INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTERS IN SIMULATION
DOI: 10.46300/9102.2021.15.26 Volume 15, 2021

System Architecture of Unobtrusive Sensors for Supporting Home Care and Independendet Living

Cvetko Pirš, Boris Cigale, Damjan Zazula
University of Maribor, Faculty of EE and CS
Smetanova 17
2000 Maribor, Slovenia
{cvetko.pirs, boris.cigale, zazula}@uni-mb.si

Dejan Usar
Gorenje d. d.
Partizanska 12
3320 Velenje, Slovenia
dejan.usar@gorenje.com

Received: August 15, 2019. Revised: September 9, 2021. Accepted: October 10, 2021. Published: November 27, 2021.

Abstract—**The paper deals with an implementation of unobtrusive sensors installed in home environment for continuous monitoring of functional-health parameters of the observed persons. A multi-tier architecture links sensory devices through sensor-data concentrators to a home server. Automated sensory measurements are supported by a concept of sensor-activated events, event-driven data transmission and processing by a dedicated application interface. Its logic and data structures are revealed. Examples of three typical execution scenarios are given and a short description depicts clinical installation of proposed system for testing purposes.**

Keywords—sensory networks; unobtrusive sensors; home server; system architecture; functional health; home care; independent living

I. INTRODUCTION

By 2050, as much as 30% of people will be elderly, among them 11% aged over 80 years [1]. National health systems are investing more than 1.5% of economic output in the provision of long-term health care, which means 150 billion annually. In three decades, expenditures will increase even up to 2.8%. Already by preventing falls, which occur with serious injuries in 40% of the elderly, substantial savings may be expected. A large proportion also applies to chronic diseases and inability to stay independent.

The solution for these problems should focus on the individuals inside their everyday living environment. However, it can be achieved optimally only if an automated and unobtrusive assessment of the functional-health parameters (FHPs) is feasible. Monitoring of daily living activities helps detecting changes in residents' daily routines, which is one of the key supporting features of a smart home. Today's inexpensive low-power sensors, embedded processors, and wireless communications are available technologies that are typical building blocks for larger networks of sensors. These assist unobtrusive home healthcare [2].

Several research and development projects have reached significant level of smart-home solutions and support to independent living. System architectures applied connect environmental sensors with data loggers, servers, and data bases. The authors of [3] studied the ISO X73 upper-layer substandards, i.e., nomenclature specification, domain information model, application profiles, and vital sign device descriptions to verify suitability for smart homes. They measured body temperature and weight, blood oxygenation, and electrocardiograms (ECG) by using data loggers. The CASAS project reveals “a smart home in a box” by the architecture that controls data flows from the physical components through the middleware to the software applications, and vice versa [4]. Sensors in a smart home generate events that consist of a date, a time, a sensor identifier, and a sensor message. The goal of the system is to recognize the residents' activity and to map a sequence of sensor data to a corresponding activity label. Elite Care environment [5] introduces infrared and radio frequency sensors for locator badges that also help caregivers in the in- and outdoor alert situations to react rapidly. Higher level of awareness of the residents' situation is obtained by combining location and movement sensors by video cameras, such as in the TigerPlace project [6]. The House_n project initiated by MIT builds on hundreds of sensing components that are installed in nearly every part of test home [7]. The sensors are, among others, being used to monitor activity in the environment so that researchers can carefully study how people react to new devices, systems, and architectural design strategies in the complex context of the home.

All smart-home projects implement ubiquitous sensing and pervasive computing, mainly to monitor residents' behavior and remotely detect critical situations. There are components as well that collect data on the residents' health status, although these are, in general, neither overwhelming nor exclusively unobtrusive. On the other hand, we designed and developed system architecture that supports a flexible system of distributed, unobtrusive sensors ready to be installed in home environments with a primary goal of monitoring and assessing residents' FHPs. Its hierarchical concept is revealed in this paper. Section II depicts basic system requirements and design principles. Prototype system architecture is described in Section III and exemplified by typical cases and implementation in Section IV, while Section V concludes the paper.

Authors acknowledge the financial support of the Slovenian Ministry of Education, Science, and Sport, and the European Regional Development Funds for the Biomedical Engineering Competence Centre.

II. System Requirements and Design

The architecture of proposed system consists of multi-tier design that connects sensory devices with home server and database [8]. At the hardware level, this requires additional computational power provided by sensor data concentrators (SDCs) and home server (HS).

SDCs are used as a bridge between sensory devices (SDs) and HS (Fig. 1). SDCs are linked to HS by the USB or wireless connection. Sensory data are transmitted in a proprietary event-driven protocol as described below. HS provides software support for data acquisition, data processing, and connection to a healthcare centre (HC).

Distributed SDs are installed in dwelling environments. They have additional analogue inputs which can be used for the connection of external sensors, such as reference measuring devices, which can also serve as system validation during development.

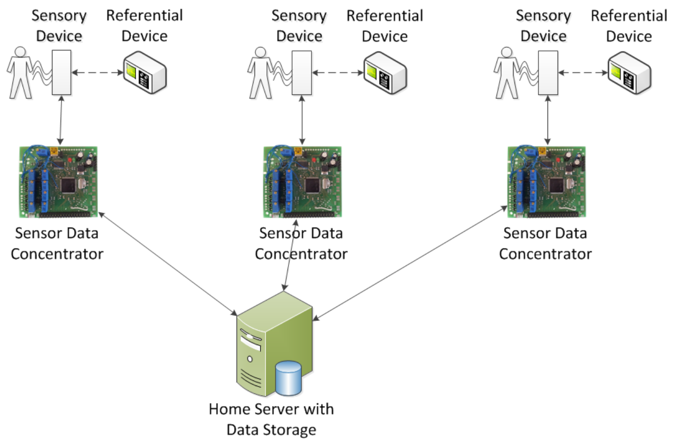

Fig. 1. Conceptual design of connections between sensory devices and home server for data storage and processing

Data from the SDs are collected by a corresponding SDC adapted to individual sensors and events before they are transmitted to HS. Events are triggered by SD when it detects a person is present and the acquired data are related to that person. Events mark the start or end of a measurement. They are triggered automatically when, for example, the person touches measuring electrodes in the environment or when the person appears in the visual or detection field of the sensor. HS assigns a unique numerical value to each event at the beginning of measurement. The same value is used as a part of the filenames when the visualization and acquisition programs that run on HS store the data into separate files. These are supplemented by the event time, SD name, and the unique event number.

All SDCs are identified and connected to the HS by its communication service. HS also offers an API to access and transfers data from temporary storage on SDCs. This functionality is used by the server program running on HS, which can asynchronously request a transfer of sensory data and FHPs computed by the software routines residing in SDCs.

Application interface on HS can communicate with multiple devices. It also incorporates support for the interpretation of FHP, locking of files with sensory measurements (protection against deleting), postponed events, query for SD properties and metadata about individual events.

FHPs appear as separate entities that have similar structure to the files containing measured data. Actually, they contain the results obtained by extracting FHPs from the data measured, and are always related to a specific event. Individual FHPs can be generated by the program for acquisition of sensory data on-line, but it can also result from a postponed processing of measurements that are protected against deletion in the meantime. To queue such postponed operation, special postponed events are used. They link the computed FHPs to one of previous events and measurements. API provides the following functionalities to the external client: (a) query for the list of SDs; (b) query for the status of individual SDs (presence, readiness, etc.); (c) query for the list of events generated by the SD; (d) acquisition of measurement data sent by SD with every new event; and (e) deletion of the events.

The client uses the API to: (a) obtain a list of identified SDs in the form of alphanumeric identifiers; (b) obtain integer identifiers of events related to the device–the range of search can be provided; (c) access the raw data for all new events on all devices; (d) delete an event from the internal buffer on HS when necessary.

III. System Architecture

A. *Sensor Data Concentrator*

To connect various sensors to HS, we developed microcontroller-based SDCs. The main task of SDC is to transmit the acquired data from different SDs in a uniform way. The SDCs also take care that all acquired data are synchronized.

The SDC's set-up depends on the connected SD. Although their architecture stems from the same concept, implementations may vary and include also wireless communication channels to SDs and a different number of analogue and digital inputs. An SDC is based on the PIC32MX534F064L microcontroller which is connected to HS via USB by using the FT245RL communication chip. The latter means a single-chip USB-to-parallel FIFO bidirectional data transfer interface [9]. Using the royalty-free D2XX drivers provided by the FTDI producer, the data transfer rates of up to 1 MByte/s can be achieved. Drivers are supported for all popular operating systems. SDCs support all the developed SDs, except video cameras that are, when used, connected directly to the HS.

B. *Application Program Interfacing on Home Server*

We developed an API for connecting multiple SDCs to a HS which supervises SDCs and process data. API is written in C to be compatible with different operating systems (Windows, Linux).

The sensory architecture which is assumed by the API is composed of one or more SDs with one or more sensors. SDs are connected to the HS through SDCs (except video cameras, as mentioned). HS has a permanent buffer, where sensory data is stored until an API request from the HC is sent. HC determines when the data, related to an event, are not needed anymore and can be removed.

Described system architecture is schematically shown in Fig. 2.

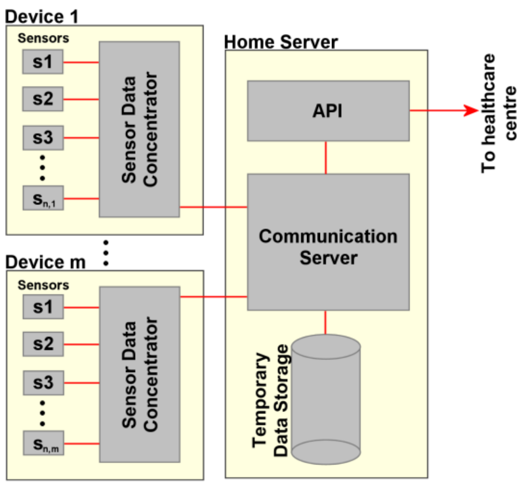

Fig. 2. System architecture of sensory devices: a home server connects to *m* devices, each *k*-th device comprises n_k sensors.

C. Intermediate data storage

All sensor and measurement data is written into the HS files. Data are organized in a tree (hierarchical) directory structure which enables this system to be implemented on top of any platform that supports similar structures. The root of the structure contains only one folder with only one file named *devices.dat*. The rows of this file describe names of devices that, at the same time, are also the names of subfolders in the root folder. Each subfolder contains all information about one device.

Every subfolder contains a cluster of files. The most important files are:

- `properties.dat`, which describes the device, its sensors and possible results (FHPs),
- `events.dat`, which is used to index all completed events or events waiting to be processed.

Beside these two files, there are also all files belonging to all known events. The file names follow a strict syntax:

```
    <event  identification  mark>-<ID|lock|
sensor{n}|result{n}>.dat
```

where the identification mark stands for increasing integers, `ID` for an event file, `lock` for a locked event file, and `sensor` and `result` denote files with sensory measurements and files with computed results (FHPs), respectively, both linked to a sensor number (written as *n* in the above syntax).

D. Gorenje Interface Event Handler

This API is shortly called GIEH and is implemented as a dynamically shared library. It offers functions as defined in Section II. Beside that it provides: (a) a list of involved sensors for a specific event; (b) a list of possible FHPs for a specific device; (c) a list of possible FHPs for a specific event; (d) a path to the file with specific device properties; and (e) a path to the file with metadata about an event.

The access to API is serialized to prevent concurrent modification of critical files. The synchronization is achieved on the operating system level of the HS [10][11].

To describe all possible sensors, results and connections between them, we introduced files with device properties. The format of files is depicted in Fig. 3.

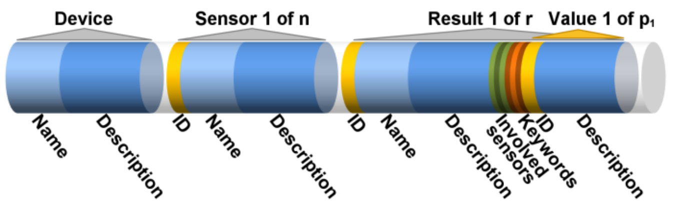

Fig. 3. File structure containing devices' properties: exemplified by a device with *n* sensors and *r* resulting FHPs, each of them having $p_{[1,r]}$ values.

E. Data-Stream and File Structures

All data obtained by SDCs from sensors and connected external devices, are processed and wrapped in the same way. Data from each sensor are collected in data packages. Size of the packages is 128 bytes and they comprise a head of 8 bytes, and data part of 120 bytes. The head provides the package generation time, the package serial number, and the sensor or SD identification mark.

The package data part contains raw data, exactly the same as collected from the sensory data flows. SDCs do not check the data flow contents, they only cut them to proper size, wrap them in packages, and send them to HS.

In general, each sensor has its own data structure which defines the format of measured data. These are processed by the algorithms for the FHPs computation. We have analyzed various possible FHP structures and created common formats that can be read and interpreted by the same software routine.

Internal formats of `properties.dat`, `events.dat`, identification event files, and files with FHPs are organized in rows and can be read in the same way as the conventional configuration files (key=value). Values are of the type that depends on the key.

The `properties.dat` file describes properties of devices. All sensors and FHPs related to a device are characterized. The description format is exemplified by a particular FHP:

```
    result_id=hr1
    result_name=Heart rate
    result_description=The  person's  heart  rate
in beats per minute
    result_sensors=hr_monitor
    result_keywords=hr|heart|rate|circulatory|ca
rdiovascular
    result_value_id=hr
    result_value_type=integer
    result_value_unit=BPM
```

The example above shows how heart rate is described. This FHP is a computation result that has an ID of `hr1`, and is called `Heart rate`, which describes a `person's heart rate in beats per minute`. The processed data are acquired by a sensor called `hr_monitor`. The FHP can be found elsewhere by using various keywords, including `hr`, `circulatory` and `cardiovascular`. This FHP comprises a single value with an ID of `hr`. It is an integer with units of `BPM`.

IV. Examples of Possible Scenarios and System Implementation

Tables in this section describe typical processes for creating new events. Table I refers to when an event is created by a single program that captures the event data and processes the data immediately after (synchronous mode). Table II shows an event that is handled by two independent programs or threads, where the first program/thread acquires data from sensors and the second one processes the data asynchronously. Table III describes a situation when the acquisition program recognizes events of long measurements and starts processing the data in parallel, in real time.

Handling of sensory data in synchronous or asynchronous mode is depicted by a flow chart in Fig. 4.

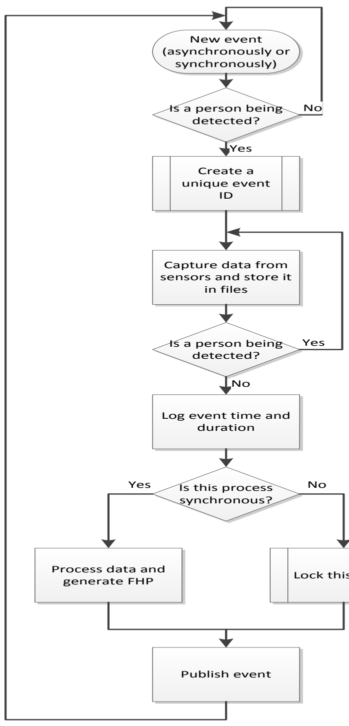
New event (asynchronously or synchronously)
Is a person being detected?
No
Yes
Create a unique event ID
Capture data from sensors and store it in files
Is a person being detected?
Yes
No
Log event time and duration
Is this process synchronous?
Yes
No
Process data and generate FHP
Lock this event
Publish event

Fig. 4. Flow chart for the procedure of detecting and creating the events and processing the capruted data either synchronously or asynchronously.

TABLE I. CREATION OF A NEW EVENT BY USING SYNCHRONOUS ACCESS

Step	Step description
1	A unique event ID (called ID1) is created by using GIEH.
2	Data is captured from sensors and stored in files. Files are created on demand.
3	An ID file is created for event ID1 which contains event time and duration logs.
4	Data is processed and FHPs are created as dictated by the device's properties.
5	All files are finalized. Event ID1 is published.

TABLE II. CREATION OF A NEW EVENT BY USING ASYNCHRONOUS ACCESS

Step	Step description
1	A unique event ID (called ID1) is created by using GIEH.
2	Data is captured from sensors and stored in files. Files are created on demand.
3	An ID file is created for event ID1 which contains event time and duration logs.
4	Event ID1 is locked by creating a lock file. Event ID1 now resists deletion.
5	All files are finalized. Event ID1 is published. This event is now known as ID2. There are no FHPs for it yet and it is still locked. This process has now finished its work.
6	Using an asynchronous process, a new locked event ID2 is detected and its data is processed. FHPs are created as dictated by the device's properties. Files are named with temporary names while writing results. They are renamed to a proper name at the end of writing.
7	A unique event ID (called ID3) is created by using GIEH.
8	An ID file is created for event ID3 which contains event time and information about the updated event ID2.
9	Event ID2 is unlocked. It can now be deleted.
10	All files are finalized. Event ID3 is published.

TABLE III. CREATION OF A NEW, LONGER EVENT IMPLYING A CALCULATION OF FHPS IN REAL TIME

Step	Step description
1	A unique event ID (called ID1) is created by using GIEH.
2	An ID file is created for event ID1 which contains information such as event start time.
3	Event ID1 is locked by creating a lock file. Event ID1 is now marked as unfinished. It also resists deletion.
4	All files are finalized. Event ID1 is published. This event is now known as ID2. There are no FHPs for it yet and it is still locked.
5	Data is processed and FHPs are created for event ID2 as dictated by the device's properties. Care is taken that files can be opened and read by multiple processes at the same time.
6	All files are finalized. Event ID2 is unlocked. Event is now marked finished. It can also be deleted.

A. System Implementation

The architecture revealed in previous sections was used for an implementation of home-server connected unobtrusive sensors. The sensors are coupled in sensory devices and these are connected with data concentrators (except for video cameras). SDCs prepare data packages to be sent to a HS.

In our implementation, sensors were built into the different household appliances and home devices. Fig. 5 shows schematically which sensors and which devices were involved.

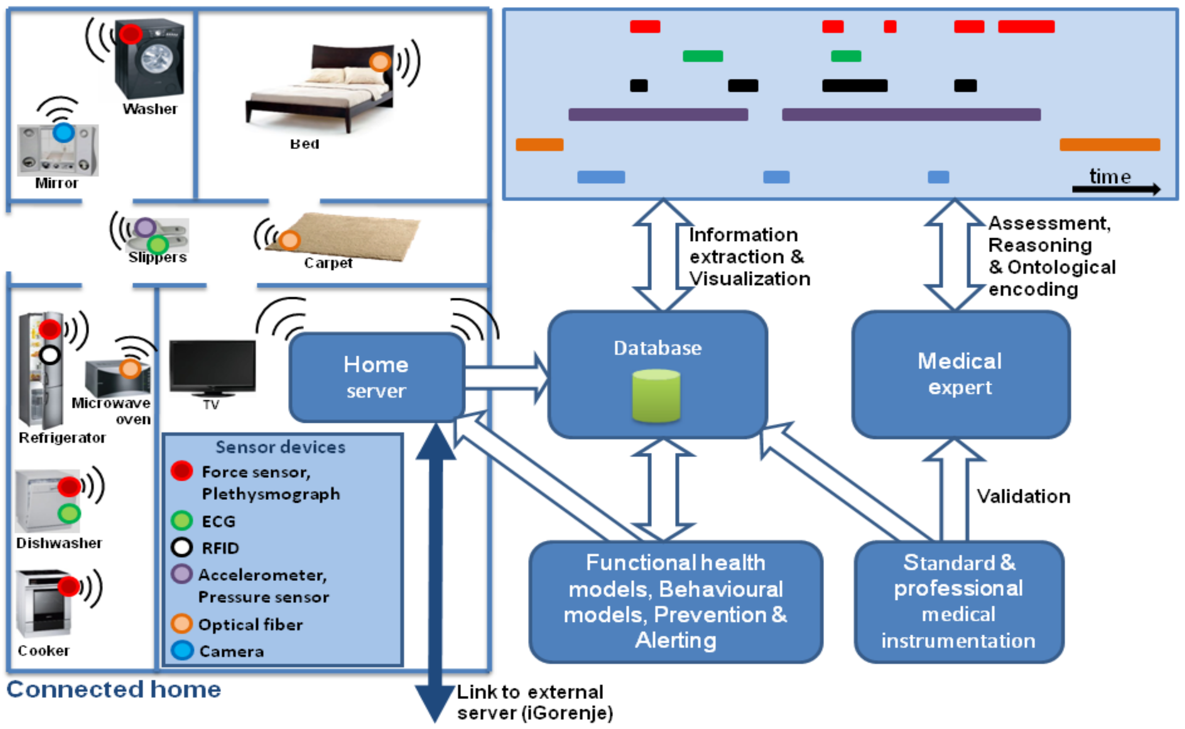
Washer
Mirror
Bed
Slippers
Carpet
Refrigerator
Microwave oven
TV
Dishwasher
Cooker
Home server
Sensor devices
Force sensor, Plethysmograph
ECG
RFID
Accelerometer, Pressure sensor
Optical fiber
Camera
Connected home
Link to external server (iGorenje)
time
Information extraction & Visualization
Assessment, Reasoning & Ontological encoding
Database
Medical expert
Validation
Functional health models, Behavioural models, Prevention & Alerting
Standard & professional medical instrumentation

Fig. 5. A scematic set-up of sensory devices mounted in dwelling environment to capture unobtrusively human vital signs and compute accordingly the functional-health parameters.

For the time being the following sensory devices have been constructed and tested:

- sensors in the refrigerator lever for detecting and computing heartbeat, blood oxygenation and pressure, body temperature and physical condition
- optical sensors in the bed for detecting and computing heart rhythm, arrhythmias, respiration curve, respiration abnormalities, and movement
- dry electrodes with an ECG measuring device built in the oven handle to analyze ECG features
- sensors in the slippers for detecting and computing heartbeat, blood oxygenation, gait features, and stabilogram
- video camera behind the mirror to detect and analyze psychophysical condition, i.e. emotions, and skin and the eye white color
- floor carpet with optical sensor for detecting the residents' location, movement, and possible incident of falling.

Prototypes of all the developed sensors, SDs, and SDCs were installed and connected into a test system at the University Rehabilitation Institute Soča of Ljubljana. Several experiments have been conducted with healthy and also diseased people. Some of the experiments involved only one single combination of sensors, SD and SDC, such as those in the refrigerator lever (Fig. 6). Another type of experiments was designed with a sequence of daily activities that involved handling with several sensory devices, such as opening the refrigerator and oven, passing the mirror, and lying in bed. All the FHPs acquired and computed were saved into the database along with the corresponding referential measurements. These incorporated standard medical devices that measured the same parameters as tested unobtrusive sensors did at the same time, so that comparisons and validation were feasible afterwards.

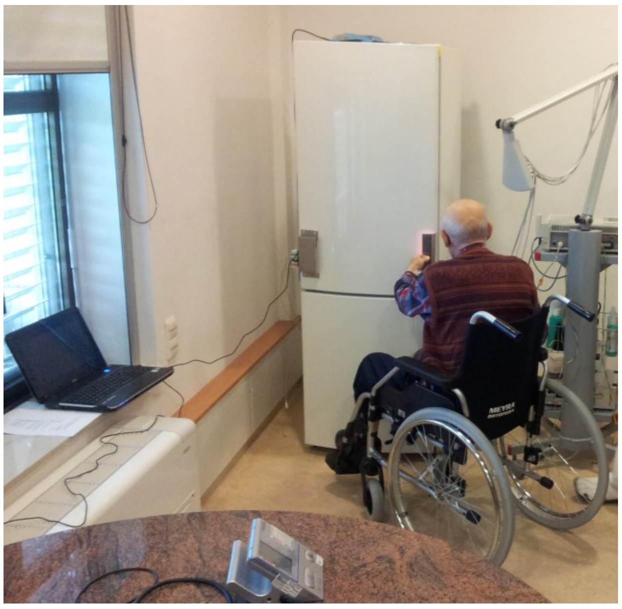

Fig. 6. Exeprimental environment for testing the sensors in the refrigirator lever installed at the University Rehabilitation Institute Soča of Ljubljana

V. Conclusion

Current functionality of API is ready to be extended by additional sensors and/or sensory devices. Any additional sensor or device needs additional software routines that must be able to handle any particular requirements when acquiring and storing data according to the basic rules implemented in the API.

The system of delayed events introduces a simple way of computing new FHPs from previously stored measurements if an error is detected in data-processing algorithms or the algorithms have been modified.

The described system supports ubiquitous computing in health care delivered at home. It has been designed and implemented as a network of distributed, unobtrusive sensors that can be built in different home devices, but also in many other environments, such as residential homes for elderly, clinics, or public places.

References

[1] M. Chan, D. Estève, C. Escriba, and E. Campo, “A review of smart homes—Present state and future challenges,” *Computer Methods and Programs in Biomedicine*, vol. 91, no. 1, pp. 55–81, Jul. 2008

[2] D. Ding, R. A. Cooper, P. F. Pasquina, and L. Fici-Pasquina, “Sensor technology for smart homes,” *Maturitas*, vol. 69, no. 2, pp. 131–136, Jun. 2011

[3] J. Yao, and S. Warren, “Applying the ISP/IEEE 11073 standards to wearable home health monitoring systems,” *Journal of Clinical Monitoring and Computing*, vol. 19, pp. 427-436, 2005

[4] D. Cook, A. Crandall, B. Thomas, and N. Krishnan, “CASAS: A smart home in a box.,” *IEEE Computer*, to appear

[5] V. Stanford, “Using pervasive computing to deliver elder care,” *Pervasive computing*, pp. 10-13, 2002

[6] M. J. Rantz, R. T. Porter, D. Cheshier, D. Otto, C. H. Servey, R. A. Johnson, M. Aud, M. Skubic, H. Tyrer, Z. He, G. Demiris, G. L. Alexander, G. Taylor, “TigerPlace, A state-academic-private project to revolutionize traditional long-term care,” *Journal of Housing for the Elderly*, vol. 22, no. 1/2, pp. 66-85, 2008

[7] House_n Research Group, “House_n,” MIT Department of Architecture, http://architecture.mit.edu/house_n/

[8] L. C. De Silva, “Multi-sensor based human activity detection for smart homes,” *Proceedings of the 3rd International Universal Communication Symposium IUCS ’09*, Tokyo, Japan, pp. 223-229, Dec. 2009

[9] FT245RL USB FIFO(USB-Parallel)I.C. ©Future Technology Devices Intl. Ltd. 2010: 1-37

[10] Microsoft. (2012, Dec 18). Using Mutex Objects (Windows) [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms686927%28v=vs.85%29.aspx

[11] V. Shukla. (2007, May 24). Semaphores in Linux [Online]. Available: http://www.linuxdevcenter.com/pub/a/linux/2007/05/24/semaphores-in-linux.html