Sensorem - An Efficient Mobile Platform for Wireless Sensor Network Visualisation

Jin Ming Koh*, Marcus Sak*,
Hwee-Xian Tan†, Huiguang Liang†, Fachmin Folianto† and Tony Quek‡
*NUS High School of Mathematics and Science, Singapore
†Institute for Infocomm Research (I^2R), Singapore
‡Singapore University of Technology and Design (SUTD), Singapore

***Abstract*—We design and implement an Android application, Sensorem, for efficient retrieval and visualization of wireless sensor network (WSN) data. In light of data distribution and visualization being important developmental keystones of smart cities, we seek to enhance sensor data accessibility by developing a user-friendly mobile application (Sensorem) for meaningful visualization of sensor data, targeted at maintenance personnel. Sensor selection can be made with reference to geographical location through an embedded Google Map fragment, as well as a sensor list with sensors in order of WSN node ID. Sensor data is presented through interactive graphs, and graph overlaying functions are offered for easy comparison of data trends. We also employ a Bayesian prefetch algorithm and caching mechanisms to minimize sensor data access latency, such that the app system is able to cope with network and back-end bottlenecks.**

I. Introduction

In light of the current and projected expansion of the urban population [1], there is a need for smart cities that employ Internet of Things (IoT) to maintain high living standards [2]. Wireless sensor networks (WSNs), as the sensing-actuation arm of the IoT, integrates into the urban landscape and provides sensory input such as temperature, Pollutant Standards Index (PSI) levels, noise levels and humidity [3].

In this project, we extend the visualization and management of sensor data from such networks to mobile platforms by developing an app named Sensorem, specifically targeting the Android platform. We build on concepts established on the UNISENSE desktop application developed by the Sense & Sense-Abilities group for the WSNoise project. We aim for *Sensorem* to be a tool used in the field by maintenance personnel who can access sensor details and data on demand without the hassle of desktop hardware. Ultimately *Sensorem* may be extended to the public as a mobile front-end platform to information collected by their cities' sensory networks. We envision *Sensorem* to increase urban lifestyle transparency and ensure widespread smart citizen participation by serving as an efficient network and data visualization platform, meeting the key aspects of smart city approaches.

II. System Architecture

A. Overall design and GUI

The app is designed to be lightweight on device resources and functions across a wide range of screen sizes, with 7 to 11 inches being the designed optimal. These ensure portability and ease of integration into existing or future infrastructure.

The key design pillars of *Sensorem* include:

1) *Accessibility.* Technical expertise should not be a prerequisite for easy usage.
2) *User-friendliness.* Feature actuation and navigation in the app should be intuitive and responsive.
3) *Information-rich Interfaces.* There should be minimal compromises on the amount of information presented to achieve the goals above.
4) *Expandability and Scalability.* The app should allow easy integration of future functionalities and preserve user experience whilst coping with significant back-end bottlenecks.

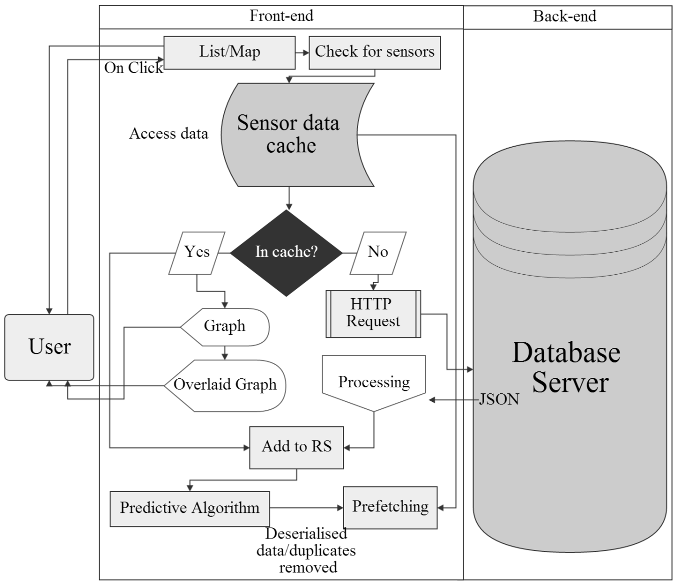

Fig. 1. Simplified system block diagram for *Sensorem*.

Figure 1 shows a simplified system block diagram of *Sensorem*. The design goals of accessibility and user-friendliness calls for a simple app framework. App start-up entails the loading of network information, after which a menu with 4 options appear - *Sensor Map*, *Sensor List*, *Preferences* and *About*.

The *Sensor Map* visualises the geographic distribution of network nodes. Battery levels are presented as colour codes, and basic node information is available upon selection. In contrast, the Sensor List presents nodes in a vertical list arranged in a selectable ascending or descending node index order. Filter and search functionalities allow for nodes to be queried based on location, type, battery level, and node index.

Sensor data and detailed specifications are presented upon clicking on a map or list entry, with the former presented in measurement-time graphs with gesture-based zoom and pan functionalities, and the latter in a concise scrollable table. Graphs of different modalities can be accessed by swiping horizontally. In addition, selected graphs can be overlaid for the visualization of trends (Bottom left of Figure 2).

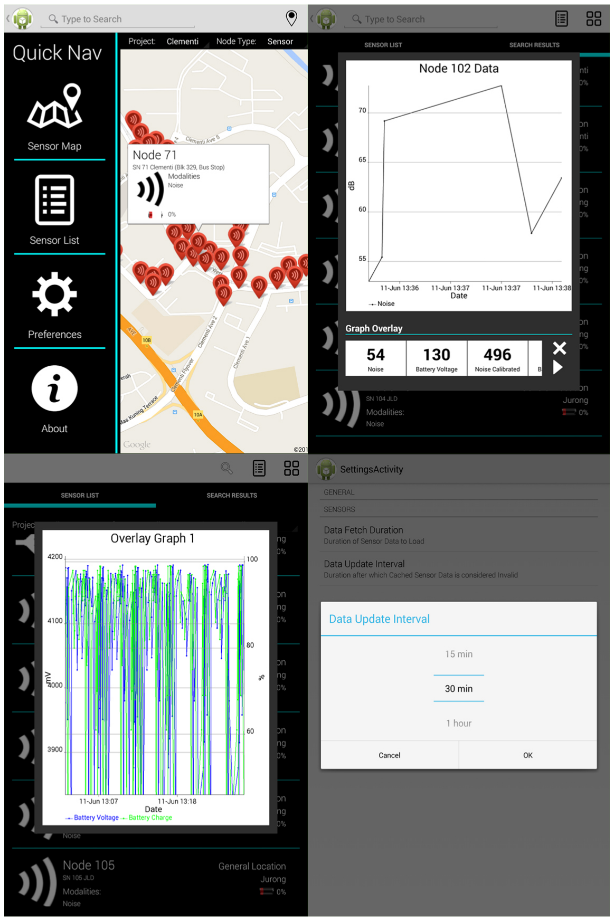

Fig. 2. Selected screenshots of *Sensorem*. Clockwise from top left: *Sensor Map* with quick navigation menu; *Sensor List* with single data graph; Overlaid graph; *Preferences* screen.

Throughout the app, a quick navigation menu is accessible by swiping from the left bezel.

Key app behaviours can be modified under the *Preferences* section. Adjustable parameters include the amount of data to load by default and information freshness thresholds (Section II-C). A particularly notable option is the prefetch mode. We recognize that the prefetch mechanism implemented may consume network data and introduce network fees should the device rely on data plans, and therefore introduced a 'smart prefetch' feature. In this mode, prefetch will only be activated when the device is connected to Wi-Fi.

B. Implementation

The *Sensor Map* was implemented using the Google Maps platform, on the premises of general ease of use and user familiarity. Customized MapFragments present the primary map view, whilst dynamic response to screen orientation was provided through view containers with custom event capture methods. The *Sensor List*, on the other hand, was implemented entirely using Android native libraries and consists of a base assembly of ViewPager and PagerTabStrip upon which ListViews and other UI elements are layered.

AChartEngine was chosen as the primary graphing library on the basis that it has excellent support for inheritance-based customizations, an important aspect for the implementation of graph overlays. The stock functionalities of the engine is by themselves sufficient for the formatting of graphs, however, performance issues with the library called for customized implementations of internal data processing and rendering. These include the modification of data structures within the engine to reduce redundancy and the lowering of anti-aliasing quality.

A third-party library, named GSON, was utilized for the serialization and deserialization of JSON-formatted information for communication with the network servers. Server requests are based on a standardized API used for the UNISENSE servers. Notably, the communication process is considerably wasteful, for the device has to wait for the server to respond. To reduce stall times, several HTTP requests are made and processed in parallel through the use of AsyncTasks.

C. Scalability

The scale of WSNs required for the development of a smart city is estimated to be orders of magnitude greater than the typical testbed at current times. To preserve applicability in the future, the developed app system must be able to maintain performance and user experience in the event of network and back-end bottlenecks that arise due to WSN expansion.

To achieve this scalability goal, we implement prefetching and caching mechanisms into *Sensorem*. A prefetch algorithm based on Bayesian inference makes predictions on data likely to be requested next by the user, prefetches the batches of data, and stores them in cache [4]. If the prefetch is accurate and is successfully accomplished before the user accesses the data, then the need to contact the server for data download is negated, and therefore stall-time is reduced. The algorithm draws on previous user selections and corresponding prediction accuracies in tandem with an accuracy-based weightage system to intelligently adapt to usage style and generate predictions. We achieve a 48.4% reduction in user-perceived data retrieval latency using this method [4].

References

[1] Department of Economic and Social Affairs, United Nations, "World urbanization prospects: The 2011 revision," 2011.

[2] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, "Smart Cities of the Future," *The European Physical Journal Special Topics*, vol. 214, no. 1, 2012.

[3] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami, "Network architecture and QoS issues in the internet of things for a smart city," in *ISCIT*, 2012.

[4] J. M. Koh, M. Sak, H. Tan, H. Liang, F. Folianto, and T. Quek, "Efficient data retrieval for large-scale smart city applications through applied Bayesian inference," in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2015.