Large-Scale Participatory Sensing Experimentation Using Smartphones within a Smart City

Evangelos Theodoridis
Tech. Educ. Inst. of Central Greece, Lamia, Greece
and
Comp. Tech. Inst. & Press "Diophantus", Patras, Greece
theodori@cti.gr

Georgios Mylonas
Comp. Tech. Inst. & Press "Diophantus"
Patras, Greece
mylonasg@cti.gr

Veronica Gutierrez Polidura, Luis Munoz
Universidad de Cantabria
Plaza de la Ciencia s/n
Santander, Spain
{veronica, luis}@tlmat.unican.es

ABSTRACT

A number of Future Internet testbeds are being deployed around the world for research experimentation and development. SmartSantander is an infrastructure of massive scale deployed inside a city centre. We argue that utilising the concept of participatory sensing can augment the functionality and potential use-cases of such a system and be beneficiary in a number of scenarios. We discuss the concept of extending SmartSantander with participatory sensing through the use of volunteers' smartphones. We report on our design and implementation, which allows for developers to write their code for Android devices and then deploy and execute on the devices automatically through our system. We have tested our implementation in a number of scenarios in two cities with the help of volunteers with promising results; the data collected enhance the ones by fixed infrastructure both quantitatively and qualitatively across the cities, while also engaging citizens more directly.

1. INTRODUCTION

A large body of work is being dedicated to bridge the digital and physical domain, through a combination of technological developments. More specifically, the integration of low-cost sensors in an increasing number of application domains, along with the extraordinary growth in the use of mobile devices and embedded systems, has already led to the omnipresence of connected computing devices of all kinds around us. In order to better understand the mechanics of the "Future Internet" and start implementing innovative systems and applications, a number of experimentation "testbeds" have surfaced around the world. Such testbeds are available, freely or through some other cost model, to the research community and companies in order for them to be able to deploy their applications on real devices on a large scale, an option that would otherwise be too expensive to deploy and maintain for each team individually.

The SmartSantander project [23] has developed one of the largest Future Internet infrastructures globally, located at the center of the city of Santander in Spain, complemented with a set of smaller federated testbeds in other European cities. The testbed includes over 20000 sensing points, offering a duality of operation modes; on the one hand, they operate as part of the city's monitoring system for e.g., car parking slots, environmental parameters, traffic conditions, etc., and on the other hand they are available to developers for experimentation, i.e., executable code can be uploaded to them. Projects like SmartSantander have already provided the community with useful results and insights with respect to actual issues and intricacies in implementing such systems in real-life settings. However, there are limitations to the scale that can be achieved when following such an approach, due to cost and maintenance limitations, and also the type of applications that can be supported through such infrastructures.

Moreover, we believe that innovation is no longer the result produced solely by single entrepreneurs or small groups; instead, collaborative efforts by researchers, companies, municipalities and end-users are all equally important sources for social and technological innovation. Therefore, all of these parties should get involved in providing a meaningful setting to deploy and test Future Internet systems, and also get sensible feedback from both citizens and companies/organizations alike. We have recently seen related approaches become popular, such as social computing and participatory sensing. Social computing involves the combination of social networking and pervasive computing, proposing even the embodiment of computation in human society. Participatory sensing takes also the concept of crowdsourcing and adapts it to the context of wireless sensor networking, harnessing the proliferation of ubiquitous networking technologies such as smartphones.

Another important aspect that should be taken into account is the omnipresence of smartphones and the possibilities they enable. Such devices already encompass an impressive range of integrated sensors: accelerometers, cameras, gyroscopes, microphones, thermistors, proximity sensors, etc. Some of the recent high-end smartphones even feature dedicated ambient temperature sensors, or specialised software to infer context awareness. Apart from this, the latest breed of smartphone operating system also offers enough flexibility for adding external sensing units directly or communicating wirelessly with them. Furthermore, with each iteration in the evolution cycle of smartphone operating systems, additional functionality is integrated, supported by more potent hardware.

Having the above in mind, in this work we discuss the design and implementation of a participatory sensing component for SmartSantander, aiming to augment the functionality of the testbed with new application use-cases and also provide a broader area cover-

MOBIQUITOUS 2014, December 02-05, London, Great Britain Copyright © 2014 ICST 978-1-63190-039-6
DOI 10.4108/icst.mobiquitous.2014.258016

age compared to the one provided by the existing infrastructure. The main concept is to have a number of volunteers install a smartphone application on their devices, which allows for automatic registration of the devices and their sensing capabilities in the SmartSantander system. The system uses a multi-layer architecture for:

- sending "experiments" in the form of executable code to the volunteers' smartphones, which utilize the integrated sensors, computing and communication resources,
- producing data traces that are subsequently uploaded to the SmartSantander data repository.

The smartphone software part is based on the Ambient Dynamix framework [2], which eases the online distribution and update of executable code in the form of plugins. Essentially, our implementation provides a means for developers to deploy their experimental applications to a large number of smartphone users in an almost transparent way, bypassing many hurdles and limitations usually associated with the deployment of experimental applications in large scale, even more so in a research/academic context.

We have conducted experiments to evaluate our approach in the context of a large-scale experimental Future Internet infrastructure, such as SmartSantander. We targeted two indicative scenarios, that showcase the complementarity of our system and the existing one. Although the scenarios themselves have been implemented previously on smartphones, they targeted specific application domains, and were not meant to serve as extensions to experimental infrastructures. They show that the extensions offer potential advantages in terms of augmenting the capabilities of fixed sensors, without sacrificing much in the way of development costs.

Regarding the structure of this work, in the following section we discuss previous work and draw comparisons with our approach, while we continue with a discussion on participatory sensing and the related requirements. On Section 3, we discuss the architecture of SmartSantander and our system, while also explaining the Ambient Dynamix Framework design. We include some insights to our implementation in Section 5, while Section 6 details our experiments and respective results. We conclude in Section 7.

2. RELATED WORK

Participatory sensing utilising smartphones and other mobile devices is being adopted fast by the research community. As mentioned, there is a multitude of approaches currently used, with some systems on the one hand focusing on tackling very specific problems and application scenarios, and on the other hand systems trying to offer a more generic framework in order for developers to build their own systems upon. The former type of systems has been up till now also the most popular one. Some of the first representative participatory sensing systems focused on monitoring urban environmental conditions using vehicles and smartphones or other mobile devices, mapping out potholes on the road surface [9], or creating maps with the most efficient or healthy routes for bicyclists [22]. Other systems focus on monitoring noise levels in urban area, like Earphone [20], utilising the microphones of smartphones to obtain readings. Opensense [1] is a community sensing platform comprising a set of various participatory sensing scenarios, e.g., air pollution monitoring [13]. [7] is another example of participatory sensing utilising vehicles to monitor air pollution levels in urban areas. All of these examples have, to a certain extent, provided evidence that mobile devices mounted on vehicles or carried by volunteers can produce measurements that are not far off from the ones produced by more sophisticated (and expensive) systems. [8] specifically addresses the issue of participatory sensing data quality, based on a urban noise levels scenario.

AnonySense [5] discusses issues regarding privacy in collaborative opportunistic sensing and presents an architecture for distributing sensing tasks while ensuring the anonymity of participants. [16] also discusses in detail anonymity for mobile sensor datasets and indicates that there are real de-anonymization threats to users' privacy and that the community should take them seriously in terms of design and implementation of future systems. [4] discusses extensively the privacy concepts associated with mobile participatory sensing applications, along with the respective threats to users' privacy. It presents a multitude of systems addressing such issues. Another detailed survey of algorithms, applications and systems utilizing smartphones in a participatory sensing setting is provided in [15]. [25] argues that scale is the key to the success of crowdsensing and provides a discussion on the related barriers. Overall, there are many different approaches utilised in such systems.

[24] and [14] discuss participatory sensing applications for monitoring noise levels in urban areas and in-house air quality, respectively, while also shedding light on the use of incentivisation as a means of engagement for the participants of both studies. [21] is another work focusing on the participants' part in the study and its effect on the design of participatory sensing systems. In this work we did not focus on the use of incentive mechanisms, although we plan to integrate such aspects in our future work.

Bubble-Sensing [17] was one of the first approaches for distributing sensing tasks among smartphones that opportunistically pass through certain "bubble-sensing" locations. [19] also discusses extensively the concept of utilising smartphones as data mules in order to provide opportunistic networking in a variety of settings. Closer to the work presented here is the PRISM system [6], aiming to provide a generic framework for easily disseminating sensing applications to potential volunteers' smartphones to be executed. PRISM presents certain similarities to our own work, e.g., sensing tasks are relayed to end devices in binary form, akin to stand-alone applications, running on top of a software framework and essentially securely isolating such code from the rest of the phone applications. However, PRISM is closed-source, is not offered currently publicly as a system, and has also been evaluated with a rather small actual number of participants (although the authors conducted simulations with large numbers of users). We also interface to one of the largest sensor testbeds currently available on a global scale. [3] discusses a system based on smartphones for experimentation, along with a set of use cases focused more on information retrieval aspects. Although the authors are using similar technologies, they target a different use-case domain, with smartphones essentially operating in a more controlled environment, and not in the context of a large-scale IoT smart city scenario. PhoneLab [18] is another system aiming to utilise the potential of smartphones as a testbed platform; it shares many goals with our system, though it uses an implementation based on AOSP that limits its applicability in a more generic, less-controlled, user setting. We also aim towards integration within a large-scale FI testbed.

Regarding currently available Future Internet testbeds, [11] reports on a number of architectures and implementations. Also, with respect to the concepts discussed in this work, to the best of our

knowledge, there has not been another similar attempt to combine participatory sensing with a large-scale Future Internet experimentation infrastructure. As mentioned, we utilise the Ambient Dynamix Framework [2] for a large part of our core functionality. This is however the first use of the framework in the context discussed here, along with the significant extensions that were required. With respect to prior work in SmartSantander, participatory sensing has been utilised [12], however it was constrained to certain specific domains, and all provided sensing functionality was predefined, i.e., there was little or no flexibility in terms of IoT experimentation.

Overall, in contrast to the majority of the participatory sensing systems available, we aim to provide a base system, on top of which, and within certain limitations, researchers can deploy their code which will be executed on volunteers' smartphones. The system also provides a more holistic solution in terms of measurement data storage, availability, interfacing, etc. Moreover, such code will be written for the Android software platform, which is currently the most popular smartphone ecosystem. Also, we have focused on technological aspects of such a system, leaving out user interaction and engagement aspects for future work.

3. PARTICIPATORY SENSING - REQUIREMENTS WITHIN SMARTSANTANDER

Having discussed existing approaches in participatory sensing without or with smartphones, in this section we discuss the possibilities created by integrating volunteers' smartphone devices into an existing IoT experimentation platform, and also the overall research challenges associated with realising such a vision. More specifically, with respect to the integration of smartphones with experimentation testbeds, we envision the following sensing affordances, in addition to the ones already available in current FI testbeds:

- *Augment existing FI testbeds by smartphones interacting with existing testbed infrastructure*: smartphones could be utilized to extend the overall capabilities of existing infrastructure, crowdsourcing experimentation in a city-wide context. E.g., by utilizing delay-tolerant communication schemes, smartphones could be used to deliver data between different parts of the testbeds, or they could be used to preprocess (compress, filter) parts of the data generating from the testbed nodes before delivering them to other system layers.
- *Utilize integrated smartphone sensors as sensor nodes*: smartphones already encompass an impressive range of sensors, or technologies like NFC (Near Field Communication). Such technologies are often not supported in the IoT sensor nodes used in experimentation testbeds. Essentially, in this fashion smartphones could increase sampling points and overall sensor number, even if the quality of readings is worse than static nodes. Also, their potential in producing data on human mobility and activity currently cannot be rivaled, being utilized in places where it would not be possible.
- *Utilize additional unique sensors for novel applications*: smartphone ecosystems have added support for adding external sensing units directly or interfacing wirelessly. Such sensors could be commercial add-on products, e.g., radiation sensors, or custom boards built by hackerspace communities, especially in applications with social or artistic dimensions. One could imagine e.g., studying mobility patterns in city streets, and simultaneously monitor environmental parameters, e.g., noise pollution.

Having the above in mind, we can identify a set of associated challenges, that have not been fully answered yet by neither the research community, nor the industry.

Programming and application model: current smartphone ecosystems support rather unique programming models compared to more potent systems, due to various constraints. One major difference is the concurrency model supported on each platform. There are additional restrictions in space, e.g., users or application restrictions could require saving data only in the cloud, or issues with sharing information with other applications, etc. Thus, applications should be tailored for such specific execution context. Also, with Web technologies' advances, software for experimentation purposes could even comprise applications executed within browsers, since smartphones' computational capabilities have increased drastically.

Dynamic nature - Participation, resource discovery and control: there is an inherent opportunistic nature when including smartphones in an experimentation platform. Users decide on providing their resources for experimentation or not, and there is also the issue of networking availability. Moreover, limitations are not just on whether volunteers actually offer their smartphones for experimentation, but also in the ways they actually carry their devices, e.g., if they place it in their pockets. Such limitations may seem insignificant but could be as important as the calibration of sensors in a controlled-environment testbed.

Enabling meaningful interaction with end-users and incentivization: focus should be placed on the ways mobile platforms are changing our everyday computing usage patterns and try to utilize such trends in the context of FI. Also, long-term user engagement is another point of consideration; without continuous user commitment, large-scale results may be hard to achieve. Mechanisms such as incentivization or social networking could be considered in this direction, since they have been proven effective in other areas.

Enabling novel application contexts - Collective Awareness: the social and environmental impact of such activities can be seen from an outer perspective; imagine tools for monitoring changes in civic areas and their impact they have in social interactions or the environment. Such insight can be of great help to city planners for discovering the outcome of implemented changes in much shorter time, or interest certain business sectors. Furthermore, pervasive games and human mobility patterns/models could also be used to provide insights for human activity or ways to test/provide collective awareness. Such mechanisms could provide combined correlation of various sensing sources, for promoting behavioural changes or increasing awareness about environmental conditions.

Security and Privacy management: monitoring environmental parameters and behaviour tracking with smartphones raises personal and public security/privacy concerns. There is also the additional implication that volunteers should have feedback on data they produce during experiments, or data produced by other contributors, e.g., for increasing participants' motivation. However, the fact those end-users are volunteers, and assuming that there is transparent handling of the data produced by their devices and the ways that such data are produced, should reduce such concerns.

4. SYSTEM ARCHITECTURE

SmartSantander [13] follows a 3-tiered network node architecture (Fig. 1) consisting of an IoT device tier, an IoT gateway (GW) tier and server tier. The IoT node tier consists mainly of IoT devices providing the experimentation substrate. These are resource-constrained devices, exporting sensing or actuating capabilities. This tier accounts for the majority of the devices in the testbed infrastructure. The IoT GW tier links IoT devices to a core network infrastructure. These nodes are also part of the programmable experimentation substrate, allowing experimentation for different networking and integration scenarios. These devices are typically more powerful than IoT nodes but more resource-constrained than devices in the server tier. The server tier comprises devices with high availability, directly connected to core network infrastructure. They are used to host IoT data repositories and application servers that can be configured to realize a variety of different IoT services and applications or to investigate approaches for real world data mining and knowledge engineering. The server tier benefits from virtualization in a cloud infrastructure, ensuring high reliability and availability of all components and services.

We aimed to complement this platform by including smartphones in the experimentation process, enabling them to co-exist with SmartSantander's infrastructure. The existing model for experimentation, in short, requires developers to write code for embedded platforms (e.g., Libelium nodes), then select and reserve for a certain time interval via a provided GUI the desired IoT nodes to execute the experiment. Finally, the system submits the experiment to the IoT nodes and delivers generated data to developers. With our system, the overall flow (see Fig. 2) between various high-level experimentation lifecycle phases is as follows:

- Registration of smartphones to SmartSantander, via a specific application on smartphones that manages the registration in the Resource Directory (RD) component [13] , the configuration of the device by end-users, etc.
- Uploading of smartphone experimentation code to SmartSantander, along with a set of experimentation parameters (execution period, etc.).
- Validation of the code by SmartSantander, in order to maintain uninterrupted smartphone use and anonymity of users before making it available for deployment and experimentation.
- Dynamically deploy experimentation code to the volunteers' smartphones and execute it at smartphone level for a specific time interval.
- Retrieval of data generated by the experimentation code on the smartphones, uploaded to the SmartSantander servers and made available to the experimenters.

For deploying, executing and managing experimentation code on smartphones, we rely on the Ambient Dynamix Framework (ADF) for Android, having extended it in order to support our system. ADF is a plug-and-play context framework for Android, enabling mobile applications to install plugins during runtime, including sensing and actuation plugins. Furthermore, ADF provides at server level a plugin repository architecture, which enables developers to create, share and publish new plugins. ADF at the smartphone level runs as a background service, providing access for other applications to install, access and execute already installed ADF plugins.

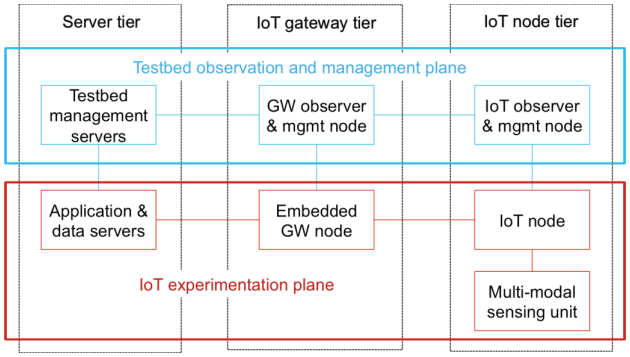

Figure 1: Logical separation of 3-tier node architecture into a testbed observation and management and an experimentation plane.

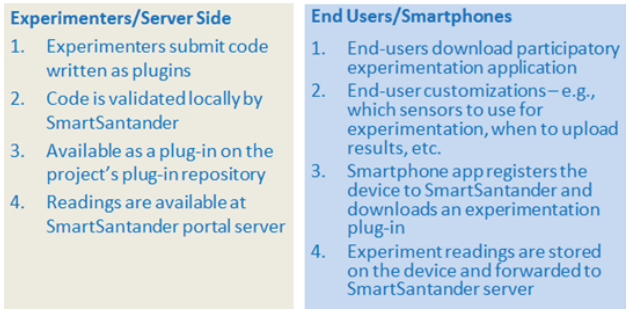

Figure 2: Summary of the procedure for submitting experiments to a IoT testbed infrastructure using our approach.

It is also possible for native applications and browser-based Web applications to request context support from a local ADF instance. ADF automatically discovers, downloads and installs the plugins needed for a given sensing or actuation task. New or updated plugins can be deployed to the device at runtime, without the need to restart the application or framework, transparently from the smartphone user. Ambient Dynamix plugins are written in native Android code, making it possible for programmers to access through the ADF other plugins or resources.

Essentially, smartphone experiments in our platform are published on a web repository as ADF plugins, i.e., developers write code adhering to ADF plugin restrictions. When volunteer smartphone owners register and connect to the system, such a plugin will be installed locally to their device. Such plugins will be executed and managed later on (paused, resumed, stopped, etc.) by a special SmartSantander smartphone application. This app, identified as the SE manager (Smartphone Experimentation) is responsible for:

- registration of the smartphone to SmartSantander,
- scheduling, execution and management of an experiment,
- management of access rights of the experimentation code to the smartphone resources according to the preferences of the device owner,
- recording of experimentation data and forwarding them to SmartSantander data repository.

We could map the above requirements to a set of implementation components on a server and a client/smartphone tier. On the server side, An *SE Configuration* component is responsible for the experiment configuration, reusing and extending the functionality of the existing *Configuration* module of SmartSantander (synthesis, specification, file upload, sanity check, resource selection). Also, a *Smartphone Experimentation Server* component manages the scheduling of the experiments, the delivery of the experimentation code to the smartphones, the registration of Smartphones to RD, the gathering of experimentation data from the smartphones.

On the client/smartphone side, a *SE and User Preferences Configuration* module manages the configuration of the experimentation sharing of the smartphone by the end-user (e.g., allow the use of specific sensors, specify the time of day to be used, etc.). The module registers the Smartphone to the system. Also a *SE Manager* module manages the experimentation code delivery to the smartphone, the execution, the monitoring of the experiment and experimental data recording. Finally, it will deliver the produced data to the Smartphone Experimentation Server.

Smartphones, in contrast to other IoT devices in SmartSantander platform, are being carried by users/owners. Therefore, special components are required that will not interfere with the normal operation of the smartphone and guarantee the privacy of the participating users as well. Several issues regarding the integration with the SmartSantander architecture have been identified and realized. For the time being, we have focused on the Android smartphone platform for our implementation, due to restrictions posed on multitasking in other popular smartphone platforms and Android's overall popularity as a platform among smartphone users.

In order to implement such functionality, we designed and implemented several modifications and extensions to ADF.

Portal Server Level: The existing Ambient Dynamix Plugin Web repository has been extended with a special API responsible for inserting, deleting and updating plugins. This interface is used by the experimentation user client app in order to insert, delete or update an experiment. Furthermore, this extension ensures the integrity among the published plugins (i.e., SmartSantander-related experiments) and the main experimentation database.

Smartphone Level: The Ambient Dynamix Context Service has been extended with the following components:

- an *Access Rights Manager*, which is a manager for giving to the plug-ins (experiments) the rights to access certain smartphone resources according to end-user owner. The issue here is that smartphone owners may decide to restrict access to specific integrated sensors of the smartphone, so that e.g., the experimentation code cannot utilise the GPS of their device
- *Communication Interfaces* in order to access smartphone storage for persisting experimentation data. At the end of the experiment, the SE Manager forwards the stored data to the SmartSantander experimentation DB.
- An *Android application* for starting, pausing, or resuming the execution of a plugin/experiment.

Registration of mobile phone devices into the SmartSantander platform. We developed a specific application for Android devices that registers a smartphone to the platform and manages (run/pause/stop) the deployed experiment. This application instantiates the *"SE and User Preferences Configuration"* module, where the end-user gives consent to contribute his smartphone computing, sensing and communication resources and configure several parameters of experiment execution (e.g., which ones of the integrated sensors to share, when and how to upload experiment readings, etc.). This application also registers through the *"Resource Registration"* component the device to the system. Registration will be handled by the *"Smartphone Experimentation Manager"* at the portal server level.

Uploading of smartphone experimentation code to the SmartSantander system. Through an Experimentation Client, experimenters will specify a range of devices (number, type, sensors, etc.), reserve them for a time period and define an experiment for smartphones. The identified component *"Experiment Manager"* will consume all the parameters of an experiment and will prepare the experiment as a file to be deployed on smartphones. The experiment is published as a plugin at the *Plugin Experiment Web repository*.

Deployment of the experimentation code to smartphones and execution. The smartphone level *SE manage*r interacts with the *Plugin Experiment Web repository* and installs the experiment plugin. Furthermore, it manages the execution of the experiment and also the interventions of the users (e.g., stop/resume whenever required).

Retrieval of Experimentation Data. SE manager interacts at the smartphone with the experiment plug-in and retrieves the captured experimentation data. These data are stored locally in a local database. At the end of the experiment, the reporting module pushes back to the server all captured data.

5. IMPLEMENTATION DETAILS

In this section, we discuss several aspects regarding the implementation[1] of the architecture discussed previously.

5.1 Integration with SmartSantander Platform

Regarding integration with SmartSantander (Fig. 3(a)), all operations and communication interfaces from smartphones to SmartSantander servers have been developed using SOAP Web Services (JAX-WS). The registration of devices has been done in the RD, the main SmartSantander component for storing IoT devices of the infrastructure along with various attributes and characteristics of the devices. By design, RD can support dynamic attribute set and characteristics, covering a wide range of devices. During registration, the smartphone vendor/type is submitted along with users' preferences regarding which sensors are permitted to be used. These sensors are registered as capabilities of the device. The device is registered by a unique integer ID, sent and stored in smartphones for subsequent data registration.

Regarding experiments, another component of SmartSantander platform has been used, the Experimentation Database (ED). ED is a database where experiments and various attributes (like time interval of execution, desired devices, number of devices, user which submitted experiment, source code and binary files of the experiment, etc.) are stored. Concerning delivery of experimented, each one of the registered devices are polling the main server for an active experiment and if its characteristics are matching the device capabilities (user preferences), then the binary code (in our case, a OSGi plugin) of it is delivered to the polling device.

[1]Repository: https://github.com/theodori/AndroidExperimentation

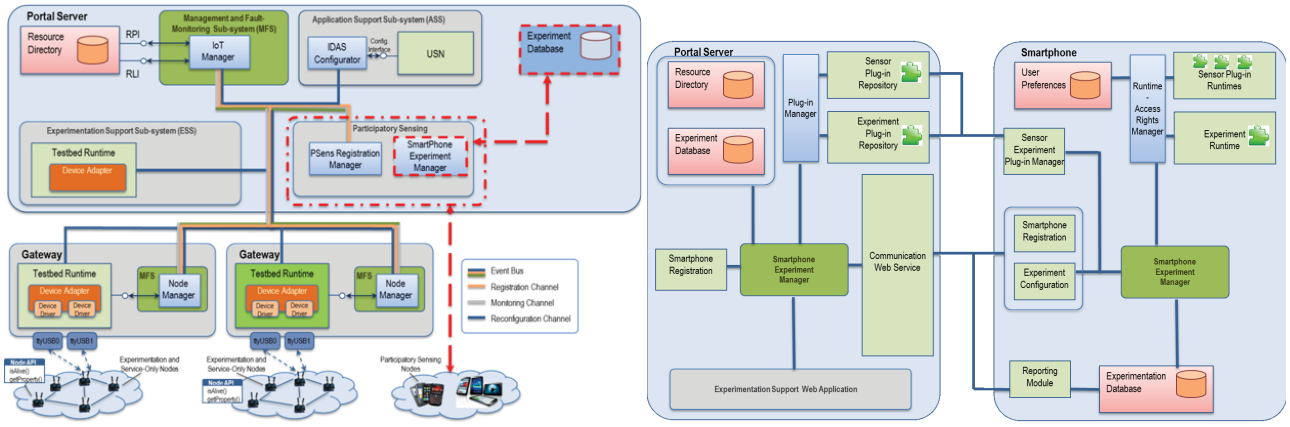

Figure 3: (a) Integration with SmartSantander, (b) overall architecture of the system and interconnections between the experimentation portal and smartphone devices.

Finally, regarding data gathering of experiments, each one of the experiments is producing data. These data are posted through the web interfaces to main system and eventually are stored to the Experimentation Database. In this database its is possible either to store a plain observation (e.g., noise levels) described by an observation name, a primitive data type (e.g., float, string) and an observation value or it can be flexible enough and experimented can store observation value or it can be flexible enough and experimented can store arbitrary payload messages as JSON strings.

5.2 How to implement a plugin

ADF is based on OSGi architecture using an Apache Felix implementation[2], a framework enabling the development of Java applications based on dynamic components, i.e., application functionality can be dynamically installed, updated or executed if the respective code follows the OSGi specification, in a plugin fashion. Using such an architecture allows changes to the composition of the smartphone application dynamically, without requiring application restarts. It also decouples components, and enables them to dynamically discover each other for interaction, i.e., components can utilise the functionality of each other

Any framework that implements the OSGi standard provides an environment for the modularization of applications into smaller bundles, the basic components in OSGi, which are tightly coupled, dynamically loadable collection of classes, JARs, and configuration files, declaring their dependencies. The standard bundle composition includes a detailed manifest file declaring all its contents and additional services required. OSGi framework bundles can be dynamically installed, started, stopped, updated and uninstalled.

5.3 Smartphone application UI

As mentioned previously, building upon our extensions of the ADF, we developed a new user interface for the smartphone app running on the volunteers' devices. The overall goals for such an app are the following:

- Give the end-users the ability to easily initiate or terminate their participation in the experimentation process, and enable/disable their devices at will at any time, while having a clear overview of the whole process.
- All plug-ins and respective sensors can be activated/deactivated selectively at any time, making clear to the system what experimentation capabilities the end-users are willing to contribute, and also make clear to end-users what kind of data they can actively offer.
- The user interface provides a live view of the data generated by the device when there is an active experiment and the device has joined in. Along with the data, there is a short description of the experiment itself available. The content of the readings messages, produced by the app in order to be sent to the experimentation server, are bundled in readable JSON format, so it is relatively easy to examine the actual data sent to the rest of the system.
- A number of statistics are provided regarding their overall participation and contribution to the project, such as the number of readings produced and experiments participated, total online time (i.e., connected to the system and producing readings), etc. There is also a bar chart available, displaying the users' activity over the previous week.
- Apart from being able to inspect the content of the readings messages sent to the rest of the system, there is an debug screen available for developers to use, where lots of system-generated information is displayed continuously, so there is little need for developers to add custom debug mechanisms.

5.4 Web Portal

Apart from the user interface targeted towards the experimentation volunteers, we have also implemented a web portal in order to simplify the whole process of experimentation for the developers. As discussed, after having first developed their code adhering to the guidelines/rules set by the system, researchers can submit their code to be validated and then deployed over to the volunteers' devices. This web interface offers the following functionality:

- Developers can upload their generated experimentation plug-ins through a simple web form. Researchers also must pro-

[2]Apache Felix, http://felix.apache.org/site/index.html

vide both short and analytic descriptions of the aim their experiments, define which sets of device sensors and networking interfaces they wish to use (also in the form of a list of utilising other existing ADF and experimentation plug-ins), and also set a number of restrictions on the date and time they wish to see their experiments deployed and performed.

- If any experiments with public results are already available on the system (i.e., they have been performed and concluded in the past), developers can have a look at such data, and also get an overview of the associated experiment statistics (i.e., number of devices that participated, time of execution, etc).

- Researchers can have an overview of the execution of their submitted experiments, and the respective generated sensor readings. They can view such data both in raw format or in simple chart figures; they can also select to see the output of specific devices (by selecting a device ID that has been anonymously associated with each volunteer's device).

6. EXPERIMENTS - RESULTS

In order to validate our system and overall approach, we had 2 experimentation scenarios implemented in the form of plug-ins for our system, as explained in the previous sections. We carried out our experimentation in two cities, Santander (location of SmartSantander infrastructure) and Patras, Greece with very similar characteristics in landscape and population size (i.e., around 200K citizens, a large port and a city centre area similar in size). The first scenario involved the detection of WiFi networks available at street level (wardriving), while the second one involved the monitoring of noise levels in city centres. This set of scenarios was not chosen to showcase new innovative applications, but rather to present different ways of integration with an already installed IoT infrastructure, and how easy and efficient such a system architecture can be to develop for, at least compared to the existing alternative approaches.

With respect to the actual experiment runs, there were 30 volunteer users in total that took part in the experiments described here; however, only the second scenario was performed in both locations simultaneously. A large total number of 130346 reading messages were produced (59287 for the wardriving scenario and 71059 for the noise level monitoring scenario) during 7 days of experiments. In terms of actual contribution to the experiments, some users provided data for a single day while others contributed for longer time periods, while 3 users just registered their device and never actually produced readings (i.e., briefly interacted with the application and then quit). The vast majority of the volunteers used devices with a version of Android from 4.0.3 and above. We now proceed to discuss separately each scenario and the respective results.

6.1 Mapping WiFi across a City Center

The aim here is to showcase the easiness with which developers can utilise the volunteers' devices' networking interfaces; we chose this specific scenario due to its simplicity and straightforward concept. The devices scan for available WiFi networks together with GPS coordinates and periodically send measurements back to the experimentation server. The implementation took 386 lines of code and users were required to download a 1277KB experiment plug-in to start taking measurements. Essentially volunteers carry their Android smartphones in order to map free WiFi networks along the city streets. One could imagine all kinds of experimentation setups based on our system, where someone could e.g., monitor Bluetooth/WiFi interfaces to establish the number of mobile device users in a certain area, or infer face-to-face encounters between people visiting an exhibition or working in the same company.

The experiment was conducted only in Patras for 2 days, with 8 volunteer users, mainly as a test for the implementation of the system. The results gathered spanned across an area of 3 Km^2 (estimated), depicted in Fig. 6. The depicted area is almost 7 Km wide. Readings were taken by volunteers both when walking and while driving. Overall, a total number of 2728 WiFi networks were reported, of which only 18 were open (i.e., not requiring a password to join). With such data in mind, it seems a bit like searching for a needle in a hay stack; however, it is safe to say that is feasible to create the map of WiFi availability over a city in just a few days using this automated experimentation procedure. By creating a city coverage map, such data could also be used to make new routes for mobile IoT nodes, e.g., belonging to SmartSantander infrastructure, in order to utilise the open WiFi networks available in a city centre.

6.2 Urban Noise Levels Monitoring

This scenario aimed to detect ambient noise levels in a city center utilizing the volunteers' smartphone microphones. As mentioned in Section 2, this is an application scenario that has been implemented in several parts of the world the last decade; it is a characteristic example of participatory sensing and also an important one. Specifically in the EU, it is mandatory for all cities above a certain size to periodically monitor noise levels in certain important areas, as defined in [10]. The usual procedure followed is to have a number of calibrated microphones installed in specific locations, taking measurements for a day, and then relocating the equipment to other city spots to repeat the procedure. Thus, this procedure takes considerable effort and time to be completed, and by design it covers only the designated locations. Sound propagation models are then used to provide a more complete picture over a whole urban area.

Participants were instructed specifically to carry their devices in a way that leaves their microphones exposed (e.g., in a jacket pocket while walking), in order to have more reliable readings. The plugin implemented for the experiment calculates sound pressure levels and outputs decibel levels (dBA) together with GPS coordinates. It was implemented using a total of 382 lines of Java code, while volunteers need to download a total of 1171 KB for the experiment to begin. A total of 27 volunteers submitted readings during the run of the experiment in both locations. Some of the measurements are depicted in Fig. 6 and 5, illustrating differences in noise levels between busy streets and other areas, and between different periods of the day. In Fig. 6, noise levels are shown in the center of Patras and a suburban/campus location; there are clear differences in noise levels. Also, in Fig. 5 busy areas of Santander, i.e., the University of Cantabria campus and the commercial center are clearly depicted.

In the case of Santander, 45 IoT nodes equipped with microphones have been installed throughout the city center, aiming to provide a continuous stream of data regarding noise levels inside the city. However, these are stationary nodes, whose microphones[3] are equivalent to ones present in typical Android smartphones. Moreover, they are calibrated to return values between 50 and 100 dBA, filtering value both below and above these thresholds. By comparing readings returned by these IoT nodes and the devices in our experiments, we can see that the range of values from both sources is similar; the difference in average noise levels measurements between static nodes and smartphones in the same or nearby areas

[3]Libelium Smart Cities Board, Panasonic WM-61A microphones.

Scenario	Lines of code	Size of plug-in (KB)	#Participants	#Readings	Duration	Coverage
Wardriving	386	1277	8	59287	2 days	3 Km^2
Noise Levels monitoring	382	1171	27	71059	5 days	6.8 Km^2

Table 1: Overview of the experiments and respective results

ranges from 3 to 6 dBA. However, smartphones can also produce values below 50 dBA, e.g., in some "quiet" routes average values of 47 dBA, while in "busy" ones quite larger values. Also, many of the infrastructure nodes are placed on rooftops, explaining why the measurements from smartphones to a certain extent present greater variation. With respect to the area covered, in 5 the rectangle in the two rightmost figures shows the approximate location of the fixed infrastructure; it is clear that with just 5 days of an experiment run, we managed to gather measurements from a larger area.

In terms of the time periods that were monitored, in Santander there were measurements available for all hours of the day, while for Patras there were none available for the time period from midnight to 6:00am. Most readings were gathered between 12:00 and 20:00. In terms of total area covered, in the case of Santander an area close to 4 Km^2 was monitored, while in Patras the area covered was close to 2.8 Km^2. In other words, a fairly large area was covered in just a few days, and with longer experiment duration, it could have provided a fairly comprehensive noise levels map of both cities' centers almost round the clock.

6.3 Discussion - Limitations

The Wardriving scenario illustrates mainly how a smartphone experimentation platform can augment the networking capabilities of a given IoT city infrastructure. Although similar results can be achieved by using e.g., mobile IoT installed on taxis or buses, the scalability enabled by using the described approach cannot be understated. Furthermore, in the case of monitoring interactions through networking interfaces, stationary/mobile IoT nodes belonging to a FI testbed cannot monitor their designated areas continuously. However, volunteers that participate in a smartphone experimentation platform can fill in the "gaps" that are present in this case and therefore enhance the overall functionality of the platform.

Similarly, the noise level monitoring scenario, illustrates how to aid stationary infrastructure deployments with measurements covering broader area and filling in potential "gaps". However, although current smartphones integrate many sensing components, their characteristics pose certain challenges to integrate them within an IoT sensing infrastructure. E.g., microphones utilised are (naturally) not very accurate, and microphone calibration profiles are required in order to provide more meaningful values. Also, in the case of wind measurements produced do not correspond directly to noise levels and should be filtered somehow; areas that shared similar noise characteristics produced higher noise levels only due to being more exposed to winds than nearby areas.

Regarding the concept of experimentation, we could comment that there is a certain tradeoff between the number of experimentation volunteers and their commitment, the period of time to perform the experiment, and the quality of the results produced. The problem is that if we want good resolution throughout the day, e.g., as in the case of noise levels, this approach might not provide adequate results on its own but require combined effort with some fixed infrastructure. E.g., in the 00:00 - 06:00 time zone, there were no noise level measurements produced in Patras. However, in many cases this procedure can cover large areas to at least provide some indications/trends in limited time.

With respect to development effort, on the one hand with our approach researchers avoid the complexity of developing for an embedded highly specialised platform and instead use popular development tools for smartphone platforms. On the other hand, using such experimentation procedures can lead to creating an abundance of additional data, that further complicate the situation of making sense out of this data; additional effort may be needed for filtering and data mining in order to produce meaningful information.

7. CONCLUSIONS - FUTURE WORK

We presented in this work a system extending a well-established city-wide IoT experimentation platform. Such a system augments the existing paradigm for IoT testbeds in a logical and practical manner, augmenting the existing experimentation functionality with smartphones carried by volunteers inside urban areas. Our results show that it is easy to provide sensing coverage for wide areas by using smartphones and confirm previous results showing that sensors embedded in smartphones can produce results that can at least correlate with the respective results produced by much more expensive and sophisticated systems. On top of that, we demonstrate that such results can be produced in a relatively easy manner, developing for the popular Android platform and distributing the code to multiple clients transparently from both the experimenters and the smartphone users/volunteers.

With respect to our immediate plans for extending the current platform, we aim to make our system more efficient and simplify the process of producing sensor calibration profiles and integrating them to the system in order to produce more reliable results from the volunteers' devices. Enabling a more dynamic interaction with end-users is another direction we intend to pursue. We also plan to delve into matters such as gamification and user incentivisation, which are mostly absent from this work. Finally, we look into providing a more sophisticated DTN communication component between smartphones and IoT devices, in order to enable complex experimentation scenarios. We expect that the next phases in the evolution of experimentation in IoT will inevitably rely, to a certain extent, on the voluntary involvement of non-experts in data collection and large-scale experiment participation. These are all aspects that should be more aggressively encompassed in order to identify key research challenges, future directions and major scientific questions common to inspire future, long-term research.

Acknowledgments

This work has been partially supported by the EU research project SmartSantander, under contract number FP7-257992. We would also like to thank all of the volunteers that participated in the reported experiments.

8. REFERENCES

[1] K. Aberer, S. Sathe, D. Chakraborty, A. Martinoli, G. Barrenetxea, B. Faltings, and L. Thiele. Opensense: Open community driven sensing of environment. In *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming*, IWGS '10, pages 39–42, New York, NY, USA, 2010. ACM.

[2] D. Carlson and A. Schrader. Dynamix: An open plug-and-play context framework for Android. In *Internet of Things Conference*, pages 151–158. IEEE, 2012.

[3] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 16(5):36–44, 2012.

[4] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *J. Syst. Softw.*, 84(11):1928–1946, Nov. 2011.

[5] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 211–224, New York, NY, USA, 2008. ACM.

[6] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. Prism: platform for remote sensing using smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 63–76, New York, NY, USA, 2010. ACM.

[7] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, and B. Nath. Real-time air quality monitoring through mobile sensing in metropolitan areas. In *Proceedings of the 2Nd ACM SIGKDD International Workshop on Urban Computing*, UrbComp '13, pages 15:1–15:8, New York, NY, USA, 2013. ACM.

[8] E. D'Hondt, M. Stevens, and A. Jacobs. Participatory noise mapping works! an evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, 9(5):681 – 694, 2013. Special issue on Pervasive Urban Applications.

[9] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 29–39, New York, NY, USA, 2008. ACM.

[10] EU Directive 2002/49/EG, Official Journal of the European Communities, 2002.

[11] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49:58–67, 2011.

[12] V. Gutierrez, J. Galache, L. Sanchez, L. Munoz, J. M. Hernandez, J. Fernandes, and M. Presser. Smartsantander: Internet of things research and innovation through citizen participation. *Book chapter, Future Internet Assembly 2013, Dublin, Ireland*, 2013.

[13] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. Participatory air pollution monitoring using smartphones. In *Proc. 1st Int'l Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, 2012.

[14] S. Kim and E. Paulos. Inair: sharing indoor air quality measurements and visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1861–1870, New York, NY, USA, 2010. ACM.

[15] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *Comm. Mag.*, 48(9):140–150, Sept. 2010.

[16] N. D. Lane, J. Xie, T. Moscibroda, and F. Zhao. On the feasibility of user de-anonymization from shared mobile sensor data. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, PhoneSense '12, pages 3:1–3:5, New York, NY, USA, 2012. ACM.

[17] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell. Bubble-Sensing: A New Paradigm for Binding a Sensing Task to the Physical World using Mobile Phones. In *Proceedings of the International Workshop on Mobile Device and Urban Sensing (MODUS)*, Apr. 2008.

[18] A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen. Phonelab: A large programmable smartphone testbed. In *Proceedings of First International Workshop on Sensing and Big Data Mining*, SENSEMINE'13, pages 4:1–4:6, New York, NY, USA, 2013. ACM.

[19] U. Park and J. Heidemann. Data muling with mobile phones for sensornets. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 162–175, New York, NY, USA, 2011. ACM.

[20] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 105–116, New York, NY, USA, 2010. ACM.

[21] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive'10, pages 138–155, Berlin, Heidelberg, 2010. Springer-Verlag.

[22] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava. Biketastic: sensing and mapping for better biking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1817–1820, New York, NY, USA, 2010. ACM.

[23] L. Sanchez, L. MuÃśoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61(0):217 – 238, 2014. Special issue on Future Internet Testbeds - Part I.

[24] I. Schweizer, C. Meurisch, J. Gedeon, R. Bärtl, and M. Mühlhäuser. Noisemap: multi-tier incentive mechanisms for participative urban sensing. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, PhoneSense '12, pages 9:1–9:5, New York, NY, USA, 2012. ACM.

[25] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan. Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, pages 9:1–9:6, New York, NY, USA, 2013. ACM.

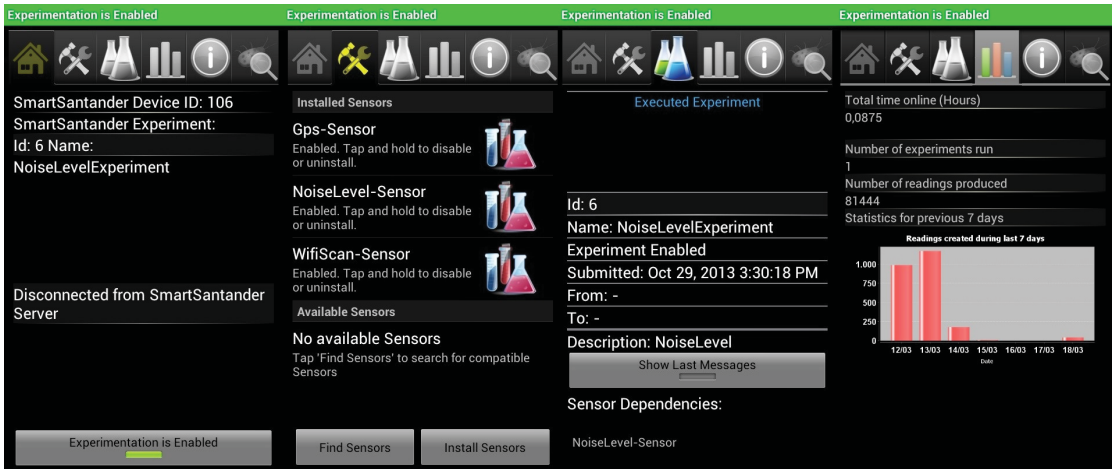

Figure 4: Sample screens from the smartphone user interface - the first screen (leftmost) depicts overall info about the device and connection status to the system, the second displays user preferences regarding the use of specific sensors, the third shows information about the experiment currently executing on the device, and lastly, some statistics about the specific device (e.g., number of readings contributed).

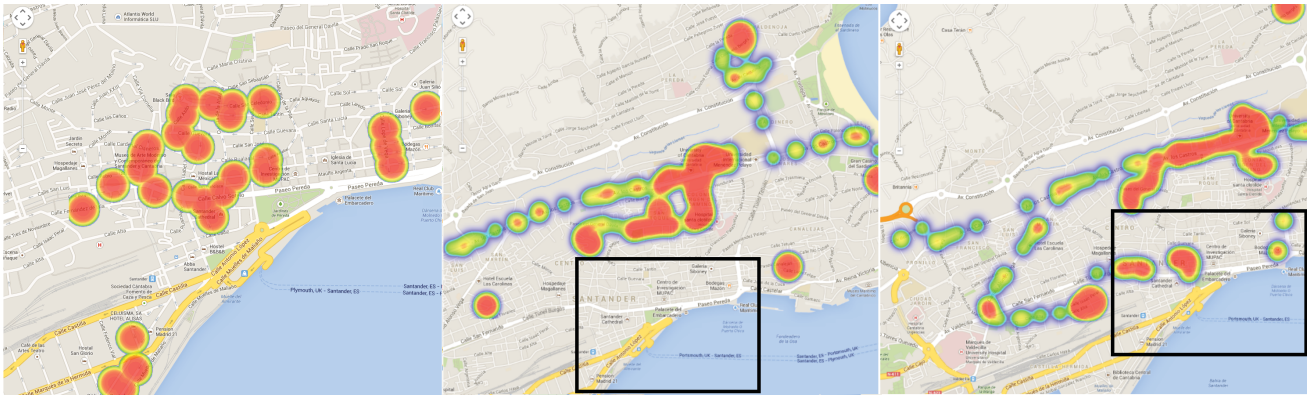

Figure 5: Results generated by experiments run by volunteers at Santander - the first figure (left) shows measurements by the stationary IoT nodes of SmartSantander. The center figure shows a heatmap of average noise levels measurements produced by smartphones between 12:00 and 18:00, while the rightmost figure shows measurements between 18:00 and 24:00. The rectangle shows the approximate location of the static infrastructure.

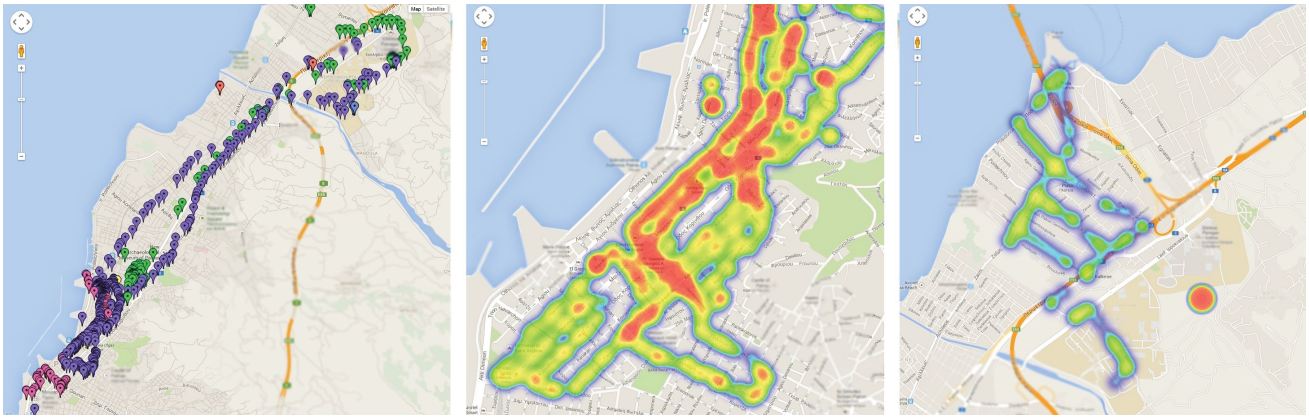

Figure 6: Results generated by experiments run by volunteers at Patras - the leftmost figure shows the WiFi measurements as placemarks. The center (city center) and rightmost (suburb and university campus) figures show average noise levels between 18:00 and 24:00.