A Google Android based communication interface with Smart Homes sensing devices

Dimitrios Archontis, Spyros Kehagias, Vasilis Maglogiannis, Nikos Makris,
Virgilios Passas, Ilias Syrigos, Kostas Hounos and Lefteri H. Tsoukala
Dept. of Electrical and Computer Engineering
University of Thessaly, Volos, Greece
Email: {arhontis, spkehagi, vamaglog, nimakris, vipassas, ilsyrigo, hounos}@uth.gr | tsoukala@ecn.purdue.edu

Abstract—**The rapid evolution of Wireless Sensor Networks (WSN) lately, has been the stimulus for the establishment of Smart Houses. A Smart House (SH) is a house that has highly advanced automatic systems for lighting, temperature control, multi-media, security, window and door operations, and many other functions. Wireless technologies that are exploited in a SH installation are usually the WiFi and ZigBee protocol, due to the freedom of operating devices in the unlicensed band of 2.4 GHz. In this paper, we utilize the WSN devices developed by the NITLab laboratory in University of Thessaly and a Plugwise Home Basic kit, and design and implement an Android application able to retrieve data and measurements from these sensors, and represent them to the end user providing a user-friendly GUI.**

I. Introduction

Over the past few years, the idea of Smart Homes (SHs) has become more and more attractive. SH connect all the devices and appliances in an ordinary home so they can communicate with each other and usually with the owner. Anything in a home that uses electricity can be put on the home network and at ones command. In the case that interaction with an SH user is available, this comes usually with several interfaces; whether a command is given by voice, remote control or computer, the home reacts. So far, almost all of the developed applications relate to lighting, home security, home theater and entertainment and thermostat regulation [1].

Therefore, SHs are homes which utilize information appliances and a home-based network to connect household appliances to each other and to the outside Internet world. Using sensors mounted on devices and various coordinating protocols, SH try to enhance urban living by automating ordinary life.

SH technology first allowed compatible products to talk to each other over the already existing electrical wires of a home. A key technology that still utilizes such interfaces is the Power Line Communication (PLC) protocols, developed by several technology vendors. All the appliances and devices are receivers, and the means of controlling the system, such as remote controls or keypads, are transmitters.

Instead of going through the power lines, some systems use radio waves to communicate, which is also how WiFi and cell phone signals operate. However, home automation networks don't need all the features of a WiFi network because automation commands are short messages. The use of the unlicensed bands of 2.4GHz, has been a strong motivation for the design and development of several new protocols, specially designed for satisfying the needs of wireless communication in a SH.

Wireless networking has developed rapidly in the last years towards this direction and has contributed a great deal to this field. An applied technology of wireless networking is the WiFi based on the IEEE 802.11 standards. The obvious advantage of wireless transmission is a significant reduction and simplification in wiring and harness. Many communication technologies, such as IrDA, Bluetooth and ZigBee, GSM/GPRS (General Packet Radio Service), etc., have been developed for different situations. Nowadays, a kind of real time systems in which multiple sensors connected simultaneously to one gateway unit become necessary, and they are transformed into wireless sensor networks (WSNs).

A real life application of WSNs are the SHs [2]. In such a case, a WSN consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location [3]. Usually, a server there aggregates and processes the data, and using sophisticated techniques may schedule the operation of many SH appliances, in the most efficient way.

In the following sections we give a brief overview of the most common wireless technologies that are used in a SH system. The common point between them is the band they transmit (usually they make use of the unlicensed bands), and their extended use worldwide. The most common are

- The WiFi protocol
- The ZigBee protcol

A. The WiFi protocol

The radios used for WiFi communication are very similar to the radios used for walkie-talkies, cell phones and other devices. They can transmit and receive radio waves, and they can convert 1s and 0s into radio waves and convert the radio waves back into 1s and 0s.

But WiFi radios have a few notable differences from other radios. They transmit at frequencies of 2.4 GHz (or 5 GHz). This frequency is considerably higher than the frequencies used for cell phones, walkie-talkies and televisions. The higher frequency allows the signal to carry more data. This frequency also belongs to what are called unlicensed radio frequency bands. These are used by anyone who wants to use them, without fee. They use 802.11 networking standards, which come in several standards. The main standards transmitting at unlicensed frequency bands are 802.11b and 802.11g.

- 802.11b is the slowest and least expensive standard. For a while, its cost made it popular, but now it's becoming less common as faster standards become less expensive. 802.11b transmits in the 2.4 GHz frequency band of the radio spectrum. It can handle up to 11 megabits of data per second, and it uses complementary code keying (CCK) modulation to improve speeds.
- 802.11g transmits at 2.4 GHz like 802.11b, but it's a lot faster – it can handle up to 54 megabits of data per second. 802.11g is faster because it uses the same OFDM coding as 802.11a.

Other protocols/standards operating in different frequency bands include 802.11a, 802.11n and 802.11ac [4].

B. *The ZigBee protocol*

ZigBee is a low-cost, low-power, wireless mesh network standard. The low cost allows the technology to be widely deployed in wireless control and monitoring applications. Low power-usage allows longer life with smaller batteries. Mesh networking provides high reliability and more extensive range. ZigBee chip vendors typically sell integrated radios and microcontrollers with between 60 KB and 256 KB flash memory. ZigBee operates in the industrial, scientific and medical (ISM) radio bands; 868 MHz in Europe, 915 MHz in the USA and Australia and 2.4 GHz in most jurisdictions worldwide. Data transmission rates vary from 20 kilobits/second in the 868 MHz frequency band to 250 kilobits/second in the 2.4 GHz frequency band [3]. The ZigBee network layer natively supports both star and tree typical networks, and generic mesh networks. Every network must have one coordinator device, tasked with its creation, the control of its parameters and basic maintenance. Within star networks, the coordinator must be the central node. Both trees and meshes allows the use of ZigBee routers to extend communication at the network level.

ZigBee builds upon the physical layer and media access control defined in IEEE standard 802.15.4 (2003 version) for low-rate WPANs. The specification goes on to complete the standard by adding four main components: network layer, application layer, ZigBee device objects (ZDOs) and manufacturer-defined application objects which allow for customization and favor total integration. Besides adding two high-level network layers to the underlying structure, the most significant improvement is the introduction of ZDOs. These are responsible for a number of tasks, which include keeping of device roles, management of requests to join a network, device discovery and security.

ZigBee is not intended to support powerline networking but to interface with it at least for smart metering and smart appliance purposes such as the ones we find in SHs. Because ZigBee nodes can go from sleep to active mode in 30 ms or less, the latency can be low and devices can be responsive,

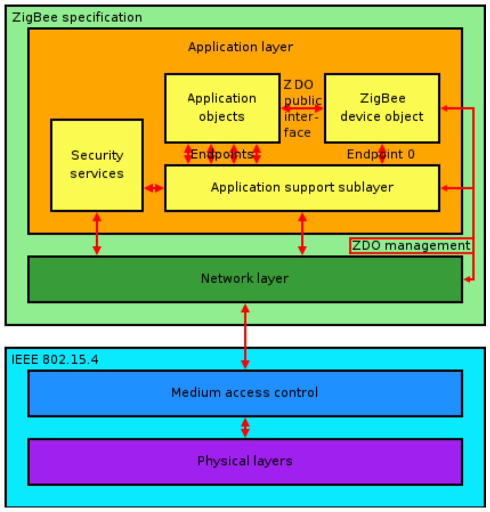


Fig. 1: The ZigBee protocol stack

particularly compared to Bluetooth wake-up delays, which are typically around three seconds [2]. Because ZigBee nodes can sleep most of the time, average power consumption can be low, resulting in long battery life. A Bluetooth SMART device can exchange data and disconnect in 3 ms, when advertising is pushed to maximum connect. This significantly enhances the experiences for HID devices [5].

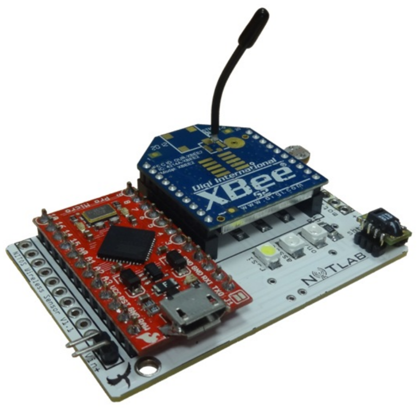
User - SH appliance communication may be integrated into the functionality of already popular devices such as smart phones operating under various mobile operating systems. We developed a Google Android application, one of the world's most popular mobile platform. It is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices [6]. Details about the design and implementation of the system are presented in the following sections.

II. Sensing Platforms

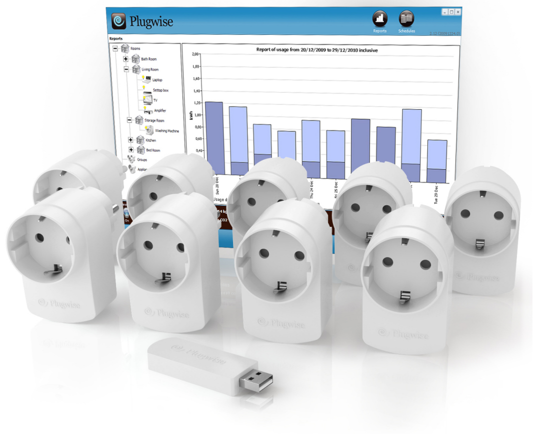
For this paper, we used two different technologies, able to be incorporated in a SH, and developed the Android application for retrieving the measurements from them. The first platform is an experimental one, completely designed and developed by University of Thessaly. The second one is the well known commercial platform of PlugWise, ready for integration with already existing home appliances. A brief overview of the platforms is presented in the following subsections.

A. *NITOS WSN platform*

To implement our solution we utilized the NITOS wireless sensor platforms. Towards the direction of enabling sensing based on WSN solutions, NITLab [7] / UTH has developed a prototype wireless sensor platform comprised of open-source and configurable modules. The main part of the aforementioned platform is a Pro Micro [8] board, fabricated by Sparkfun Electronics [9], a well-known industrial company, which



(a) NITOS Sensing Platform



(b) The PlugWise platform

Fig. 2: Sensing Platforms used in our setup

provides open-source hardware platforms that are fully configurable by Arduino's [10] open-source software. The on-board AVR micro-controller developed by Atmel [12], runs at 8Mhz and coordinates the overall platform operation. Moreover, the platform is equipped with an Xbee [13] radio interface that enables communication with a respective gateway. The Xbee module is a tiny device ideal for setting up mesh networks and has a defined rate of 250 kbps. This module uses the IEEE 802.15.4 stack (the basis for Zigbee [14]) and wraps it into a simple to use serial command set, allowing a very reliable and simple communication with Pro Micro's micro-controller.

The developed platform currently features a number of sensing modules, such as air temperature and humidity, light intensity and human presence. Various types of sensing modules and actuators can be further integrated utilizing Arduino libraries that implement several existing communications protocols, such as I2C, LIN, SPI, TWI, USI, etc. Firmware can be easily uploaded through the on-board USB connection. In addition to this, the developed platform supports over-the-air-programming (which is visible through) achieved by a special circuit already integrated in the PCB, allowing firmware to be uploaded wirelessly. Apart from the pluggable Xbee module, the developed platform can be equipped with WiFi or Bluetooth radio interfaces compatible with Xbee footprint, thus enabling communication utilizing different technology standards. By exploiting open-source firmware, developers can define different network topologies according to their experimentation setup, while numerous aggregation schemes can be used for gathering information reducing the total communication cost. Finally, the developed platform is a low-cost and small-sized one that can be powered by low-voltage sources, making it ideal for extended deployment at almost any place.

B. *The PlugWise platform*

Our developed platform incorporates support for a PlugWise - Home Basic kit. Plugwise has developed and produced wireless systems for energy management and appliance control since 2006. They have released several products, mainly power consumption metering devices, which can be easily configured and integrated with existing home appliance equipment. Such a kit is the Home Basic Kit, which consists of Circle+, eight Circles, the Plugwise Stick and the Source software. The components of the kit are depicted in Figure figure 2(b), and can be easily operated by inexperienced users.

Plugwise Circles communicate with each other wirelessly, utilizing the 802.15.4 networking stack and a slightly altered ZigBee protocol. All Circles form a wireless mesh network with each other, but in case they should transmit data back to a server device, they communicate via the Circle+ plug.

Although the Plugwise platform is a commercial one, and therefore the company does not provide an official Application Interface (API) for experimentation with their products, we used a reversed-engineered python library that has become available to the community. However, although the API is clearly defined in this library, it is not that mature yet and extra programming effort is required. More details upon the problems we had to cope with when setting up our platforms concerning the software used are given in the next section.

III. Implementation Details

In this section, we describe some implementation details, and the tools we exploited for the overall setup of the platform. We initially describe the software tools we used, and subsequently the networking setup that was required for the demonstration of our application.

A. *Software Tools*

Towards rendering the described hardware platform into a functional framework, we developed appropriate software in order to control wireless sensor platforms(WSPs) and the Plugwise Home Basic Kit. The four basic software components that were used are:

We exploited the capabilities of the provided source package [15] so as to achieve the desired functionalities of the Home Basic Kit. We are able to manage the status of the connected plugs, meaning that we have the ability to switch them on or off. Moreover we can retrieve the instantaneous power consumption(in Watts) of every plug, as well as the average consumption of each plug and of the whole Kit.

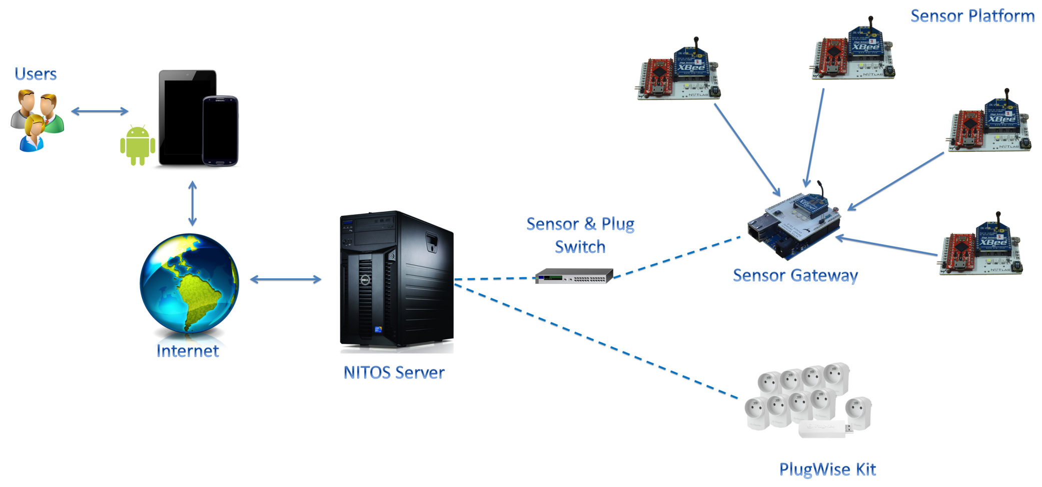


Fig. 3: Our Experimental Architecture

Concerning our implementation, we decided that in order for our approach to have some feasible results, we needed to write several python scripts that would operate in the backend of our system, and would actually instrument the Plugwise hardware operation. The operations that we needed to support are: switching on/off the PlugWise Circles, get the instantaneous power consumption and the average power consumption in Watts. Since the average consumption values are not supported by the API we used, we wrote a daemon process that wakes up periodically, and asks the device for the value of its current power consumption.

1) Arduino IDE: The Arduino integrated development environment (IDE) [11] is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface. A program or code written for Arduino is called a sketch. Arduino libraries are written in C or C++. The Arduino IDE comes with a software library called ”Wiring” from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings
- loop() : a function called repeatedly until the board powers off.

We employed the Arduino IDE in order to manage the wireless sensor platforms and the gateway. The former‘s implemented functionality is to collect periodically the values of the existing attached sensors and send them to the gateway through the Zigbee Protocol 802.15.4. The gateway remains active and waits for receiving packets from WSPs. Every incoming packet triggers another function of the gateway which sends the packet to NITOS server using the ethernet interface attached on it.

2) Android API - Appccelerator IDE: Android SDK is a software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel. Android SDK includes several APIs each one according to a specific Android OS version. Our application has been developed with API level 14 and it supports every Android version newer than 3.2 . Due to our previous experience in developing Android applications using Appccelarator we chose to take advantage of the various capabilities offered by it and develop our application.

Appcelerator Titanium Mobile is one of several phone web based application framework solutions allowing web developers to apply existing skills to create native applications for iPhone and Android. Instead of using the familiar JavaScript syntax, developers can also use Ajax, JQuery, HTML and CSS frameworks. For the evaluation of our developed application we utilized Appccelarator emulator, so as to preview and test application for possible errors. The objective of the developed application is to retrieve data and measurements from the sensors and the Plugwise platform presenting them to the end user through a user-friendly interface.

3) Server Software: For the database creation and management, the SQLite platform was installed and used on the server side. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. The source code for SQLite is in the public domain. The values received from the gateway are stored to a specific database. Furthermore, the connecting link between the Android application and the SQLite database is a PHP daemon that handles the requests from the application.

B. Networking Setup

The networking setup we used for the demonstration of the Android application, consisted of the following:

1) A wireless Access Point (AP)

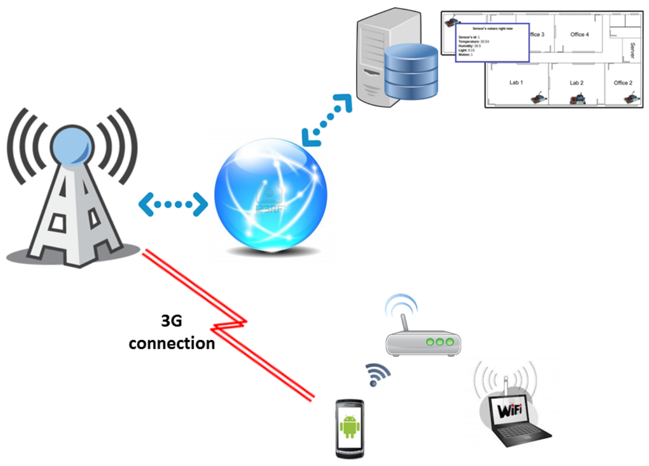


Fig. 4: Networking setup for demonstration needs

2) A broadband 3G connection to the Internet
3) Two Web servers for the backend of our system

For the demonstration needs, we needed to utilize two different connections to the two servers:

- A local server was setup, that runs the PlugWise platform software we developed and handles HTTP requests, which exploits the wireless link through the AP to the wireless interface of the Android phone
- A second remote server that retrieves temperature, humidity, light intensity and human presense measurements from the NITOS wireless sensors.

The second server is using an Internet connection and thus the 3G interface of the Android phone was used for connecting to it. A schematic of the overall architecture is depicted in Figure 4.

IV. EVALUATION

Our developed application has been developed with the aim to retrieve measurements from the NITOS WSN platform, as well as the remote management of the Plugwise Home Basic Kit. To this aim, we designed a user friendly GUI, making even the inexperienced users able to operate our platforms.

In Figure 5 we present some indicative screeshots from our application. As one can see, the main application window features three separate tabs;

- Testbed
- PlugWise
- Graph

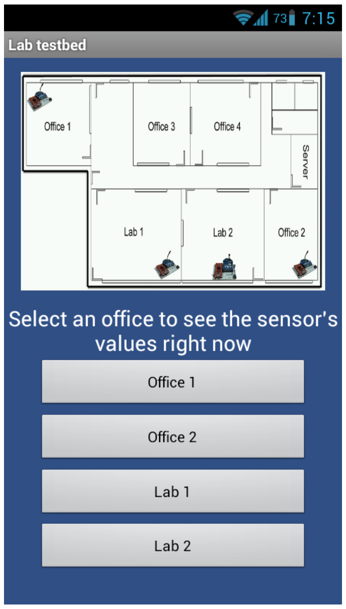
The first tab refers to the WSN testbed, located at NITLab premises, where the topology of the four wireless sensors are deployed. As we have mentioned, these four sensors communincate with a dedicated gateway, responsible for collecting and forwarding the sensed measurements. Using the dedicated buttons for every room, we can retrieve the measurements from the sensors. The path taken by each request for the sensed data is the following; When the respective button is pressed, it invokes an HTTP request to the NITLab server. The handling function in the backend of our system, issues a request to the sensor gateway for the measurement retrieval. The sensor gateway broadcasts this request to the dedicated sensors, which push back to it the aggregated data. The sensor gateway sends this data back to the server, and subsequently back to the phone. We use a popup window to display the data measured. The aforementioned described functionality is presented in Figures 5(a) and 5(b) respectively.

Concerning the plugwise platform manipulation, we use a dedicated tab for it (Figure 5(c)). A user can use the interface to interact with the Plugwise ”Home Basic Kit”, and turn on/off each plug or retrieve information about instantaneous power consumption of each plug, average power consumption of all the plugs and their status. We utilize a home WLAN connection for this purpose, since in the notion of a SH, all the data generated and aggregated are usually within a single LAN. An Internet connection could also be utilized for controlling the plugs, however this setup is not so common and requires a server running in the house 24/7.

The data path followed in the case of the Plugwise platform is: The HTTP request from each button is sent to a local server running in the SH, which handles it and invokes particular scripts on the server responsible for the plugwise platform manipulation. A key feature of the software developed on the server is a daemon process that retrieves power consumption from the plugs and averages it. The Plugwise stick is attached to the server, and therefore has full control of the Plugwise Circles. The data that our server processes produce are then pushed back to the Android application, and presented to the user, using a popup window similar to Figure 5(b).

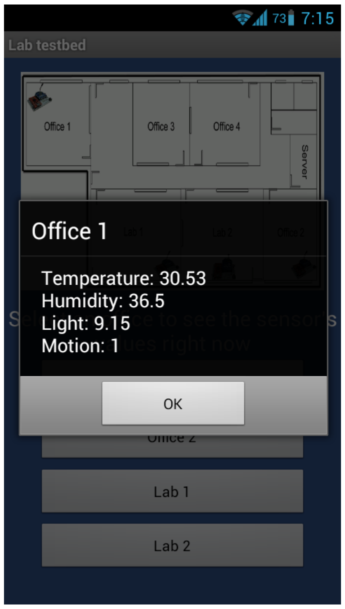
Through the Android application, the user can have access to specially created charts, presenting the last measurements from the WSN platform. The total number of measurements is selected by the user, and presented through a line chart, with different colors from the data sensed by each dedicated sensor. A sample line chart is presented in Figure 6.

The user has the option of selecting each unique point of the graph, and by a popup window in the Android application, dedicated meassurements are presented back to him/her. Each



Select an office to see the sensor's values right now

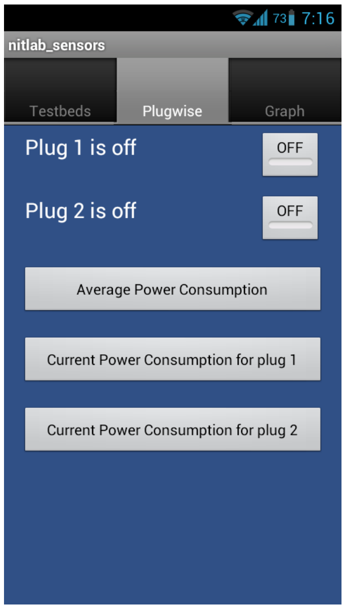
(a) Sensor measurement retrieval



Office 1

Temperature: 30.53
Humidity: 36.5
Light: 9.15
Motion: 1

(b) Sensor measurement retrieval


7:16
nitlab_sensors
Testbeds
Plugwise
Graph
OFF
OFF
Average Power Consumption
Current Power Consumption for plug 1
Current Power Consumption for plug 2

Plug 1 is off

Plug 2 is off

(c) Plugwise manipulation and measurement retrieval

Fig. 5: Application Screenshots

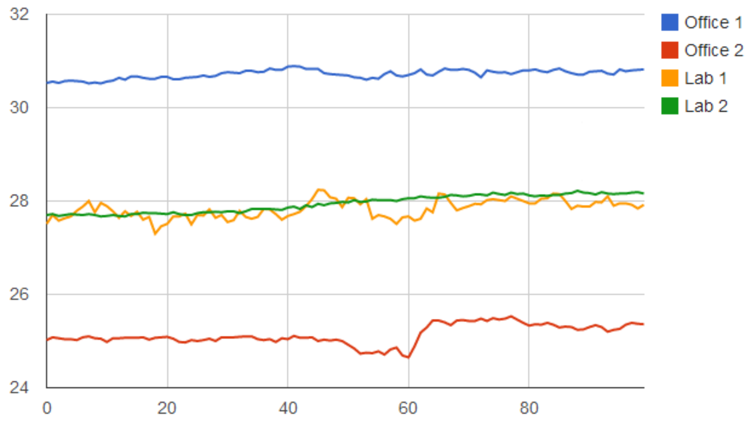


Fig. 6: Sample line chart presenting data from NITOS WSN

measurement is accompanied with the exact time and date that the specified measurement was taken.

V. Conclusion

In this paper we have presented our Android application used to operate and retrieve measurements from SH commercial and experimental sensors. The overall platform has been evaluated under realistic conditions.

In this age of economic recession, this state-of-the-art technology can provide the opportunity to build an economic business model for application area such as smart homes. Building smart homes is a big challenge for worldwide increasing elderly populations which are the largest demographic group of developed countries.

Smart homes obviously have the ability to make life easier and more convenient. They also provide some energy efficiency savings. Because wireless systems put some devices at a reduced level of functionality, they can go to ”sleep” and wake up when commands are given. Some devices can track how much energy each appliance is using and command it to use less. Communication between the user and the smart home appliances can be facilitated through the use of extensively used devices such as smart phones using the Android OS.

Acknowledgment

The authors would like to thank the NITLab laboratory for their cooperation, and the fact that they let us use the equipment developed by them. We would also like to thank Prof. Thanasis Korakis and Prof. Leandros Tassiulas, directors of the lab, who, with their permission, made this work feasible.

References

[1] How Smart Homes Work, http://home.howstuffworks.com/smart-home.htm/.

[2] Design and implementation of wireless Smart-home sensor network based on ZigBee protocol, Lili Liang and Lianfen Huang and Xueyuan Jiang and Yan Yao, In the proceedings of ICCCAS 2008.

[3] Wireless Sensor Network, http://en.wikipedia.org/wiki/Wireless_sensor_network/.

[4] How WiFi Works, http://computer.howstuffworks.com/wireless-network.htm/.

[5] ZigBee protocol, http://en.wikipedia.org/wiki/ZigBee/.

[6] Android Operating System, https://en.wikipedia.org/wiki/Android_(operating_system).

[7] Network Implementation Testbed Laboratory, http://nitlab.inf.uth.gr/NITlab/.

[8] Pro Micro Board, https://www.sparkfun.com/products/10999/.

[9] Sparkfun Electronics, https://www.sparkfun.com/.

[10] Arduino, http://www.arduino.cc/.

[11] Arduino-IDE, http://arduino.cc/en/Main/Software/.

[12] Atmel Corporation, http://www.atmel.com/.

[13] Xbee radio interfaces, http://www.digi.com/xbee/.

[14] ZigBee Alliance, http://www.zigbee.org/.

[15] Plugwise Source, http://www.plugwise.com/idplugtype-f/source.