PESTO: Data Integration for Visualization and Device Control in the SmartCare Project

Nicholas "Brent" Burns[1], Peter Sassaman[1], Kathryn Daniel[2], Manfred Huber[1], and Gergely Záruba[1]
The University of Texas at Arlington; [1]Department of Computer Science and Engineering, [2]College of Nursing
nburns@mavs.uta.edu, sassaman@mavs.uta.edu, kdaniel@uta.edu, huber@uta.edu, zaruba@uta.edu

***Abstract*—The SmartCare project is to design, develop, and evaluate an intelligent sensor-driven living environment for the elderly. The core objectives are to provide emergency detection, improve quality of life, extend independence for the elderly, and detect patterns of behavior that could suggest early signs of a physical or cognitive issue, all in an unobtrusiveness manner. This paper specifically focuses on the development of the infrastructure integration component: PESTO and two of its sub components: a 3D visualization of the resident's smart apartment (VISMA) and providing everyday task assistance through the Z-Wave home automation technology (ZAPS).**

Keywords—eldercare, home healthcare, pervasive computing, ubiquitous computing, smart technology, sensor networks, z-wave, home automation, virtual environment, 3D modeling

I. Introduction

Older adults are the fastest growing population segment in developed countries. We are living longer due to many improvements in our health, often the result of modern pharmacologic interventions which have significantly reduced early death due to infections. As we reach older age, however, we tend to accumulate chronic conditions, such as high blood pressure, diabetes, or arthritis. In the U.S., for example, it is expected that the number of people above 65 will double over the next 25 years, reaching 72 million by 2020. In this population, over 80% suffer from at least one chronic disease and at least 50% suffer from two or more chronic conditions that require careful, regular monitoring and treatment.

As the prevalence of conditions that require careful monitoring increases, we overwhelmingly desire to remain as independent as possible for as long as possible. To address this challenge, our research group is establishing a federally funded live-in laboratory where technology designed for independent living and elder care can be tested directly with the target population. This laboratory is part of our SmartCare project aimed at the use and evaluation of existing and the develop new technology to design holistic, integrated systems for aging in place, including environmental and health sensors, automated data analyses, decision support, and robust communication. In order to prevent disease progression and provide early identification of deterioration of our chronic conditions, SmartCare is a testbed for the development of integrated, unobtrusive technology applications designed to preserve the dignity and independence of older adults by monitoring their complex health needs and well-being. We are working with older adults in a retirement living environment [1] to develop embedded technology applications that can rapidly identify unsafe situations related to their health and safety and automatically alert caregivers so that treatment can be initiated and adverse outcomes such as hospitalization can be prevented or mitigated. Devices will be non-intrusive, easy to use, and easily deployable to existing environments, secure, reliable, and lower cost than current technologies. The SmartCare apartment is, along with other sensor systems, equipped with a a network of Z-Wave based sensors and interfaces that monitor outlets, lights, and other environmental conditions and a pressure sensing floor over the entire 850sqft apartment with a 1 sensor/sqft. Resolution (the description of this floor is outside the scope of this paper).

SmartCare brings together the expertise of researchers from Nursing and Engineering to integrate advanced sensors, wireless communications technologies, software, intelligent systems, and modern care and treatment methods to optimize preventive health monitoring, treatment, and continuous care for those with disabilities due to aging. SmartCare will facilitate the collaboration with industry to ensure the applicability of the technologies in practical settings and to minimize the time between their development and deployment.

This paper will discuss the main infrastructure framework of SmartCare: PESTO (Pervasive Environment STructure for the Old). In addition two modules will be discussed in detail: 1) VISMA (Visualization for the SmartCare Apartment) the 3D visualization component that depicts data received from the SmartCare apartment as if the information was coming from cameras; and 2) ZAPS (Z-Wave APplication for SmartCare): the smart device control component. The remainder of this paper is organized as follows: background information about the tools and methods used in this part of the project are detailed in Section II. Section III discusses other ongoing work related to SmartCare and the use of visualization and home automation for such "smart home" projects. The specific tools and resources used for the work presented in this paper are specified in Section IV. Section V delves into the architecture of PESTO and some of its subcomponents. Conclusions are drawn and future work is presented in Section VI.

II. Background

Visualizations are great for communicating large amounts of data in a manner in which the brain can process them more easily. For example charts can show correlations or the lack thereof between two sets of paired data. In the case of smart environments, visualizations can be pictorial representations of what is actually occurring in the physical space. Here data is often collected to be analyzed out of context and a good visualization tool can be instrumental in determining what actually happened and caused the collected data (and thus to preform reverse engineered from the collected data).

For the SmartCare apartment we envisioned a tool that could show what exactly is happening (or has happened) in the apartment from real-time streaming data coming from the apartment as well as from previously recorded data streams. For example, if people are moving around (data indicating such movement comes from the embedded floor pressure sensors) we would like to show a person moving around in a 3D model of the apartment. In addition, the visualization tool should be able to show the current status of all smart appliances (small and large) and electrical consumers while also enabling a technician to alter the status of these smart devices. The SmartCare apartment does not contain any video recording devices (i.e., no cameras) but we would like to reassemble the data into a visualization that looks as if it was coming from cameras as this provides an intuitive and efficient means for a human to analyze the event occurring in the environment. In order to accomplish this we enlisted the help of a Game Engine called *Unity 3D* along with several graphics manipulation tools to build a representation of the actual apartment. This way 3D models that display what is occurring can be made visible to authorized individuals (e.g.., a nurse with permission to see such data) monitoring the state of the apartment without showing actual video, as to not invade the resident's privacy.

A. Z-Wave Home Automation Technology

Z-Wave [2] is a wireless technology geared towards home automation devices and communication that has grown tremendously over the past 10 years. It is quickly becoming the new standard amongst professionals and hobbyists due to the increasing availability of off-the-shelf Z-Wave enabled devices and customizable software for developers. The decision to integrate Z-Wave devices and software into the SmartCare apartment was two-fold: to provide instant reactionary assistance and preventative care through early detection.

1) Assistive: The SmartCare apartment should be state-of-the-art in terms of ease of living for the resident by making their entire environment assistive through the use of home automation. Allowing common functionality (turning off lights, opening doors or blinds, etc.) to be easily controlled and automated can greatly increase the quality of living for the elderly. Something as simple as automatically turning on lights and illuminating a path for a resident waking up in the middle of the night can help prevent a devastating fall [3]. Similarly, automation could also be used to automatically cut power to an appliance or water from a faucet accidently left on to prevent fire or water damage.

2) Pattern Recognition: Since the Z-Wave devices not only allow wireless control, but can report usage and power consumption, a model of the resident's daily patterns and routines can be constructed. This model can then be used and integrated for anomaly detection and routine deviation within SmartCare's main health monitoring infrastructure. If an anomaly of urgent severity is discovered, instant notification would be sent to their care-giver to provide assistance. In addition, using the resident's daily routine model over an extended period of time could be useful in relation to cognitive health and memory function by shedding light on smaller non-urgent deviations.

III. VISMA Related Work

This section highlights some other researchers' projects that share a similarity to either SmartCare, visualization of a health monitoring environment, smart home automation, or some form of a combination. This is by no means a complete review of visualization approaches to smart environments as most projects have their homegrown visualization.

The Smart Condo [4] is a project with similar goals to SmartCare; it has a variety of sensors deployed around a condo including infrared motion sensors to approximate the activity of an occupant in a physical space. Using this data, visualization in a game called Second Life is able to show what is taking place in a virtual representation of the condo. In contrast to this global, net-centric approach, VISMA uses a local data representation with local models and is thus arguably more compliant to inhabitant privacy. VISMA also enables the tracking of multiple occupants within a space instead of assuming the presence of only one occupant. The Smart Condo was not the first to use Second Life for visualization. An earlier approach at the Razorback Hospital has modeled hospital inhabitants and equipment [5]. The CenceMe project has also evolved to show sensor data using Second Life [6].

More recently an elder care endeavor at The University of Milan has used a 2D engine called Siafu to build a 2D visualization for their smart kitchen [7] where activity recognition of the elderly can take place.

IV. Tools Used

This section specifies the particular tools, software, and hardware used to build the virtual environment and Z-Wave home automation aspects of SmartCare.

A. Visualization

1) Unity 3D: Unity 3D [8] was selected as the engine of choice for visualization as it is one of the most widely used game engines today. Some of the major reasons for its acceptance by game developers are its intuitive interface and its use of C# and Java Script to create very powerful scripts for controlling the flow of simulations. In addition Unity can generate executables which may run on many platforms such as: iOS, Android, Windows, Mac, Linux, and in browsers just to name a few. In addition to the aforementioned, the vast majority of the Unity tool is available free of use as long as the developer does not make profit in excess of USD100k.

2) Blender: Blender [9] is a 3D modeling tool used by many artists in the 3D mdeling industry. Blender allows for both high and low level modelling through both object operations, as well as manipulation of individual vertices, lines, and faces composing a model. This tool also allows for the application of materials and the mapping of textures to a file prior to exporting them to be used in Unity 3D.

3) Gimp: Gimp [10] was the primary tool used in generating and manipulating textures for use in Blender. Gimp has a wide variety of existing textures and tools for modifying them to be imported in Blender and Unity 3D.

4) MonoDevelop: MonoDevelop [11] is the environment used to edit C# and Java Script scripts to be applied to objects

In Unity 3D. Other editors can be used in its place however this tool comes free with the installation of Unity 3D.

5) MakeHuman: MakeHuman [12] is an "open source tool for making humanoid 3D characters". This makes it an ideal program for generating avatars to put into the SmartCare apartment visualization. The program allows for easy adjustment of the appearance of an avatar. A skeleton can also be applied which makes it easy to animate in Unity 3D.

B. Z-Wave Home Automation

Z-Wave technology is a wireless hardware and software specification that can be integrated with various household devices in order to allow them to communicate, report usage information, and be remotely controlled for the purpose of home automation and lowering energy consumption.

1) Types of Devices: Z-Wave Devices may be battery or mains powered. All devices can initiate and relay messages by sending and receiving indiscriminately amongst themselves due to Z-Wave's *Mesh Network Architecture*. This adaptive routing scheme provides a more reliable transmission of messages throughout the network. Z-Wave devices generally fall into two categories: Replacment and Addition.

a) Replacement: A device is manufactured and pre-installed specifically with the Z-Wave hardware and software protocol integrated so it can replace a regular non-Z-Wave device. For instance, a Z-Wave enabled LED light fixture can simply just replace your existing light bulb and automatically report its power usage through the rest of the Z-Wave network while also allowing wireless control (on, off, dimming).

b) Addition: A device designed to work in conjunction with existing household items without Z-Wave capabilities that when used together can report useful information wirelessly and add remote control. For instance, instead of the Z-Wave LED bulb, one could put a Z-Wave based controller (e.g., a dimmer switch) that connects the light to the power outlet. These types of devices allow you to keep existing or old technology and retrofit them with a Z-Wave device to add wireless on/off/dim and data gathering capabilities.

2) Device Classification in the Network: A single Z-Wave network consists of only one *Primary Controller* in addition to multiple *Secondary Controllers* (if desired), and up to 232 *Slave* devices (including the number of controllers).

a) The *Primary Controller* is the head of the network and responsible for assigning unique IDs (inside the network) to all other nodes, delegating tasks, and storing routing information. Usually the *Primary Controller* is a single static controller connected to a computer (USB stick, hub, or Rspberry Pi shield) that is the portal for messages sent to and from commercial or custom Z-Wave software. Custom controllers allow much more customization, low-level data knowledge, and specific control over the network and are for the more experienced developers and hobbyists. These controllers can be used with existing or custom-built home automation software (discussed in detail in a later section).

b) Secondary Controller(s) are devices such as a handheld remote controller, key fob, or touch pad that may be used as an additional source of resident control for the *Slave Devices*. The remote controllers allow simple ease-of-use for the more average user who simply wants the handheld power of dimming lights or turning outlets and appliances on and off. For instance, a USB static controller may act as the primary (running the Z-Wave related integration software) but the resident may use *Secondary Controllers* to directly turn *Slave* devices on and off to bypass the Primary Controller's actions.

c) Slaves are The actual Z-Wave devices integrated into the living environment that report information and can be controlled by the above controllers.

3) Z-wave Specification and Protocol Overview: The routing scheme and topology control protocol amongst all the meshed nodes (controllers and slaves) is abstracted from the user for plug-and-play simplicity. The user only has to worry about injecting commands and requests into the network and waiting for replies. The user does not have to configure which nodes will communicate directly; it is a self-healing and self-learning network from the initial startup. However, if a certain node is not responding often the user may tell the controller of the specific issue and the network will attempt to resolve the matter and refine the routing scheme for certain nodes.

4) Z-Wave Devices Used: There are various Z-Wave devices currently deployed within the SmartCare apartment. The two main types of devices (that account for 80% of the total number of devices/nodes used) are the Aeon Labs Micro Smart Energy Switch [13] and the Micro Smart Dimmer Switch [14] used for power outlet and light control, respectively. Some other devices include the Aeon Labs 4-in-1 Multi-Sensor, Repeater, Curtain Control Module, Flood Detector, Thermostat, Recessed Door Sensor, ZVent AC duct, and Smoke/CO Alarm. The detailed specifications and usage of these devices are described in Section V-B.

5) Z-Wave Software Used: There are many well-made existing software packages used for controlling, monitoring, and configuring Z-Wave networks and devices. They can range greatly in cost (some are free), interface (browser or traditional), and whether they are open source. The most popular include InControl [15], OpenRemote [16], OpenHAB [17], HomeGenie [18], OpenZwave [19], HomeSeer [20], and Zensys' Z-Wave PC Controller [21].

We used OpenZwave as an aid to develop our own Z-Wave integration tool. OpenZwave is an effort to offer development support for those who do not wish to purchase the Sigma Designs software development kits. It is a large online collaboration with many contributors frequently adding and refining the libraries to implement Z-Wave from a software level. It is very powerful and capable, however it can have a slow response for basic commands at times and can freeze/deadlock when not used or understood perfectly. ZAPS was designed based on lessons learned from using OpenZwave; the underlying foundation was created from the digiwave [22] bare bones tutorial.

Using this approach and structure combined with Z-Wave technical documents and the Pepper Z-Wave device library

[23] that provides specific command details, our framework was developed as a light-weight interface to ensure faster response time, minimal code complexity, and custom error handling. We also used HomeGenie's and Zensys' software periodically for one-time configuration of certain Z-Wave devices in lieu of using the custom built software to save time.

V. System Architecture and Design

This section provides a thorough description of the underlying architectural design of the SmartCare apartment's virtual environment, Z-Wave automation configuration, and SmartCare integration (PESTO) in general.

A. VISMA

We have built a framework inside Unity 3D that enables the creation of 3D virtual smart home environments. All the programmer has to do is enlist the services of a modeler for the various objects (e.g., doors, curtains, stoves, etc.) and insert them into our framework within Unity 3D, using Unity 3D's editor for placing the models:

1) Hierarchy: Unity 3D's hierarchy was used to structure all of the objects contained in the visualization.

2) Scene: Another useful functionality are the Scene settings which allow for easy viewing and manipulation of the virtual environment.

3) Game: This tab allows a preview as to what the visualization will look like prior to running the program.

4) Animator: The animator tab allows for the creation of state machines built from individual animations.

5) Animation: The animation window allows for editing animations pertaining to an object. It is used to control aspects of an object over time.

6) Design: The virtual apartment as seen in VISMA (see Figure 1) is composed of six camera windows (esily reconfigurable in our framework) providing a top down view of the virtual representation of an actual physical SmartCare apartment. The first five views are perspective views from within the apartment while the last view is an orthographic top down view of the apartment. The visualization tool provides a network service, running a TCP server that allows the reception of external data to be displayed and the transmission of user requested changes to the environment.

Figure 1 - Visualization using six virtual cameras.

For example, in Figure 1 there are green discs that provide a visualization of pressures being applied to actual pressure sensors in the physical SmartCare apartment. When information about new pressure readings is received, the appropriate function in our Unity framework is called to move the avatar to the new position. In another example, information could indicate that a group of lights in the apartment has changed brightness level; the appropriate objects in the framework will receive this information, change their properties and trigger Unity's engine for an update of the GUI. Figure 2 shows all lights turned off in VISMA.

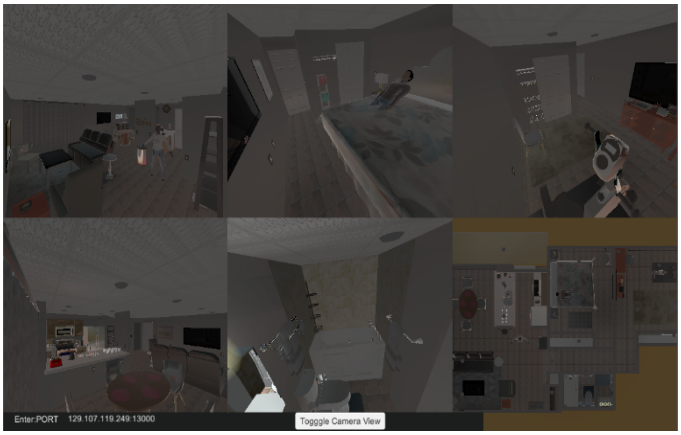

Figure 2 - Lights turned off in the Apartment.

The design of VISMA is such that anyone with sufficient 3D modelling, texturing, and basic programming skills could easily expand upon or modify it for their own purposes (e.g., existing walls in the environment could be removed and replaced with other walls to be appropriate for a new apartment, additional Z-Wave controled device abstraction could be added for controling or receiving status information, etc.) We are hoping that this framework could find wide use in future smart living environments to help with the elderly.

B. ZAPS (Z-Wave APplication for SmartCare)

1) Overview: in the early stages, ZAPS was built as a stand-alone C# application. Once effective control was established for the Z-Wave network and its devices, functionality was added to provide external data flow to other SmartCare software, making ZAPS into a network service.

*2) Devices Used and Their Purpose: (*we only focus on a few of the devices installed.)

a) The *Primary Controller* used was the Aeon Labs Z-Stick Series 2 (Figure 3). It has a USB adapter that plugs into the computer running the Z-Wave software and acts as the wireless communication portal between the software and the network's nodes in the apartment.

Figure 3 - Z-Stick Series 2

Figure 4 - MSES and MSDS

Figure 5 - Multi-Sensor

Figure 6 - Recessed Door Sensor

b) MSES and MSDS (Power and Light Control): The most used Z-Wave slave nodes are the Aeon Labs Micro Smart Energy Switch (MSES) (Figure 4) and the Micro Smart Dimmer Switch (MSDS). They are both packaged identically and function as general-purpose Z-Wave enabled microcontrollers to use with exisiting household devices. They are embedded in the wall boxes behind switches and power outlets and thus are invisible to the inhabitants. Similar devices also control the power supply to the single-board computers (that control the pressure floor circuits and sensors). The MSES can also report energy consumption.

c) Multi-Sensor (Temperature,Motion, Humidity, Light): A useful device with multiple features offered by Aeon Labs is the 4-in-1 Multi-Sensor (Figure 5). It can measure and report temperature, motion detected via IR, humidity, and amount of light. These are placed strategically throughout the apartment in every room.

d) Recessed Door Sensor (Door State): This device (Figure 6) can be polled for the open/close state and it automatically sends a notification if the door state changes.

Since we want to know about any activities that take place within the apartment, the Z-Wave devices need to be able to respond with their data when asked by the software. All devices are capable of responding to a request made by the Primary Controller asking for various metrics such as current state (on, off, level), battery level (0-100%), device type (class), voltage (V), current (A), power consumption (watts), and energy usage (kWhs).

Z-Wave devices may also be configured to automatically report to the Primary Controller (software) when a particular action occurs such as, water alarm triggered, current drawn over a certain threshold, etc. They may also be configured to periodically report information every 30 seconds, 10 minutes, 24 hours, etc. This method of automatic slave node reporting reduces strain on the Z-Wave network.

3) Node Data Structure Variables: ZAPS provides a class object for each node (slave device) within the network. The variables' values used for each node are sent and stored to the SmartCare system's main database to provide an archive or log of all events that occurred with Z-Wave devices.

4) JSON Socket Communication: To allow the main SmartCare software access to Z-Wave control, data is exchanged between the Z-Wave software and the Central Software's database via JSON. JSON (JavaScript Object Notation) is a data structure that can easily transmit object data in a human-readable text format. It simplifies multiple different types of numeric and string/character data types into Numbers and Strings, respectively, to stream-line the communication process. To lessen the amount of data stored in the database and free-up network traffic, not all JSON object members/variables are sent every time an event occurs or when the client requests data information.

For testing purpuses ZAPS has a local 2D user interface (Figure 7) that can be turned off when used in conjunction with PESTO. This interface also enables debugging of ZAPS and communication issues within PESTO and VISMA.

Figure 7 – 2D developmental ZAPS Interface

C. PESTO: The Heart of SmartCare

In this section we discuss PESTO, the generalized structure of the SmartCare software and hardware organization that organizes and establishes the iteraction amongst the core software, visualization, and Z-Wave automation components, the apartment floor, and any additional existing instances of experimental smart and assistive technologies tested in the apartment (Figure 8).

The heart of the system is located on a central server. Any data collected from the SmartCare apartment is received and stored on this server. At the core we can find a MongoDB NoSQL database providing data storage service to PESTO. Other SmartCare components such as ZAPS and VISMA could be running (and are indeed running) on other computers as long as they provide a network service (e.g., a TCP server). Then additional PESTO components running on the central server can connect to these services, receiving information from them to be placed into the database, or receiving the latest information from the database to relay to the services. For example if a user of VISMA clicks on a light in the visualization interface indicating that the light should turn on, a PESTO component attached to VISMA is going to receive information from VISMA that is going to be placed into the database. In addition the PESTO component attached to ZAPS will see that a new entry in the database suggests that it will need to send a command to ZAPS. ZAPS in turn turns the light on, notifies the PESTO ZAPS interface about the change which in turn updates the database with the new information; this in turn is picked up by the PESTO-VISMA component, the information is relayed to VISMA which will update the visualization component to show the light turned off.

In another example, information comes from the SmartFloor. Individual BeagleBones in charge of a floor area provide a network service. A PESTO-SmartFloor component running on the central server connects to all BeagleBones, receives raw SmartFloor data from them (or issues commands to the floor), and places such information in JSON format into the PESTO database. The PESTO-VISMA component then takes this information and relays it to VISMA for the

visualization of the inhabitants.

With this "communication through a database" approach all previous information is available to analysis and datamining tools. In addition, a past stream of database entries can be easily relayed again to a (possibly separate instance of) VISMA to display what has happened during a past epoch.

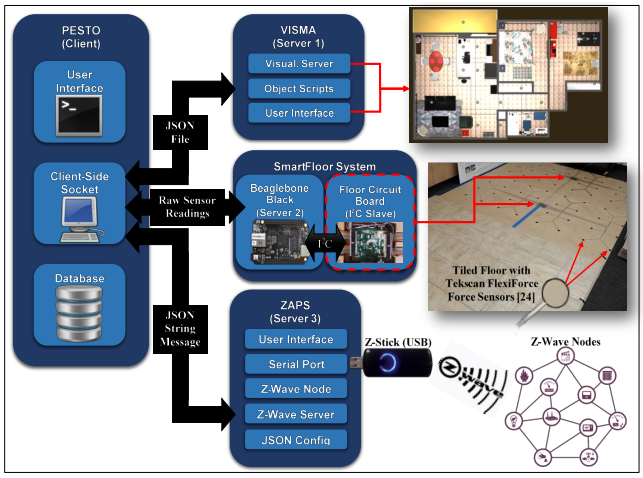

Figure 8 - High-Level SmartCare System Organization

VI. Conclusions, Lessons Learned, and Future Work

While the SmartCare project has received a significant amount of work and attention, the real work is just beginning. The integration software discussed in this paper is essential for the SmartCare vision to be realized. The combination of the technologies detailed here could prove to be extremely useful for the future of developing state-of-the-art living environments for the elderly. One of the goals of SmartCare is to see what technologies are actually useful or provide enough added information to be of benefit. The PESTO framework has been designed to be able to obtain and store information from a vast amount of sensors and to be able to control many actuators. Any additional services can be implemented as modules between the central database and network services provided by the new service or technology.

Some of the Z-Wave devices that we want to integrate are still being tested and configured in many cases due to lack of detailed documentation regarding which commands and configuration settings they allow. Communication between different pieces of software needs to have one rigid format for data exchange to ensure no data is lost. In addition, although Z-Wave is seen by many as an inclusive standard, it is not tailored to people who want to develop their own slave devices. With the USB and Raspberry Pi masters, custom built controllers have become possible, yet there is no simple and small communications module with which a custom Z-Wave device could be built (e.g., for our curtain control we are using Z-Wave control modules that are large, and can only issue commands but do not allow the reading of sensors that would relate the state of the curtains back to our Z-Wave controller).

One of the biggest tasks in our future work is the development of modules to be integrated into PESTO that can anlyze raw data in order to be able to retrieve high level information from such data. For example, currently the avatars of VISMA are moved based on simple pressure signatures. Instead we are developing a module that would read the pressure data, use it as an input to gait models and show an appropriate animation of a human gait in VISMA (i.e., a limping person would show up limping in VISMA).

References

[1] Lakewood Village Senior Living Community. Christian Care Senior Living Communities, Fort Worth, TX. christiancarecenters.org/communities/lakewood-village-ft-worth-tx/

[2] Z-Wave. Sigma Designs Inc., Milpitas, CA. http://www.z-wave.com/

[3] K. Sada et al., "Effects of clear visual input and change in standing sequence on standing sway related to falls during night toilet use," *Int. Journal Of Older People Nursing,* vol. 5 (1), pp. 34-40, Mar. 2010.

[4] N. Boers et al., "The smart condo: Visualizing independent living environments in a virtual world," *Proc. 3rd Int. Conf. Pervasive Comput. Technol. Healthcare (PervasiveHealth)*, Apr. 2009.

[5] C. W. Thompson, F. Hagstrom, "Modeling Healthcare Logistics in a Virtual World," *IEEE Internet Computing,* vol. 12, no. 5, Sept. 2008.

[6] M. Musolesi, E. Miluzzo, N. D. Lane, S. B. Eisenman, T. Choudhury, and A. T. Campbell, "The Second Life of a sensor: Integrating real-world experience in virtual worlds using mobile phones," in *Proceedings of the 5th Workshop on Embedded Networked Sensors* (EmNets'08), Charlottesville, Virginia, Jun. 2-3 2008

[7] Gabriele Civitarese, Zaffar Haider Janjua, Daniele Riboni, Claudio Bettini, "From lab to life: Fine-grained behavior monitoring in the elderly's home". PerCom Workshops 2015

[8] Unity3d.com, 'Unity - Game Engine'. N.p., 2015. Web. 26 Nov. 2015.

[9] Blender.org,. 'Blender'. N.p., 2015. Web. 26 Nov. 2015

[10] Gimp.org,. 'GIMP - GNU Image Manipulation Program'. N.p., 2015. Web. 26 Nov. 2015.

[11] MonoDevelop.com,. 'Cross Platform IDE for C#, F# and More'. N.p., 2015. Web. 26 Nov. 2015.

[12] MakeHuman.org,. 'MakeHuman | Open Source Tool for Making 3d Characters'. N.P., 2015. Web. 26 Nov. 2015

[13] Micro Smart Energy Switch. *Aeotec by Aeon Labs*. Available: http://aeotec.com/z-wave-in-wall-switches/848-micro-ses-2e-manual-instructions.html

[14] Micro Smart Dimmer Switch. *Aeotec by Aeon Labs*. Available: http://aeotec.com/z-wave-in-wall-switches/877-micro-sei-2e-manual-instructions.html

[15] InControl. *InControl Home Automation (InControl Console)* [Online]. Available: http://incontrolzwave.com

[16] OpenRemote. *Open Source Automation Platform* [Online]. Available: http://www.openremote.org

[17] OpenHAB. *Empowering the Smart Home* [Online]. Available: http://www.openhab.org

[18] HomeGenie. *Open Source Home Automation Server in a "Internet Of Things" world* [Online]. Available: http://sourceforge.net/p/homegenie

[19] OpenZWave. *A Open and Free Library that talks to Z-Wave devices via a Dongle and the Z-Wave Serial API* [Online]. Available: http://www.openzwave.com/

[20] HomeSeer. *Home Automation Systems* [Online]. Available: http://www.homeseer.com

[21] Zensys Controller Software. *Z-Wave PC Controller* [Online]. Available: http://zensys.software.informer.com

[22] H. L. Jørgensen. (2009, Nov 30). *An introduction to Z-Wave programming in C#* [Online]. Available: http://www.digiwave.dk/en/

[23] Pepper-One GmbH. *Z-Wave Device Library* [Online]. Available: http://www.pepper1.net/zwavedb/

[24] Tekscan. *Pressure Mapping, Force Measurement & Tactile Sensors* [Online]. Available: https://www.tekscan.com/