?"*. RFID systems consist of four main components:

- A transponder or a *tag* to carry data. Tags can be passive, semi-passive or active, based on their power source and the way they are used.
- Microwave *antenna* or coil and a *microchip data* located on the object to be identified.
- An *interrogator* or *reader*. Compared with tags, readers are larger, more expensive and power-hungry: that can be read-only, read/write or read/write/re-write, depending on how their data is encoded.
- *Middleware*, which forwards the data to another system, such as a database, a personal computer or robot control system, depending on the application.

Sensing and actuating technology

The sensor answers the question "*how*?" A sensor is an electronic device that detects senses or measures physical values. Some sensors also provide actuation functionality and they called sensors/actuators.

The interconnecting network is generally wireless and is known as WSNs. The sensing and computation nodes are considered part of the sensor network. Because of the large quantity of data collected, algorithmic methods for data management play an important role in sensor networks. There are four basic components in a sensor network [24]:

(i) An assembly of distributed or localized sensors.
(ii) An interconnecting network.
(iii) A central point of information clustering.
(iv) A set of computing resources at the central point to handle data correlation, event-trending, querying, and data mining.

Sensors can be classified according to the parameters they measure:

- Mechanical (e.g. position, force, pressure, etc.).
- Thermal (e.g. temperature, heat flow).
- Electrostatic or magnetic fields.
- Radiation intensity (e.g. electromagnetic, nuclear).
- Chemical (e.g. humidity, ion, gas concentration).
- Biological (e.g. toxicity, presence of biological organisms), etc.
- Military – enemy tracking or battlefield surveillance.

Many scientific and research groups are working to develop more efficient and feasible sensor networks. The main technical constraints are:

- Power, size, memory and storage capacity.

- Trade-off between power and size.
- Communication model.
- The environment where the sensors are deployed (underwater, land field, etc.).

The extended scope of WSN is the USN (Ubiquitous Sensor Network), a network of intelligent sensors that could one day become ubiquitous. This USN is also a unified "invisible," "pervasive," or "ambient intelligent" IoT. [25]

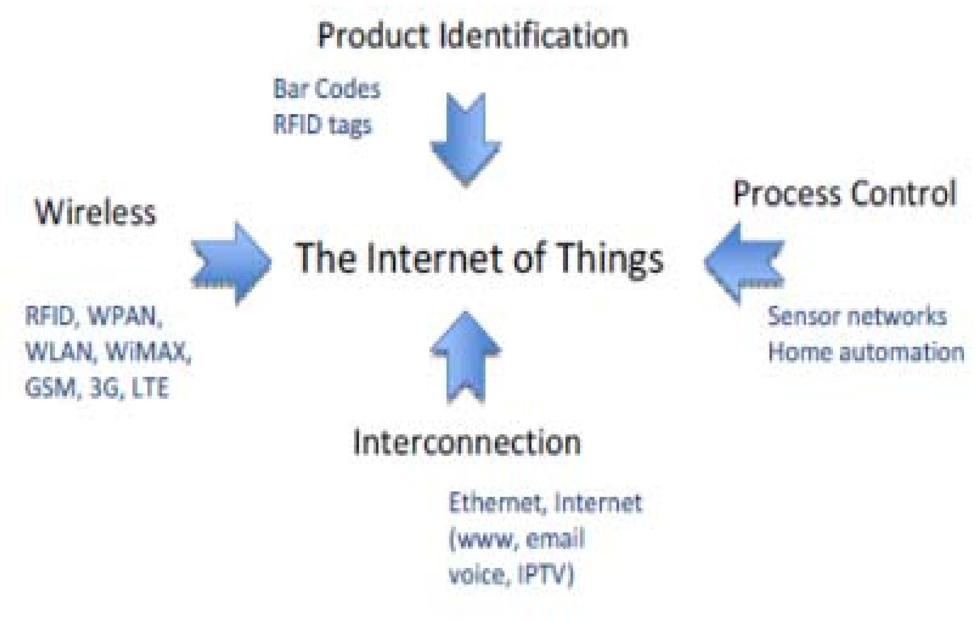


Figure 20 New participants in the IoT value chain [25]

Access Technology

There are two main types of access technology: *wireless* and *wired*.

- *Wired* access technologies require a physical connection to a cable such as a telephone line, the cable company's coaxial cable, Ethernet and xDSL.
- *Wireless* or radio access technologies do not require a physical connection and use radio waves to relay information [26]. One way to classify a wireless access technology is by the distance at which a wireless link can be maintained. These subcategories are called WPAN (Wireless Personal Area Networks), WLAN (Wireless Local Area Networks), and WWAN (Wireless Wide Area Networks)

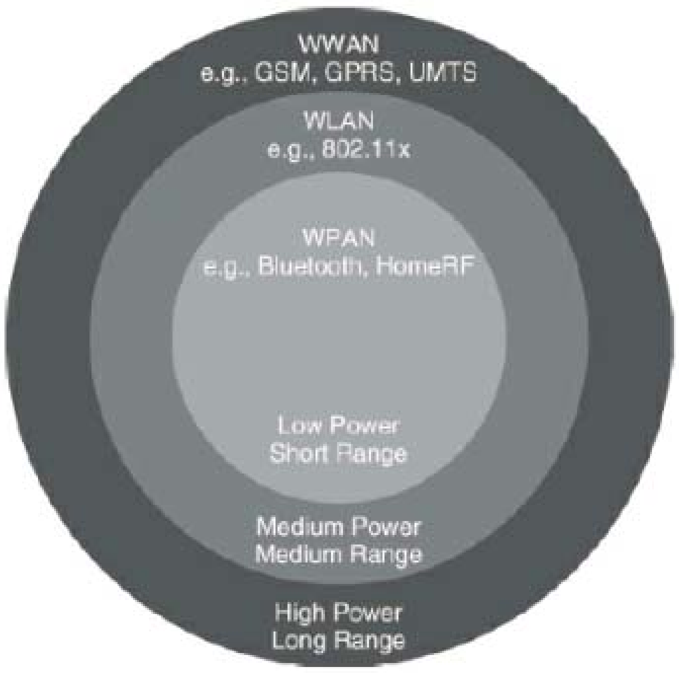


Figure 21 WPAN, WLAN and WWAN [26]

3.2 M2M – An overview

There are "six pillars" of M2M technology, representing market segments that involve networking physical assets and integrating machine data into business systems. The six pillars of M2M are as follows:

1. *Remote monitoring* is a generic term most often representing supervisory control, data acquisition, and automation of industrial assets.
2. *RFID* is a data-collection technology that uses electronic tags for storing data.
3. A *sensor network* monitors physical or environmental conditions, with sensor nodes acting cooperatively to form/maintain the network.
4. The term *smart service* refers to the process of networking equipment and monitoring it at a customer's site so that it can be maintained and serviced more effectively.
5. *Telematics* is the integration of telecommunications and informatics, but most often it refers to tracking, navigation, and entertainment applications in vehicles.
6. *Telemetry* is usually associated with industrial, medical, and wildlife-tracking applications that transmit small amounts of wireless data.

3.2.1 The Internet of Devices

The current M.Sc. Thesis considers that the term M2M is restricted to refer to device connectivity technologies, products, and services relevant to the cellular wireless networks operated by telecommunication companies.

Industry	**Example Application**	**Benefits**
Medical	Wireless medical device	Remote patient monitoring
Security	Home alarm and surveillance	Real-time remote security and surveillance
Utility	Smart metering	Energy, water and gas conservation
Manufacturing	Industrial automation	Productivity and cost savings
Automotive	Tracking vehicles	Security against theft
Transport	Traffic systems	Traffic control for efficiency
Advertising and public messaging	Billboard	Remote management of advertising displays
Kiosk	Vending	Remote machine management for efficiency and cost savings
Telematics	Fleet management	Efficiency and cost savings
Payment systems	Mobile transaction terminals	Mobile vending and efficiency
Industrial automation	Over-the-air diagnosis and upgrades	Remote device management for time savings and reduced costs

Table 3 Application areas for Cellular M2M [27]

3.2.2 ETSI M2M Architecture

The key elements of the ETSI M2M architecture are described below: [27]

- M2M device.
- M2M Area Network (MAN).
- M2M gateway.
- M2M communications networks.

- M2M application server.

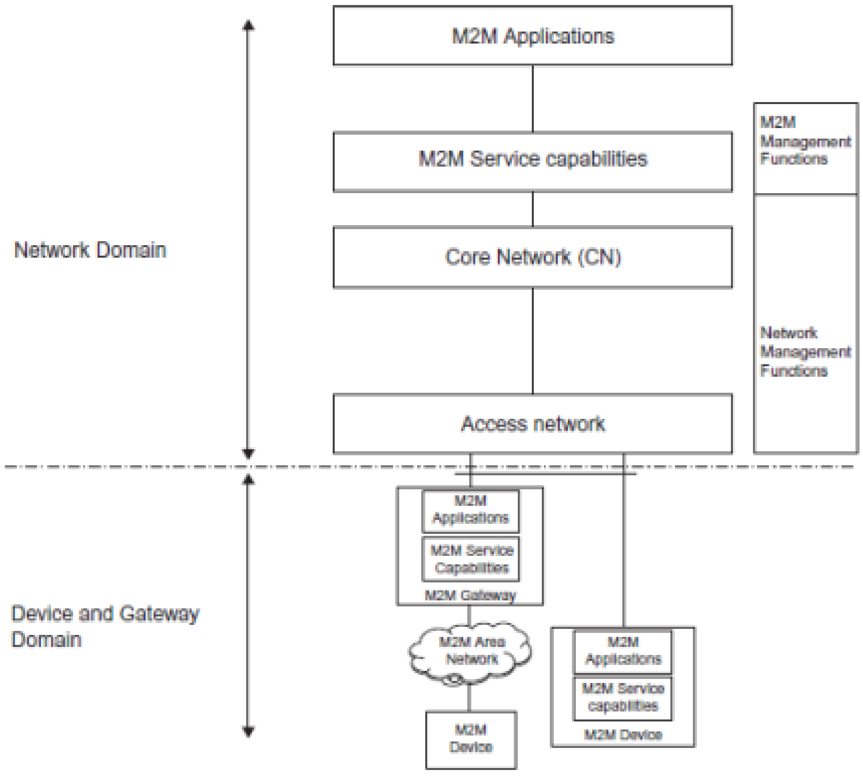


Figure 22 ETSI M2M Functional Architecture [27]

The high level architecture for M2M includes a Device and Gateway Domain and a Network domain. The **Device and Gateway Domain** is composed of the following elements:

- **M2M Device** that runs M2M Application using M2M Service Capabilities. M2M Devices connect to Network Domain in the following manners:
  - ✓ **Case 1 "Direct Connectivity":** M2M Devices connect to the Network Domain via the Access network. The M2M Device performs the procedures such as registration, authentication, authorization, management and provisioning with the Network Domain. The M2M Device may provide service to other devices connected to it that are hidden from the Network Domain.
  - ✓ **Case 2 "Gateway as a Network Proxy":** The M2M Device connects to the Network Domain via an M2M Gateway. M2M Devices connect to the M2M Gateway using the M2M Area Network. The M2M Gateway acts as a proxy for the Network Domain towards the M2M Devices that are connected to it.
- **M2M Area Network** provides connectivity between M2M Devices and M2M Gateways. Examples of M2M Area Networks include: Personal Area Network

technologies such as IEEE 802.15.1, Zigbee, Bluetooth, IETF ROLL, ISA100.11a, etc. or local networks such as PLC, M-BUS, Wireless M-BUS and KNX.

- **M2M Gateway:** A gateway that runs M2M Application(s) using M2M Service Capabilities. The Gateway acts as a proxy between M2M Devices and the Network Domain. The M2M Gateway may provide service to other devices connected to it that are hidden from the Network Domain.

The **Network Domain** is composed of the following elements:

- **Access Network:** Network which allows the M2M Device and Gateway Domain to communicate with the Core Network. Access Networks include (but are not limited to): xDSL, HFC, satellite, GERAN, UTRAN, eUTRAN, W-LAN and WiMAX.
- **Core Network** provides:
    - ✓ IP connectivity at a minimum and potentially other connectivity means.
    - ✓ Service and network control functions.
    - ✓ Interconnection (with other networks).
    - ✓ Roaming.
    - ✓ Different Core Networks offer different features sets.
    - ✓ Core Networks (CNs) include (but are not limited to) 3GPP CNs, ETSI TISPAN CN and 3GPP2 CN.
- **M2M Service Capabilities:**
    - ✓ Provide M2M functions that are to be shared by different Applications.
    - ✓ Expose functions through a set of open interfaces.
    - ✓ Use Core Network functionalities.
    - ✓ Simplify and optimize application development and deployment through hiding of network specificities.
- **M2M applications:** Applications that run the service logic and use M2M Service Capabilities accessible via an open interface.
- **Network Management Functions:** consists of all the functions required to manage the Access and Core networks: these include Provisioning, Supervision, Fault Management, etc.
- **M2M Management Functions:** consists of all the functions required to manage M2M Service Capabilities in the Network Domain. The management of the M2M Devices and Gateways uses a specific M2M Service Capability.
    - ✓ The set of M2M Management Functions include a function for M2M Service Bootstrap. This function is called MSBF (M2M Service Bootstrap Function) and

is realized within an appropriate server. The role of MSBF is to facilitate the bootstrapping of permanent M2M service layer security credentials in the M2M Device (or M2M Gateway) and the M2M Service Capabilities in the Network Domain.

- ✓ Permanent security credentials that are bootstrapped using MSBF are stored in a safe location, which is called M2M Authentication Server (MAS).

3.2.3 Key application areas

Existing M2M solutions cover numerous industry sectors and application scenarios. The largest segment is currently Telematics for cars and vehicles. Typical applications include navigation, remote vehicle diagnostics, pay-as-you-drive insurance schemes, road charging, and stolen vehicle recovery [28].

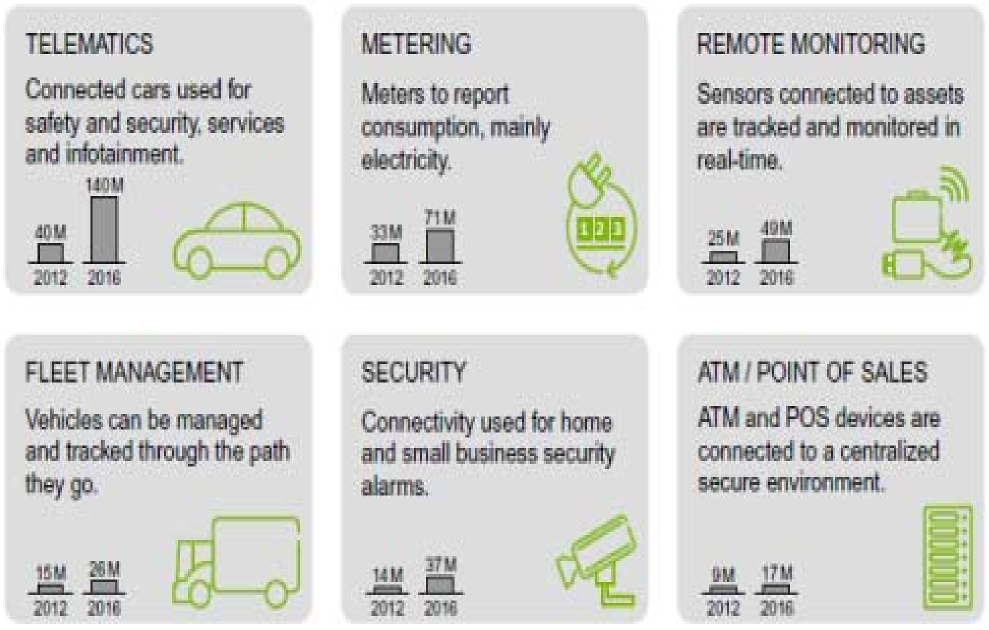


Figure 23 Summarized cellular M2M market situation [28]

Aspect	**M2M**	**IoT**
Applications and services	Point problem driven	Innovation driven
	Single application – single device	Multiple applications – multiple devices
	Communication and device centric	Information and service centric
	Asset management driven	Data and information driven
Business	Closed business operations	Open market place
	Business objective driven	Participatory community driven
	B2B	B2B, B2C
	Established value chains	Emerging ecosystems
	Consultancy and Systems Integration enabled	Open Web and as-a-Service enabled
	In-house deployment	Cloud deployment
Technology	Vertical system solution approach	Horizontal enabler approach
	Specialized device solutions	Generic commodity devices
	De facto and proprietary	Standards and open source
	Specific closed data formats and service descriptions	Open APIs and data specifications
	Closed specialized software development	Open software development
	SOA enterprise integration	Open APIs and web development

Table 4 A comparison of the main characteristics of M2M and IoT [28]

3.3 "Smartworld" components

3.3.1 PanStamp

The PanStamp project is designed for automation using small low-power wireless motes transmitting in the sub-GHz frequency band and achieving ranges of more than 200 meters. It includes the following components [29]:

- **PanStamps** are PCB boards with integrated microcontrollers and wireless communication capabilities at 868/915MHz frequency band.
- **SWAP** which is an application layer protocol for the wireless devices.

- **Lagarto** an M2M server for monitoring and controlling.
- **Base boards** are PCB boards with a battery case and integrated sensors and actuators. They are designed to connect to end-devices in order to provide them with power supply and uncomplicated connection to sensors.

The PanStamp project is aimed at creating ICT solutions for measuring and controlling things wirelessly. The central aspect of the project is the PanStamp, which is a small wireless board designed to fit in low-power applications. PanStamps are suitable for any kind of automation including energy metering, weather monitoring, home automation and robot control. The project is open-source and all parts of the code, the protocol stack and the hardware designs are released and available for download from the online repository.

PanStamps are small wireless boards designed for monitoring and controlling things wirelessly which contain [29]:

- An Atmega328p microprocessor, which makes them reprogrammable and configurable.
- An integrated CC1101 IC RF transceiver from Texas Instruments, which provides communication ranges of more than 200 meters in open spaces with transmissions at sub-GHz frequency bands.

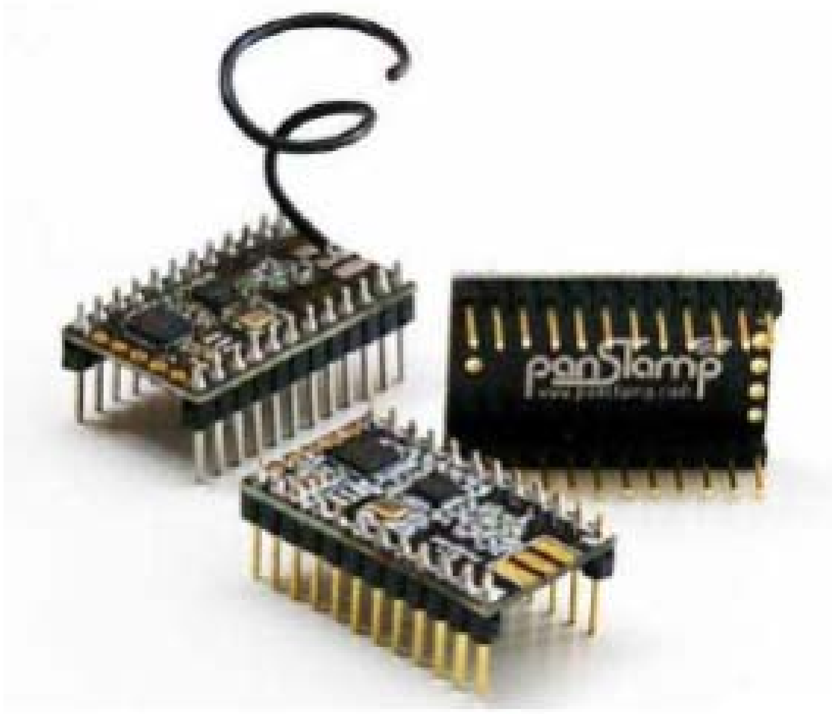


Figure 24 PanStamp modules

Dimensions	0.7 x 1.2 in (17.7 x 30.5 mm)
MCU	Atmega328p
Radio chip	CC1101 (Texas Instruments)
Bus Speed	8MHz
Flash Memory	32KB
RAM	2KB
EEPROM	1KB
Voltage range	from 2.5VDC to 3.6VDC
Rx current	24 mA max
Tx current	36 mA max
Sleep current	1 uA
Maximum Tx power	+12 dBm
RF bands	868/905/915/918 MHz ISM bands
Modulation	GFSK
Data rate	38 Kbit/s
Communication length	200m in open spaces

Table 5 PanStamp specifications

PanStamps use the CRC (Cyclic Redundancy Check) mechanism to detect errors in messages and use CCA (Clear Channel Assessment) in order to switch from Receive mode to Transmit mode to avoid collisions. Both CRC and CCA are integrated in the TI CC1101 RF transceiver. The PanStamp itself does not support collision detection. If two PanStamps begin transmitting at the same, there will be a collision. However, they do have a retransmission mechanism if CCA detects the carrier busy, in which case the PanStamp will try again three times delayed by the number of the PanStamp's ID multiplied by 2 in milliseconds. This guarantees that if two PanStamps sense the carrier simultaneously and both detect it busy, they will not try again at the same time.

PanStamps are programmed through the Arduino IDE that is an open source environment for programming microcontroller boards based on processing, another open source programming language built to teach computer programming fundamentals [30]. The Arduino IDE programming language is merely a set of C and C++ functions that are called from the code. The Software scripts written using the Arduino IDE are called sketches. Sketches are written in the text editor and are saved with the file extension .ino. The IDE also offers a serial monitor for direct bidirectional serial communication with the board.

Base Boards

Base Boards are PCB boards that can accommodate PanStamps and easily connect them to sensors, actuators and power supply. Base boards take the power from a single AA battery and can host multiple sensors, including temperature, humidity and pressure. Using the proper programming techniques, this board can run for months from a single battery, thus being suitable for building automation and environmental monitoring applications [31].

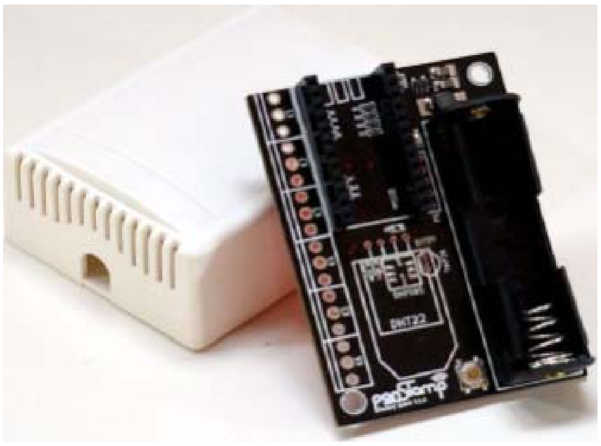


Figure 25 PanStamp battery-board [31]

Base boards generally remain simple since most of the electronics are contained in PanStamps. The core of the board is a step-up voltage regulator which transforms 0.8 - 1.5 volts supplied by the AA battery into 3.3V, a voltage accepted by most sensors nowadays. They are perfect for prototyping since most pins are available so that users can solder external sensors or mount 3.5 mm terminals for further comfort.

Lagarto Servers

Lagarto is an open source software server written in the Python programming language. It is designed to be used as an M2M server that runs on the gateway to translate end-point messages to routable form and automate physical tasks by controlling actuators. Lagarto servers provide an HTTP GET/POST interface used to control values from clients or any other application with HTTP client capabilities (Web browser). They use a common ZeroMQ Publishing socket over TCP to notify events to clients so that they do not have to continuously request updates. This dual communication mechanism (ZeroMQ + HTTP) results in a functional solution for automation. There are two types of Lagarto servers, both explained in detail in the next sections [32]:

1. **Lagarto SWAP** is a software M2M server designed to monitor and control end-device networks.
2. **Lagarto MAX** is a software server designed to communicate with multiple M2M servers over TCP using the ZeroMQ message queue.

Lagarto processes provide a web interface for configuration and basic monitoring purposes. Every process binds to a different port number, allowing the existence of multiple servers on the same computer. The HTTP port number is set manually from an XML file.

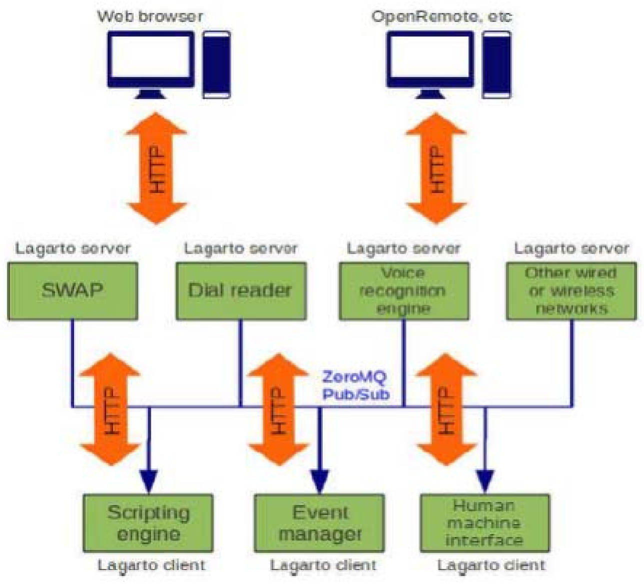


Figure 26 Lagarto architecture [32]

Simple Wireless Abstract Protocol (SWAP)

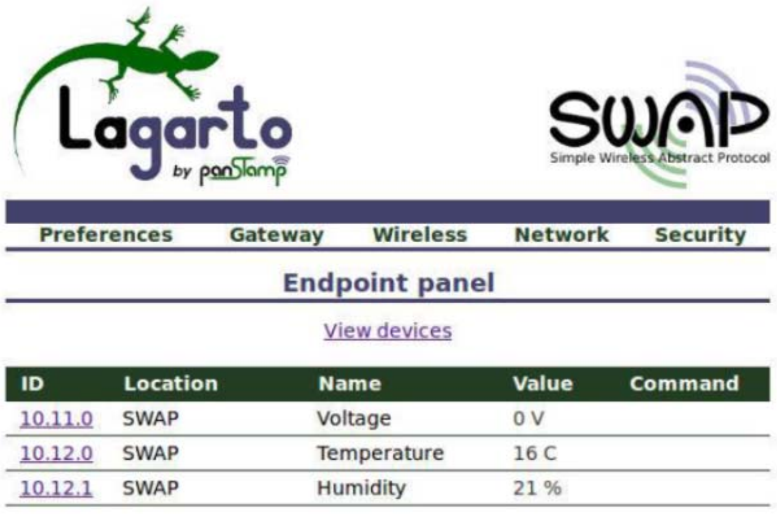
SWAP or Simple Wireless Abstract Protocol [33] is an application layer lightweight protocol for the wireless communication of PanStamps used by the Lagarto servers. SWAP is designed to be used on Texas Instruments CC11XX-based radios and it relies on their packet structure which is based on abstract registers and was inspired by Modbus. SWAP provides mechanisms for sending, requesting and controlling abstract data registers. Freeing the protocol from data types simplifies a lot the implementation. Instead, end-applications are accessing the definition folders. The definition folders are central repositories containing the registers' specifications in form of XML files. Each register used in SWAP devices is described in detail in an XML file on the Lagarto SWAP's directory and not in the OS of the end-device. This way SWAP end-devices remain lightweight and only maintain a small amount of mandatory registers containing basic information like communication channel, device address and network id.

Lagarto SWAP

Lagarto SWAP [34] is the M2M server that communicates directly to the PanStamps. This process is written in Python and relies on pySWAP, a library than contains the implementation of the SWAP protocol for Python applications. At the time of writing of the current Thesis, there are no libraries to support other programming languages.

Lagarto SWAP runs on the gateway and requires a PanStamp to be connected to the serial port and act as a modem for the network traffic. The server is bound to listen to the serial port for SWAP traffic and is able to send a command to the serial port that will be transferred to the PanStamp for wireless transmission. This way all the SWAP messages transmitted from the PanStamp network are received from the PanStamp which is connected to the gateway, sent to the serial port and captured by the Lagarto SWAP server.

The SWAP protocol defines that the first transmission of any SWAP device includes the Product ID. This is how Lagarto SWAP auto-detects new devices. When a device is auto detected from Lagarto for the first time, the server uses the product ID to find the device in the definition folders and uses the device's XML file to add its registers in the monitor interface.



ID	Location	Name	Value	Command
10.11.0	SWAP	Voltage	0 V	
10.12.0	SWAP	Temperature	16 C	
10.12.1	SWAP	Humidity	21 %	

Figure 27 Lagarto SWAP monitor interface [34]

The server is configured from a web interface that provides a simple interface for viewing and controlling SWAP values and for setting up the network with options like serial port connection speed, Broadcast TCP/IP channel and HTTP server port. The default HTTP port for the interface is 8003 and can be accessed from the computer running the server or any other computer inside the LAN. The URL for the default port is: *http:\\ip_address:8003*

Lagarto MAX

Lagarto MAX [35] is a software process that runs on the gateway and communicates with one or more M2M servers running on the same network using the ZeroMQ message queue. It is a basic component to the Lagarto architecture because it's able to collect end-device values belonging to different networks and present them to the user in a homogeneous way. Moreover, being user-programmable, Lagarto MAX can send to the end-devices automated commands based on time or sensor values.

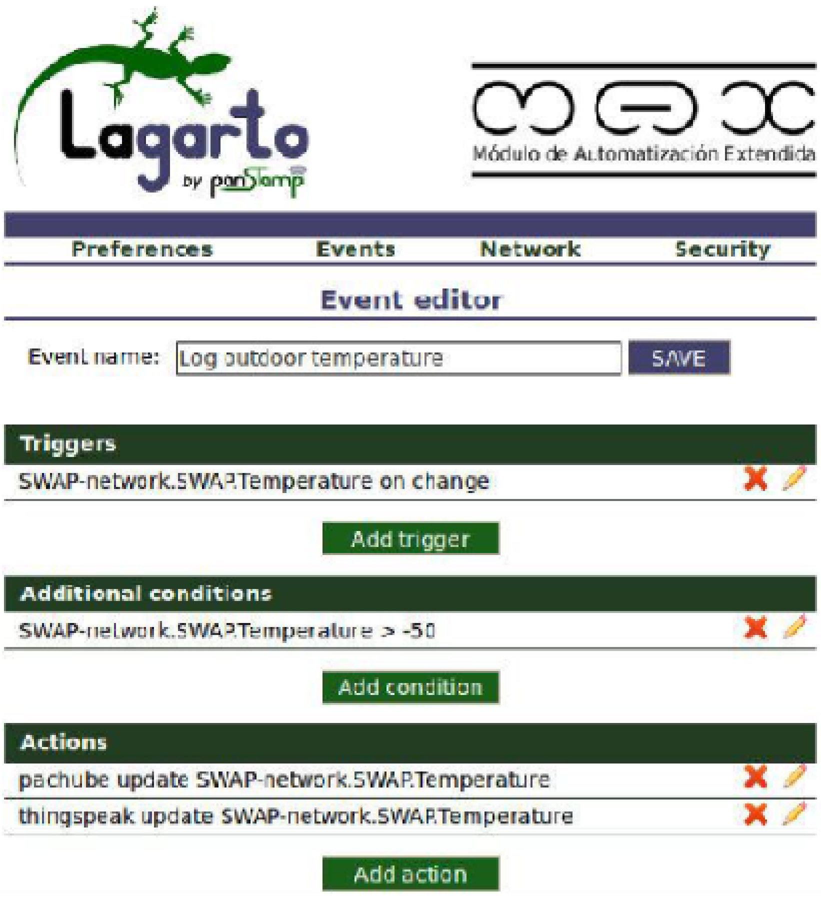
The Lagarto MAX server has a web interface for monitoring and controlling any Lagarto endpoint on the same network and it offers an event manager which can be programmed from the web interface to interact with the end-devices. The event manager can also be programmed to push end-device values to the supported virtual clouds (ThingSpeak, OpenSense, Xively). The default HTTP port for Lagarto MAX is 8002 so the URL for accessing the web interface is: *http:\\ip_address:8002*.

There are three sections in the web interface regarding programing events:

1. **Triggers** are initial conditions required to start an action. Multiple triggers can be added on a single event but only one of them has to be fulfilled in order to run the subsequent actions. There are two different types: network endpoint conditions and time conditions.

2. **Additional conditions** don't trigger the event by themselves but all of them must be fulfilled. There are two different types, network endpoint and time conditions.

3. **Actions** can be of two types: endpoint commands and pushing network values to a virtual cloud platform.



Event editor

Event name: Log outdoor temperature

Triggers

SWAP-network.SWAP.Temperature on change

Additional conditions

SWAP-network.SWAP.Temperature > -50

Actions

pachube update SWAP-network.SWAP.Temperature

thingspeak update SWAP-network.SWAP.Temperature

Figure 28 Lagarto MAX event interface [35]

More complicated events are also possible by altering and adding more code to the file where the events are programmed (webscripts.py), to the file that handles the HTTP requests (clouding.py) and to the file that hosts the network configuration (api.py). All these files are inside the Lagarto MAX directory and can also be found in the available online release of the PanStamp project. The webscripts.py file contains all the programmed events and is initially empty.

3.3.2 Zolertia Z1

Z1 by Zolertia [36] is a low-power WSN module that serves as a general purpose development platform for WSN developers, researchers, enthusiasts and hobbyists. This module has been designed from the beginning with two goals:

- Maximum backwards compatibility with the successful Tmote-like family motes while improving the performance in several aspects.
- Maximum flexibility and expandability with regards to any combination of power-supplies, sensors and connectors.

The Z1 is a low power wireless module compliant with IEEE 802.15.4 and Zigbee protocols intended to help WSN developers to test and deploy their own applications and prototypes with the best tradeoff between time of development and hardware flexibility. Some characteristics of the Z1 are:

- The Z1 mote is equipped with a second generation MSP430F2617 low power microcontroller, which features a powerful 16-bit RISC CPU @16MHz clock speed, built-in clock factory calibration, 8KB RAM and a 92KB Flash memory. It also includes the well-known CC2420 transceiver that is IEEE 802.15.4 compliant and operates at 2.4GHz with an effective data rate of 250Kbps.
- Z1 hardware selection guarantees the maximum efficiency and robustness with low energy cost.
- Z1 comes with built-in digital sensors ready to work: a digital programmable accelerometer and a digital temperature sensor (TMP102) are on the main board
- The Z1 mote does not require additional hardware to program, just an USB cable.
- You can power the Z1 mote with batteries (3.3V) or using 5V from the USB port.

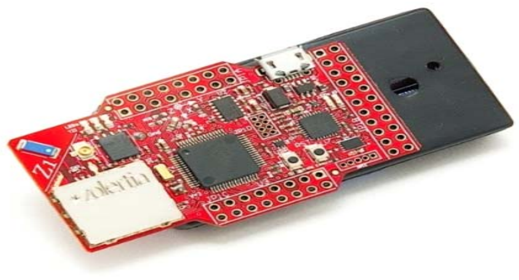


Figure 29 Zolertia Z1 mote

3.3.3 Raspberry Pi B+

The Model B+ [37] replaced the original Model B in July 2014. The RPi Model B+ [37] incorporates a number of enhancements and new features. Improved power consumption, increased connectivity and greater Input/Output are among the improvements to this powerful, small and lightweight ARM based computer.

Comparing to the Model B, it has:

- More GPIO**.** The GPIO header has grown to 40 pins, while retaining the same pinout for the first 26 pins as the Model B.
- More USB**.** We now have 4 USB 2.0 ports, compared to 2 on the Model B.
- Micro SD**.**
- Lower power consumption**.** By replacing linear regulators with switching ones we've reduced power consumption by between 0.5W and 1W.
- Better audio**.** The audio circuit incorporates a dedicated low-noise power supply.

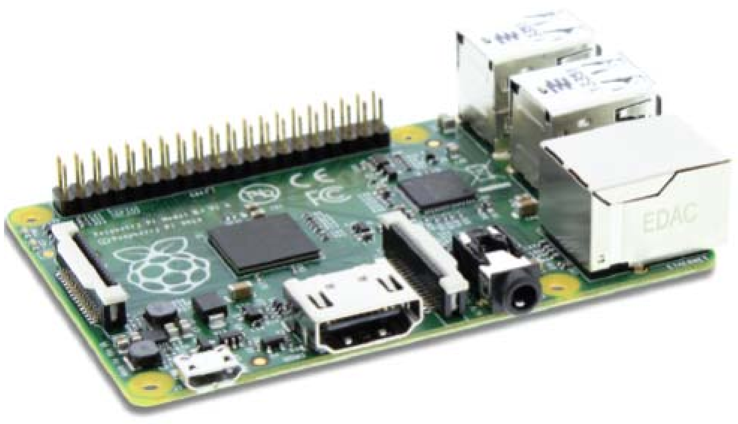


Figure 30 RPi B+ [37]

Chip	Broadcom BCM2835 SoC
Core architecture	ARM11
CPU	700 MHz Low Power ARM1176JZFS Applications Processor
Memory	512MB SDRAM
Operating System	Boots from Micro SD card, running a version of the Linux operating system
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V, 2A
Ethernet	10/100 BaseT Ethernet socket

Table 6 RPi B+ Specifications [37]

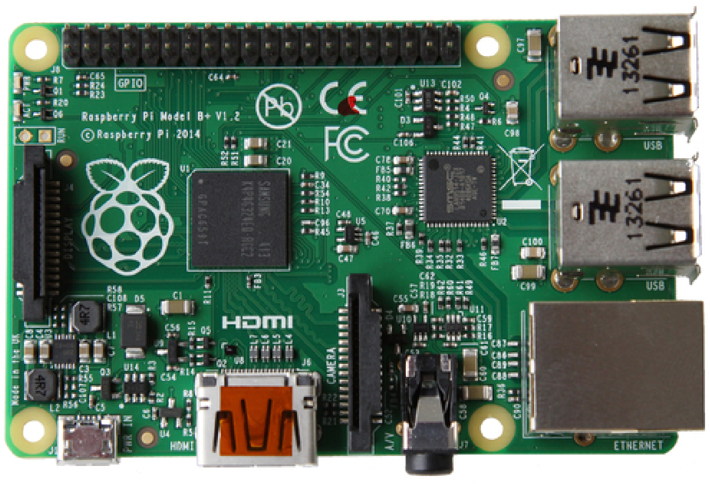


Figure 31 RPi B+ [37]

The Raspberry Pi as a Gateway

Small boards with integrated microcontrollers like PanStamps are low-power and low-cost, suitable to be deployed to a broad area. However, they are not able to produce packets that can be routed to the Internet. Thus, a regular gateway that only routes incoming and outgoing traffic is not appropriate. Constrained resource end-point networks require a gateway to receive sensor data, translate them to routable packets and then route them to the Internet. This means that the OS running on the gateway allows the implementation of servers that include network libraries and run constantly in order to translate messages to routable packets in real time.

The RPi is a very low-power computer with many capabilities that create ideal conditions to make it the gateway for end-device networks. Raspbian is the OS that runs on the RPi and it based on Debian. Also, it supports a plethora of programming languages and libraries to implement network protocols [38]. In addition, the RPi has an Ethernet interface and can also connect to Wi-Fi and cellular networks by using extra modules. For this reason, it can upload information to the Internet with various ways such as 2G/3G/LTE/Wi-Fi/Ethernet. It is extremely low power and only draws few watts of electricity. It has a huge community supporting it and there are many forums that help new developers accomplish their tasks. Another important reason that makes the RPi a great gateway is that there are many modules that have been developed to fit in RPi projects such as Camera module, GPS module, cellular Internet shields etc.

The Lagarto SWAP and the Lagarto Max servers are successfully set up on the Raspbian OS and a PanStamp shield is attached to the GPIO of the RPi. The PanStamp shield relays SWAP messages to the Lagarto SWAP server, enabling the RPi to connect to the SWAP network.

There are also some PanStamps attached to base boards. Each base board has an AA battery to supply power to the PanStamps and is connected to a temperature sensor, a humidity sensor and a LED light which operates as actuator. The SWAP protocol is implemented in the PanStamps which are programmed to send frequent updates of the sensor values to the Lagarto SWAP server and are able to receive commands at any moment for the actuators. The Lagarto Max server can create events that based on the network values can turn on/off an LED light or upload sensor measurements to a virtual cloud for online monitoring. The polling mechanism is implemented on the Lagarto MAX source code in order to add online control capabilities to the end-device network.

At this point, the end-device network is connected to a gateway (the RPi), which routes information to and from the Internet. The network is connected to a server, the Lagarto, that monitors and controls the sensor values locally and can also create automated events. Online monitor and control is also achieved through the online virtual clouds that receive values through HTTP requests and send commands using the polling mechanism. One virtual cloud is connected to the server and this is Open.Sen.se. Any computer inside the LAN can access the Lagarto servers and gain access to the end-device network through the RPi's IP and the Lagarto's port number.

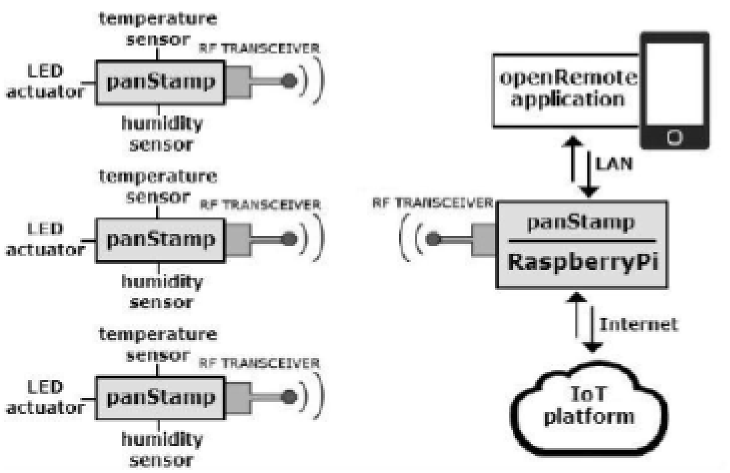


Figure 32 Implemented end-device network with Internet Connectivity [38]

3.3.4 LTE

LTE (Long Term Evolution) or the E-UTRAN (Evolved Universal Terrestrial Access Network), introduced in 3GPP Release8, is the access part of the Evolved Packet System (EPS). The main requirements for the new access network are high spectral efficiency, high peak data rates, short round trip time as well as flexibility in frequency and bandwidth [39].

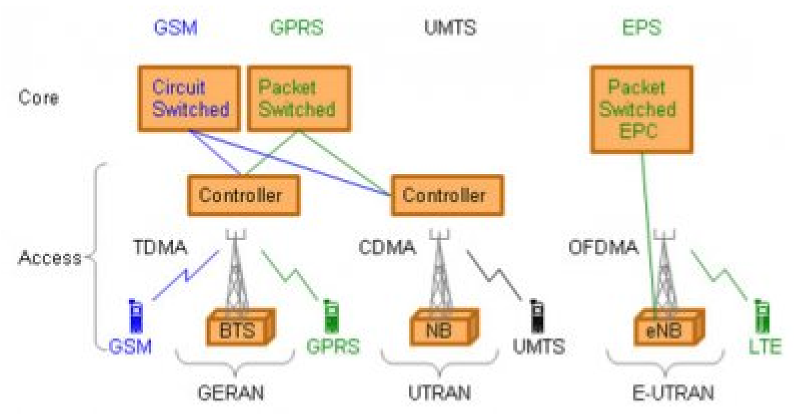


Figure 33Network Solutions from GSM to LTE [39]

The new access solution, LTE, is based on OFDMA (Orthogonal Frequency Division Multiple Access) and in combination with higher order modulation (up to 64QAM), large bandwidths (up to 20 MHz) and spatial multiplexing in the downlink (up to 4x4) high data rates can be achieved. The highest theoretical peak data rate on the transport channel is 75 Mbps in the uplink and downlink, , the rate can be as high as 300 Mbps by using spatial multiplexing. The LTE access network is simply a network of base stations, evolved NodeB (eNB), generating a flat architecture. There is no centralized intelligent controller, and the eNBs are normally inter-connected via the X2-interface and towards the core network by the S1-interface. The reason for distributing the intelligence amongst the base-stations in LTE is to speed up the connection set-up and reduce the time required for a handover. For an end-user the connection set-up time for a real time data session is in many cases crucial, especially in on-line gaming. The time for a handover is essential for real-time services where end-users tend to end calls if the handover takes too long.

Another advantage with the distributed solution is that the MAC protocol layer, which is responsible for scheduling, is represented only in the UE and in the base station leading to fast communication and decisions between the eNB and the UE. In UMTS the MAC protocol, and scheduling, is located in the controller and when HSDPA was introduced an additional MAC sub-layer, responsible for HSPA scheduling was added in the NB. The scheduler is a key component for the achievement of a fast adjusted and efficiently utilized radio resource.

3.3.5 SIGFOX

SIGFOX [40] provides an end-to-end solution for your communication chain, from your objects through to your information system, with unprecedented pricing models and low energy consumption. As a network operator SIGFOX operates fixed-location transceivers enabling your objects to be connected “out of the box”. The SIGFOX transceivers and the entire SIGFOX connectivity solution has been developed, built and deployed to only serve the low throughput M2M and IoT applications. As an operated long range network, SIGFOX provides connectivity without the need to deploy specific network infrastructures for each application. Moreover, SIGFOX does not require customers to invest in network equipment, the SIGFOX network is simply available to any object equipped with the certified connectivity solutions.

Radio technology: SIGFOX uses *a UNB (Ultra Narrow Band)* based radio technology to connect devices to its global network. The use of UNB is key to providing a scalable, high-capacity network, with very low energy consumption, while maintaining a simple and easy to rollout star-based cell infrastructure. The network operates in the globally available ISM bands (license-free frequency bands) and co-exists in these frequencies with other radio technologies, but without any risk of collisions or capacity problems. SIGFOX currently uses the most popular European ISM band on 868MHz. An important advantage provided by the use of the narrow band technology is the *flexibility it offers in terms of antenna design*. On the network infrastructure, it allows the use of small and simple antennas.

Coverage: SIGFOX is being rolled out worldwide.

SIGFOX Cloud and API: The SIGFOX Cloud provides a web application interface for device management and configuration of data integration, as well as standards based web APIs to automate the device management and implement the data integration. The APIs are based on *HTTPS REST* requests, as *GET or POST* and the payload format is *JSON*. Upon request we can also provide IPv6 proxy interfaces for the devices as well as the MQTT protocol.

The SIGFOX network is not targeting specific industries and can virtually be used in any context in which there is a need for a low throughput connectivity solution. The definition of low throughput on the SIGFOX network can characterized as follows:

- Up to 140 messages per object per day.
- Payload size for each message is 12 bytes.
- Wireless throughput up to 100 bits per second.

Integrating SIGFOX with an IT system: To receive the messages sent from the objects, the IT system needs to be integrated with the SIGFOX servers. SIGFOX provides a web application, accessible through a regular web browser, which allows you to register HTTPS addresses of the IT application that needs to receive the messages. Whenever an object sends a message, it is forwarded to the configured HTTPS address.

- *Managing objects*: SIGFOX will alert the customer's IT system whenever there is a communication problem with an object and can furthermore provide operational information such as the ambient temperature and power supply status. Through the SIGFOX web application the customer can easily configure when and where to receive this information through standard HTTPS messages.
- *Web application*: A standard web based managed service provides a clear overview of the managed objects and provides access to the history of messages next to other relevant data.
- *Radio frequencies*: The SIGFOX network operates in the unlicensed ISM radio bands. The exact frequencies can vary depending on national regulations, but in Europe the frequency is generally 868MHz and in the US it is 915MHz.
- *Uplink and downlink*: SIGFOX provides mono and bi-directional communication. The capacity to provide mono-directional communication is very unique and allows extremely low power consumption in use cases where bi-directional communication is not required.
- *Reliable connectivity*: From the very beginning SIGFOX solutions have been designed to provide high reliability for applications looking to connect large numbers of objects. Several antennas receive each message and the network backbone has a redundant and continuously monitored infrastructure to guarantee a high level of service.
- *Security and privacy*: SIGFOX employs several techniques for securing the communication in order to avoid privacy issues and other security related risks. SIGFOX does not impose specific data formats and only the customers know what they transmit and in which format.
- *Standardization*: SIGFOX is collaborating with ETSI on the standardization of low throughput networks.

3.3.6 Open.Sen.Se

Open.Sen.se [41] is an online IoT platform that receives and processes data through HTTP requests, and stores its context so that it can be retrieved at any time. The Open.Sen.se interface is based on a three-option menu which includes Sense board, Applications and Channels. The Sense board tab is the main way to visualize the received data. It is a customizable board where graphs, meters and switches can be added for monitoring the end-devices and their surroundings. The Applications tab includes various processes that can be added to the Sense board for displaying data send notifications or add computing functions. Finally, the Channels tab is where new devices, real or virtual, can be created and configured.

Open.Sen.se Features:

- Real-time data collection.
- Data processing counters, sum, average.
- Data visualizations with graphs.
- Virtual switches that represent end-device actuators.
- Email notifications.
- Social Network integration with Twitter and Facebook.

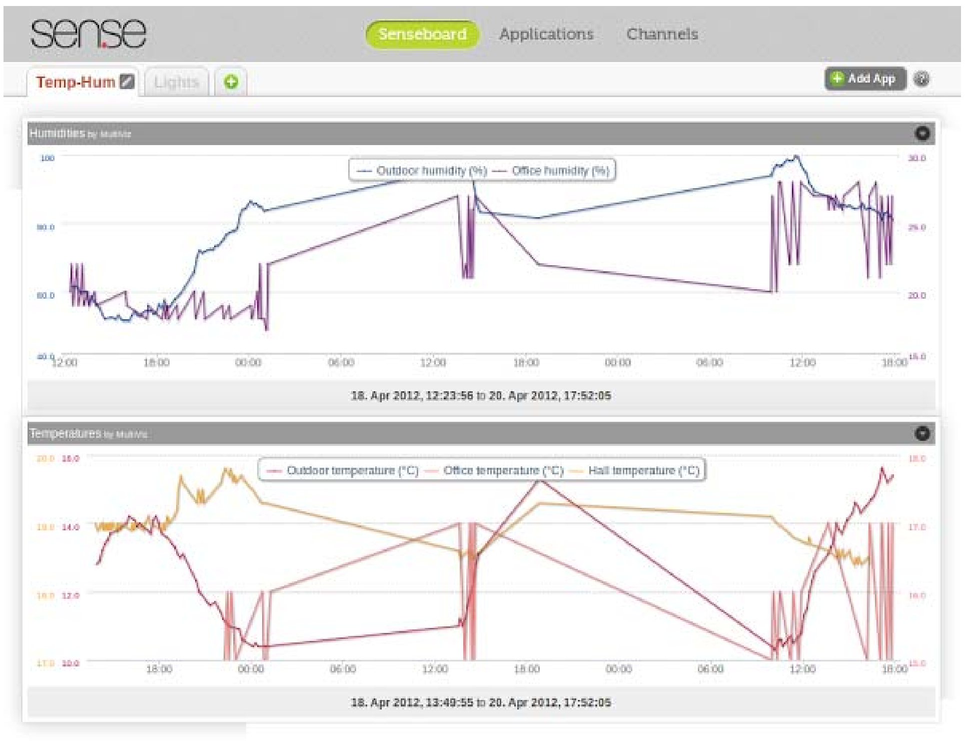


Figure 34 Open.Sen.Se Sense-board [41]

3.3.7 Android

After implementing a complete End-to-End M2M communication network, next step is the integration of representative applications of the IoT. This chapter describes Android, the targeted OS for the applications along with two smart phone applications, designed to improve the quality of life in modern urban environments.

The applications are Smart Parking and Geo Fencing. Smart Parking provides an information system regarding the status of common parking spaces. Geo Fencing is a process that allows remote automation based on the geographical position of the user. Both applications are implemented targeting the Android OS so that users can carry them in their mobile phones and use their services at any moment.

To connect an Android application to an online IoT platform and thus, to the end-device network, the application uses HTTP GET requests to extract sensor data and HTTP POST requests to send commands for the actuators. This way the mobile devices obtain full control over the sensor network and can provide user-friendly and functional applications.

4. “SmartWorld”: Development of a Heterogeneous Internet-of-Things testbed

4.1 Introduction

«SmartWorld» Testbed pretends to be a M2M testbed including different technologies and protocols (SWAP, Zigbee). Different motes with different sensors(PanStamp, Zolertia) ends up uploading data in a Cloud in the Internet, and from there being able with a smartphone application to use this data. The aim of this testbed is to support heterogeneity in Wireless Sensor Network and in the connectivity with the internet.

At the beginning of October 2014, two type of technologies are integrated: PanStamp, and Zolertia Z1 using Contiki Operative System. Vasileios Karagiannis[42] was in charge of the PanStamps and Giacomo Genovese[43] added another technology (Zolertia Z1 motes) to this initial testbed.

In my master Thesis, I reproduce the entire testbed in RPi B+ Model. Also, I connect another sensor (Ultra Sonar Sensor) in the current system and implement a procedure to support Cellular Connectivity (LTE, SIGFOX) in order to send the data to a Cloud Application.

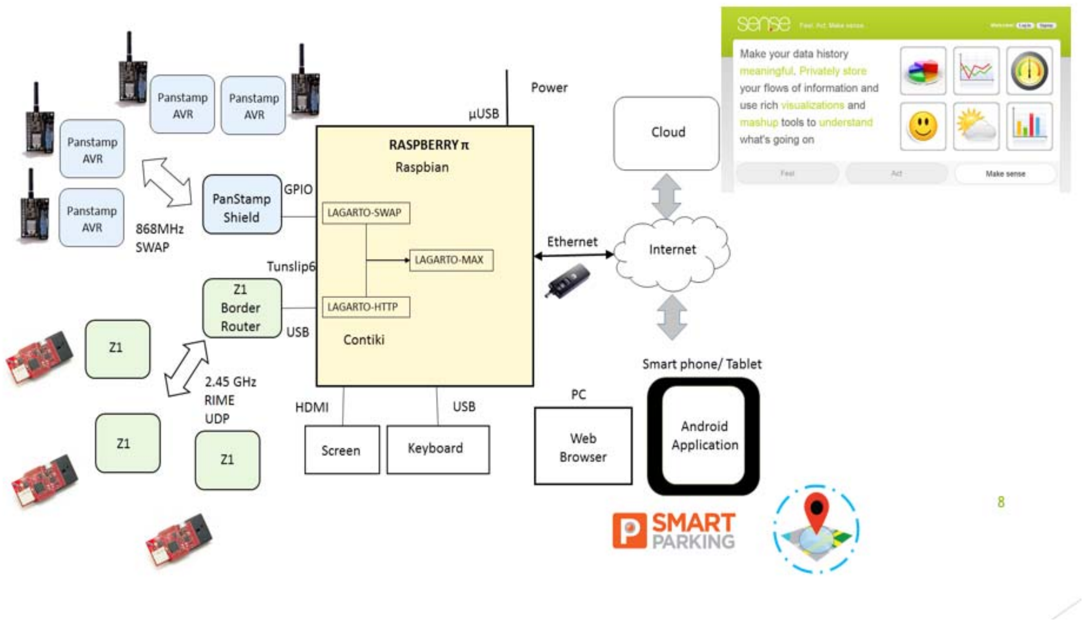


Figure 35 «SmartWorld» testbed – scheme

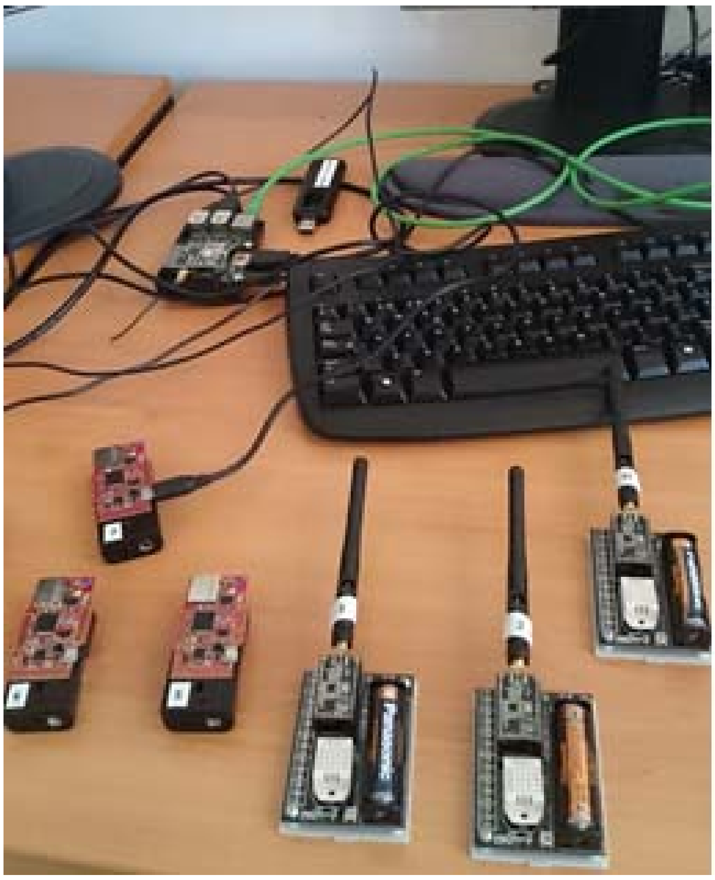


Figure 36 «SmartWorld» testbed

4.2 Wireless Sensor Network

4.2.1 PanStamp

PanStamp is an open source project created for the enthusiasts that love measuring and controlling things wirelessly. PanStamps are small wireless boards specially designed to fit in low-power applications, simple to program and simple to work with. With PanStamps, you can measure almost everything by simply connecting your PanStamp to the sensors, placing a battery and sending wireless data from the first moment. The core of the PanStamp project is PanStamps. These small wireless boards are extremely programmable and configurable. They can be plugged into any of the available base boards containing sensors and actuators, or new boards can be developed according to your needs. Base boards generally remain quite simple since most of the electronics are contained in PanStamps. The boards run a compact stack and communicate with each other using a very simple protocol called SWAP. PanStamp also offers sensors and battery modules where the Wireless PanStamp module has a socket to be connected.

Panstick is a USB mother board for PanStamps. Used to program PanStamps, it also acts as a serial gateway to the wireless network. Simply place a PanStamp on the Panstick, program the PanStamp with the modem application, and plug the dongle to your computer.

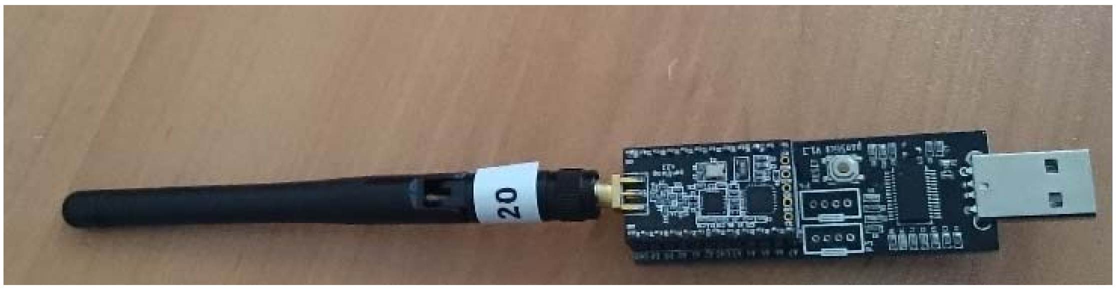


Figure 37 «SmartWorld» - Panstick

Panstick is also a compact development board. Thanks to the available contacts, you can solder pin headers under the board and plug the Panstick and PanStamp dongle on a breadboard, having access to PanStamps pins for your developments. You will then be able to connect your sensors directly to your Panstick whilst keeping your PanStamp accessible via USB. Panstick includes an on-board voltage regulator which takes the power directly from the USB bus. Thanks to this regulator, Panstick is able to power your sensors up to 250 mA always at 3.3 VDC. The USB board also includes a reset button that will let you restart your application without having to unplug the board.

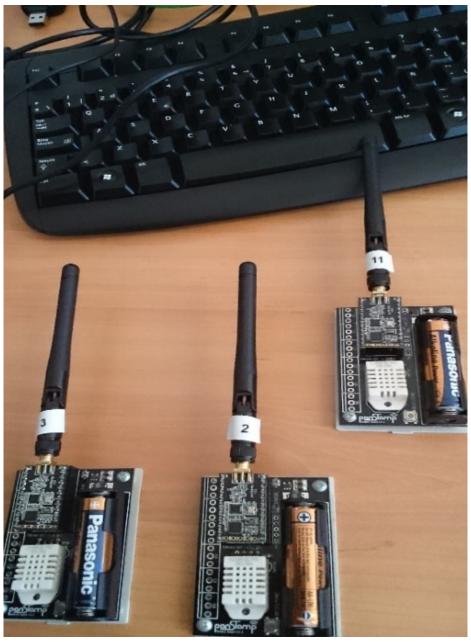


Figure 38 «SmartWorld» - PanStamps

In the Figure 40, we can observe 3 PanStamp Boards with the PanStamp Microcontrollers. PanStamps can be programmed using the Arduino IDE. How to upload your Arduino code in your PanStamps:

1. Download Arduino 1.5
2. Install Arduino 1.5
3. Download PanStamp-Arduino files
4. Unzip PanStamp Arduino files
5. Move the folder "arduino-1.5_patch" to the directory where Arduino-1.5 is installed.
6. Open Command Prompt (in Windows) and go to "Arduino-1.5_patch" folder
7. Run: install.bat in order to Enable PanStamp or you can make right click to file "install.bat" and executed as administrator.
8. After this make a reboot to your pc.
9. Connect the PanStamp to the Panstick.
10. Insert the Panstick to a USB port in your pc.
11. Now, you go to "Arduino-1.5_Patch/libraries/" and copy all the folder. Paste them to the Arduino/libraries folder (in Program Files/Arduino/libraries).
12. Open "Smartparking_prem.ino" file.
13. Tools -> Boards -> Arduino pro or pro mini
14. Tools -> processor -> ATmega328(3.3, 8Hz)

15. Port -> COM4 (go to devices and check in which port is configured the Panstick. Usually, is at COM3. But you have to change it, to "COM4").

16. Click on "Verify/Compile"

17. Click on "Upload"

18. You can disconnect Panstick. Now your PanStamp is programmed.

The following schema shows a system overview:

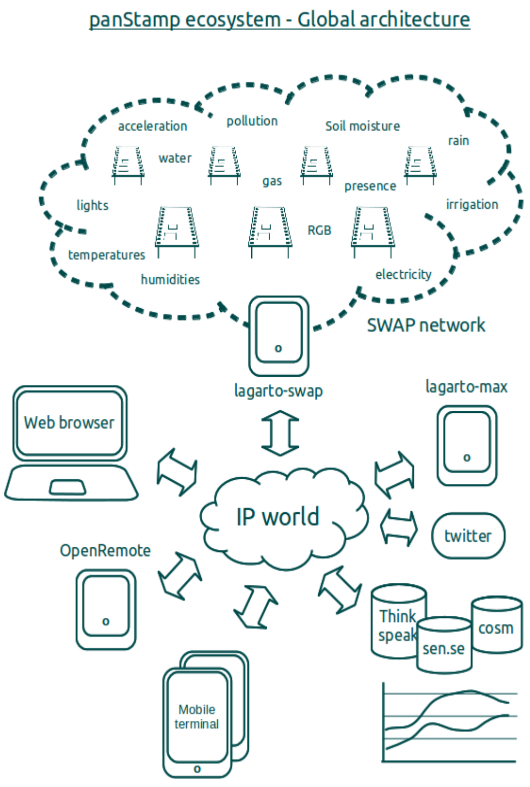


Figure 39 PanStamp ecosystem - Global architecture [32]

Lagarto is written using Python and has an open architecture. New Lagarto servers and clients can be developed using any language. Moreover, Lagarto servers can be queried using simple HTTP GET/POST commands so communicating with existing applications should not be a problem. As result, we can say that your ideas can be developed on both sides of the above diagram: the low-power wireless network and the IP side. ***Lagarto*** *is* an open platform designed to automate physical tasks in homes, buildings and industrial plants.

4.2.2 Zolertia Z1

Z1 mote from Zolertia is a low power sensor mote, based on Texas Instruments MSP430F2617 microcontroller and CC2420 radio, which is a 2.45GHz transceiver. It provides support to two open sources operating systems /stack to wireless sensors networks, Contiki, the one it will be used in our case, and TinyOS. The board includes a temperature sensor and a 3 axis accelerometer. It also senses the battery level, and includes 3 LEDs. It is compatible with IEEE 802.15.4, 6LowPan and Zigbee. Z1 can be connected to internet with IPv6.

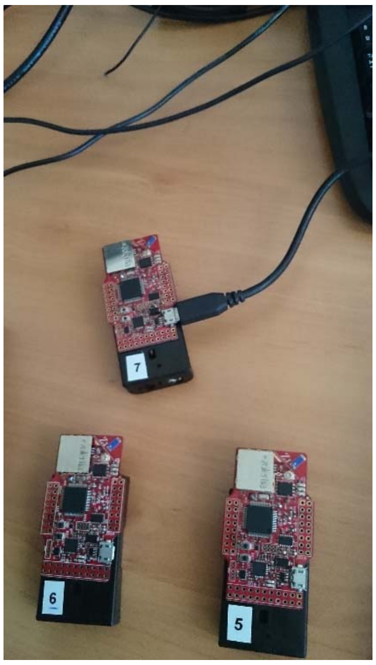


Figure 40 «SmartWorld» - Zolertia Z1 motes

Z1 motes that are used now in the testbed are powered with two AA batteries. In order to power and un-power it is needed to put and remove at least one battery. When connecting the battery, if there is good power, the red led it is turn on for a second.

Contiki is an open source Operating System for IoT, provides powerful low power Internet communications. Contiki provides a full IP network stack, with standard IP protocols such as UDP, TCP, and HTTP, in addition to the new low-power standards like 6lowpan, RPL, and CoAP. The Contiki IPv4 and IPv6 stack, is fully certified.

4.2.2.1 Working environment - Download and install Contiki

Contiki should be installed both in your development PC (Ubuntu) and in RPi, where the Z1 border router (mote 7) will be connected. In this link you can find the Contiki library, it is important to download the newest one.

https://github.com/contiki-os/contiki/wiki/Setup-contiki-toolchain-in-ubuntu-13.04

(1) Install the following packages:

sudo apt-get install build-essential binutils-msp430 gcc-msp430 msp430-libc binutils-avr gcc- avr gdb-avr avr-libc avrdude openjdk-7-jdk openjdk-7-jre ant libncurses5-dev doxygen git

sudo pip install requests (extra package)

(2) Clone the Contiki sources from Github:

git clone git://github.com/contiki-os/contiki.git contiki

(3) The same should be installed in RPi.

https://github.com/contiki-os/contiki/wiki/Setup-contiki-toolchain-in-ubuntu-13.04

Contiki can be implemented in a range of motes of different vendors. For this reason the first time we enter in a folder where the code is, we want to compile and upload in our mote (json-ws or ipv6 in our case).

4.2.2.2 Select the right mote target

The following command should be executed to select the right mote target.

>> cd contiki/examples/z1

>> make TARGET=z1 savetarget

When you write 'make' on the window, you call the make file inside the folder that you are using for your code. Each Contiki folder has its make and configuration file.

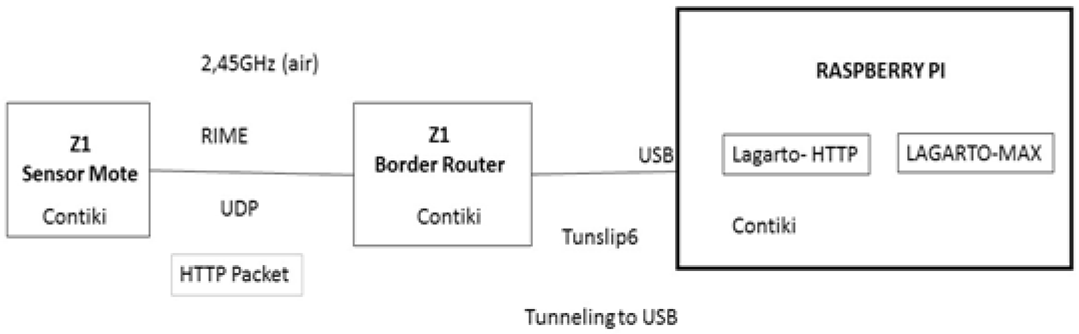


Figure 41 Tunneling to USB

4.2.2.3 Contiki code for Zolertia Z1 motes in Smart World Testbed

Folder “mycontiki” is the Contiki folder in RPi B+, where the code for *the border router* and the *Contiki mote* can be found. At this moment three motes are on use and with the appropriate code uploaded. They are designed with numbers and are motes 7, 6, 3 and 5. Contiki includes a wide range of libraries. Some of the most important libraries that we are going to use are: json, http, slip.

Border Router:

Z1 mote number 7 is the border router/modem, which is connected to RPi via USB-microUSB. It should be power on with the two AA batteries to work.

Border router code (Mote 7) is in the following path:

“Mycontiki/examples/ipv6/rpl-border-router/border-router.c”

Rpi_border-router.c code is implemented to connect Raspberry with the contiki WSN.

To compile and upload code to the border router: (connect via USB to the PC)

```
>> make border-router.upload
```

*Change tab to the Terminal to continue

Some basic explanations of the code files:

Border_router.c includes dev/slip.h library ,which includes tunslip, which is what is used to tunneling the USB, and get HTTP packets at the RPi.

Tunslip6 does a tunneling in the USB for IPv6, in such a way that the Raspberry sees the USB port as an Ethernet port. The process is similar to the Ethernet one but it’s not the same, Raspberry will see a new virtual network interface “*Tunslip creates a virtual network interface (tun) on the host side and uses SLIP (serial line internet protocol) to encapsulate and pass IP traffic to and from the other side of the serial line. The network element sitting on the other side of the line does a similar job with it’s network interface*”. This tunneling should be activated in the RPi executing two commands:

```
>> cd contiki /examples/ipv6/rpl-border-router
>> make z1 –reset && make connect –router&
```

Sensor motes:

Z1 number 6, 3 and 5, have all the same code (at this moment they read temperature sensor) but with different identifiers. The ID of the mote is the same as the number which is marked on it.

Z1 at this moment has 3 possible options of things to do:

(1) Led on or off,

(2) Temperature monitoring,

(3) Battery level monitoring or an accelerometer. Code should be added to read the battery level or the accelerometer, examples can be found.

Mote code can be founded in this path:

"mycontiki/…/json_ws/mywebsense2.c"

To compile and upload code in the mote should be executed the command:

```
>> make mywebsense2.upload WITH_UDP=1
```

This means that we are using UDP.

Some basic explanations of the code files:

- *mywebsense2.c:* is the code that formats data the way is needed.
- *json_ws.c:* it calls json_ws
  - *HOST*: address was mote's data is send. It was chosen because it's the address of the virtual network Interface created by the border router and we set our Lagarto-HTTP to listen to that address. At this moment is aaaa::1
  - *SEND INTERVAL*: now value is 120, in seconds, each 2 minutes. It can be modified. It's set each 30 second, you can find that value inside json-ws.c

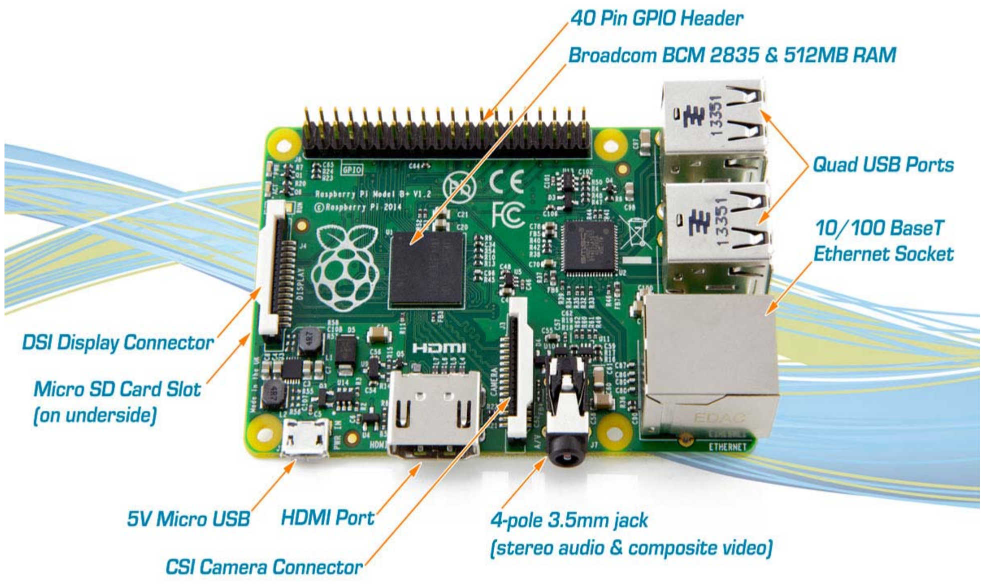
4.2.3 Ultra-Sonic Sensor

The LV-MaxSonar-EZ1 [44] is a great choice for use where sensitivity is needed along with side object rejection. The LV-MaxSonar-EZ products were the first low cost, high quality ultrasonic distance sensors to offer easy to use outputs, no sensor dead zone, calibrated beam patterns, stable range readings, low power demands, and a host of other features. The LV-MaxSonar-EZ1 is the most popular indoor ultrasonic sensor and is a good, low-cost sensor to use. I connect this sensor in the PanStamp Board in order to perceive the existence of an obstacle in the sensor area. To do this, I reprogram the PanStamp microcontroller in Arduino IDE. We measure if there is an obstacle in 41cm or not. The code to do this is the following:

```
byte i=0;
byte stateLowByte,laststate = 0 ;
byte
coordinates0,coordinates1,coordinates2,coordinates3,coordinates4,coordinates5,coordinates
6,coordinates7;
//digital 9
uint8_t binaryPinI[] = {1};//, 0, 1, 2, 3, 4, 5};     // Binary pins (Atmega port bits)
volatile uint8_t *binaryPort[] = {&PINB,};
//, &PINC, &PINC, &PINC, &PINC, &PINC, &PINC};   // Binary ports (Atmega port) //it`s
the PINB in the atmega pins
 // Initial pin states
uint8_t binaryPinO[] = {3, 4};//{8, 14, 15, 16, 17, 18, 19, 7}; //analog 3 = PC3 (Arduino
digital pins)
const int sensorPin=3; //arduino Digital 9 pin is connected to MAXSensor
long value=0;
int cm=0;
int inches=0;
...
void loop()
{
 pinMode(sensorPin,INPUT); //MAXSonar - input - digital 9
 value = pulseIn(sensorPin,HIGH); //MAXSonar value is HIGH (like off)
 inches = value/147; //convert the value in inches
 cm = inches * 2.54;  //convert the inches to cm
 if (cm < 41) //check if the cm is less than 41cm the value will be ON - SensorIN=false
 {
  pulseIn(sensorPin,LOW);
 }
 else
 {
  pulseIn(sensorPin,HIGH); // if the cm is more than 41 the value will be OFF -SensorIN=true
 }...
```

Figure 42 LV-MaxSonar®-EZ1™ High Performance Ultrasonic Rangefinder [44]

4.3 Gateway: Raspberry Pi B+



40 Pin GPIO Header
Broadcom BCM 2835 & 512MB RAM
Quad USB Ports
10/100 BaseT Ethernet Socket
DSI Display Connector
Micro SD Card Slot (on underside)
5V Micro USB
HDMI Port
CSI Camera Connector
4-pole 3.5mm jack (stereo audio & composite video)

Figure 43 RPi B+ [37]

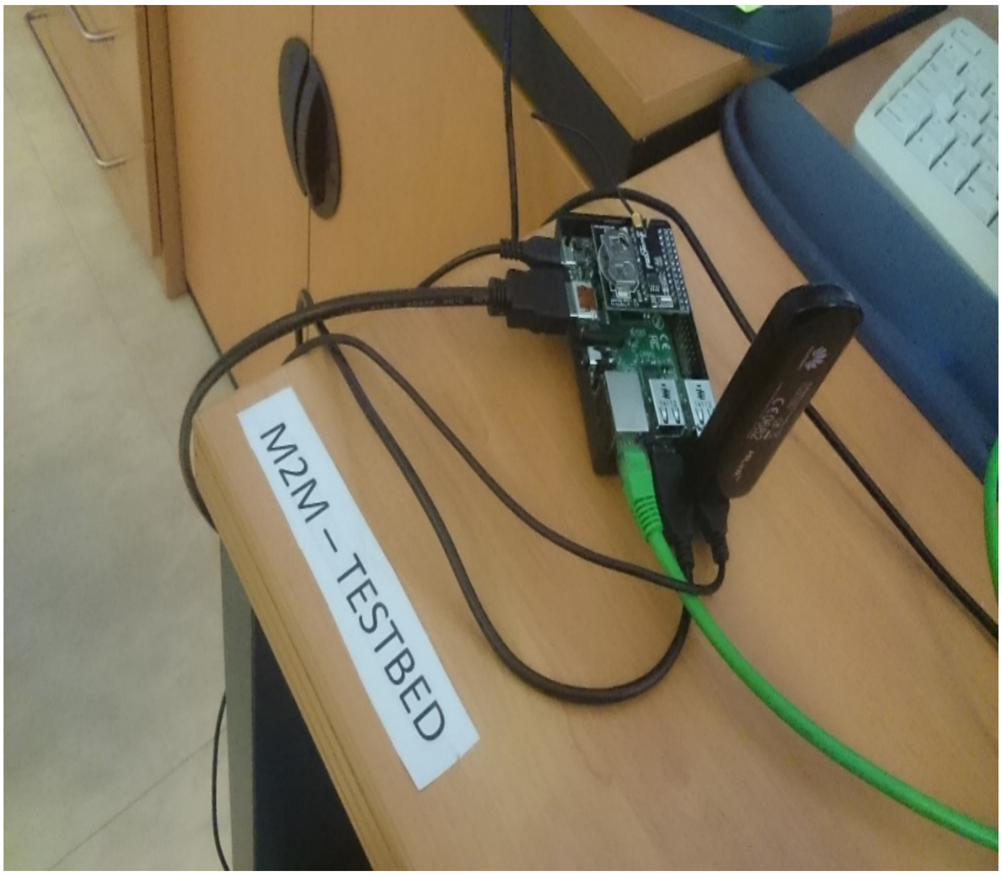


Figure 44 «SmartWorld» - RPi B+

4.3.1 Raspberry Pi B+ in «SmartWorld» – Installation steps

It would be useful to have these things, in order to start the procedure:

- Ubuntu, as Operating System.
- PC with SD card reader.
- PC with access to the Internet (recommended Ethernet)

4.3.2 Install Noobs and OS to Rarspberry Pi

It will be useful to install NOOBS in the SD Card. With NOOBS is easy to install or restore the Raspbian (OS for RPi). Advanced users may wish to install a specific Operating System image. Download an image below and follow the image installation guides in our documentation.

4.3.3 Login to RPi

Username: pi
Password: raspberry
Load GUI: startx -> press Enter

4.3.4 Install Lagarto Servers

(1) Pyswap & Python Applications.

Download and extract the pyswap and rest of python applications for PanStamp.

(2) Pyserial

Download Pyserial (In case you have Raspbian Wheezy, you should download version 2.7).

```
wget "https://pypi.python.org/packages/source/p/pyserial/pyserial-2.7.tar.gz"
tar xvfz pyserial*.gz
cd pyserial*
sudo python setup.py install
```

(3) ZeroMQ

There are two ways to install ZeroMQ:

(a)
```
Sudo apt-get install libzmq-dev
*install libzmq-dev for series 2.2 (because we have Wheezy)
```

(b)
```
apt-get install uuid-dev libtool autoconf automake pkg-config build-essential
wget "http://download.zeromq.org/zeromq-2.2.0.tar.gz"
tar xvfz zeromq*.gz
cd zeromq*
./configure
sudo make
sudo make install
ldconfig
```

(4) Python Bindings

```
Sudo apt-get install libpgm-5.1-0
Sudo apt-get check
apt-get install python-dev
wget "https://pypi.python.org/packages/source/p/pyzmq/pyzmq-13.0.2.tar.gz"
tar xvfz pyzmq*.gz
cd pyzmq*
sudo python setup.py configure
sudo python setup.py install
```

(5) Barrel

```
wget https://pypi.python.org/packages/source/b/barrel/barrel-0.1.3.tar.gz
tar xvfz barrel*.gz
cd barrel*
sudo python setup.py install
```

(6) pyephem

```
wget "https://pypi.python.org/packages/source/p/pyephem/pyephem-3.7.5.1.tar.gz"
tar xvfz pyephem*.gz
cd pyephem*
sudo python setup.py install
```

4.3.5 Network configuration and DNS configuration

Change the Network Configuration

Sudo nano /etc/network/interfaces

```
Auto eth0
Iface eth0 inet static
Address 10.1.3.53
Gateway 10.1.3.1
Netmask 255.255.255.0
Network 10.1.3.0
Broadcast 10.1.3.255
```

Change DNS configuration

Sudo nano /etc/resolf.conf

```
Nameserver 84.88.62.194
Nameserver 84.88.62.220
```

Save and exit. (ctrl + C , Yes to Save)

Reboot your RPi

4.3.6 Activate the Sensors & de-activate Updates of Devices

Insert the batteries to the PanStamp sensors and locate the PanStamps somewhere close to the RPi. (RPi must be off when you do this). If the servers start for first time push “RESET/SYNC” button.

Declare the Devices you have:

lagarto/lagarto-swap/config/devices/devices.xml

<developer id="2" name="developerName">

<dev id="1" name="temphum12" label="TempHumVolt2Dinput"/>

<dev id="2" name="temphum13" label="Temp13"/>

</developer>

De-activate the "Update" of devices
lagarto/lagarto-swap/config/settings.xml
Change the "*Update*" to "*False*"

4.3.7 Access and Test the Servers

Note that, we must run first Lagarto-MAX and then Lagarto-SWAP.

4.3.7.1 Lagarto-MAX

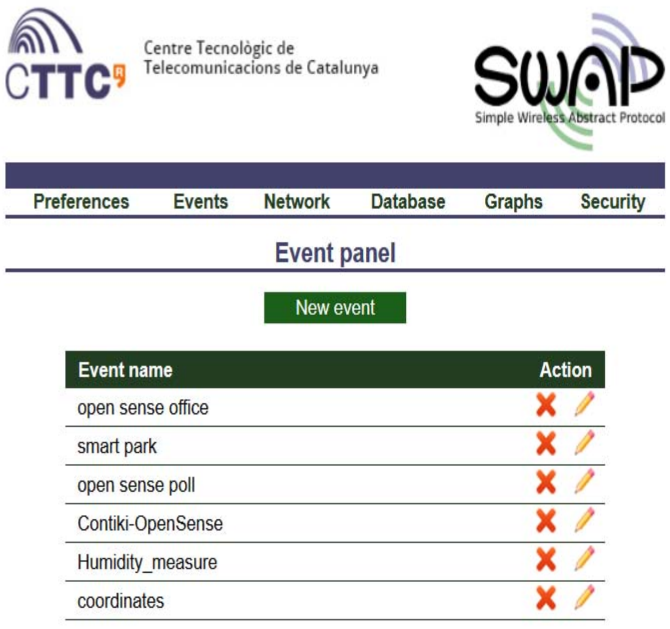
Lagarto process

- Receives, resends and processes events transmitted from any Lagarto server running on the same IP network.
- Able to integrate real values belonging to different networks and present them to the user in a homogeneous way.
- Being user-programmable: can take decisions based on time or network events.

Start the server:

sudo python ~/PanStamp/lagarto/lagarto-max/lagarto-max.py

To access the servers, run a browser on the same computer or on another one in the local network and type the following URL: http://local_IP_address:8002

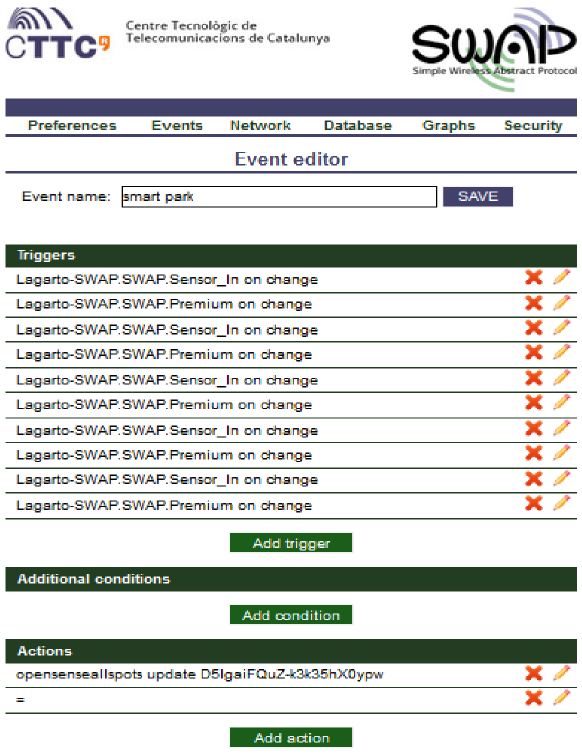

Centre Tecnològic de Telecomunicacions de Catalunya
CTTC
SWAP
Simple Wireless Abstract Protocol
Preferences
Events
Network
Database
Graphs
Security
New event

Event panel

Event name	Action
open sense office	
smart park	
open sense poll	
Contiki-OpenSense	
Humidity_measure	
coordinates	

Figure 45 LAGARTO-MAX

From the above Events we worked with “Smart Park” and “open sense poll” in order to send the data from the PanStamp network to the cloud Service (Open.Sen.Se). The first one is connected with “SmartParking” application and the second one with “Geofencing” Application.



Event editor

Event name: smart park

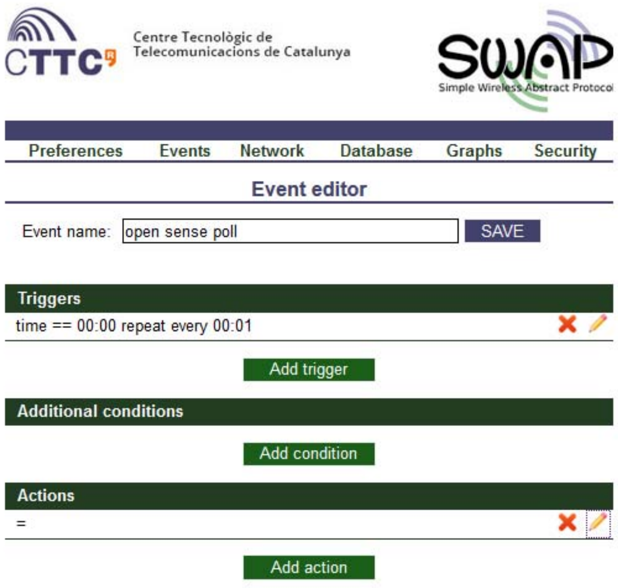
Triggers
Lagarto-SWAP.SWAP.Sensor_In on change
Lagarto-SWAP.SWAP.Premium on change
Lagarto-SWAP.SWAP.Sensor_In on change
Lagarto-SWAP.SWAP.Premium on change
Lagarto-SWAP.SWAP.Sensor_In on change
Lagarto-SWAP.SWAP.Premium on change
Lagarto-SWAP.SWAP.Sensor_In on change
Lagarto-SWAP.SWAP.Premium on change
Lagarto-SWAP.SWAP.Sensor_In on change
Lagarto-SWAP.SWAP.Premium on change

Additional conditions

Actions
opensenseallspots update D5lgaiFQuZ-k3k35hX0ypw
=

Figure 46 Lagarto-MAX "Smart Park" Event

In the figure we can observe that we have configured a Trigger in order to send data to Open.Sen.Se service. If the "Premium" or "Sensor_in" values change then an Action starts. The Action sends the data to the Open.Sen.Se via "opensenseallspots" method (which is a method in the Lagarto-MAX source code).


Centre Tecnològic de Telecomunicacions de Catalunya
SWAP Simple Wireless Abstract Protocol
Preferences
Events
Network
Database
Graphs
Security
SAVE
Add trigger
Add condition
Add action

Event editor

Event name: open sense poll

Triggers

time == 00:00 repeat every 00:01

Additional conditions

Actions

=

Figure 47 Lagarto-MAX "open sense poll" Event

In the Figure 49 we have configure the Event "Open sense poll". It has one Trigger with timer. It sends data to the Open.Sen.Se service every second. There is an Action, too. But we can not see it in the Browser, because it is in source code of Lagarto-MAX.

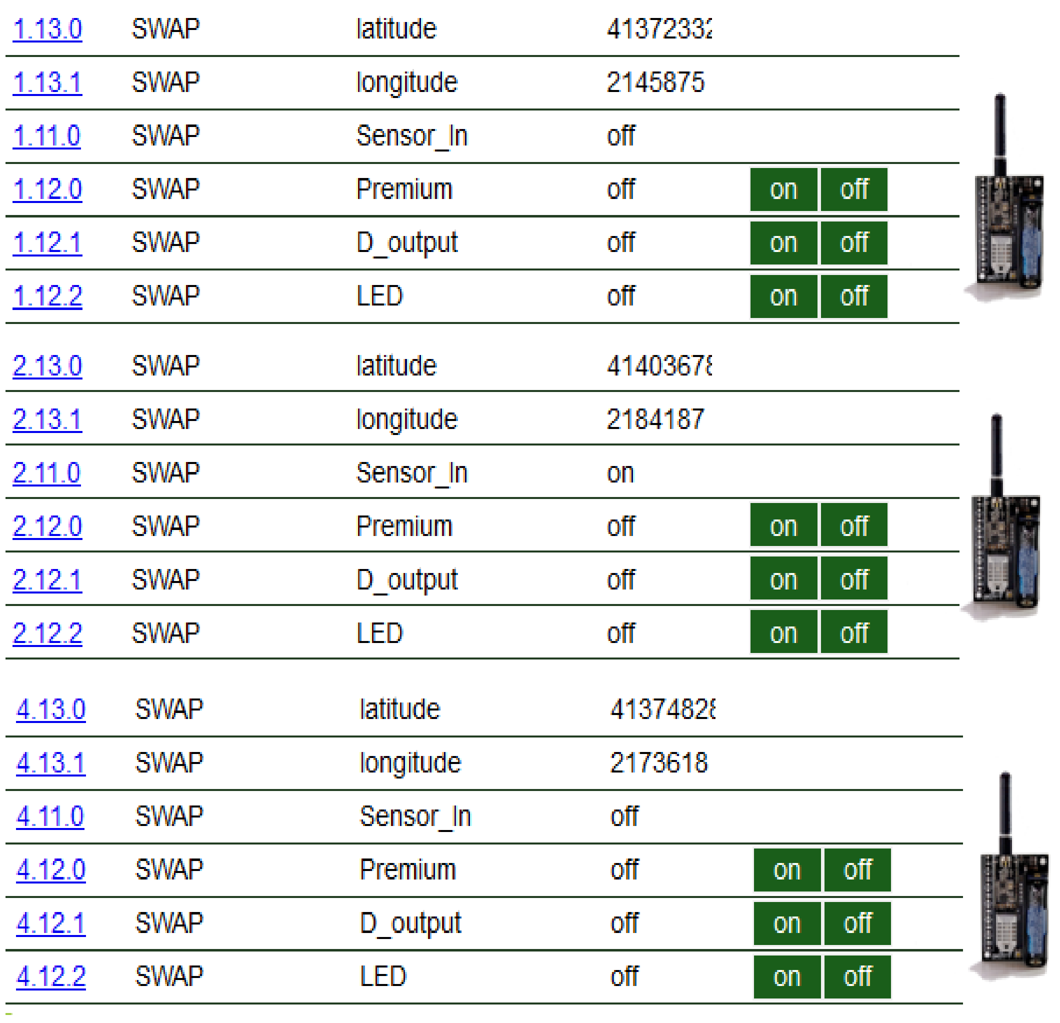
4.3.7.2 Lagarto-SWAP

Lagarto-SWAP is the server that talks to PanStamps and relies on pyswap.

Start the server:

sudo python ~/PanStamp/lagarto/lagarto-swap/lagarto-swap.py

To access the servers, run a browser on the same computer or on another one in the local network and type the following URL: http:// local_IP_address:8003



1.13.0	SWAP	latitude	4137233		
1.13.1	SWAP	longitude	2145875		
1.11.0	SWAP	Sensor_In	off		
1.12.0	SWAP	Premium	off	on	off
1.12.1	SWAP	D_output	off	on	off
1.12.2	SWAP	LED	off	on	off
2.13.0	SWAP	latitude	4140367		
2.13.1	SWAP	longitude	2184187		
2.11.0	SWAP	Sensor_In	on		
2.12.0	SWAP	Premium	off	on	off
2.12.1	SWAP	D_output	off	on	off
2.12.2	SWAP	LED	off	on	off
4.13.0	SWAP	latitude	4137482		
4.13.1	SWAP	longitude	2173618		
4.11.0	SWAP	Sensor_In	off		
4.12.0	SWAP	Premium	off	on	off
4.12.1	SWAP	D_output	off	on	off
4.12.2	SWAP	LED	off	on	off

Figure 48 Lagarto-SWAP Control and monitoring Page

In the figure above we observe 3 PanStamps. For each we observe an ID (e.g. 1.13.0), the protocol that uses (e.g. SWAP) and a set of attributes (e.g. latitude, longitude, etc) with their values (e.g. 41403678). The "Latitude" and "Longitude" attributes are static. We have declared them in the Arduino code, when we program the PanStamps. The "Sensor_In" value we can change it via Ultra Sonar Sensor or if we connect a wire in the "A3" port in PanStamp. The "Premium", "D_output" and "LED" attributes we can change them via the virtual switches from the Browser.

4.3.7.3 Lagarto – HTTP

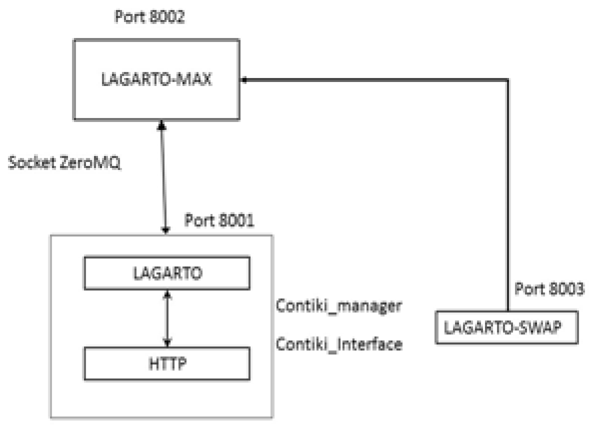


Figure 49 Lagarto Server [43]

Lagarto –HTTP, developed in M2M department at CTTC. It follows the same structure as Lagarto-SWAP to be able to communicate with Lagarto-max. Lagarto-HTTP is also called Lagarto-Contiki. The code is the "*contiStamp*" folder. This is the Lagarto-HTTP server. This is the folder tree. We will find files *.pyc*, which are generated by .py, are the last ones the ones that we are interested.

4.3.8 How to use the servers

Once the servers are running, they will auto-detect any PanStamp that start transmitting SWAP messages. If the PanStamps address is not stored in the server, details about the PanStamp, its address and its registers will appear in the server's logs, otherwise only the PanStamps ID address will appear. The PanStamp can update the servers extremely fast but the servers interface updates itself every 3 seconds.

When powering off a PanStamp, it stops updating the servers, but information about its ID address and its registers are still available in the server. This information can be deleted manually from the server's interface. Lagarto servers use two xml files to autocreate a new device. The files are stored at *devices.xlm and application_name.xlm.*

4.4 Connectivity with Internet

4.4.1 Procedure to turn on the testbed (PanStamp and Zolertia) via Ethernet

1) Connect to RPi the keyboard, screen, USB, Ethernet, PanStamp shield, etc.

2) Power on RPi: connect it to power.

3) Log in the RPi.

4) Connect Border Router Zolertia via USB (Mote with ID 7).

5) Ifconfig (display information about all network interfaces and observe the IP address)

6) Turn on Tunslip:

- *cd contiki /examples/ipv6/rpi-border-router*
- *make z1 –reset && make connect –router&* (the last & it to run in background)
  At the end of the screen will appear the router address: aaaa::1 because it's set inside the Border-router 's code.
  **Change tab to the Terminal to continue*

7) Now is the moment to start the severs

- *ssudo python PanStamp/lagarto/lagarto-max/lagarto-max.py&*
- *sudo python PanStamp/lagarto/lagarto-swap/lagarto-swap.py&*
- *sudo python contistamp/lagarto/lagarto-contiki/lagarto-contiki.py&*

8) Power on motes**:** connect batteries (PanStamp and Z1). A new mote appears it is added, it is turn off it only disappears in restarting Lagarto-MAX.

9) In other PC in the same network, in the browser write (IP= local_IP_address RPi B+)

- IP.IP.IP.IP:8002-- It should appear Lagarto-MAX web server.
- IP.IP.IP.IP:8003-- It should appear Lagarto-SWAP web server.
- IP.IP.IP.IP:8001-- It should appear Lagarto-HTTP web server.

4.4.2 Cellular Connectivity with LTE to Raspberry Pi B+

4.4.2.1 Operator: Telefonica M2M

Telefónica, S.A. [45] is a Spanish broadband and telecommunications provider with operations in Europe, Asia, North America and South America. Operating globally, it is the sixth-largest mobile network provider in the world. The company started as a public telecommunications company. Telefónica is the second largest corporation in Spain.

Figure 50 «SmartWorld» - Telefonica SIM cards

4.4.2.2 4G USB Dongle: Huawei E3276 LTE modem

Dimensions	Height 92mm Width 32mm Depth 14mm Weight <50g
Form	USB Stick
Communication System	FDD/TDD UMTS/HSUPA/HSPA+ GSM/GPRS/EDGE
Speed	High-speed LTE FDD packet data service of up to 150/50 Mbit/s High-speed LTE TDD packet data service of up to 112/10 Mbit/s
microSD card slot	Yes
Operation System	Windows XP, Windows Vista, Windows 7, Windows 8, Mac OS X 10.5, Mac OS X 10.6, Mac OS X10.7, Mac OS X10.8

Table 7Huawei E3276 LTE modem specifications [46]

Figure 51 Huawei E3276s-150 model [46]

Huawei E3276 modem modes

1. 12d1:1f01 CD-ROM Mode
2. 12d1:14db HiLink Mode (appears as network interface with fixed address. No need to install drivers)
3. 12d1:1442 Serial mode

Figure 52 «SmartWorld» - 4G USB Huawei

HiLink features being able to use the modem without any dashboard software I.e. just plug in and control device thru 192.168.1.1 in browser. Huawei HiLink modems are a special class of USB 3G / 4G modem that appears as a network interface rather than a standard serial port. These modems have a fixed IP address and the current Huawei software does not allow this ip address to be changed. Care should be taken to ensure than neither you local network, or any of your WAN networks do not conflict with the IP address of your HiLink modem.

__Getting to Know Your Mobile Internet Key__

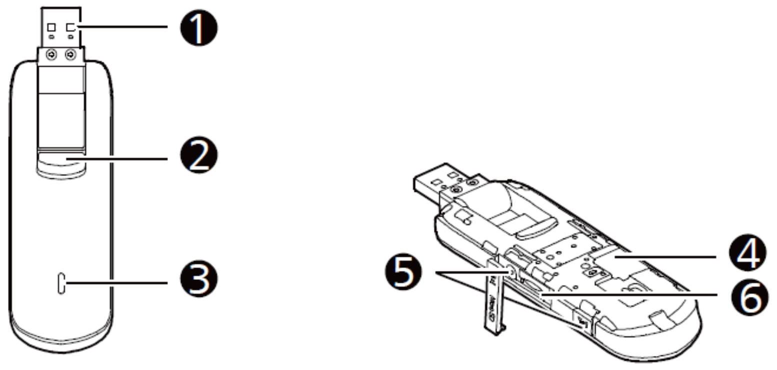


Figure 53Mobile Internet Key [46]

1. USB Connector
2. It connects the Internet Key to a PC, folds for safe transport, and rotates to help you get the best signal.
3. USB Connector Release Button.
4. The LED light indicates if the internet key has found a network and if you have successfully connected to a mobile high speed internet network.

    • *Green*, blinking twice every two seconds: the modem is powered on.
    • *Green*, blinking once every 0.2 seconds: the modem's software is being upgraded.
    • *Green*, blinking once every two seconds: the modem is registering with a 2G network.
    • *Blue*, blinking once every two seconds: the modem is registering with a 3G/3G+ network.
    • *Cyan*, blinking once every two seconds: the modem is registering with a 4G network.
    • *Green*, solid: the modem is connected to a 2G modem.
    • *Blue*, solid: the modem is connected to a 3G network.
    • *Cyan*, solid: the modem is connected to a 3G/4G network.

    • *Off*: the modem is removed.
5. SIM card slot
6. MicroSD Card Slot
7. External Antenna Ports
8. The Internet Key can support 2 external antennas for better signal reception.

Getting your Computer Ready

The steps to install the Connection Manager depend on your computer's operating system (OS). It is recommended you uninstall any existing programs used with a mobile internet key prior to completing the next steps. [46]

Step 1: Connect the Mobile Internet Key to your computer.

Plug the connector of the Internet Key directly into a USB port of your computer. Do not use a USB hub.

Step 2: Installing the Mobile Internet Key Modem

1. Plug the modem into one of the USB ports.

2.
```
sudo apt-get update
sudo apt-get install libusb-dev
sudo apt-get install sg3-utils
```

3. In a terminal session: *#lsusb*

*Note the ID of the device (e.g. 12d1:1f01, value of "Default Product"

You'll see something like:

```
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.

Bus 001 Device 005: ID 12d1:1f01 Huawei Technologies Corp.
```

4. *sudo apt-get install usb-modeswitch*

Download usb-modeswitch-2.2.0.tar.bz2,

```
cd usb-modeswitch-2.2.0

sudo make install
```

5. Download the usb-modeswitch-data package

```
cd usb-modeswitch-data

sudo nano make install
```

6. Install libusb-1.x.

```
cd libusb-1.0.9
```

./configure

make

sudo make install

Reboot RPi and open a terminal session

7. Plug the modem in, leave it for a second or two for it to be recognized and then: *#lsusb*
   *The output will now indicate that the modem has been recognized.
   (e.g. 19d1:14db, value of "Target Product")

```
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.

Bus 001 Device 004: ID 12d1:14db Huawei Technologies Co., Ltd.
```

8. Display the network interface list: *#ifconfig –a*
   You will see: eth0, lo, eth1
   Eth1 IP address will be 192.168.1.100 because it is fixed from Huawei.
   Driver load automatically network interface eth1.
9. Acquire an IP address: *#sudo nano /etc/network/interfaces*

```
Auto eth1
Allow-hotplug eth1
Iface eth1 inet dhcp
```

10. Reboot the RPi
11. *#sudo nano /etc/resolv.conf*

    Note that name servers had changed to 192.168.1.1.

Step 3: Create custom usb-modeswitch config file

1. Create a file /usr/share/usb_modeswitch/12d1:14fe with the following contents.

```
DefaultVendor= 0x12d1
DefaultProduct=0x1f01
TargetVendor=  0x12d1
TargetProduct= 0x14db
MessageContent="55534243123456780000000000000011062000000100000000000
000000000"
Enablelogging=1
```

2. Create a file /lib/udev/usb_modeswitch_huawei containing the following lines.

```
#! /bin/sh
/sbin/modprobe option
/bin/echo '12d1 1f01 > /sys/bus/usb-serial/drivers/option1/new_id
```

3. In the file /lib/udev/rules.d/40-usb_modeswitch.rules, insert the following lines.

```
# Optus (Huawei) E3276 v2
ATTRS{idVendor}=="12d1",                ATTRS{idProduct}=="14db",
RUN+="usb_modeswitch '    %b/%k'", RUN+="usb_modeswitch_huawei"
```

4. *#Sudo usb_modeswitch –c /etc/usb_modeswitch.conf*

Step 4: Add an extra module to system`s modules

Sudo nano /etc/modules

Add this: *cdc_ether*

**cdc_ether is the driver/module that used from RPi to recognize the 3G USB dongle. We have to add it to system`s modules, in order to recognized when the system is power on.*

Step 5: Configure network settings

Access to *192.168.1.100* and configure the appropriate settings with your SIM operator. Note:

- Software version: 22.436.09.00.1080 (22.xxx.xx.xx.xxxx means that is Hi Link. If it is 11.xxx.xx.xx.xxxx, that means that it is Normal mode)
- Web UI version: 13.100.04.00.1080
- WAN IP Address: 10.43.4.204 (when Ethernet cable was connected)

Step 6: Activate SIM Card

https://movistar.jasperwireless.com/provision/jsp/login.jsp

Username: CTTCSpain

password: ************

APN: m2mkit.telefonica.com

4.4.2.3 Access to Lagarto – Servers via Cellular connectivity

Step 1: Change the IP address of Lagarto-MAX:

PanStamp2/Lagarto/Lagarto-max/config/ lagarto.xml

Change the IP address to: 192.168.100

Keep the port: 8002 the same.

Step 2: Change the IP address of Lagarto-SWAP:

PanStamp2/Lagarto/Lagarto-swap/config/ lagarto.xml

Change the IP address to: 192.168.100

Keep the port: 8003 the same.

Step 3: Run Lagarto-MAX

Cd PanStamp2/Lagarto/Lagarto-max

Sudo python Lagarto-max.py

Open a new terminal session.

Step 4: Run Lagarto-SWAP

Cd PanStamp2/Lagarto/Lagarto-swap

Sudo python Lagarto-swap.py

Step 5: Access to Open.Sen.Se

Login with your username and password.

See the data from the PanStamp and Zolertia.

4.4.3 Cellular Connectivity with SIGFOX to Raspberry Pi B+

The SIGFOX Ready [40] certification process ensures that the optimal radio capacity of the devices can be expected for the SIGFOX devices. Only certified devices can claim to be SIGFOX Read. SIGFOX provides a qualification process that tests the radio qualities of a device and determines the SIGFOX Class. SIGFOX Ready modems operate with low transmission power and no continuous network synchronisation. This enables unprecedented gains in energy efficiency whilst guaranteeing years of autonomy.

5 steps to get connected to SIGFOX Network

1. Design device.
2. Get device certified.
3. Sign subscriptions agreement.
4. Get devices unique Ids.
5. Power on device: you're done !

We connect the SIGFOX M2M Gateway (TD1208 Module) via FDDI (Fiber Distributed Data Interface) cable to a USB port in Rasperry Pi B+. Then, the data are forward to the SIGFOX Backend and from there to our Application Server (Open.Sen.Se).

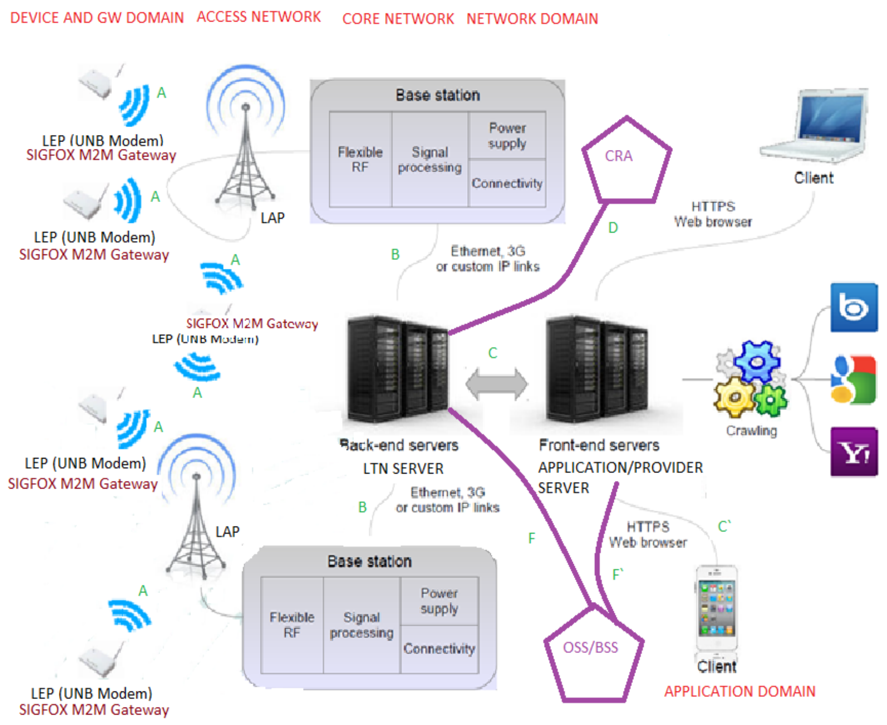


Figure 54 SIGFOX network

4.4.3.1 Telecom Design 1208 Module

In order to connect to SIGFOX network we use the Telecom Design 1208 Module. The TD1208 evaluation board provides a rich development platform for the Telecom Design TD1208 SIGFOX Gateway module. [47]

Certification	SIGFOX
Frequency range	ISM 868 MHz
Receiver sensitivity	-126 dBm
Modulation	G)FSK, 4(G)FSK, GMSK / OOK
Max output power	+14 dBm
Low active radio power consumption	13/16 mA RX , 37 mA TX @ +10 dBm
Power supply	2.3 to 3.3 V
GPIOs pins	Up to 12
bus interface	I2C

Table 8 Telecom Design 1208 Module[47]

The TD1208 module provides an integrated dual AT command interpreter for interfacing with an external application over a serial link.

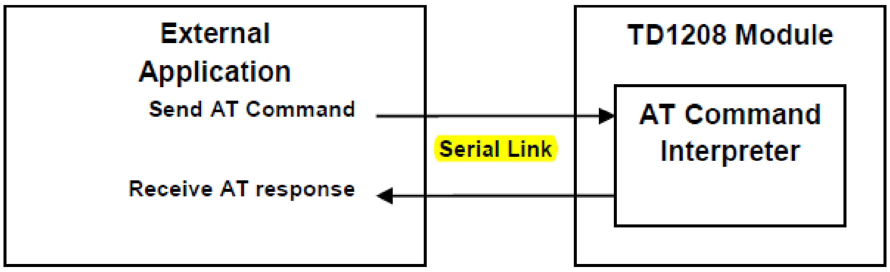


Figure 55 AT Command Interpreter[47]

Telecom Design's TD1208 devices are high performance, low current SIGFOX gateways. The combination of a powerful radio transceiver and a state-of-the-art ARM Cortex M3 baseband processor achieves extremely high performance while maintaining ultra-low active and standby current consumption. The TD1208 device offers an outstanding RF sensitivity of – 126 dBm while providing an exceptional output power of up to +14 dBm with unmatched TX efficiency. The TD1208 device versatility provides the gateway function from a local Narrow Band ISM network to the long-distance Ultra Narrow Band SIGFOX™ network at no additional cost. The broad range of analog and digital interfaces available in the TD1208 module allows any application to interconnect easily to the SIGFOX™ network. The LVTTL low energy UART, the I2C bus, the multiple timers with pulse count input/PWM output capabilities, the high-resolution/high-speed ADC and DAC, along with the numerous GPIOs can control any kind of external sensors or activators. Featuring an AES encryption engine and a DMA controller, the powerful 32- bit ARM Cortex-M3 baseband processor can implement highly complex and secure protocols in an efficient environmental and very low consumption way.

The TD1208 Evaluation Board (EVB) provides access to the different TD1208 SIGFOX gateway module interfaces, USB connectivity using a standard FTDI LVTTL RS232 _ USB cable, and development flashing/debugging facility using the standard ARM™ SWD debug interface, as well as an integrated regulated power supply. The TD1208 EVB can be powered from USB or from the dedicated power pins on the available headers, with the capability to measure the current consumption of the target TD1208 module.

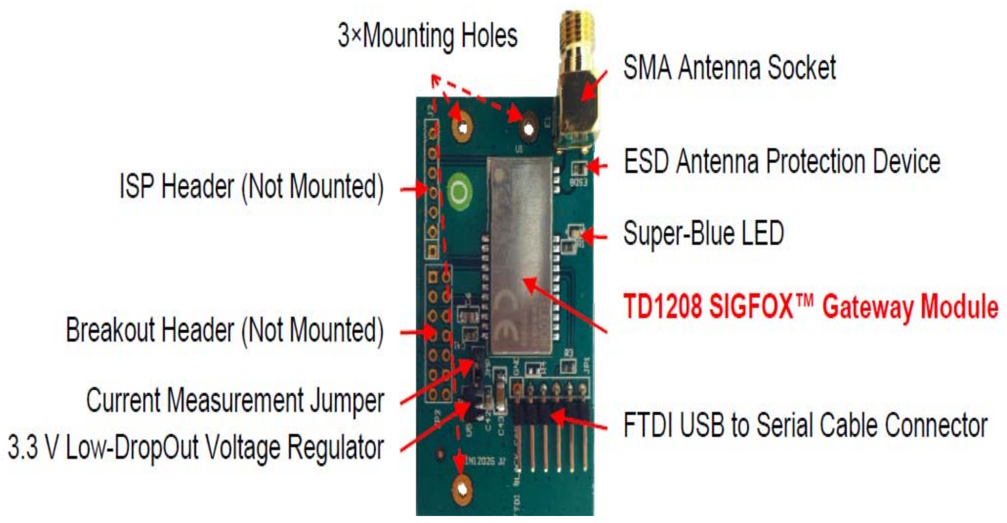

3×Mounting Holes
SMA Antenna Socket
ESD Antenna Protection Device
ISP Header (Not Mounted)
Super-Blue LED
TD1208 SIGFOX™ Gateway Module
Breakout Header (Not Mounted)
Current Measurement Jumper
3.3 V Low-DropOut Voltage Regulator
FTDI USB to Serial Cable Connector

Figure 56 TD1208 SIGFOX Gateway Module [47]

4.4.3.2 Get started with SIGFOX TD1208 Module

The TD1208 SIGFOX Gateway module on the TD1208 EVB evaluation board is pre-installed with a firmware allowing an easy set up. This firmware contains a Hayes-compatible “AT” command interpreter that also understand the SIGFOX compatible commands, making it easy to type in control commands and getting the corresponding answers using a simple serial terminal emulator.

In order to verify that the device is functional:

- Connect the SMA antenna to the TD1208 onboard SMA socket and rotate the antenna so that it stands up, perpendicular to the TD1208 EVB board top surface.
- Make sure that the current measurement strap is placed across the 2-pin header on the TD1208 EVB board.

- Connect the FTDI cable 0.1" Female Molex connector into the onboard R/A 6-pin header, so that the FTDI black wire is aligned with the label on the TD1208 EVB board.
- Connect the FTDI cable USB A plug into an available USB host port on the PC.

The onboard "Super Blue" should flash briefly upon connection, indicating a Power-On Reset (POR) condition.

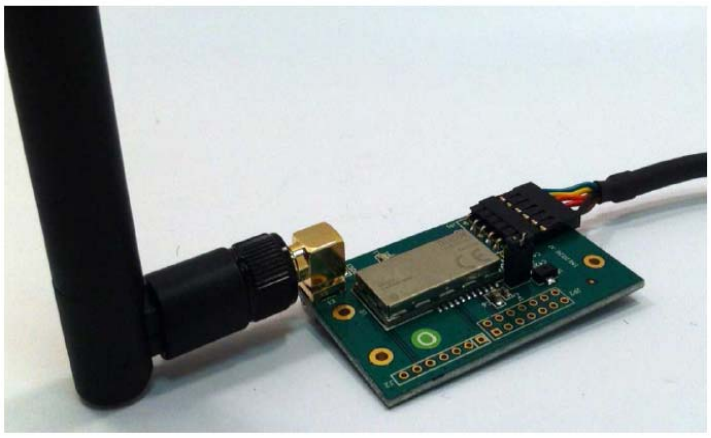


Figure 57 TD1208 Gateway module [47]

As the serial terminal emulation software will require the virtual port corresponding to the newly attached device, the best way to get it is to:

1. Use Window's "Device Manager" from the Control Panel
2. Click on the "System" icon.
3. Select the "Hardware" tab and press the "Device Manager…" button.
4. Locate and unfold the "Ports (COM & LPT)" entry into the device tree list: "USB Serial Port (COM*x*)" entry corresponds to the newly attached TD1208 EVB device.
5. Close Windows's "Device Manager" window.
6. Launch the selected serial terminal emulation software, with the following serial parameters:

   - Port as obtained from Window's "Device manager"
   - LVTTL electrical level
   - 9600 bps
   - 8 data bits
   - No parity
   - 1 stop bit

• No hardware/software flow control

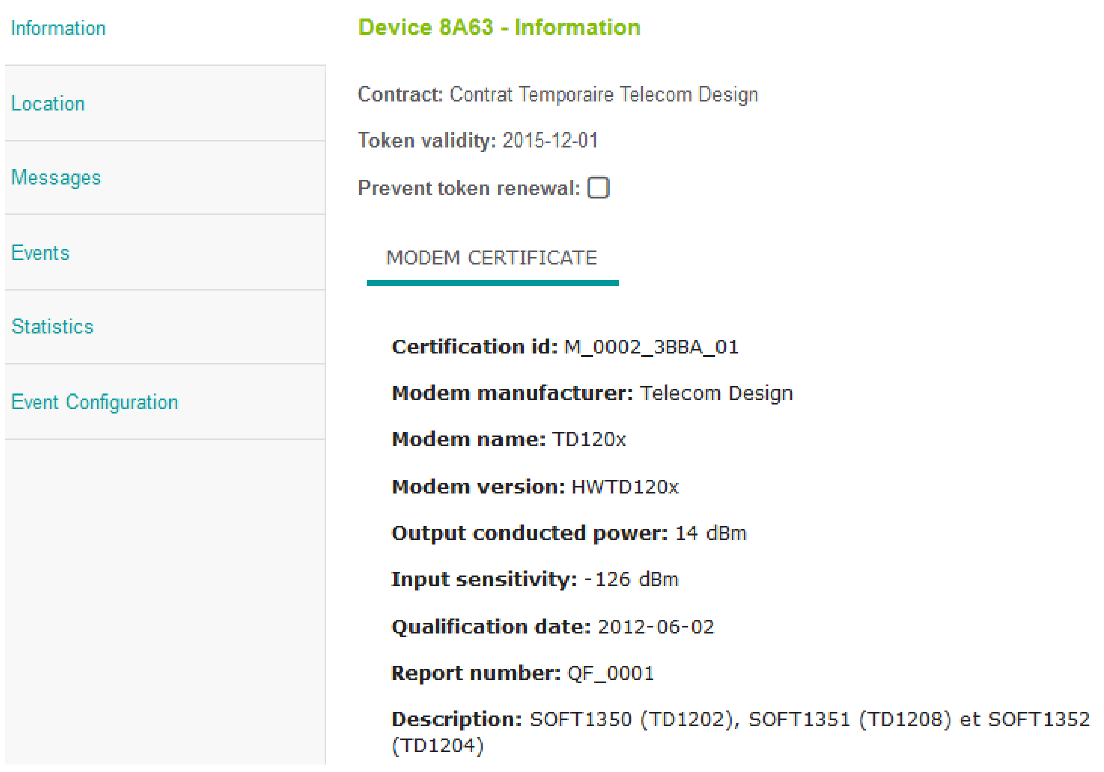
Upgrading the Firmware

Your TD1208 module is always evolving and so is our Web portal. The TD1208 SIGFOX Gateway modules contain a built-in bootloader able to perform a full firmware upgrade locally while connected to a Windows PC computer over its UART/USB interface. There is no need to have a full tool chain set up to upgrade at TD1208 SIGFOX Gateway module, as only the Telecom Design provided "TD1208Loader.exe" utility is required. This utility can be obtained from [48].

DEVICE MODE	ATS500
HELP	AT?
SAVE	AT&W
RESTART	ATZ
DEVICE INFO	AT&V
TD LAN ADDRESS	ATS400
TD LAN MASK	ATS401
TS LAN FREQUENCY	ATS403
TD LAN POWER	ATS404
SEND TDLAN MESSAGE	AT$SL = DEX
SEND SIGFOX MESSAGE	ATS$SS = DEX
REGISTER TRANSMITTER ON SENSOR	AT$REG
DEVICE CLASS	ATS501=0x1111
REGISTER DEVICE IN GW	AT$RL=1
ADDRESS AND MASK VALUES ON SENSOR LAN	AT$LA?

Table 9 Transmission Test – AT&T Commands

With these AT Commands we can see the information about our module and send messages to the network or to SIGFOX backend.



Device 8A63 - Information

Contract: Contrat Temporaire Telecom Design

Token validity: 2015-12-01

Prevent token renewal: ☐

MODEM CERTIFICATE

Certification id: M_0002_3BBA_01

Modem manufacturer: Telecom Design

Modem name: TD120x

Modem version: HWTD120x

Output conducted power: 14 dBm

Input sensitivity: -126 dBm

Qualification date: 2012-06-02

Report number: QF_0001

Description: SOFT1350 (TD1202), SOFT1351 (TD1208) et SOFT1352 (TD1204)

Figure 58 SIGFOX Backend [49]

In “SIGFOX Backend”, we can observe information about our modules like name, version, description of software, etc. Also, we can see the coverage area of the SIGFOX in “Location’, in “Messages” we can see the messages we sent via SIGFOX network, in “Statistics” we can observe some graphs with statistics and we can configure Events in “Event Configuration”.

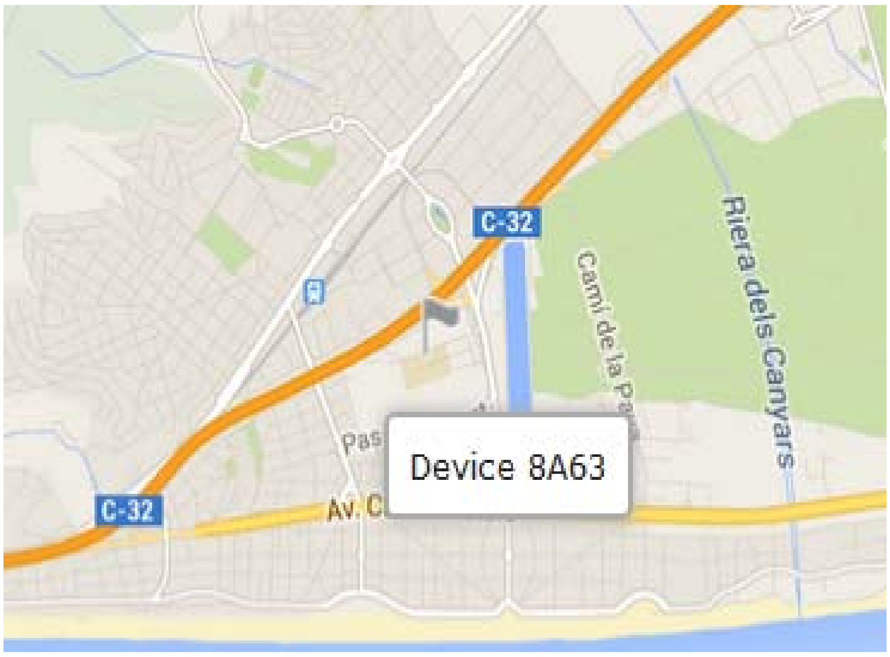
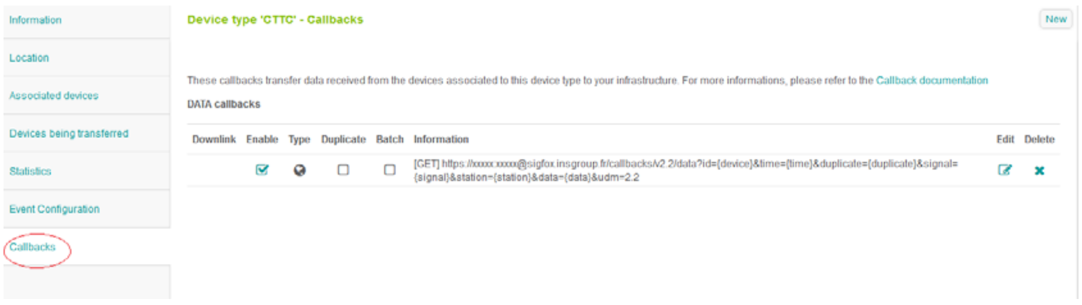


Figure 59 SIGFOX Device 8A63 - Location



Device type 'CTTC' - Callbacks

These callbacks transfer data received from the devices associated to this device type to your infrastructure. For more informations, please refer to the Callback documentation

DATA callbacks

Downlink	Enable	Type	Duplicate	Batch	Information	Edit	Delete
	✓	●	☐	☐	[GET] https://xxxxx:xxxxx@sigfox.insgroup.fr/callbacks/v2.2/data?id={device}&time={time}&duplicate={duplicate}&signal={signal}&station={station}&data={data}&udm=2.2	✎	✗

Figure 60 SIGFOX - Device type – Callbacks [49]

These **callbacks** transfer data received from the devices associated to this device type to your infrastructure. The backend can automatically forward some events using the «callback» system. The configuration of callbacks is done in the device type page. The callbacks are triggered when a new device message is received or when a device communication loss has been detected. There are 3 callback types. You have a set of available variables for each type of callback. These variables are replaced by their value when a callback is called. When receiving a callback, the client system must return an HTTP 2xx code within 10 seconds. If the client system fails to process the callback during this time, an automatic retry will be scheduled 1 minute later. We can configure all the Callbacks from [50].

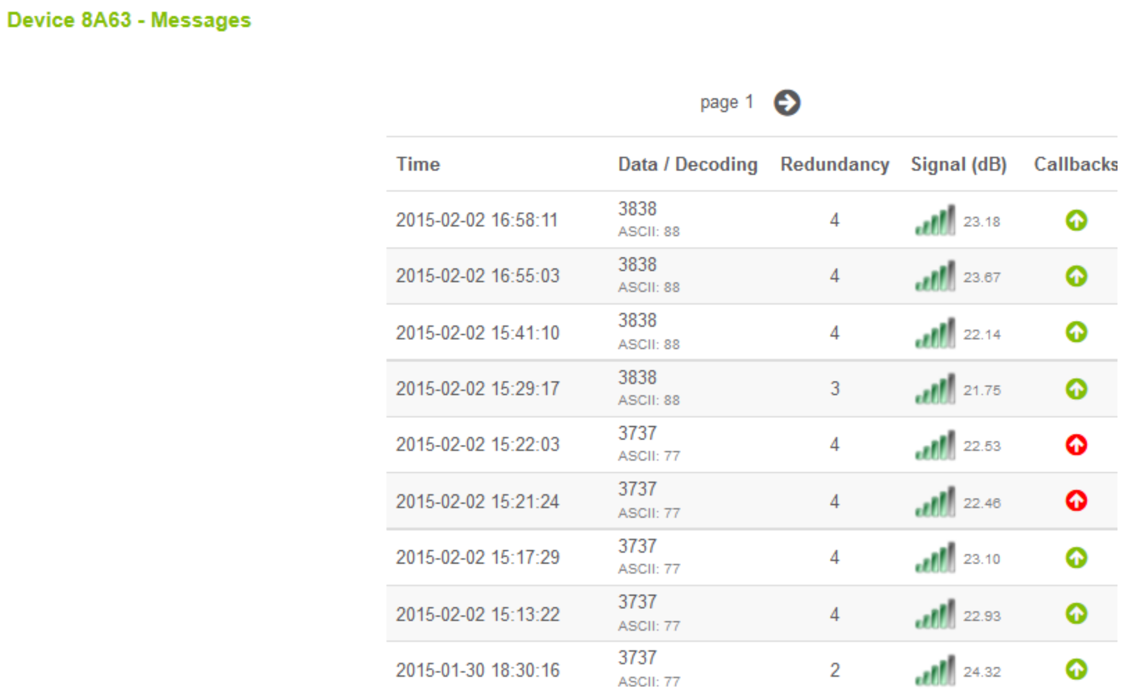
Configure TD1208 Module

In order to send data via SIGFOX we create a Python procedure. In the following section you can see the main parts of this.

```
…
SIGFOX_ENABLE = 1
# SIGFOX device serial device
SIGFOX_DEVICE = "/dev/ttyUSB0"
# SIGFOX device baud rate
SIGFOX_DEV_BR = 9600
# SIGFOX device byte size (5 to 8)
SIGFOX_DEV_BS = 8
# SIGFOX device parity ("[N]one", "[E]ven", "[O]dd", "[M]ark", "[S]pace")
SIGFOX_DEV_PAR = "N"
```

```
# SIGFOX device stop bits (1, 1.5, 2)
SIGFOX_DEV_SB = 1
# SIGFOX push rate (ms, MUST be a multiple of REFRESH_RATE)
SIGFOX_PUSH_RATE = 300000
# Refresh rate (ms)
REFRESH_RATE = 1000
# Display some debug values when set to 1, and nothing when set to 0
DEBUGMSG = 1
…
def sigfoxData():
   #return string(endp_value)
   return str(77)
print 2
sigfox = None;
sigfoxPush = 0;
if SIGFOX_ENABLE == 1:
   sigfox = serial.Serial(port=SIGFOX_DEVICE,
                          baudrate=SIGFOX_DEV_BR,
                          parity=SIGFOX_DEV_PAR,
                          stopbits=SIGFOX_DEV_SB,
                          bytesize=SIGFOX_DEV_BS)
# Push data to SIGFOX network
if SIGFOX_ENABLE == 1:
   if sigfoxPush % (SIGFOX_PUSH_RATE / REFRESH_RATE) == 0:
      import datetime
      today = datetime.datetime.today()
      sigfox.write("at$ss=" + sigfoxData().encode("hex") + "\r")
```

```
            if DEBUGMSG == 1:
           print    today.strftime("%Y-%m-%d    %H:%M:%S    -    Sent    to    SIGFOX:    "+
sigfoxData().encode("hex")+ " a.k.a " + sigfoxData() + "(V) in hexadecimal")
        sigfoxPush = sigfoxPush + 1
        # You don't want the counter to become too big :)
    if sigfoxPush > (SIGFOX_PUSH_RATE / REFRESH_RATE):
           sigfoxPush = 1
        # Pause before starting loop once again
 #time.sleep(600)
```



Device 8A63 - Messages

page 1

Time	Data / Decoding	Redundancy	Signal (dB)	Callbacks
2015-02-02 16:58:11	3838 ASCII: 88	4	23.18	🟢
2015-02-02 16:55:03	3838 ASCII: 88	4	23.67	🟢
2015-02-02 15:41:10	3838 ASCII: 88	4	22.14	🟢
2015-02-02 15:29:17	3838 ASCII: 88	3	21.75	🟢
2015-02-02 15:22:03	3737 ASCII: 77	4	22.53	🔴
2015-02-02 15:21:24	3737 ASCII: 77	4	22.46	🔴
2015-02-02 15:17:29	3737 ASCII: 77	4	23.10	🟢
2015-02-02 15:13:22	3737 ASCII: 77	4	22.93	🟢
2015-01-30 18:30:16	3737 ASCII: 77	2	24.32	🟢

Figure 61SIGFOX Received Messages from 8A63 Device [49]

In Figure 64 we can observe the Timestamp, the data of the message, the signal of the antenna and if the Callback function worked or not. The Data is in Hexadecimal but we can also see the decoding (ASCII number).

4.5 Cloud Applications

The cloud platform that was selected to connect the M2M servers and the applications is called open sense "http://open.sen.se". Open.Sen.se where infographic of different data streams can be displayed, accessed and shared from anywhere with Internet connectivity. The collected data from the sensor nodes are processed, stored and analyzed on Open.Sen.se server via an API.

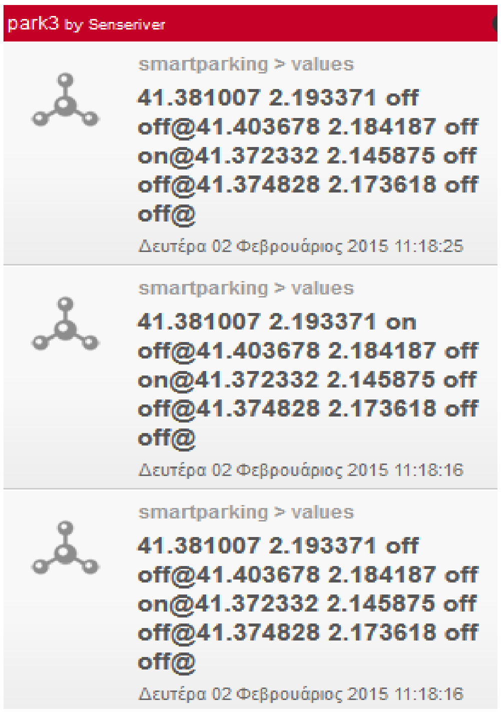
User: *****@******.com

Password: ********

Once logged in, the already created devices can be found at >Channel>mydevices

To Create and ADD new devices >Channels>Custom made devices >register >get started

The communication among the servers, the cloud and the applications is achieved through http requests (GET, POST). For security purposes every http request must contain a unique key in the header of each http packet. Once logged in, you will see the current applications.


park3 by Senseriver

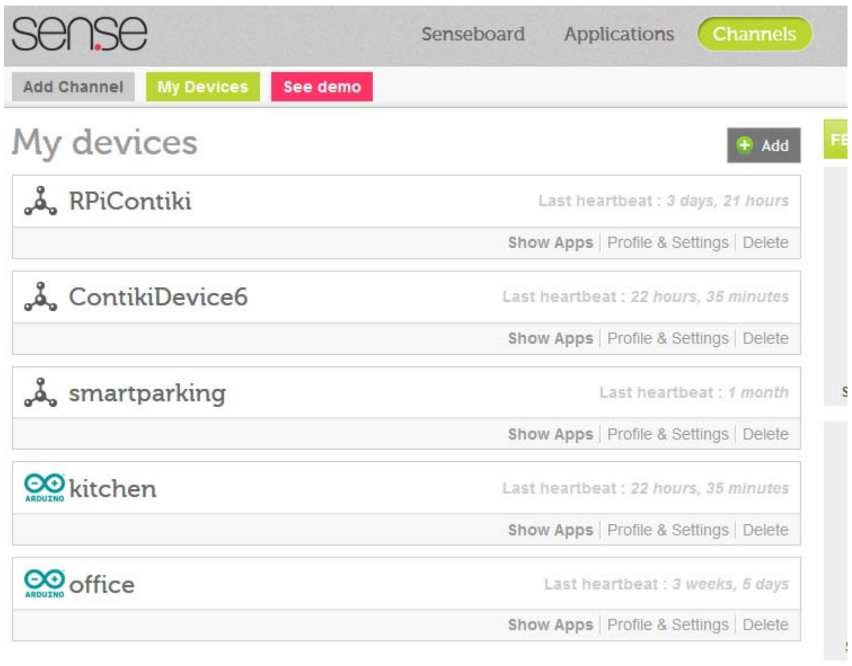
smartparking > values
41.381007 2.193371 off off@41.403678 2.184187 off on@41.372332 2.145875 off off@41.374828 2.173618 off off@
Δευτέρα 02 Φεβρουάριος 2015 11:18:25

smartparking > values
41.381007 2.193371 on off@41.403678 2.184187 off on@41.372332 2.145875 off off@41.374828 2.173618 off off@
Δευτέρα 02 Φεβρουάριος 2015 11:18:16

smartparking > values
41.381007 2.193371 off off@41.403678 2.184187 off on@41.372332 2.145875 off off@41.374828 2.173618 off off@
Δευτέρα 02 Φεβρουάριος 2015 11:18:16

Figure 62 Sense board “Park 3” application retrieves information for "Smartparking" android application

In “Sense Board” figure, we can observe the data had received from 4 sensors. These sensors are connected to “SmartParking” android application. There are 4 values for every sensor. These values are: *Coordinates* (longitude, latitude), *Sensor_In, Premium*. *Coordinates* are decimal numbers and they are static. We have pre-configured them in the PanStamp microcontroller. *Sensor_In* and *Premium* are Boolean values. We can change the *Sensor_In* value via Ultra Sonic Sensor or via connecting a Wire in the PanStamp Board. We can change the *Premium* value manually via Lagarto-SWAP server.



My devices

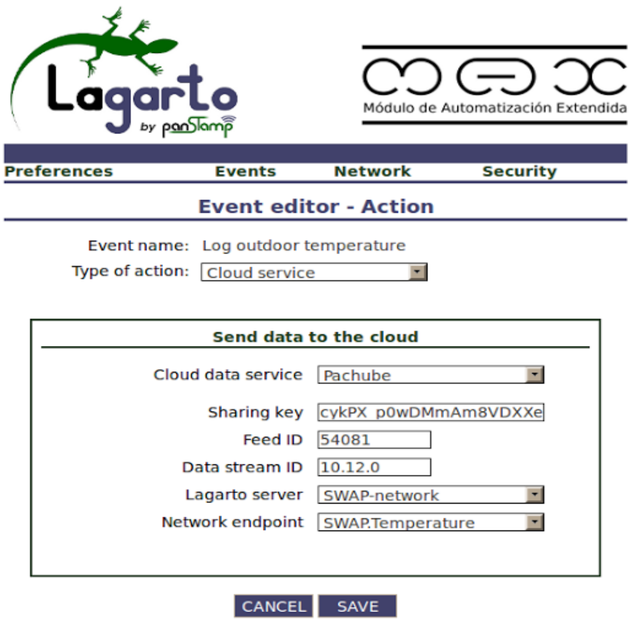
RPiContiki	Last heartbeat : 3 days, 21 hours	Show Apps | Profile & Settings | Delete
ContikiDevice6	Last heartbeat : 22 hours, 35 minutes	Show Apps | Profile & Settings | Delete
smartparking	Last heartbeat : 1 month	Show Apps | Profile & Settings | Delete
kitchen	Last heartbeat : 22 hours, 35 minutes	Show Apps | Profile & Settings | Delete
office	Last heartbeat : 3 weeks, 5 days	Show Apps | Profile & Settings | Delete

Figure 63 Channels -> MyDevices [41]

In "Channels" figure, we can observe and edit the devices we had created. We use the "RPiContiki" device for "Smartparking" application and "ContikiDevice6" for Zolertia motes, in order to power on/off the LED.

You can see your API key -> Profile & Settings-> API Key

Now, we can access the Lagarto-MAX via a browser. (http://local_IP_address:8002)



Event editor - Action

Event name: Log outdoor temperature
Type of action: Cloud service

Send data to the cloud

Cloud data service	Pachube
Sharing key	cykPX_p0wDMmAm8VDXXe
Feed ID	54081
Data stream ID	10.12.0
Lagarto server	SWAP-network
Network endpoint	SWAP.Temperature

Figure 64 Lagarto-MAX Event editor

- Type of Action: Cloud Service
- Cloud data service: Open.sen.se
- Sharing key: Sense Key
- Feed ID: Code of the Feed (from the Open.sen.se)

4.5 Android Applications

Two following applications were developed targeting the Android OS:

SmartParking: This application receives information about parking spots from the Lagarto server and displays them in a map using the Google maps API. The server, when it detects a change in the spots, updates the cloud using a POST request. The application updates itself by sending a GET request to the cloud. The parking spaces are noted in the map using different colors depending on the spot availability.

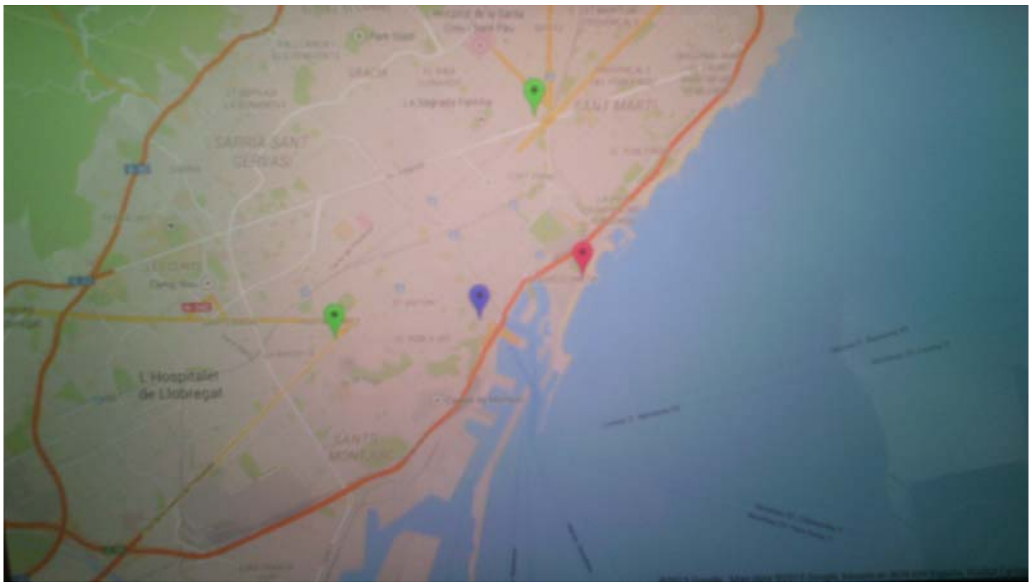


Figure 65 "SmartParking" android application

In the Figure 65, we can see 4 spots that are 4 sensors. If the spot is “Green” this means that it is available. If it is “blue” it means that it is Premium spot. That means you are Premium user and you have pre-paid for a parking spot. If it is “Red” means that the parking spot is occupied.

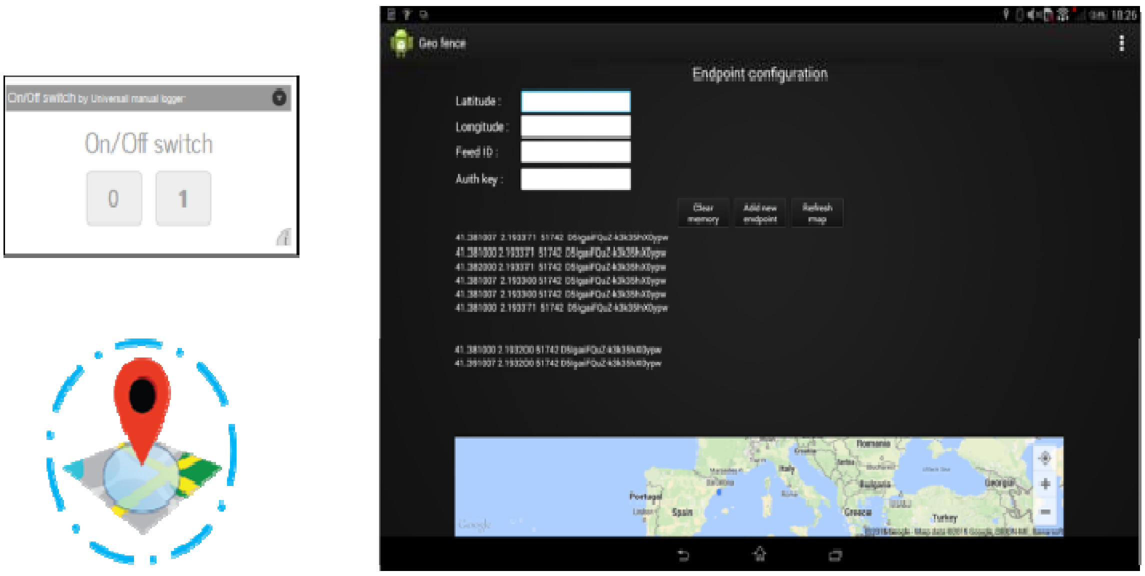
PanStamp are programmed to transmit to the M2M server the values of the sensors that decide if there are any cars in the parking spots and the coordinates of their exact location. Every time a value changes, the PanStamps send the new value to the M2M server so it is kept updated. The M2M server has a configured automated event and every time there is an update from the PanStamps that represent parking spaces, it uploads all the information to a virtual cloud. The Open.Sen.se platform is used for online storing of the data.

The Android application is used mostly for translating the data obtained from the IoT cloud into a simplified and user-friendly interface. It implements an HTTP library so it can send HTTP GET requests to the virtual cloud and acquire information about the coordinates and states of the parking spaces. The application also implements the Android Google Maps API v2 which enables the display of a map based on Google Maps. The specific API is selected because Android users are already familiar with the Google Maps interface so using the SmartParking application is simple. The application processes the HTTP GET response and creates markers on the map with different colors and explanatory labels depending on the sensor values.

A premium membership feature is also implemented by adding an output register on the PanStamps. The register is configured as output so that it can be turned on and off dynamically. This way the premium membership status is not integrated in the firmware and it can be controlled from the M2M server. It is a virtual output that corresponds to no physical pin of the microcontroller. The M2M server updates the cloud about the state of the switch and the application translates it to premium membership if the switch is on and normal membership if the switch is off.

GeoFence: Application that changes a value in the Lagarto server judging from the geological position of the android device. The application asks for the necessary information (coordinates, security key, endpoint ID) and every time the geological position fulfills certain conditions, it sends a POST request to the cloud. The Lagarto server is configured to poll values from the cloud every minute (configurable) and detects the change.

For the Geo Fencing application a PanStamp is programmed to control an actuator. The actuator used for the needs of the application is an LED light, but in real life the actuator could be integrated in the heating system, the garage door or the coffee maker. The M2M server communicates with the panStamp and can send commands to the actuator at any time. The polling mechanism is used so that the server receives status updates regarding the state of the actuator from a virtual switch on the Open.Sen.se cloud. The rest is handled by the Android application.



Endpoint configuration

Latitude :
Longitude :
Feed ID :
Auth key :

```
41.381007 2.193371 51742 D5igaiFQuZ-k3k35hXDypw
41.381000 2.193371 51742 D5igaiFQuZ-k3k35hXDypw
41.382000 2.193371 51742 D5igaiFQuZ-k3k35hXDypw
41.381007 2.193300 51742 D5igaiFQuZ-k3k35hXDypw
41.381007 2.193300 51742 D5igaiFQuZ-k3k35hXDypw
41.381000 2.193371 51742 D5igaiFQuZ-k3k35hXDypw

41.381000 2.193200 51742 D5igaiFQuZ-k3k35hXDypw
41.391007 2.193200 51742 D5igaiFQuZ-k3k35hXDypw
```

Figure 66 "Geofencing" Application

The application can be used in various scenarios. Could be to turn on the heating system when the user is approaching his home so that the house is already warm when he arrives, or to open the outside door automatically every time the user is outside his house. It is designed to be able to save many events with different actuators so it can automate many processes.

4.6 Problems in the development

(1) The SD card, we have to use, it has pre-installed NOOBS and Raspbian.

Followed the procedure from the PanStamp Website [51] but it didn’t work. Thus, we had to install the components of Lagarto Servers from the scratch (one by one).

*According to founder of PanStamp Daniel Berenguer, the image has not been tested yet in Rapsberry Pi B+, so we don’t know if it is compatible.

(2) The ZeroMQ has to be 2.2 because we use Raspbian Wheezy.

(3) Python bindings has to be version 2.0 (not 13.0+)

(4) The Python tools have to be updated.

(5) The RPi has to be updated.

(6) To avoid Error: *SwapException: 'SwapException occured: Unable to reset serial modem'*

Serial.xml (lagarto-swap/config/serial.xml) should be like this:

```
<?xml version="1.0"?>
<serial>
```

<port>/dev/ttyAMA0</port>

<speed>38400</speed>

</serial>

The PanStamp shield should be fit in RPi like this image (without the SMA Antenna). Replace SMA antenna with a wire. For the B+ version, you should leave pins 27-40 from the RPi unconnected.

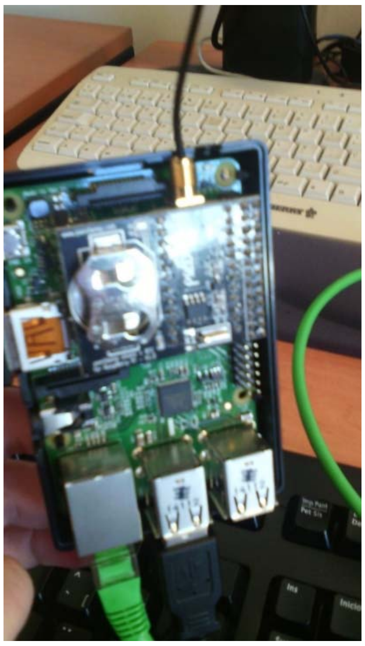


Figure 67 The way to connect a wire antenna to RPi B+

(7) Avoid Lagarto-MAX errors in these files -> *lagarto-max.py, envmanager.py, database.py, api.py*

(8) Telefonica`s signal coverage didn't reach the area of CTTC. So we send the data via 3G+.

(9) SIGFOX can send only hexadecimal data. So, we have to convert the data to ASCII.

(10) The Android code should be updated in order to support multiple screens.

5. Conclusion and Future Trends

The number of "connected devices" to Internet is expanding and is expected to continue to grow as the utilized number of devices increases. Nowadays, mobile phone subscriptions have exceeded 3 billion over the world. End-users are using multiple devices and new types of devices that allow machines to be connected to one another are emerging. These devices will communicate and offer services via the Internet. This tense growth is remarkable within not just the communications industries, but also the wider global economy.

No industry and no part of society have remained unaffected by this technical revolution. Rapidly decreasing costs of sensors and actuators is now equal to few Euros. In addition, these devices (through increases in the computational capacity of the associated chipsets) are now able to communicate via fixed and mobile networks. They are able to communicate information about the physical world in near real-time across networks with high bandwidth at low relative cost. So, while we have seen M2M solutions for quite some time, we are now entering a period of time where the uptake of both M2M and IoT solutions will increase dramatically.

The reasons for this are three-fold [28]:

1. The increased need for understanding the physical environment in its various forms, from industrial installations through to public spaces and consumer demands. These requirements are often driven by efficiency improvements, sustainability objectives, or improved health and safety.
2. The improvement of technology and enhanced networking capabilities.
3. Reduced costs of components and the ability to collect and analyze the collected data.

From a long perspective, the development trend of IoT includes three steps:

1) Embedded intelligence: We have embedded intelligences which can do actions automatically. There already have been many applications, such as the washing machine controller can make washing machine complete its work automatically
2) Connectivity: Make every smart device able to connect. From the smart connected devices viewpoint, smart devices are not smart because they are just endowed with agent capabilities and all the actions are pre-designed by human, they are smart because

they are connected. Things can be connected wired or wirelessly. In the IoT wireless connection will to be the main way. Base on the existed infrastructure, there are many ways to connect a thing. Connect smart things makes interaction possible.

3) Interaction: Things can interact, they exchange information by themselves. So the form of communication will change from human-human to human-thing to thing-thing. As IoT is application driven, new business applications should be created which can improve the innovation and development of IoT

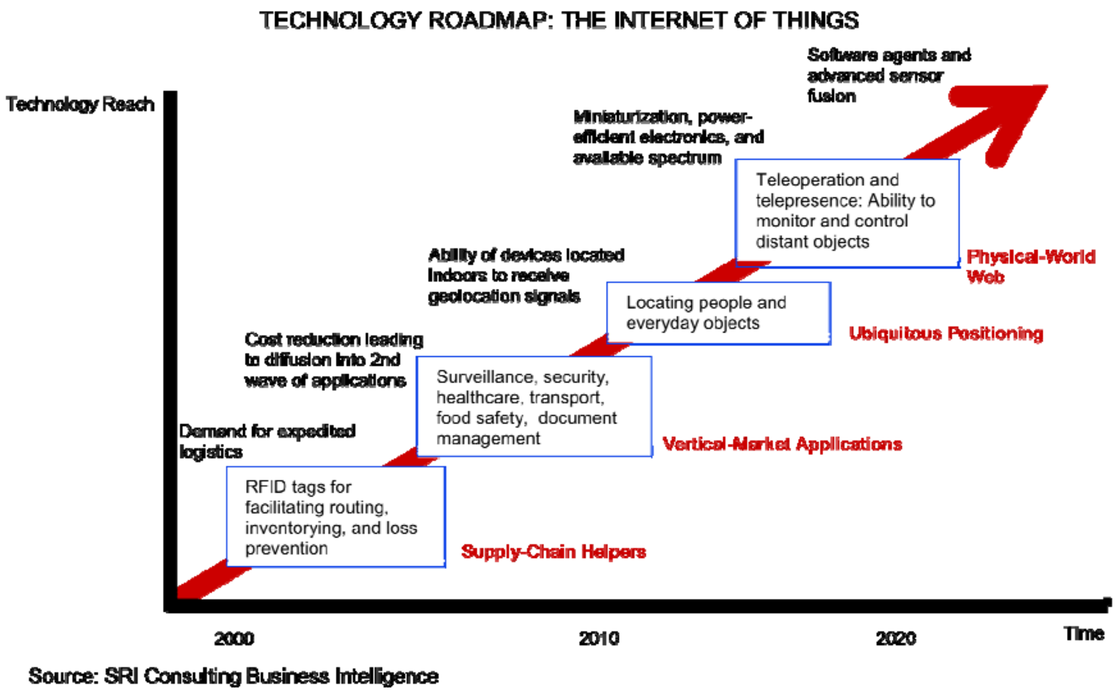

TECHNOLOGY ROADMAP: THE INTERNET OF THINGS
Technology Reach
Software agents and advanced sensor fusion
Miniaturization, power-efficient electronics, and available spectrum
Teleoperation and telepresence: Ability to monitor and control distant objects
Physical-World Web
Ability of devices located indoors to receive geolocation signals
Locating people and everyday objects
Ubiquitous Positioning
Cost reduction leading to diffusion into 2nd wave of applications
Surveillance, security, healthcare, transport, food safety, document management
Vertical-Market Applications
Demand for expedited logistics
RFID tags for facilitating routing, inventorying, and loss prevention
Supply-Chain Helpers
2000
2010
2020
Time
Source: SRI Consulting Business Intelligence

Figure 68 Technology Roadmap - The IoT [52]

The entire work that was performed during my placement in the M2M department in CTTC was divided into 2 tasks: The entire "Smartworld" testbed was reproduced in RPi B+ and cellular and SIGFOX connectivity to the gateway was realized.

At the end of the time spent at CTTC of Barcelona, I managed to surpass all planned objectives and we reproduced a complete End-to-End M2M communication network and integrate two representative applications of the IoT.

"Smartworld" project was extremely challenging, combining many different tools and technologies. We worked with short-range technologies on the sub-GHz band (PanStamps) and Zigbee-like solutions (Z1 from Zolertia), setting up a heterogeneous short range wireless network glued up through an M2M gateway (RPi B+). The RPi B+ acts as a gateway and has been connected to the Internet through both cellular LTE/3G and Low Power Wide Area

(LPWA) technologies. The connectivity has enabled the execution of a smart parking final application. In the future in the "Smartworld" project, we will be able to:

- Make over the air programming.
- Integrate with OpenMotes.
- Improve the integration with the Cloud.
- Add security (data encryption), Securing the ETSI M2M architecture is a crucial task, since we have to secure heterogeneous wireless communications (cellular, wireless, wired), devices (sensor or mobile phone) and applications (programming language, framework, database).
- Integrate with IoT Lab from SMARTECH @CTTC.
- We will become a recognized CTTC Testbed.

An idea for more research in the field:

In the end, it will be amazing to use the sensor data with Semantic Technologies [53]. Devices and sensors would register through network gateway which writes their data into semantic database. Every time the sensor would send the temperature value, the gateway writes it into the repository and then matches it with desired temperature inside the room where sensor is located. In general, the implementation of semantic annotation of M2M data enables opening of the secondary M2M market. While the primary M2M market means vertically connected sensor/applications which target a specific business need, re-using of data collected in primary market with semantic information which helps to understand the original data opens up the secondary M2M market. By providing means to understand M2M data, the available business models can be greatly enhanced. A possible business case that can be realized could be to provide derived information from the provided raw data through intelligent processing, e.g. analyzing the data, aggregating data across many different data sources, or to provide interpreted data as an additional service. In this system, adding semantic information can be done in different ways and constitutes another very interesting future research direction.

Bibliography

[1] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelfflé, European Commission, και Directorate-General for the Information Society and Media, *Vision and challenges for realising the Internet of things*. Luxembourg: EUR-OP, 2010.

[2] D. Fensel, Επιμ., *Foundations for the Web of Information and Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[3] C. Kalalas, 'SmartWorld testbed: A comprehensive technical description', *think about the iot*. . [Retrieved March 2015]

[4] M. Imran, A. M. Said, και H. Hasbullah, 'A survey of simulators, emulators and testbeds for wireless sensor networks', in *Information Technology (ITSim), 2010 International Symposium in*, 2010, s. 2, pp 897–902.

[5] J. Kowalczuk, M. C. Vuran, και L. C. Perez, 'A Dual-Network Testbed for Wireless Sensor Applications', in *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, 2011, pp 1–5.

[6] E. Ertin, A. Arora, R. Ramnath, M. Nesterenko, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, και H. Cao, 'Kansei: a testbed for sensing at scale', in The Fifth International Conference on Information Processing in Sensor Networks, 2006. IPSN 2006, 2006, pp 399–406.

[7] F. Hermans, O. Rensfelt, P. Gunningberg, L.-Å. Larzon, και E. Ngai, 'Sensei-UU — A Relocatable WSN Testbed Supporting Repeatable Node Mobility', in *Testbeds and Research Infrastructures. Development of Networks and Communities*, T. Magedanz, A. Gavras, N. H. Thanh, και J. S. Chase, Επιμ. Springer Berlin Heidelberg, 2011, pp 612–614.

[8] J.-P. Sheu, C.-J. Chang, C.-Y. Sun, και W.-K. Hu, 'WSNTB: A testbed for heterogeneous wireless sensor networks', in *2008 First IEEE International Conference on Ubi-Media Computing*, 2008, pp 338–343.

[9] Z. Zhao, G.-H. Yang, Q. Liu, V. O. K. Li, και L. Cui, 'EasiTest: A multi-radio testbed for heterogeneous wireless sensor networks', in *IET International Conference on Wireless Sensor Network, 2010. IET-WSN*, 2010, pp 104–108.

[10] A. Reinhardt, M. Kropff, M. Hollick, και R. Steinmetz, 'Designing a sensor network testbed for smart heterogeneous applications', in *33rd IEEE Conference on Local Computer Networks, 2008. LCN 2008*, 2008, pp 715–722.

[11] 'Publications • FIT/IoT-LAB'. Available in: https://www.iot-lab.info/publications/. [Retrieved: 25 March 2015].

[12] E. Welsh, W. Fish, και J. P. Frantz, 'GNOMES: a testbed for low power heterogeneous wireless sensor networks', in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03*, 2003, τ. 4, pp IV–836–IV–839 vol.4.

[13] S. Bromage, C. Engstrom, J. Koshimoto, M. Bromage, S. Dabideen, M. Hu, R. Menchaca-Mendez, D. Nguyen, B. Nunes, V. Petkov, D. Sampath, H. Taylor, M. Veyseh, J. J. Garcia-Luna-Aceves, K. Obraczka, H. Sadjadpour, και B. Smith, 'SCORPION: A Heterogeneous Wireless Networking Testbed', *SIGMOBILE Mob. Comput. Commun. Rev.*, s. 13, issue. 1, pp 65–68, 2009.

[14] Q. Liu, Z. Zhao, και L. Cui, 'A Versatile Heterogeneous Sensor Networks Testbed', in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2010, pp 387–388.

[15] 'Demo Abstract: Twonet - Large-Scale Wireless Sensor Network Testbed with Dual-Radio Nodes'. Available in: http://www2.cs.uh.edu/~gnawali/papers/twonet-sensys13demo-abstract.html. [Retrieved: 25 March 2015].

[16] F. Hermenier και R. Ricci, 'How to Build a Better Testbed: Lessons from a Decade of Network Experiments on Emulab', στο *Testbeds and Research Infrastructure. Development of Networks and Communities*, T. Korakis, M. Zink, και M. Ott, Επιμ. Springer Berlin Heidelberg, 2012, σσ 287–304.

[17] T. Dimitriou, J. Kolokouris, και N. Zarokostas, 'Sensenet: A Wireless Sensor Network Testbed', in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, New York, NY, USA, 2007, pp 143–150.

[18] G. Werner-Allen, P. Swieskowski, και M. Welsh, 'MoteLab: a wireless sensor network testbed', in *Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005*, 2005, pp 483–488.

[19] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, και M. Welsh, 'CitySense: An Urban-Scale Wireless Sensor Network and Testbed', στο *2008 IEEE Conference on Technologies for Homeland Security*, 2008, pp 583–588.

[20] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, και T. Chiueh, 'MiNT-m: An Autonomous Mobile Wireless Experimentation Platform', in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, New York, NY, USA, 2006, pp 124–137.

[21] U. of Texas και O. 2009, 'Pharos: An Application-Oriented Testbed for Heterogeneous Wireless Networking Environments', *TechRepublic*. Available in: http://www.techrepublic.com/resource-library/whitepapers/pharos-an-application-oriented-testbed-for-heterogeneous-wireless-networking-environments/. [Retrieved: 25 March 2015].

[22] H. Mahkonen, T. Rinta-aho, T. Kauppinen, M. Sethi, J. Kjällman, P. Salmela, και T. Jokikyyny, 'Secure M2M Cloud Testbed', in *Proceedings of the 19th Annual International Conference on Mobile Computing &#38; Networking*, New York, NY, USA, 2013, pp 135–138.

[23] A.-S. Tonneau, N. Mitton, και J. Vandaele, 'A Survey on (mobile) Wireless Sensor Network Experimentation Testbeds', in *2014 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2014, pp 263–268.

[24] D. Minoli, *Building the internet of things with IPv6 and MIPv6: the evolving world of M2m communications*. Hoboken, New Jersey: Wiley, 2013.

[25] H. Chaouchi, Επιμ., *The Internet of things: connecting objects to the web*. London : Hoboken, NJ: ISTE ; John Wiley & Sons, 2010.

[26] D. Boswarthick, O. Elloumi, και O. Hersent, Επιμ., *M2M communications: a systems approach*. Chichester, West Sussex, U.K: Wiley, 2012.

[27] H. Zhou, *The internet of things in the cloud: a middleware perspective*. Boca Raton: CRC Press, Taylor & Francis Group, 2013.

[28] J. Höller., *From machine-to-machine to the Internet of things: introduction to a new age of intelligence*. Amsterdam: Elsevier Academic Press, 2014.

[29] PanStamp official website, http://www.PanStamp.com/home, Retrieved March 2015.

[30] Arduino official website, http://arduino.cc/en/Guide/Environment, Retrieved March 2015PanStamp official website, base

[31] board, http://www.PanStamp.com/products/battery-board, Retrieved March 2015

[32] PanStamp technical details, lagarto servers, http://code.google.com/p/PanStamp/wiki/lagarto, Retrieved 25 March 2015

[33] PanStamp technical details, SWAP, http://code.google.coRetrieved 25 March 2015m/p/PanStamp/wiki/SWAP, Retrieved 25 March 2015.

[34] PanStamp technical details, lagarto SWAP, https://code.google.com/p/PanStamp/wiki/LagartoSWAP,Retrieved 25 March 2015.

[35] PanStamp technical details, lagarto MAX, https://code.google.com/p/PanStamp/wiki/LagartoMAX, Retrieved 25 March 2015.

[36] Zolertia Official website, http://www.zolertia.com , Retrieved March 2015

[37] Raspberry Pi official website, help and FAQ http://www.raspberrypi.org/help/what-is-a-raspberry-pi/, http://www.raspberrypi.org/help/faqs/ Retrieved 25 March 2015.

[38] Lagarto on the RaspberryPI, https://github.com/PanStamp/PanStamp/wiki/Lagarto-on-the-RaspberryPI , Retrieved March 2015

[39] 3GPP LTE http://www.3gpp.org/technologies, Retrieved March 2015

[40] Sigfox official Website, http://sigfox.com/en/announcement/ , Retrieved March 2015

[41] OpenSense official website, http://open.sen.se/, Retrieved 25 March 2015.

[42] V.Karagiannis. "Building a Testbed for the Internet of Things" B.A thesis, Alexander Technological Educational Institute of Thessaloniki, Thessaloniki, April, 2014

[43] G.Genovese "Heterogeneous M2M Networking based on Wireless Sensor Network " M.A thesis, Universita degli Studi Mediterranea di Reggio Calabria, Calabria, 2014

[44] MB1010 LV-MaxSonar®-EZ1™ High Performance Ultrasonic Rangefinder, http://www.maxbotix.com/Ultrasonic_Sensors/MB1010.htm, Retrieved 25 March 2015

[45] Telefonica Official Website, http://www.telefonica.com/en/home/jsp/home.jsp , Retrieved 25 March 2015

[46] Huawei E3276 features, http://consumer.huawei.com/en/mobile-broadband/dongles/features/e3276-en.htm, Retrieved 25 March 2015

[47] TD 1208 Evaluation Board, https://developers.insgroup.fr/cloud-on-chip/td1208/evb.html, Retrieved 25 March 2015

[48] TD 1208 Evaluation Board Downloads, https://developers.insgroup.fr/cloud-on-chip/td1208/download.html, Retrieved 25 March 2015

[49] Sigfox Backend, https://backend.sigfox.com/, Retrieved 25 March 2015

[50] Sigfox Backend Callback API, https://backend.sigfox.com/apidocs/callback, Retrieved 25 March 2015

[51] *How to run Lagarto on Raspberry Pi* , https://code.google.com/p/PanStamp/wiki/RaspberryPi, Retrieved 25 March 2015

[52] Internet of things, http://upload.wikimedia.org/wikipedia/commons/5/5a/Internet_of_Things.png, Retrieved 25 March 2015

[53] Z. P. V. Cackovic, 'Abstraction and Semantics support in M2M communications', pp 404–408, 2013.

Appendix A : "SmartWorld" project – Developer`s Manual

A.1. PanStamp Motes

A1.1.1 Hardware

PanStamp is an open source project created for the enthusiasts that love measuring and controlling things wirelessly. PanStamps are small wireless boards specially designed to fit in low-power applications, simple to program and simple to work with. With panStamps, you can measure almost everything by simply connecting your panStamp to the sensors, placing a battery and sending wireless data from the first moment.

The core of the panStamp project is, of course, panStamps. These small wireless boards are extremely programmable and configurable. They can be plugged into any of the available base boards containing sensors and actuators, or new boards can be developed according to your needs. Base boards generally remain quite simple since most of the electronics are contained in panStamps. The boards run a compact stack and communicate with each other using a very simple protocol called SWAP over 868/915 MHz.

Hardware specifications

Size:	0.7 x 1.2 in (17.7 x 30.5 mm)
MCU:	Atmel Atmega328P at 8MHz
Flash:	32 KB
RAM:	2 KB
EEPROM:	1 KB
RF frontend:	TI CC1101
Frequency bands:	868/915 MHz
Operating voltage:	from 2.5 VDC to 3.6 VDC
Current consumption:	1 uA when in deep sleep mode. 2.5 mA whilst transmitting

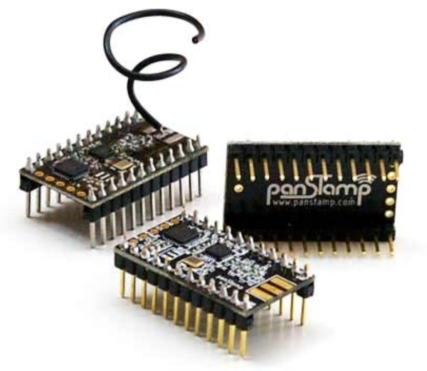


Figure A.1PanStamp

PanStamp also offers Sensors and Battery Board Modules where the Wireless PanStamp module has a socket to be connected.

Software Development Environment

PanStampscan be programmed using the Arduino IDE (http://arduino.cc/en/main/software). Before programming the panStampsthe appropriate board has to be configured in the IDE.

Tools>Board> Arduino pro or pro mini (3.3V) ATmega328

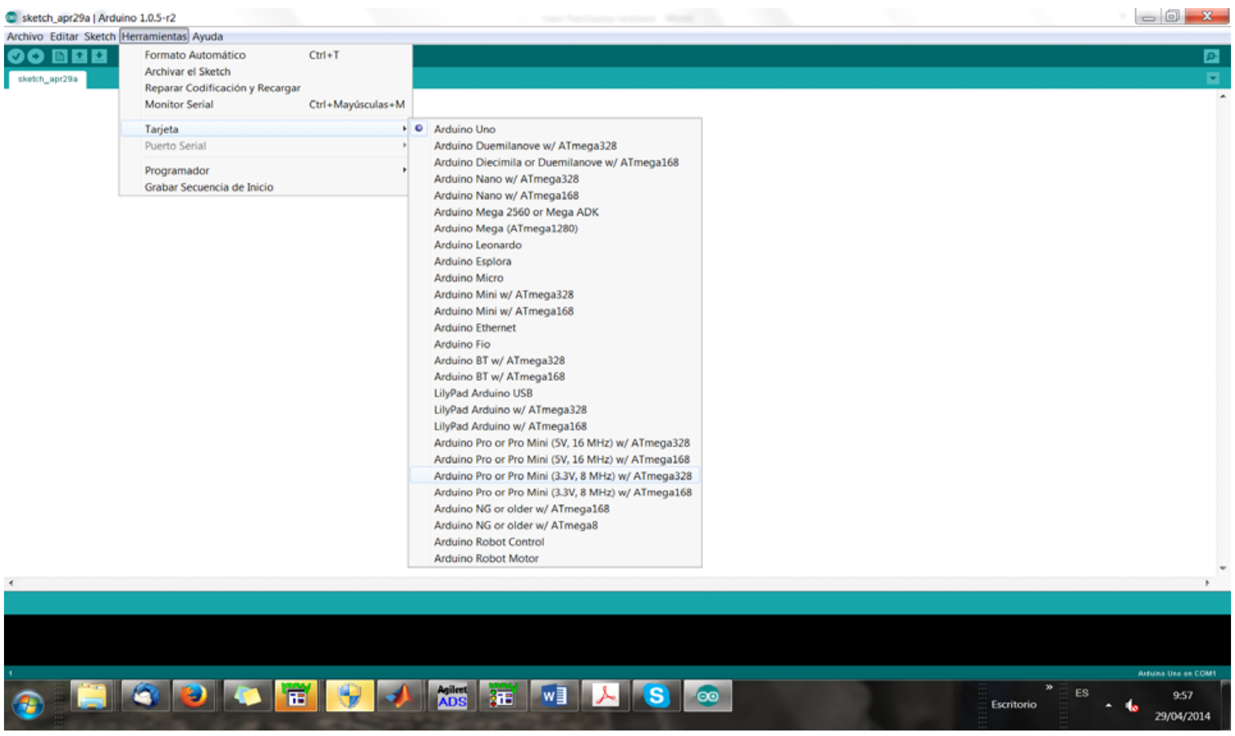

sketch_apr29a | Arduino 1.0.5-r2
Archivo Editar Sketch Herramientas Ayuda
sketch_apr29a
Formato Automático Ctrl+T
Archivar el Sketch
Reparar Codificación y Recargar
Monitor Serial Ctrl+Mayúsculas+M
Tarjeta
Puerto Serial
Programador
Grabar Secuencia de Inicio
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor
Arduino Uno on COM1
Escritorio
ES
9:57
29/04/2014

Figure A.2Arduino IDE

Software compatible with Arduinos

If you have ever played with Arduinos you will find programming panStamps quite simple. panStamps are, in fact, Arduino clones so you will be able to use any of the existing libraries for Arduino on the panStamps too.

How to upload your Arduino code in your PanStamps– STEPS

- Download Arduino 1.5
- Install Arduino 1.5
- Download PanStamp-Arduino files
- Unzip panStamp arduino files
- Move the folder "arduino-1.5_patch" to the directory where Arduino-1.5 is installed.
- Open Command Prompt (in Windows) and go to "Arduino-1.5_patch" folder
- Run: install.bat in order to Enable PanStamp or you can make right click to file "install.bat" and executed as administrator.
- After this make a reboot to your pc.
- Connect the panStamp to the PanStick.
- Insert the panStick to a USB port in your pc.
- Now, you go to "Arduino-1.5_Patch/libraries/" and copy all the folder. Paste them to the Arduino/libraries folder (in Program Files/Arduino/libraries).
- Open "Smartparking_prem.ino" file.
- Tools -> Boards -> arduino pro or pro mini
- Tools -> processor -> ATmega328(3.3, 8Hz)
- Port -> COM4 (go to devices and check in which port is configured the panStick. Usually, is at COM3. But you have to change it, to "COM4").
- Click on "Verify/Compile"
- Click on "Upload"
- You can disconnect panStick. Now your panStamp is programmed.

*** If you want to upload code for a different mote, you have to we convert the latitude and longitude from decimal to hex and replace them to the Coordinates0…coordinates 7. Also, you have to change the PanStamp address “panStamp.cc1101.setDevAddress(3);”& the Coordinates (convert them from Decimal to Hexadecimal”

Mote – Address	Latitude	longitude	
1	41.372332	02 77 4A AC	Smartparking_prem.ino
	2.145875	00 20 BE 53	
2	41.374828	02 77 54 6C	Smartparking_prem.ino
	2.173618	00 21 2A B2	
3	41.381007	02 77 6C 8F	Smartparking_prem.ino
	2.193371	00 21 77 DB	with MAXSonar
4	41.403678	02 77 C5 1E	Smartparking_prem.ino
	2.184187	00 21 53 FB	

Table A.1 Mote address - Coordinates Mapping

A.1.2 Open wireless communications

Sensor values and actuator statuses are collected by panStamps and transmitted over the air. Then, a Lagarto server typically processes this data and delivers it to the IP world using different mechanisms.

Lagarto is written using Python and has an open architecture. New Lagarto servers and clients can be developed using any language. Moreover, Lagarto servers can be queried using simple HTTP GET/POST commands so communicating with existing applications should not be a problem. As result, we can say that your ideas can be developed on both sides of the above diagram: the low-power wireless network and the IP side. PanStamp code examples implementing the SWAP protocol are available at the download section <u>http://code.google.com/p/panStamp/wiki/enddevices</u>

A.1.3 PanStick

PanStick is a USB mother board for panStamps. Used to program panStamps, it also acts as a serial gateway to the wireless network. Simply place a panStamp on the panStick, program the panStamp with the modem application, and plug the dongle to your computer.

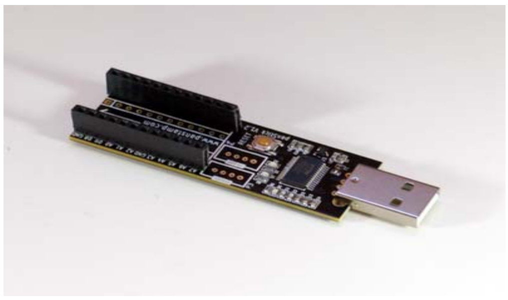


Figure A.3PanStamp USB Panstick

PanStick is also a compact development board. Thanks to the available contacts, you can solder pin headers under the board and plug the panStick and panStamp dongle on a breadboard, having access to panStampspins for your developments. You will then be able to connect your sensors directly to your panStick whilst keeping your panStamp accessible via USB.

PanStick includes an on-board voltage regulator which takes the power directly from the USB bus. Thanks to this regulator, panStick is able to power your sensors up to 250 mA always at 3.3 VDC. The USB board also includes a reset button that will let you restart your application without having to unplug the board.

A.1.4. Serial Monitor

The serial monitor is a feature of the Arduino IDE. It displays serial data being sent from the board. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial. Begin in your sketch. Note that on Mac or Linux, the Arduino board will reset (rerun your sketch from the beginning) when you connect with the serial monitor. The serial port can also be used for debugging logical errors.

A.1.5 PanStamp Code

Arduino Language is C++. It uses of two main functions:

- Setup: code that is being executed once when the panStamp is powered on (initialize panStamp, pin configuration (in/out), assigning the panStamp address (1-255)).
- Loop: code that is being executed continuously as long as it is connected to a power source (gather sensor measurements).

A.2 Raspberry Pi devices

A.2.1 Raspberry Pi B

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

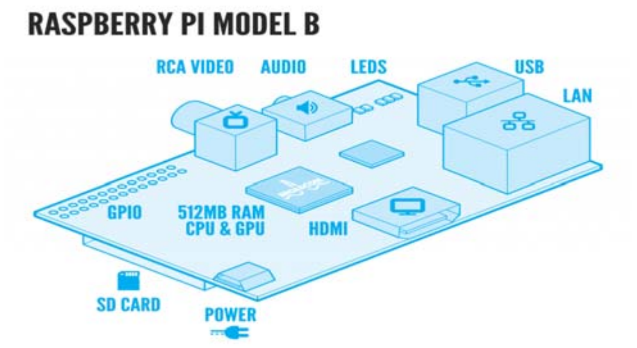


Figure A.4 Raspberry Pi B+

Developer	Raspberry Pi Foundation
Type	Single-board computer
Release date	29 February 2012
Introductory price	US$ 25 (model A) and US$ 35 (model B)
Operating system	Linux (Raspbian, Debian GNU/Linux, OpenELEC, Fedora, Arch Linux ARM, Gentoo), RISC OS, FreeBSD, NetBSD, Plan 9, Inferno, Openwrt
Power	2.5 W (model A), 3.5 W (model B)
CPU	ARM1176JZF-S (ARMv6k) 700 MHz
Memory	512 MB (Model B rev 2)
Storage	SD card slot
Graphics	Broadcom VideoCore IV
Website	www.raspberrypi.org

A.2.2 Raspberry PI B+

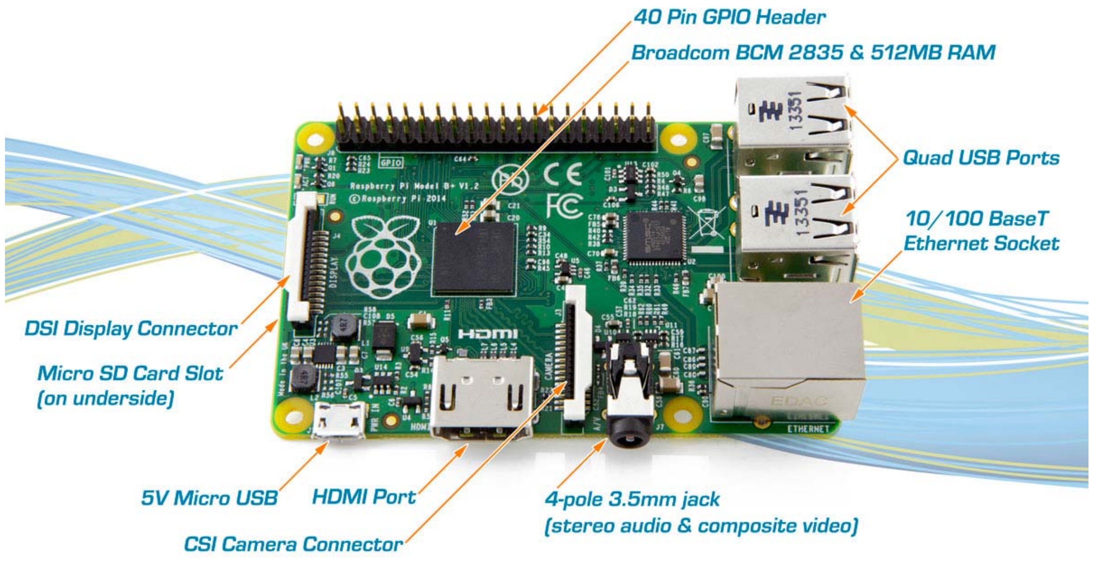

40 Pin GPIO Header
Broadcom BCM 2835 & 512MB RAM
Quad USB Ports
10/100 BaseT Ethernet Socket
DSI Display Connector
Micro SD Card Slot (on underside)
5V Micro USB
HDMI Port
CSI Camera Connector
4-pole 3.5mm jack (stereo audio & composite video)

Figure A.5 Raspberry Pi B+ (2)

The Model B+ is the higher-spec variant of the Raspberry Pi. It replaced the original Model B in July 2014. Compared to the Model B it has:

- **More GPIO**. The GPIO header has grown to 40 pins, while retaining the same pinout for the first 26 pins as the Model B.
- **More USB**. We now have 4 USB 2.0 ports, compared to 2 on the Model B, and better hotplug and overcurrent behaviour.
- **Micro SD**. The old friction-fit SD card socket has been replaced with a much nicer push-push micro SD version.
- **Lower power consumption**. By replacing linear regulators with switching ones we've reduced power consumption by between 0.5W and 1W.
- **Better audio**. The audio circuit incorporates a dedicated low-noise power supply.
- **Neater form factor**. We've aligned the USB connectors with the board edge, moved composite video onto the 3.5mm jack, and added four squarely-placed mounting holes.

A.3 Start the Development

What we need to start:

- Ubuntu, as Operating System.
- PC with SD card reader.
- PC with access to the Internet (recommended Ethernet)

A.3.1 INSTALL NOOBS AND OS IN RPi

It will be useful to install NOOBS in the SD Card. With NOOBS is easy to install or restore the Raspbian (OS for RPi) http://www.raspberrypi.org/help/noobs-setup/

Advanced users may wish to install a specific Operating System image. Download an image below and follow the image installation guides in our documentation.

Recommend to install "Raspbian Wheezy". http://www.raspberrypi.org/downloads/

- Download NOOBS
- Extract the zip file
- Copy files for the folder
- Paste them to a new SD card
- Insert the SD card to Raspberry Pi B+
- Check "Raspbian"
- Install

A.3.2 LOGIN INTO THE RPi

Username: pi

Password: raspberry

Load GUI: startx -> press Enter

If the PanStamp and RPi are new, then start from here *https://code.google.com/p/panStamp/wiki/firststeps*

INSTALL LAGARTO SERVERS TO RPi

To install Lagarto-MAX and Lagarto-SWAP from the scratch we follow this website *https://code.google.com/p/panStamp/wiki/lagarto*

Set up the RPi's UART

1. Back up and change /boot/cmdline.txt.

```
cp /boot/cmdline.txt /boot/cmdline_backup.txt
sed -i 's/ [^ ]*ttyAMA0[^ ]*//g' /boot/cmdline.txt
```

2. Back up and change /etc/inittab.

```
cp /etc/inittab /etc/inittab_backup
sed -i 's/^\(.*ttyAMA0\)/#\1/' /etc/inittab
```

3. Remove the I2C interface from the module blacklist.

```
sed -i 's/^\(.*i2c-bcm2708\)/#\1/' /etc/modprobe.d/raspi-blacklist.conf
```

4. Enable the HWClock at boot.

```
update-rc.d hwclock.sh enable
```

5. Disable the fake clock.

```
update-rc.d fake-hwclock remove
```

Reboot

Install Pyswap & Python Applications.

Download and extract the pyswap and rest of python applications for PanStamp

source: *www.panStamp.org/downloads/panStamp_python_2.6.zip*

Install Pyserial

Download Pyserial (In case you have Raspbian Wheezy, you should download version 2.7).

```
wget "https://pypi.python.org/packages/source/p/pyserial/pyserial-2.7.tar.gz"
tar xvfz pyserial*.gz
cd pyserial*
sudo python setup.py install
```

Source: *https://pypi.python.org/pypi/pyserial*

Install ZeroMQ

There are two ways to install ZeroMQ:

```
Sudo apt-get install libzmq-dev
```

source: *http://zeromq.org/distro:debian*

*install libzmq-dev for series 2.2 (because we have Wheezy)

```
apt-get install uuid-dev libtool autoconf automake pkg-config build-essential
wget "http://download.zeromq.org/zeromq-2.2.0.tar.gz"
```

```
tar xvfz zeromq*.gz
cd zeromq*
./configure
sudo make
sudo make install
ldconfig
```

Install Python Bindings

```
Sudo apt-get install libpgm-5.1-0
Sudo apt-get check
Sudo apt-get install python-dev
wget "https://pypi.python.org/packages/source/p/pyzmq/pyzmq-2.0.2.tar.gz"
tar xvfz pyzmq*.gz
cd pyzmq*
sudo python setup.py configure
sudo python setup.py install
```

Source: http://zeromq.org/bindings:python

Install Barrel

```
wget "https://pypi.python.org/packages/source/b/barrel/barrel-0.1.3.tar.gz"
tar xvfz barrel*.gz
cd barrel*
sudo python setup.py install
```

Install pyephem

```
wget "https://pypi.python.org/packages/source/p/pyephem/pyephem-3.7.5.3.tar.gz"
tar xvfz pyephem*.gz
cd pyephem*
sudo python setup.py install
```

enable Modules

```
sudo nano /etc/modules
spi-bcm2708
```

i2c-bcm2708

i2c-dev

ipv6

sudo adduser pi i2c

Reboot

Network configuration and DNS configuration

Change the Network Configuration

Sudo nano /etc/network/interfaces

Auto eth0

Iface eth0 inet static

Address 10.1.3.53

Gateway 10.1.3.1

Netmask 255.255.255.0

Network 10.1.3.0

Broadcast 10.1.3.255

Change DNS configuration

Sudo nano /etc/resolf.conf

Nameserver 84.88.62.194

Nameserver 84.88.62.220

Save and exit. (ctrl + C , Yes to Save)

Reboot your RPi

Update your RPi

Between each Raspberry Pi, Lagarto applications will probably get updated. To update your applications, ssh to the Raspberry Pi and run the following command to install svn (subversion) which is needed on the Raspberry Pi to update Lagarto. This only needs to be run once.

sudo aptitude install subversion

```
cd ~/panStamp/lagarto/
# If the version is very old, you might need to run
sudo svn upgrade
# Then update
sudo svn update
```

Activate the Sensors & de-activate Updates of Devices

Insert the batteries to the PanStamp sensors and locate them somewhere close to the RPi. (RPi must be off when you do this). If the Servers start for first time push “RESET/SYNC” button in the sensors.

Declare the Devices you have:

lagarto/lagarto-swap/config/devices/devices.xml

```
<developer id="2" name="developerName">
<dev id="1" name="temphum12" label="TempHumVolt2Dinput"/>
<dev id="2" name="temphum13" label=”Temp13”/>
</developer>
```

De-activate the “Update” of devices

lagarto/lagarto-swap/config/settings.xml

Change the “*Update*” to “*False*”

A.4 Access and test the servers

**** Start first Lagarto-MAX and then Lagarto-SWAP

A.4.1 Lagarto-MAX

Source: https://code.google.com/p/panStamp/wiki/LagartoMAX

Start the server:

```
sudo python ~/panStamp/lagarto/lagarto-max/lagarto-max.py
```

To access the servers, run a browser on the same computer or on another one in the local network and type the following URL: http://10.1.3.53:8002

A.4.2 Lagarto-SWAP

Source: https://code.google.com/p/panStamp/wiki/LagartoSWAP

Start the server: *sudo python ~/panStamp/lagarto/lagarto-swap/lagarto-swap.py*

To access the servers, run a browser on the same computer or on another one in the local network and type the following URL: *http://10.1.3.53:8001*

Note: use "&" in the end of the command to make it run in background

Stop the servers:

sudo pkill python

jobs –l (find the process numbers of the servers)

sudo kill -15 process_numbers ("-15" will ask the processes to stop, use "-9" if you want to force them to be terminated)

How to use the servers

Once the servers are running, they will autodetect any PanStamp that start transmitting SWAP messages. If the panStamp's address is not stored in the server, details about the panStamp, its address and its registers will appear in the server's logs, otherwise only the panStamps ID address will appear. The panStamp can update the servers extremely fast but the servers interface updates itself every 3 seconds.

When powering off a PanStamp, it stops updating the servers, but information about its ID address and its registers are still available in the server. This information can be deleted manually from the server's interface.

Lagarto servers use two xml files to autocreate a new device. The files are stored at

~/PanStamp/lagarto/lagarto-swap/config/ devices

- *devices.xlm:* contains all the available applications, its developers ID and its products ID. These values have to be written in the panStamp code as well in the class product.h
- *application_name.xlm:* contains details for all the registers that are used by the panStamp, their names, how many bits are using. (Template.xlm, in our case)

Details about the xml files can be found at the official site.

http://code.google.com/p/panStamp/wiki/devicexml

Note: The following files located in the Lagarto-max directory were modified by adding a few additional lines in the end in order to improve the functionality of the servers.

api.py:	poll and send values to the cloud
clouding.py:	get lagarto values, get opensense values, set lagarto values
webscripts.py:	file for triggers and events

There are two ways to make the raspberry act as a gateway for the panStamps:

- **USB**: Load the modem application to the panStamp and use the panstick to attach it to the raspberryPI.
- **GPIO**: Plug a Raspberry Pi shield to the GPIO. The panStamp soldered to the Raspberry Pi shield is preloaded with the modem application.

Note: When switching from using the GPIO to using the USB, the file *"~/panStamp/lagarto/lagarto-swap/config/serial.xml*" has to be modified from *"ttyAMA0" to "ttyUSB0"*.

A.5 Cloud Platform

The cloud platform that was selected to connect the M2M servers and the applications is called open sense "http://open.sen.se"

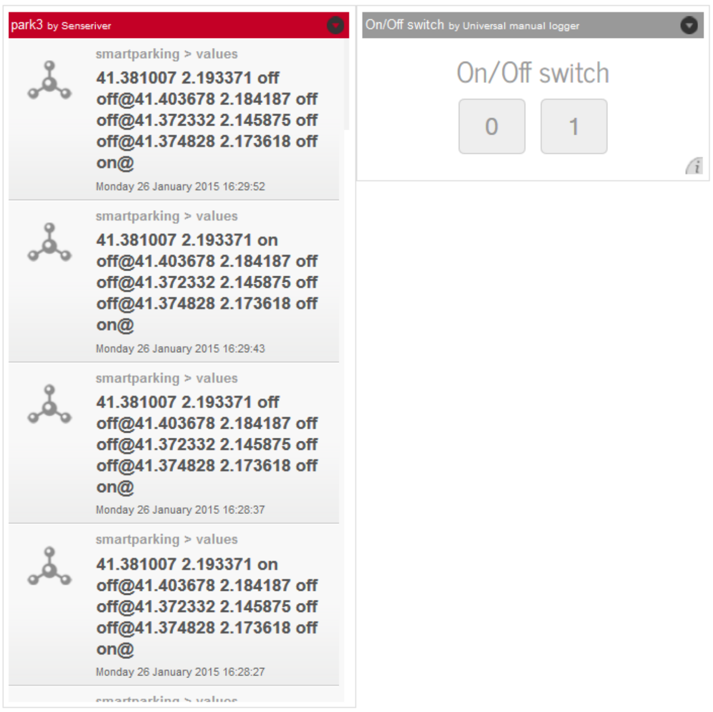
User:	*******@******.com
Password:	******

Once logged in, the already created devices can be found at >Channel>mydevices

To Create and ADD new devices >Channels>Custom made devices >register >get started

The communication among the servers, the cloud and the applications is achieved through http requests (GET,POST). For security purposes every http request must contain a unique key in the header of each http packet.

Once logged in, you will see the current applications.


park3 by Senseriver

smartparking > values
41.381007 2.193371 off
off@41.403678 2.184187 off
off@41.372332 2.145875 off
off@41.374828 2.173618 off
on@
Monday 26 January 2015 16:29:52

smartparking > values
41.381007 2.193371 on
off@41.403678 2.184187 off
off@41.372332 2.145875 off
off@41.374828 2.173618 off
on@
Monday 26 January 2015 16:29:43

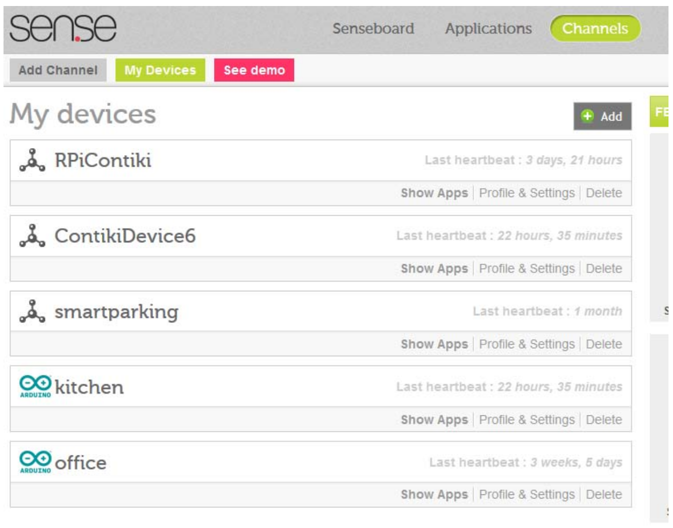
smartparking > values
41.381007 2.193371 off
off@41.403678 2.184187 off
off@41.372332 2.145875 off
off@41.374828 2.173618 off
on@
Monday 26 January 2015 16:28:37

smartparking > values
41.381007 2.193371 on
off@41.403678 2.184187 off
off@41.372332 2.145875 off
off@41.374828 2.173618 off
on@
Monday 26 January 2015 16:28:27

On/Off switch by Universal manual logger

Figure A.6 Open.Sen.Se Senseboard

In the Figure above, can observe the coordinates of 4 different PanStamps(spots), if "Sensor_in" is on/off and if "premium" is on/off.



My devices

RPiContiki — Last heartbeat : 3 days, 21 hours — Show Apps | Profile & Settings | Delete

ContikiDevice6 — Last heartbeat : 22 hours, 35 minutes — Show Apps | Profile & Settings | Delete

smartparking — Last heartbeat : 1 month — Show Apps | Profile & Settings | Delete

kitchen — Last heartbeat : 22 hours, 35 minutes — Show Apps | Profile & Settings | Delete

office — Last heartbeat : 3 weeks, 5 days — Show Apps | Profile & Settings | Delete

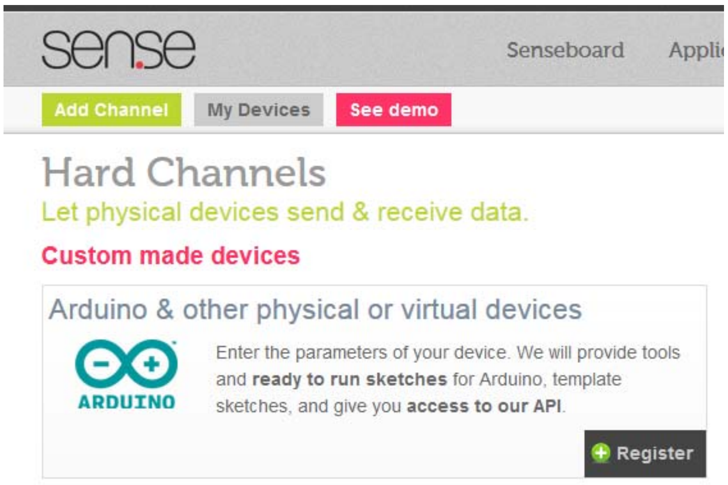
Figure A.7 Open.Sen.Se devices

Channels-> My Devices

RPi Contiki = RPi B+ (Elisavet)

ContikiDevice6 = RPi (Giacomo)

Smartparking = RPi (Vasilis)


Senseboard
Add Channel
My Devices
See demo

Hard Channels

Let physical devices send & receive data.

Custom made devices

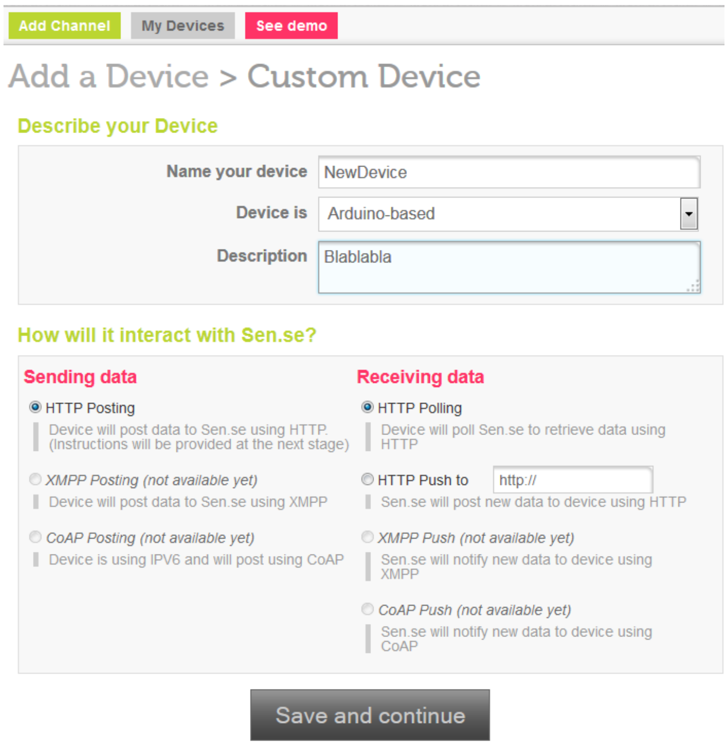
Arduino & other physical or virtual devices

Enter the parameters of your device. We will provide tools and **ready to run sketches** for Arduino, template sketches, and give you **access to our API**.

Register

Figure A.8 Open.Sen.Se Get started

Add a new Channel -> Arduino & other physical or virtual devices -> Register -> Get Started


Add Channel
My Devices
See demo

Add a Device > Custom Device

Describe your Device

Name your device	NewDevice
Device is	Arduino-based
Description	Blablabla

How will it interact with Sen.se?

Sending data

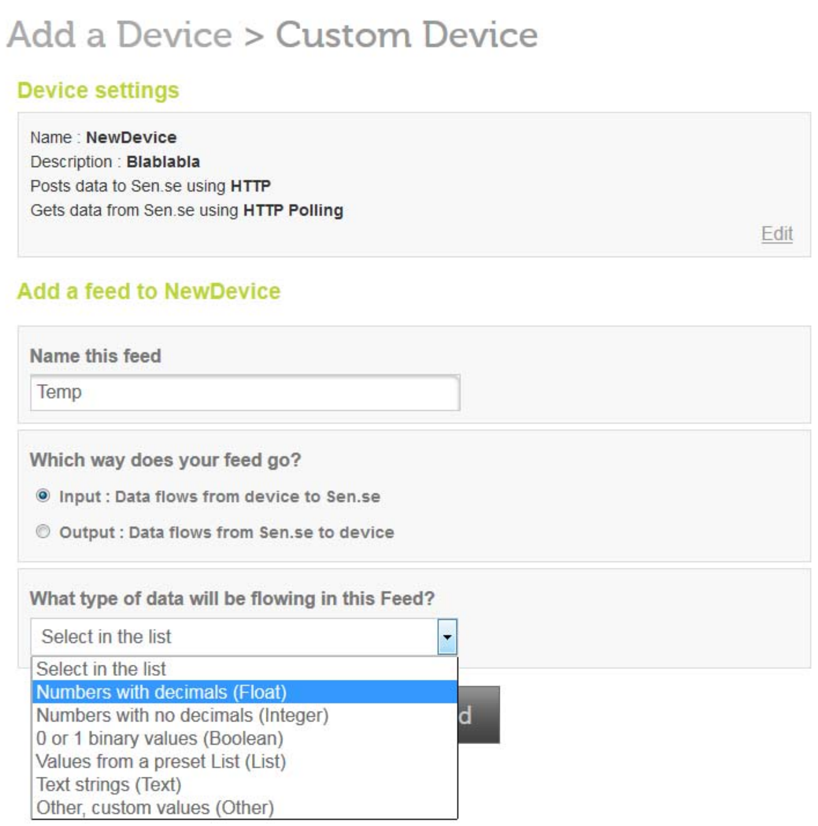
- [✓ HTTP Posting] Device will post data to Sen.se using HTTP. (Instructions will be provided at the next stage)
- *XMPP Posting (not available yet)* Device will post data to Sen.se using XMPP
- *CoAP Posting (not available yet)* Device is using IPV6 and will post using CoAP

Receiving data

- [✓ HTTP Polling] Device will poll Sen.se to retrieve data using HTTP
- HTTP Push to http:// Sen.se will post new data to device using HTTP
- *XMPP Push (not available yet)* Sen.se will notify new data to device using XMPP
- *CoAP Push (not available yet)* Sen.se will notify new data to device using CoAP

Save and continue

Figure A.9 Open.Sen.Se Custom Devices



Add a Device > Custom Device

Device settings

Name : **NewDevice**
Description : **Blablabla**
Posts data to Sen.se using **HTTP**
Gets data from Sen.se using **HTTP Polling**

Edit

Add a feed to NewDevice

Name this feed

Temp

Which way does your feed go?

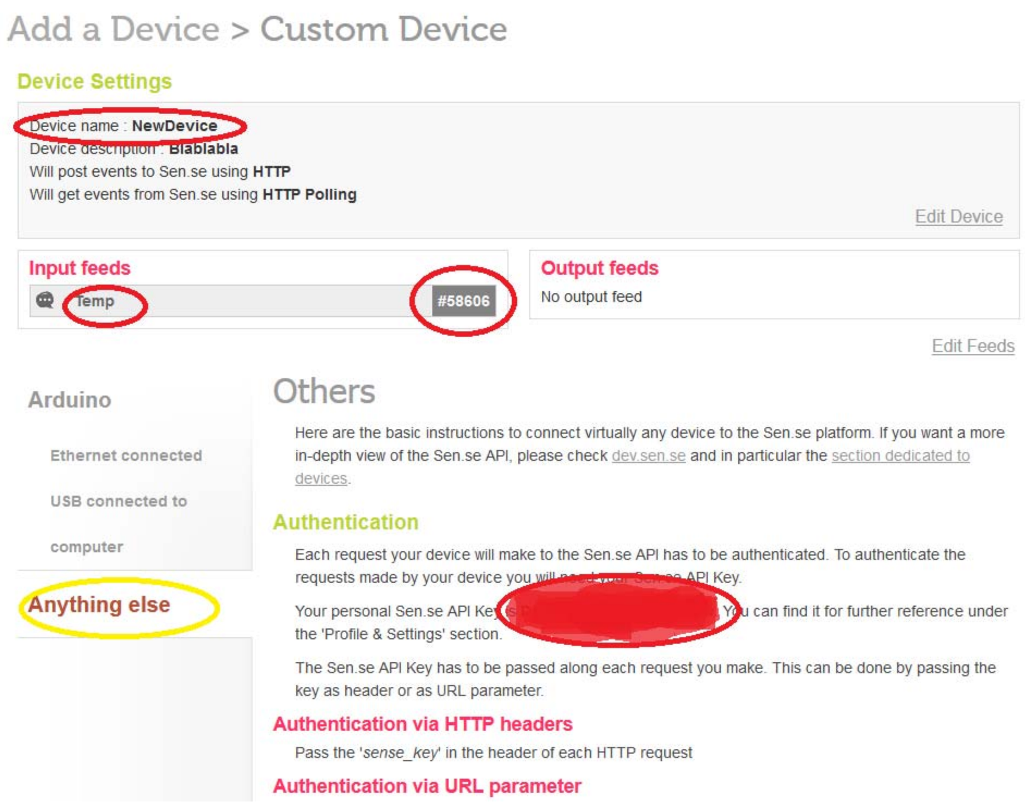
- [✓ **Input : Data flows from device to Sen.se**]
- **Output : Data flows from Sen.se to device**

What type of data will be flowing in this Feed?

Select in the list

- Select in the list
- Numbers with decimals (Float)
- Numbers with no decimals (Integer)
- 0 or 1 binary values (Boolean)
- Values from a preset List (List)
- Text strings (Text)
- Other, custom values (Other)

Name the feed (for the application) and the type of it. We choose "Input" and then according to the type of data we choose the "type of the Feed". -> Continue -> Finish



Add a Device > Custom Device

Device Settings

Device name : **NewDevice**
Device description : **Blablabla**
Will post events to Sen.se using **HTTP**
Will get events from Sen.se using **HTTP Polling**

Edit Device

Input feeds

Temp #58606

Output feeds

No output feed

Edit Feeds

Arduino

Ethernet connected

USB connected to computer

Anything else

Others

Here are the basic instructions to connect virtually any device to the Sen.se platform. If you want a more in-depth view of the Sen.se API, please check dev.sen.se and in particular the section dedicated to devices.

Authentication

Each request your device will make to the Sen.se API has to be authenticated. To authenticate the requests made by your device you will need your Sen.se API Key.

Your personal Sen.se API Key [redacted] You can find it for further reference under the 'Profile & Settings' section.

The Sen.se API Key has to be passed along each request you make. This can be done by passing the key as header or as URL parameter.

Authentication via HTTP headers

Pass the '*sense_key*' in the header of each HTTP request

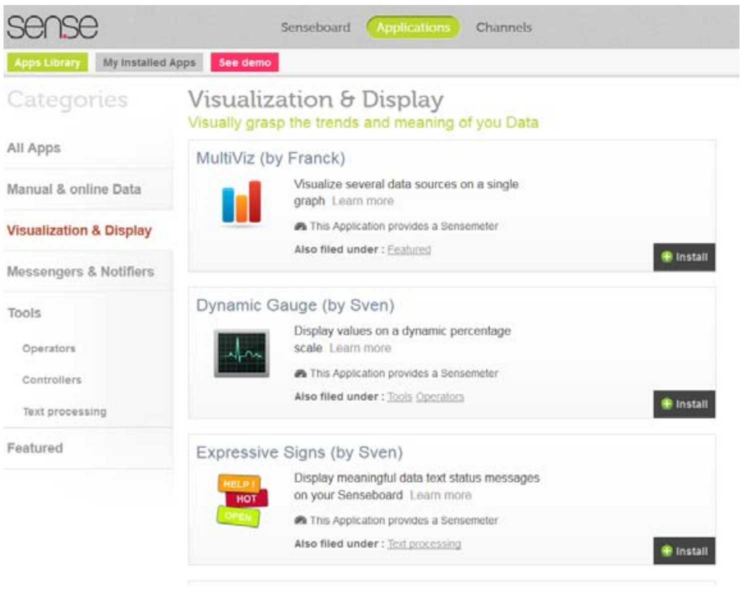
Authentication via URL parameter

Note these:

Input`s feed “Name” and code “#” and “sen.se API key”

Create an Application in Open.Sen.Se

Applications -> Apps Library-> Choose what you need. It this case we choose “SenseRiver”.



Visualization & Display

Visually grasp the trends and meaning of you Data

MultiViz (by Franck)

Visualize several data sources on a single graph Learn more

This Application provides a Sensemeter

Also filed under : Featured

Dynamic Gauge (by Sven)

Display values on a dynamic percentage scale Learn more

This Application provides a Sensemeter

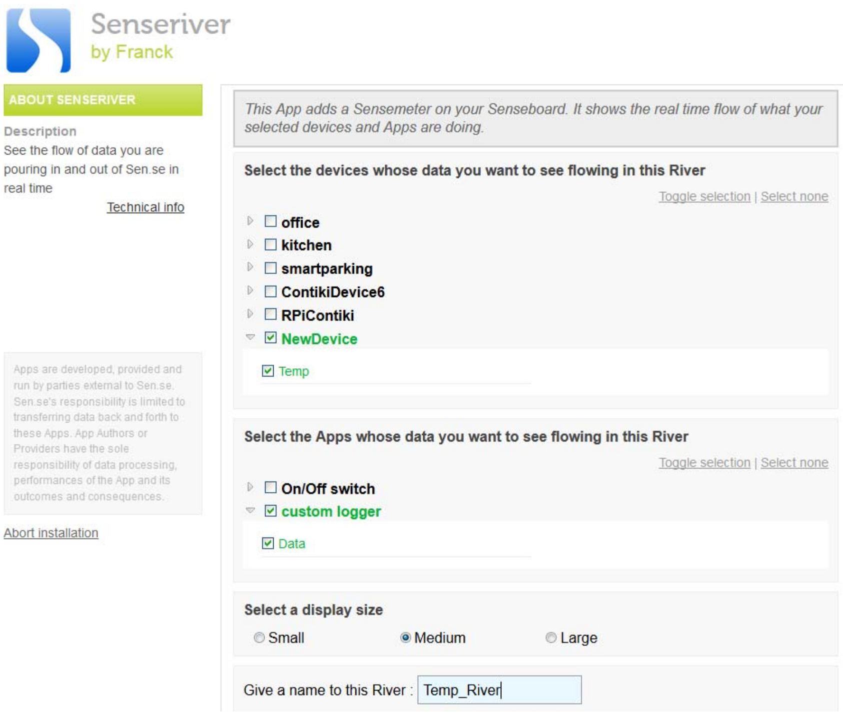
Also filed under : Tools Operators

Expressive Signs (by Sven)

Display meaningful data text status messages on your Senseboard Learn more

This Application provides a Sensemeter

Also filed under : Text processing



Description

See the flow of data you are pouring in and out of Sen.se in real time

Apps are developed, provided and run by parties external to Sen.se. Sen.se's responsibility is limited to transferring data back and forth to these Apps. App Authors or Providers have the sole responsibility of data processing, performances of the App and its outcomes and consequences.

This App adds a Sensemeter on your Senseboard. It shows the real time flow of what your selected devices and Apps are doing.

Select the devices whose data you want to see flowing in this River

- ☐ office
- ☐ kitchen
- ☐ smartparking
- ☐ ContikiDevice6
- ☐ RPiContiki
- ☑ NewDevice
  - ☑ Temp

Select the Apps whose data you want to see flowing in this River

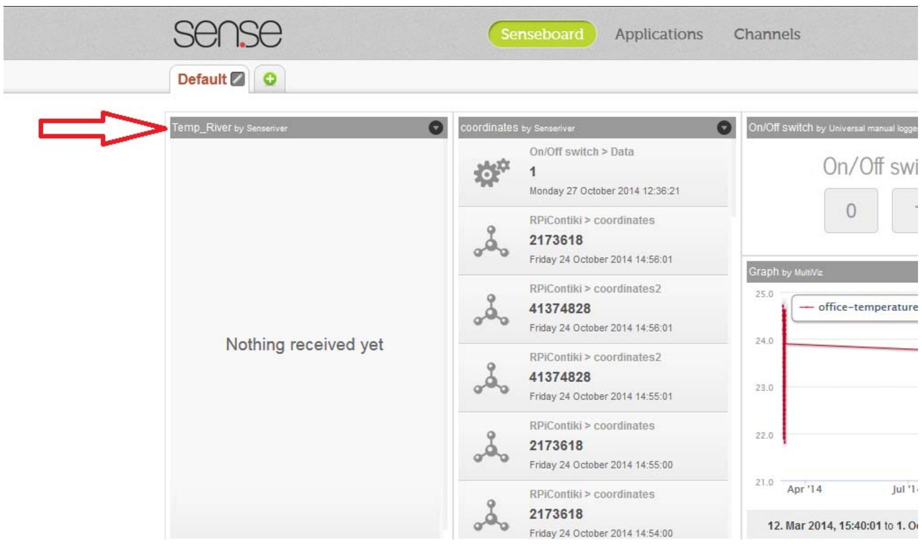
- ☐ On/Off switch
- ☑ custom logger
  - ☑ Data

Select a display size

Small [✓ Medium] Large

Give a name to this River : Temp_River

Choose the "Device" (New Device) and the "Feed" (Temp). Name the application. -> Complete installation -> Save



Temp_River by Senseriver

Nothing received yet

coordinates by Senseriver

On/Off switch > Data
1
Monday 27 October 2014 12:36:21

RPiContiki > coordinates
2173618
Friday 24 October 2014 14:56:01

RPiContiki > coordinates2
41374828
Friday 24 October 2014 14:56:01

RPiContiki > coordinates2
41374828
Friday 24 October 2014 14:55:01

RPiContiki > coordinates
2173618
Friday 24 October 2014 14:55:00

RPiContiki > coordinates
2173618
Friday 24 October 2014 14:54:00

On/Off switch by Universal manual logge

On/Off swi

0

Graph by MultiViz

office-temperature

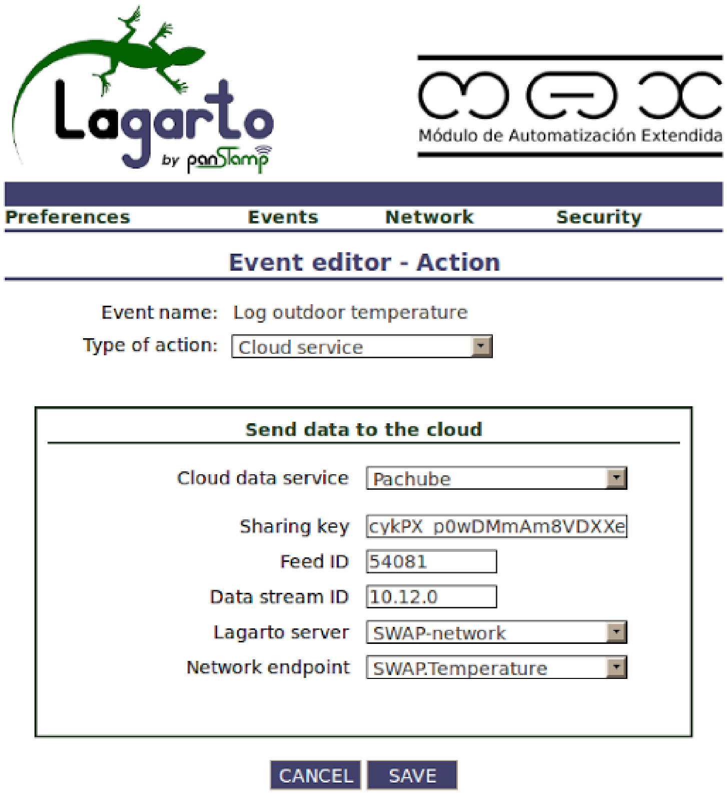
12. Mar 2014, 15:40:01 to 1. O

The New applications have showed to the Senseboard.

You can see your API key -> Profile & Settings-> API Key

Now, access via a browser the Lagarto-MAX. (http://10.1.3.53:8002)

Help: https://code.google.com/p/panStamp/wiki/LagartoMAX



Event editor - Action

Event name: Log outdoor temperature
Type of action: Cloud service

Send data to the cloud

Cloud data service	Pachube
Sharing key	cykPX_p0wDMmAm8VDXXe
Feed ID	54081
Data stream ID	10.12.0
Lagarto server	SWAP-network
Network endpoint	SWAP.Temperature

Type of Action: Cloud Service

Cloud data service:Open.sen.se

Sharing key: Sense Key

Feed ID: Code of the Feed (from the Open.sen.se)

A.6 Android Applications

Two applications were developed targeting the Android OS.

SmartParking: Application that receives information about parking spots from the Lagarto server and displays them in a map using the Google maps API. The server, when it detects a change in the spots, updates the cloud using a POST request. The application updates itself by sending a GET request to the cloud. The parking spaces are noted in the map using different colors depending on the spot availability.

GeoFence: Application that changes a value in the Lagarto server judging from the geological position of the android device. The application asks for the necessary information (coordinates, security key, endpoint ID) and every time the geological position fulfills certain conditions, it sends a POST request to the cloud. The Lagarto server is configured to poll values from the cloud every minute (configurable) and detects the change.

A.6.1 Google maps android API key version 2

A guide for implementing the Google maps android API v2 can be found at the link below, including details on acquiring the key. *https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2*

Note: the key is unique for each computer!!! if the same key is used to compile a project from another PC the application will crash.

Android Coding

To start coding in android a development environment is need. Details on installing the android development kit can be found here*: http://developer.android.com/sdk/*

A recommended tutorial for beginners including details on installing the android environment, creating android projects, installing the google maps, acquiring the API key etc can be found here: *http://www.vogella.com/tutorials/AndroidGoogleMaps/article.html*

A.7 Useful things to do/know

Install GParted in Ubuntu

To show partitions

sudo apt-get install gparted

Check drives system

Lsblk –f

SSH to Raspberry

Sudo raspi-config

Enter to “SSH”

Select “enable or disable ssh server”

Remote access to SSH from another PC

Ssh pi@10.1.3.53

Yes

Ssh

Login: …

Password:

*pi = the user of RPi.

* 10.1.3.53 = the static IP address we have in RPi

Reboot network interfaces

sudo /etc/init.d/networking restart

Reboot the RPi. I suggest two ways:

sudo shutdown –r now

sudo reboot now

Network configuration

Ifconfig –a

Start GUI Raspbian

Startx

Force reboot

Sudo reboot

Force shutdown

Useful to see if tasks running and then shut down the RPi.

Jobs –l (if there are task, this command will show them, with name and code)

Sudo kill -15 Code (to kill a specific task)

Sudo halt –h

Wait until all the LEDs except the power LED are off, then wait an additional second to make sure the SD card can finish its wear leveling tasks and write actions. You can now safely unplug the Raspberry Pi. Failure to properly shut the RPi may corrupt your SD card, and you will have to re-image it.

Install an app in Ubuntu /RPi

Sudo apt-get NameOfApp

Install pycrypto

# sudo pip install pycrypto

Install the App

Sudo apt-get install NameOfApp

Unzip a "tar.gz" file

Tar zxvf NameOfApp.tar.gz

Cd FolderOfApp

./configure

Make

Make install

Install NameOfApp

Install Git

Sudo apt-get install git

Install python

Sudo apt-get install python-setuptools

Sudo pip install virtualenv

Sudo apt-get install python-rpi.gpio

Sudo apt-get install python-smbus

Sudo apt-get install i2c-tools

Sudo apt-get install python-dev libjpeg-dev libfreetype6-dev python pip

Sudo easy-install pip

Also, install

Sudo install RPi.GPIO

Sudo install nose

Install zip unzip

Sudo apt-get zip unzip

Install a python file

Python setup.py install

Update the RPi

Sudo apt-get update

Sudo reboot

Sudo apt-get upgrade

Sudo rpi-update

How to install a package

Sudo apt-get install gdebi

Sudo dpkg –I Filename.deb

Dependences check

Sudo apt-cache showpkg vsftpd

Mount a partition in command line

Sudo mkdir /media/NameOfPartition

Sudo nano –Bw /etc/fstab

In the end of file type:

> */dev/sdb3 /media/NameOfPartition ext3 defaults o 2*
>
> Save and exit

Sudo mount –a

Refresh

Appendix B: "Smartworld" Project – User`s Manual

B.1 Smart World testbed Introduction

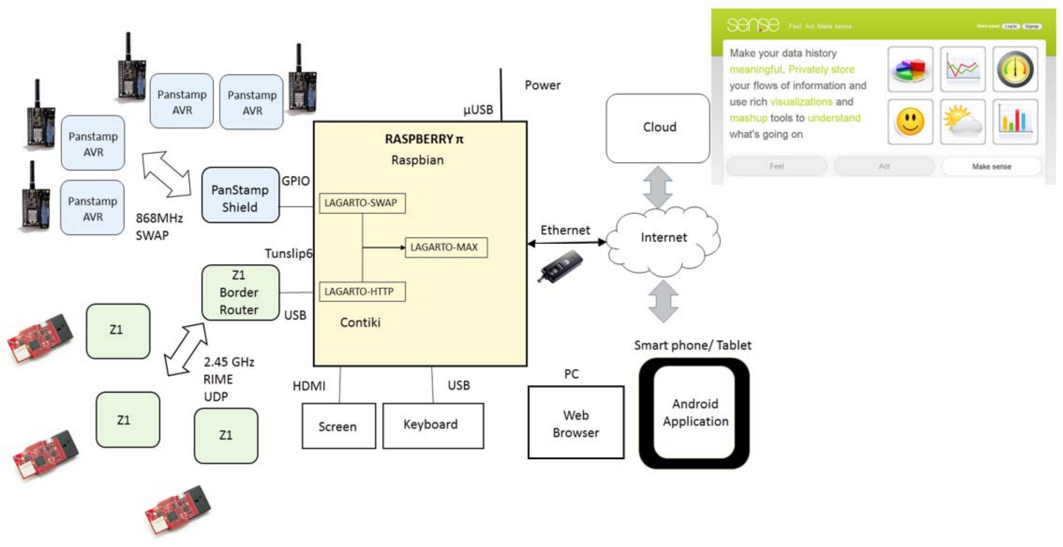


Figure B.1 Smartworld project

SmartWorld Testbed pretends to be a M2M testbed including different technologies and protocols. Different motes with different sensors ends up uploading data in a Cloud in the Internet, and from there being able with a smartphone application to use this data.

Vasileios Karagiannis was in charge of the panStamp. PanStamp provides a complete system formed by motes operating at 868MHz, which communicates with another panStamp that acts as border router. In our testbed at this moment a panStamp shield for the Raspberry Pi is being used as border router. The Raspberry Pi is the one which collects data via the servers and is able to upload this data to the cloud. PanStamp communication is done using a SWAP protocol developed by panStamp. At the raspberry there is a Lagarto- SWAP server, that collects data send by the border router, sends it to a Lagarto-Max Server, and this one takes care of the action of cloud, using OpenSense. Two android applications were created to made uploading the code to the use of all this data of OpenSense with different applications.

Giacomo Genovese added another technology to this initial testbed. He integrated Zolertia Z1 motes using Contiki and operating a 2.45GHz. In order to do this, he developed a new server called Lagarto-HTTP or Lagarto-Contiki, able to read HTTP packets received from the Z1 border Router connected to the raspberry Pi, and send them to Lagarto-MAX.

B.2 PanStamp Motes

B.2.1 Hardware

PanStamp is an open source project created for the enthusiasts that love measuring and controlling things wirelessly. PanStamps are small wireless boards specially designed to fit in low-power applications, simple to program and simple to work with. With panStamps, you can measure almost everything by simply connecting your panStamp to the sensors, placing a battery and sending wireless data from the first moment.

The core of the panStamp project is, of course, panStamps. These small wireless boards are extremely programmable and configurable. They can be plugged into any of the available base boards containing sensors and actuators, or new boards can be developed according to your needs. Base boards generally remain quite simple since most of the electronics are contained in panStamps. The boards run a compact stack and communicate with each other using a very simple protocol called SWAP over 868/915 MHz.

Hardware specifications

Size:	0.7 x 1.2 in (17.7 x 30.5 mm)
MCU:	Atmel Atmega328P at 8MHz
Flash:	32 KB
RAM:	2 KB
EEPROM:	1 KB
RF frontend:	TI CC1101
Frequency bands:	868/915 MHz
Operating voltage:	from 2.5 VDC to 3.6 VDC
Current consumption:	1 uA when in deep sleep mode. 2.5 mA whilst transmitting

PanStamp also offers Sensors and Battery Board Modules where the Wireless PanStamp module has a socket to be connected.

Lagarto is written using Python and has an open architecture. New Lagarto servers and clients can be developed using any language. Moreover, Lagarto servers can be queried using simple HTTP GET/POST commands so communicating with existing applications should not be a problem. As result, we can say that your ideas can be developed on both sides of the above diagram: the low-power wireless network and the IP side.

PanStamp Shield

Featuring a panStamp, this shield releases the computer from having to deal with the low-power wireless communications. Instead, the on-board panStamp acts as a modem connected to the RPI UART (serial port). This shield also includes a real time IC with battery backup so that we no longer depend on remote NTP servers and Internet connections to get the current time, even after an outage.

B.3 Zolertia

B.3.1 Z1 motes

Z1 mote from Zolertia is a low power sensor mote, based on Texas Instruments MSP430F2617 microcontroller and CC2420 radio, which is a 2.45GHz transceiver. It provides support to two open sources operating systems /stack to wireless sensors networks, Contiki, the one it will be used in our case, and TinyOS. The board includes a temperature sensor and a 3 axis accelerometer. It also senses the battery level, and includes 3 LEDs. It is compatible with IEEE 802.15.4, 6LowPan and Zigbee. Z1 can be connected to internet with IPv6.

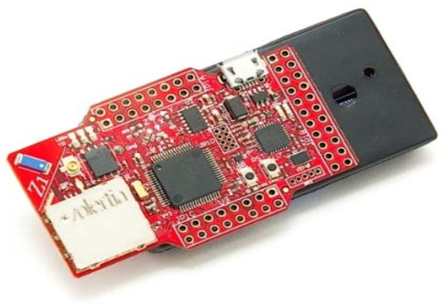


Figure B.2 Z1 mote

Z1 motes that are used now in the testbed are powered with two AA batteries. In order to power and unpower it is needed to put and remove at least one battery. When connecting the battery, if there is good power, the red led it is turn on for a second.

B.3.2 Contiki

Contiki is an open source Operating System for Internet of Things, provides powerful low power Internet communications. Contiki provides a full IP network stack, with standard IP protocols such as UDP, TCP, and HTTP, in addition to the new low-power standards like 6lowpan, RPL, and CoAP. The Contiki IPv4 and IPv6 stack, is fully certified. Contiki code is written in standard C language.

B.3.3 Z1 Contiki Network

If one mote is too far to reach directly the border router, it will hope to the closest one, and this will act as a router and send it to the border router. This is about RPL and there are different implementation, anyway, we have mutli-hop because every mote builds a neighbors' table thanks to Infopackets exchange between motes. Border router can also be called, *gateway* or *modem*.

Motes send HTTP packets, using RIME communication layer stack, and using UDP in the transport layer and IPV6.

LAYER	PROTOCOL
Application	CoAP,REST, HTTP
Transport	UDP,TCP
Network	uIP stack (IPv6, 6LowPAN, RPL), IPv4
	RIME
Data- MAC	Null MAC
Physical	2.45 GHz, IEEE 802.15.4

B.4 Raspberry Pi B+

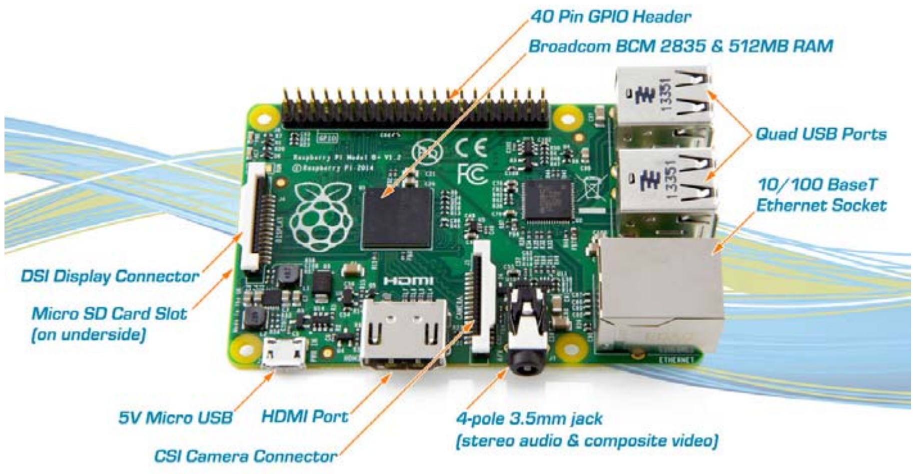

40 Pin GPIO Header
Broadcom BCM 2835 & 512MB RAM
Quad USB Ports
10/100 BaseT Ethernet Socket
DSI Display Connector
Micro SD Card Slot (on underside)
5V Micro USB
HDMI Port
CSI Camera Connector
4-pole 3.5mm jack (stereo audio & composite video)

Figure B.3 Raspberry Pi B+

The Model B+ is the higher-spec variant of the Raspberry Pi. It replaced the original Model B in July 2014. Compared to the Model B it has:

- **More GPIO**. The GPIO header has grown to 40.
- **More USB**. We now have 4 USB 2.0 ports.
- **Micro SD**.
- **Lower power consumption**. By replacing linear regulators with switching ones we've reduced power consumption by between 0.5W and 1W.
- **Better audio**. The audio circuit incorporates a dedicated low-noise power supply.

B.5 Huawei E3276 LTE MODEM

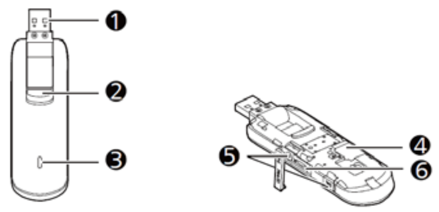
Model	E3276
Form	USB Stick

Communication System	FDD/TDD UMTS/HSUPA/HSPA+ GSM/GPRS/EDGE
Speed	High-speed LTE FDD packet data service of up to 150/50 Mbit/s High-speed LTE TDD packet data service of up to 112/10 Mbit/s
microSD card slot	Yes
External Antenna Interface	Yes
Receive Diversity	Yes
Operation System	Windows XP, Windows Vista, Windows 7, Windows 8, Mac OS X 10.5, Mac OS X 10.6, Mac OS X10.7, Mac OS X10.8

*Our model is E3276s-150

Hilink features being able to use the modem without any dashboard software I.e. just plug in and control device thru 192.168.1.1 in browser. Huawei Hi-Link modems are a special class of USB 3G / 4G modem that appear as a network interface rather than a standard serial port. These modems have a fixed IP address and the current Huawei software does not allow this IP address to be changed. Care should be taken to ensure than neither your local network, or any of your WAN networks do not conflict with the IP address of your Hi-Link modem.

B.5.1 Getting to Know Your Mobile Internet Key



USB Connector

It connects the Internet Key to a PC, folds for safe transport, and rotates to help you get the best signal.

USB Connector Release Button

The LED light indicates if the internet key has found a network and if you have successfully connected to a mobile high speed internet network.

- Green, blinking twice every two seconds: the modem is powered on.
- Green, blinking once every 0.2 seconds: the modem's software is being upgraded.
- Green, blinking once every two seconds: the modem is registering with a 2G network.
- Blue, blinking once every two seconds: the modem is registering with a 3G/3G+ network.
- Cyan, blinking once every two seconds: the modem is registering with a 4G network.
- Green, solid: the modem is connected to a 2G modem.
- Blue, solid: the modem is connected to a 3G network.
- Cyan, solid: the modem is connected to a 3G/4G network.
- Off: the modem is removed.

SIM card slot

MicroSD Card Slot

External Antenna Ports

The Internet Key can support 2 external antennas for better signal reception.

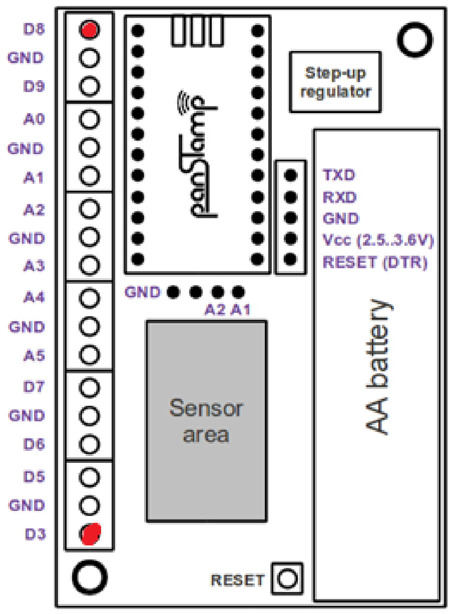
B.6 Android Applications

B.6.1 Smart parking Application

PanStamps have to be connected to sensors that can examine the parking space and successfully decide if it is occupied by a car or not. Appropriate sensors: Magnetic sensors because they can be hidden away or even buried under the parking space so they can be protected from theft or weather conditions.

<u>**No real sensors were involved during the development of this application.**</u>

They were simulated with a digital input on the microcontroller where digital 1 means that there is a car and digital 0 means that the space is available. The microcontroller can also be connected to a GPS sensor in order to know its location or its exact coordinated can be integrated in the firmware.



You have to connect a cable from D3 to D8 in panStamp Board.

PanStamp are programmed to transmit to the M2M server the values of the sensors that decide if there are cars in the parking spots and the coordinates of their exact location. Every time a value changes, the PanStamps send the new value to the M2M server so it is kept updated. The M2M server has a configured automated event and every time there is an update from the PanStamps that represent parking spaces, it uploads all the information to a virtual cloud. The OpenSense platform is used for online storing of the data.

A premium membership feature is also implemented by adding an output register on the panStamps. The register is configured as output so that it can be turned on and off dynamically. This way the premium membership status is not integrated in the firmware and it can be controlled from the M2M server (Turn on/off the Premium value in the Sensor). It is a virtual output that corresponds to no physical pin of the microcontroller. The M2M server updates the cloud about the state of the switch and the application translates it to premium membership if the switch is on and normal membership if the switch is off. Other features of the application is to show only the available spots or display the user's current location on the map.

****Smart Parking motes: 1, 2, 3, 4*

B.6.2 Geo Fencing Application

Geo fencing is a technology that uses the Geographical Positioning System (GPS) to define virtual borders around a specific region. The passing of the border can trigger actions and events programmed by the administrator. This means that by setting up virtual fences across the territory, points of interest can be created.

For the Geo Fencing application a panStamp is programmed to control an actuator. The actuator used for the needs of the application is an LED light, but in real life the actuator could be integrated in the heating system, the garage door or the coffee maker. The M2M server communicates with the panStamp and can send commands to the actuator at any time. The polling mechanism is used so that the server receives status updates regarding the state of the actuator from a virtual switch on the OpenSense cloud. The rest is handled by the Android application.

- Initially the application's user interface requires some fields to be filled.
- Latitude defines the latitude of the center of the fenced area.
- Longitude defines the longitude of the center of the fenced area.
- Feed ID is the ID number of the virtual switch that represents the actuator.

Authentication key is the key that needs to be in the header of the HTTP requests for security.

All the information is saved in a private file on the device so that they don't get erased when the application terminates. The user can create multiple virtual fences with different coordinates, ID's and authentication keys. The application implements the Android Google Maps API so the user can visualize his position and the location of the fences on a map. Finally, there is the option to enable the background service.

The background service is an autonomous process that finds the current location of the user using the Google Maps library and checks if his location is inside the virtual fence. If it is, it uses the HTTP library to send an HTTP POST request to the cloud and turn on the virtual switch. The M2M server using the polling mechanism detects the change on the switch and turns on the actuator. This way the Geo Fencing application can perform automated tasks that require no human intervention.

The application is implemented using a background service so that the user can enable the service and use his device as a regular mobile phone at the same time. The background process and the M2M server handle the rest.

The application can be used in various scenarios. Could be to turn on the heating system when the user is approaching his home so that the house is already warm when he arrives, or to open the outside door automatically every time the user is outside his house. It is designed to be able to save many events with different actuators so it can automate many processes.

This application is connected with Device “Kitchen” and channel “Light” and Feed ID “51742”.

The polling function is triggering events in the cloud.

Geo Fencing application Motes: 11 (11.14.2 Led light of mote 11)

B.7 SmartWorld Testbed – Run Step by Step (via Ethernet Connection)

STEP 1: Insert the batteries to PanStamp sensors.

STEP 2: Insert the batteries to Zolertia motes.

STEP 3: Connect Zolertia Mote “7” to Raspberry Pi.

STEP 4: Connect PanStamp Shield to Raspberry.

STEP 5: Power-on Raspberry Pi

STEP 6: Login to Raspberry Pi

Username: pi

Password: raspberry

STEP 7: write >> startx , in order to enter to GUI Raspbian

STEP 8: Configure IP

- >> sudo nano /etc/network/interfaces

  This command open the configuration file for the internet which should be modified.

  Iface eth0 inet static

  Address 10.1.3.53

  Gateway 10.1.3.1

  Netmask 255.255.255.0

  Network 10.1.3.0

  Broadcast 10.1.3.255

  To save changes done in nano editor: ctrl+X—save—yes

- Run: >>ifconfig

  Note your IP address and change the IP addresses in Lagarto servers

- *Change the IP address of Lagarto-MAX:*

panStamp2/Lagarto/Lagarto-max/config/ lagarto.xml

Change the IP address to: local_IP_Address

Keep the port: 8002

If you are connected with USB dongle: IP = 192.168.1.100 (is fixed)

If you are connected with Ethernet (in the Lab) = 10.1.3.53

- Change the IP address of Lagarto-SWAP:

  panStamp2/Lagarto/Lagarto-swap/config/ lagarto.xml

  Change the IP address to: local_IP_Address

  Keep the port: 8003

  If you are connected with USB dongle: IP = 192.168.1.100 (is fixed)

  If you are connected with Ethernet (in the Lab) = 10.1.3.53

- *Change the IP address of Lagarto-Contiki:*

  Contistamp/lagarto/lagarto-contiki/config/ lagarto.xml

  Change the IP address to: local_IP_Address

  Keep the port: 8001

  If you are connected with USB dongle: IP = 192.168.1.100 (is fixed)

  If you are connected with Ethernet (in the Lab) = 10.1.3.53

STEP 9: Configure DNS

>> sudo nano /etc/resolv.conf

Nameserver 84.88.62.194

Nameserver 84.88.62.220

To save changes done in nano editor: Ctrl+X – Save – yes

STEP 10: Configure Raspberry Modules

>> sudo nano /etc/modules

Ipv6

Cdc_ether

i2c-bcm2708

*Ipv6 – need for Contiki

** cdc_ether – 4G USB Dongle support

*** i2c-bcm2708 – PanStamp shield support

STEP 11: Define the type of mote / border-router.

Open a Terminal Session

Cd Contiki/examples/z1

Write >> make TARGET=z1 savetarget

STEP 12: Activate tunslip6

Cd ..

Cd ipv6/rpl-border-router

Make z1-reset && make connect-router&

STEP 13: Start Lagarto-MAX server

Cd panStamp2/Lagarto/Lagarto-MAX

>> sudo python Lagarto-max.py

Open a new terminal session

STEP 14: Start Lagarto-SWAP server

Cd panStamp2/Lagarto/Lagarto-swap

>> sudo python Lagarto-swap.py

Open a new terminal session

STEP 15: Start Lagarto-Contiki Server

Cd contiStamp/Lagarto/Lagarto-contiki

>> sudo python Lagarto-contiki.py

Open a new terminal session

STEP 16: Configure Servers via Web browser

Lagarto-HTTP: local_ip_address:8001

View Devices & Values

Lagarto-MAX: local_ip_address:8002

Lagarto-SWAP: local_ip_address:8003

View Devices & Values

STEP 17: Open.Sen.Se Platform

http://open.sen.se

Login: nightcrawlertool@hotmail.com

Password: cttccttc

Observe the current applications

Create new applications

The communication among the servers, the cloud and the applications is achieved through http requests (GET,POST). For security purposes every http request must contain a unique key in the header of each http packet.

Current Devices

Channels -> My Devices

RPi Contiki = RPi B+ (Elisavet)

ContikiDevice6 = RPi (Giacomo)

Smartparking = RPi (Vasilis)

Kitchen = RPi

Office = RPi

Create new Device

Channels

Custom made devices: Register

Get Started

Name the Device, Type of Device (other) , Description, Sending Data (HTTP Posting), Receiving data (HTTP Polling)

Save and Continue

Add a feed to Device: Name , Input/Output/Feed Type

Sending data

There is only one method available at this time. Your device will have to post its data on Sen.se servers using HTTP. We will provide the information on how to do this once you are done registering your device.

Receiving data

If your device doesn't embed an HTTP server then you can only use the polling method, which is checking Sen.se from time to time to retrieve possible new data. If your device embeds an HTTP server with a URL that can be accessed from outside your local network then you can choose to have new data automatically posted to your device. This solution is useful if you want real time reaction from your device.

Create a New Feed

Channels

My Devices

RPiContiki

View Recap Instructions

Edit Feeds

Add a Feed -> Name, Input/Output/Feed Type

What is a *feed* ?

A feed is a specific flow of data sent by the device or that the device expects to receive. A device might have several feeds. Ex: a device with a temperature sensor, a button and a LED will have 3 feeds (Temp, button, LED) each containing data from/to its corresponding sensor or actuator.

Why do I need to define the type of data?

Because this helps Apps understand and adapt to the type of values you are sending or expecting to receive. Not defining the type of data will make most Apps ineffective. Ex: Temperature send values like 35.2 is therefore a float number; Button sends 0s and 1 and hence is boolean; LED expects values between 0 and 255 to progressively light up and needs data of the integer number type.

Create an Application

Applications

Apps Library

Visualization & Display

Senseriver ->

Install now

Select Device & Feed , Select display size, Give a name

Save and Continue

Complete installation

See all the Applications at Senseboard

Personal API key: Profile & Settings-> API Key

STEP 18: Run Android Applications (now support multiple screens)

Smart Parking Application (PanStamp Sensors: 1,2,3,4)

Application that receives information about parking spots from the Lagarto server and displays them in a map using the Google maps API. The server, when it detects a change in the spots, updates the cloud using a POST request. The application updates itself by sending a GET request to the cloud. The parking spaces are noted in the map using different colors depending on the spot availability.

Geo Fence Application (on/off switch in Open.Sen.Se)

Application that changes a value in the lagarto server judging from the geological position of the android device. The application asks for the necessary information (coordinates, security key, endpoint ID) and every time the geological position fulfills certain conditions, it sends a POST request to the cloud. The lagarto server is configured to poll values from the cloud every minute (configurable) and detects the change