Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2011) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 3rd International Conference of Sustainable Energy Information Technology

Designing a Portable and Low Cost Home Energy Management Toolkit

David Keyson, Abdullah Al Mahmud, Marc de Hoogh, & Rob Luxen

Faculty of Industrial Design Engineering
Delft university of Technology, the Netherlands

Abstract

In this paper we describe the design of a home energy and comfort management system. The system has three components such as a smart plug with a wireless module, residential gateway and a mobile app. The combined system is called a home energy management and comfort toolkit. The design is inspired with the fact that making energy visible and able to control it will help to conserve energy. One of the key goals is to create a platform which is extendable based on the needs of the end users. Our proposed system is expected to help end users to control and manage residential energy. We also present some lessons learned while implementing the proposed system.

Key words: Sustainability, home energy management, ZigBee

1. Introduction

Energy use in home accounts for significant part of total energy consumption both in developing and western world. Residential buildings currently account for large part of the total energy demand [9]. There are several efforts to reduce home energy use such as minimize cooling and heating loads, use low energy domestic equipment and promote and achieve energy conscious behaviour among residential energy consumers [5, 6]. Therefore, it is very important to provide solutions for end users to conserve energy. Furthermore, it is equally important to create usable solutions. Home Energy Management (HEM) describes a class of technologies including sensors, smart thermostats, and feedback devices seeking to manage residential energy consumption profiles to reduce peak electric demand and consumers' electric bills [9]. Despite research showing that HEM products providing feedback on electricity consumption can reduce household electricity consumption by 4 to 12%, these products still face relatively low market penetration and those systems are relatively costly as well [4]. Therefore we would like to design a low cost HEM system. The main objectives of our system are as follows. To enable and aware the end user to make decisions concerning when a) trade-offs for switching appliances on and off at a given time of day,

b) schedule the ventilation and screens of the house based on preferences/weather condition, c) enable user to set temperature settings by different zones based on preferences/presence, d) inform the user expected and current amount of solar energy/net electricity, e) provide advice on energy storage, given planned electrical vehicle (EV) connected to a house.

In order to realize research activities we have developed a research house (i.e., living lab) called Concept House. The Concept House Prototype is a partnership between academia and several companies from the building industry together with the Interreg SusLabNWE (suslab.eu) project which provides investments in the lab infrastructure. The research house has been developed for the Dutch housing market in response to energy, environmental and engineering issues. The house has been constructed in anticipation of the legal requirement for an EPC (building-related energy consumption) of 0.0 in 2020. An integrated research approach was chosen in the design process, given the aim of developing long-term practical and result-oriented solutions.

Our proposed smart phone home energy management system involves developing the smart sensing hardware and user interface for the Concept House. The interface enables occupants to view energy consumption of all electric sources in the house and enable the user to set savings goals. Furthermore the interface supports energy storage via an electric vehicle attached to the house. The hardware unit consist of plug in modules combined with cloud computing on a dedicated server, such that it will be a plug and play system with no need to have a computer running in the house. Control options in the house are: a) switching lights on/off, b) change the intensity of the lights: low/medium/high, c) turn the plugs on/off, d) which window screens are on/off, e) show overview of the consumption and tariffs on different levels: week/month/current, f) heating control of floor zones and cooling in roof. The contributions of the work are: a) the design of a home energy toolkit which is easily extendable and deployable in regular houses b) implementation of an energy centric user interface, and c) creation of a robust database for gathering sensor values and also easy to upscale the database when additional sensor nodes are added. The paper is organised as follows. In the following section we describe related work followed by the proposed architecture and implementation of the home energy management system.

2. Background and related work

ZigBee is a an emerging network technology and used as a communication protocol. It is popular due to its low cost and wide coverage [8]. Therefore ZigBee has been used in in home network and home automation [2,3]. There have been several approaches for electric power management in homes from using ZigBee sensor networks [3, 7]. One example of real time energy monitoring and controlling system based on ZigBee is mentioned in [2]. The system had limited applications or services: monitoring of electricity consumption and remote on/off control. Smart services such as tariff comparison, recommendation for usage and time-based on/off control. Furthermore, the system utilized power sensor which need to be deployed separately whereas in our approach we created smart plug to get the value of consumption in each appliance connected to each plug in house. Another example of smart home system is based on IPV6 and ZigBee technology [1]. In this approach the main objective was to find a compatible solution between ZigBee and IPV6 networks. In addition most of the systems of smart metering for homes are only focusing on measuring the total amount of energy consumption or electricity. On the other hand in our approach we aim to measure or monitor the electricity consumption of each home appliance and provide better intelligent services such as statistical analysis, and setting energy saving goals etc. Furthermore, we aim to find a solution which is extendable and scalable and easy to deploy in regular houses.

3. System architecture: description of the first prototype

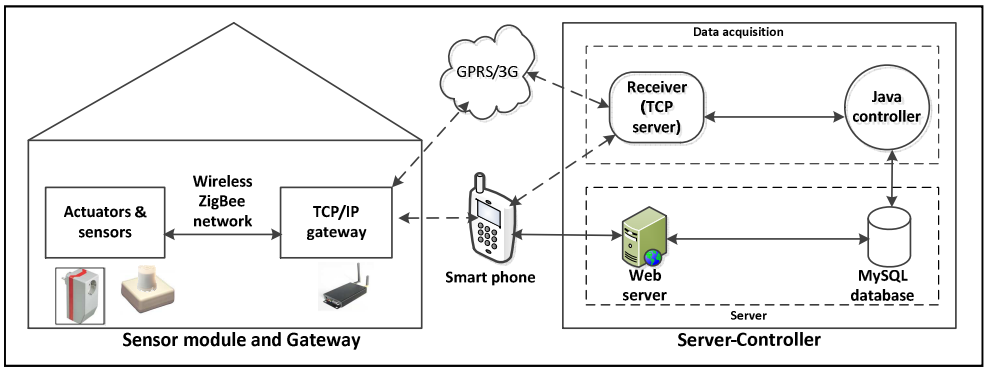

Fig 1: The first prototype- the smart plug (top left), wireless module (top right), general architecture (bottom)

The general architecture is depicted in Fig. 1 (bottom). Inside the home, sensors are monitoring the physical environment all the time. Whenever the sensor value changes it is sent wirelessly to the gateway device located in the home. The sensor gateway timestamps the sensor values and sends them over the 3G data communication network to the application server running on a computer. Using the current location of the sensor the application server determines the object the sensor data is associated with, and stores the sensor data in the associated state events table. The location and the object that contains the sensor are also stored in the database, so a sensor can easily be placed somewhere else. Applications like the one the home owner is using can register themselves with the application server and receive the current state of objects they are interested in real-time or regularly.

4. Implementation of the first prototype

4.1 Hardware components: smart plug

The smart plugs can actively participate in home monitoring and control activities. Smart plugs are able to collect metering data and implement on/off control on simple plugged energy loads.

4.2 Storing the sensor values: Database design

In defining the tables we assumed that we receive sensor values defined by two integer ids. The object id and the object input id. The object id directly matches the _ID in the OBJECTS table. The object input id is stored in the IN_INDEX column of the state variables associated with the given object. Based on the unique object input id the state variable id can be found, and the latter id is used to store the received input value. Multiple tables can be used to store these events, as every input value can have a different value type associated with it. For each value type id used for a given object a separate unique events table is (to be) created.

Every object is associated with an object type that defines the state and therefore the list of state variables of the object. The integer id of the state variable is stored in the STATEVARIABLE_ID column of the events table to identify the input source. Obviously for a given sensor architecture and single location (house) the sensors are installed in, the objects and object types will be the same. Therefore we assume that all data obtained for a specific house are stored in the same database i.e. never are data from different locations stored in the same database. The object types defining the states of the objects and the objects and all other tables that are used by the object types (see a description of the tables below) should remain immutable during the duration of an experiment. Therefore we allow the definition and therefore creation of so-called sessions, and when a session is created all tables used in the session are created or copied from the original tables and used in the session. Therefore the originals on which the session is based can be changed (e.g. to define another session) at the same time a session is running without interfering with it.

Not all object type tables are copied per se to a session. The object types e.g. are defined by 3 tables: OBJECTTYPENAMES, OBJECTTYPEPARENTS and OBJECTTYPESTATEVARIABLES, thus allowing object types ‘inheritance’. But at run time in a running session we can do with a single table defining the state variables of the object types used in the session. It is yet to be determined whether or not to create separate tables for each object type or to use a single table to store the state variables of all object types. Therefore the ‘inheritance’ is eliminated at runtime. Of course, it is easier not to use this inheritance, and eliminate the OBJECTTYPEPARENTS table altogether. That’s really up to any specific implementer. Apart from the fixed session object and object type tables a number of tables are created for storing the sensor data. In essence the received sensor data values are stored in so-called state events tables. For each object separate state events tables are defined. So for an object called SENSORBOX1 the state events tables would be called SENSORBOX1_STATEEVENTS for storing sensor data as text, or SENSORBOX1_STATEEVENTS_<valuetypeid> for every value type id defined in the state of the object type, say SENSORBOX, of SENSORBOX1. So, if no value type id is defined for all state variables defined for the object, only the base state events table will be created and used. Apart from these state events table we will also create states tables to store the complete state of an object at a given moment in time as a single record; in comparison, every record in a state events table stores a single value. State variables marked as being fixed are NOT stored in these states table. Suppose the object type defines 3 state variables with _ID 4, 7 and 8 that are not fixed. Then, the states table should contain three columns called VAR4, VAR7 and VAR8 of the right type to store the separate values. Only a single states

table is obviously required per object. Furthermore, the state variable definition also tells us whether it is external or not. If it is external the input value represents a sensor measurement. In that case it is a measurement of the object in its environment. And what is being measured could well represent something to be stored in the container of the object. This means that every object with external state variables should be able to determine its container. In our setup we assume that the object id of the container is stored in one of the other state variables therefore the object knows the container it is contained in in some of its state variables. The id of this state variable is stored in the EXTERNAL_OBJECTTYPESTATEVARIABLE_ID column. As such it is possible to determine the container at runtime. Next to the container we also need to know the state variable of the container to store the sensor value in. This is defined by the MEASURAND_ID. This id tells us what the sensor value represents. Once we know the container object id we can locate the state variable with the same measurand id in the container. This will be the state variable id to use in generating the event for the container. Finally we allow applying a stored function to the value that we store with the container. As an example consider measuring air temperature in a specific room. Both the state definition of the measurement device and the room contain a state variable with the same measurand id of air temperature. Note however that the room state variable with this measurand id should not be defined as external because in that case it would be assumed that it is measured by the room instead of received. To be able

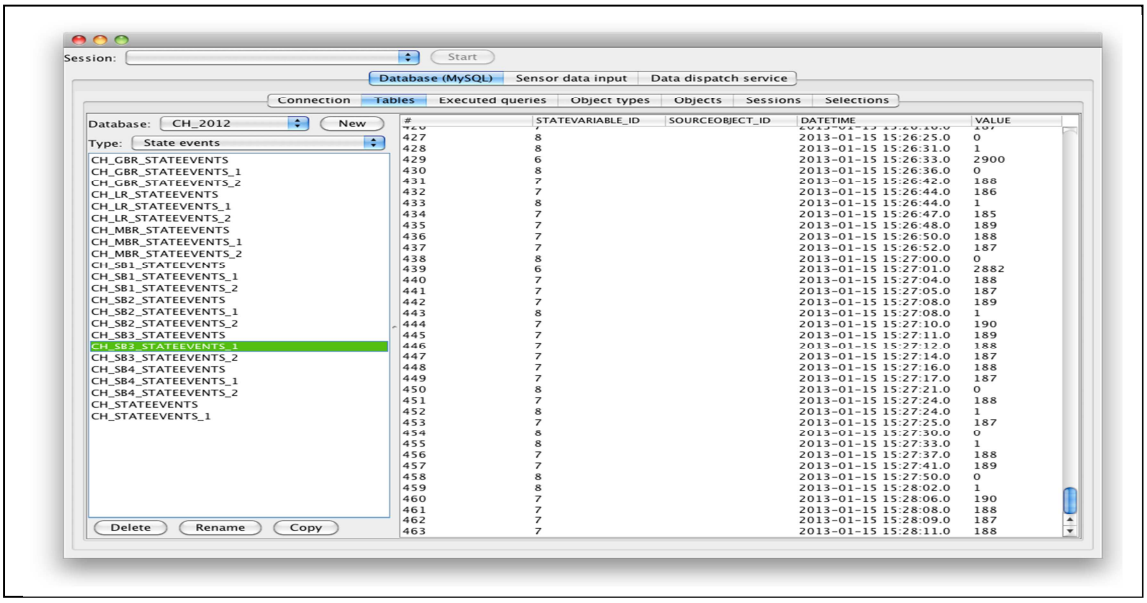

Fig 2: Snapshot of the Implementation of the sensor box state events table

to use the database for actual sensor event recording the tables to be used for that need to be defined. The design consists of defining an object ontology, like room, sensor, sensor boxes, and, subsequently, any number of (categorized) objects representative of the objects in the measurement system, so specific rooms, sensors and sensor boxes. The object ontology categories are defined and stored in the ObjectTypes table. For each category a state needs to be defined i.e. a list of state variables to be stored in a table associated with the object type. Object categories may include other object categories e.g. a box container may 'inherit' from both box and container base object types. Therefore a number of tables needs to be predefined, and it is up to the designer to define them. To start designing any measurement database system a set of initial (empty) template tables can be used as starting point. However, these tables do not need to be empty. It is therefore possible to start with any set of initial tables

and build upon that. Once the database design is completed it can be used in an active session. To this purpose the database design application allows for the definition of sessions. Once a session is defined all required tables are copied over from the database containing the definition. The tables defining the objects and object types then become read-only so that they cannot be changed anymore. The tables to contain the states and state events start out empty. Sessions are run in the application server, where they can be selected. Sessions typically encompass taking measurements regarding a fixed set of users, so the data are clearly separated. Most of the above has been implemented in a Java application that would both allow one to edit the different tables and receive and store sensor data.

4.3 User interface design

The user interface is energy centric such that each appliance or group of appliances is represented as an icon. The savings goal is indicated by the outer ring color. The thickness of lines shows the amount of flow to an icon. The size of the icon indicates how large the energy group is. This is a break from traditional home energy management controllers which are typically device centric. Under each icon the energy savings goal settings can be found. The user can also view historical energy use. The device will support speech recognition of consumption enquiries. For example, the user can ask the system “how much electricity did my washing machine and oven use last month”. The user interface is built as a web

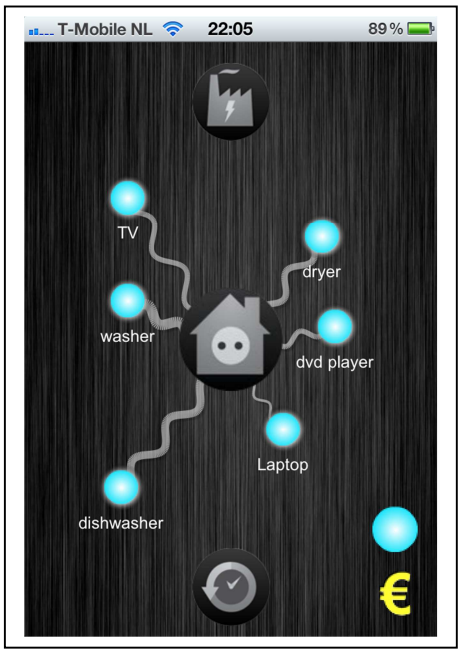

Fig 3: The main screen of the implemented UI for the proposed home energy management system

page plug-in app, being platform independent. The plug in hardware module units each feature:
a) energy consumption for product that is plugged in, b) plug on/off control for certain appliance types, c) Co2 sensing, d) light sensing, e) Motion sensing, f) temperature sensing , g) humidity sensing, h) remote communication module for additional sensors (e.g. air flow), i) wireless connection to monitor PV energy supply levels, j) UMTS link to cloud server. The additional remote sensors will be used to support the research work on indoor climate quality and comfort management.

4.4 Home Gateway

The Home Gateway represents the link between the home network and the mobile application. It is able to interface smart plugs and other user's devices through the communication protocol. We have used a ZigBee/WiFi gateway [10] for our implementation. The main objective was to find a cheap gateway for our purpose.

5. Pilot testing of the proposed system: lessons learned

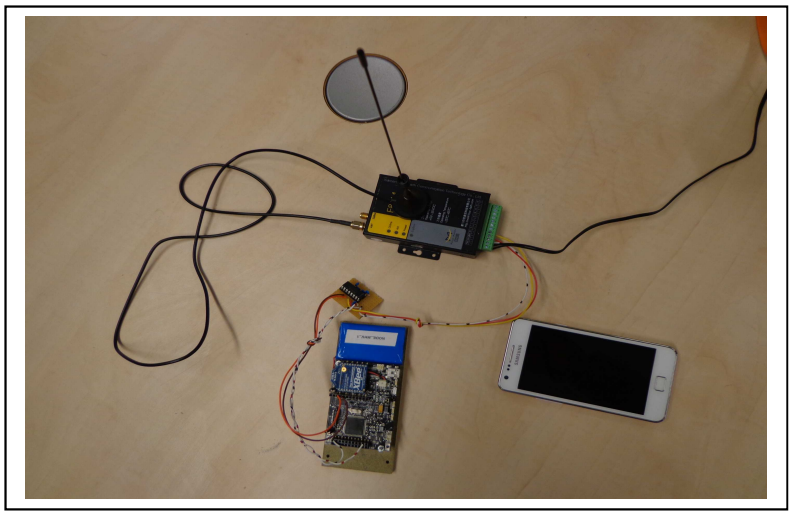

Fig 4: The test setup of the proposed system. The mobile app was used to control a light bulb in two locations using WiFi and 3G.

We have tested our home energy management toolkit in three research locations (e.g., Germany, UK and the Netherlands). The objective was to validate the proposed technical solution. During the testing we learned some technical challenges that we explore below: ZigBee is not a technical standard, so multiple implementations, so-called application profiles exist. Base Zigbee is built on top of the IEEE 802.15.4 standard that defines the physical layer and media access control layer of the wireless protocol stack. Digi sells XBee Series 1 modules that can be used in a ZigBee point to point or star network (one module functions as PAN: personal area network coordinator), and XBee Series 2 modules that can be used in a so-called mesh network, where any device is either the (single) coordinator, a router (able to that relay messages) or an end device (unable to relay messages). Star networks are much easier to set up but the distance to the central PAN coordinator should not be too large, as there are no router modules on the path to the coordinator that can relay a message. A mesh network is easy to extend because one can simply add another router at a suitable location to extend the working range of the network. One of the reasons to use ZigBee is to avoid having to use the more energy-inefficient WiFi. But getting rid of WiFi is not that easy. Response times increased noticeably to 2 to 5 seconds when using a smart phone to control a single light in the house as the command needed to travel forth and back to the central server over 3g. Given the fact that one expects response time to be almost instantaneous indoors, some WiFi access point would need to be present to facilitate apt response times. When testing abroad connecting to the server over 3G with the same SIM card would succeed, however exchanging data would fail due to unknown reasons.

6. Conclusion and future work

Awareness and feedback about electricity consumption is a key aspect to save energy. The ability to regulate energy behavior with appropriate technologies for monitoring and controlling energy consumption is a powerful vehicle for reducing energy demand. Our proposed system can monitor and measure electricity usage in real-time. With the proposed system, users can remotely control real-time electricity usage through web and other mobile devices such as smart phones or smart pads. We are now working to upgrade the whole system. Furthermore, for the database design, the session concept is not yet implemented, in that all data is still stored in the original tables. So, the ability to create sessions needs to be implemented in the near future. Also, it would seem to make sense to actually split up the application in two parts, a design application in which one would define the sessions and an operational application in which one could select a session to run. Currently we are building the next version of our smart plug to support a mesh network in a home environment. Finally, we will test our system in our research house (suslabnwe.eu). Based on the test results we will deploy the system in regular Dutch and other European houses.

Acknowledgements
We thank all the sponsors for supporting this project.

References

[1] Zou, Z., K.-J. Li, et al. (2011). "Smart Home System Based on IPV6 and ZIGBEE Technology." Procedia Computer Science 15: 1529-1533.

[2] Kim, W. H., S. Lee, et al. (2011). "Real-time Energy Monitoring and Controlling System based on ZigBee Sensor Networks." Procedia Computer Science 5: 794-797.

[3] Gill, K., S.-H. Yang, et al. (2009). "A ZigBee-based home automation system." Consumer Electronics, IEEE Transactions on 55(2): 422-430.

[4] Fischer, C. (2008). "Feedback on household electricity consumption: a tool for saving energy?" Energy Efficiency 1(1): 79-104.

[5] Hargreaves, T., M. Nye, et al. (2012). "Keeping energy visible? Exploring how householders interact with feedback from smart energy monitors in the longer term." Energy Policy.

[6] Hargreaves, T., M. Nye, et al. (2010). "Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors." Energy Policy 38(10): 6111-6119.

[7] Bayindir, R., E. Irmak, et al. (2011). "Development of a real time energy monitoring platform." International Journal of Electrical Power & Energy Systems 33(1): 137-146.

[8] Lee, J.-S., Y.-W. Su, et al. (2007). A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE, IEEE.

[9] Sioshansi, S. (2011). Energy, Sustainability and the Environment. ISBN: 9780123851369

[10] Gateway. http://www.rexense.com/en/products_det.php?classid=13&idd=70