Partial Report

Project Title: Prescient: Context-Aware Home Monitoring and Control Central
ID JEMS: 155008

Students: Rodrigo Schmitt Meurer
Rafael Eriberto Mariot Scarduelli
Daniel Manzoni Seerig

Professor: Antônio Augusto Medeiros Fröhlich

University: Federal University of Santa Catarina

Declaration of Responsibility

We, students from project ID 155008, hereby declare that the provided information in this report is true, under supervision of our professor who signed below.

Professor's name: Antônio Augusto Medeiros Fröhlich

Professor's signature: ______________________________

Date: 16/09/2016

Proposed Scheduled

Date	Task
04/04	Send proposal
20/05	Receive board
20/05	Start documentation
20/05 - 10/06	Development of Galileo's case with board sensors
20/05 - 25/05	Board Sensors Implementation and Tests
25/05 - 30/05	Integration of Galileo with LISHA's Smart Room
01/06 - 10/06	Integration Validation
10/06 - 15/07	Hybrid application Development for smart devices control
15/07 - 20/07	Implementation and Evaluation of user on network discovery
20/07 - 30/07	Interface Galileo Gateway with Web and Database
01/08	Start development of context-aware algorithm
21/08	Start development of data visualization in hybrid application
20/09	Finish data visualization in hybrid application
20/09	Send partial report
05/10	Conclude context-aware algorithm
06/10 - 16/10	Final Tests, bug corrections and implementation validation
20/10	Prepare for presentation
20/10	Finish documentation
30/10	Submit final report
01/11 - 04/11	Presentation at SBESC

Update Scheduled

Date	Task	Percentage
04/04	Send proposal	100 %
20/05	Receive board	100 %
20/05	Start documentation	25 %
22/05 - 30/05	Development of hybrid application base graphical interfaces based on HTML, JavaScript and Jquery Mobile	100 %
01/06 - 05/06	Integration and validation of the graphical interface with charting libraries for data visualization	100 %
05/06 - 07/06	Study of board capabilities and embedded operating system choice	100 %
07/06 - 10/06	Development of smartphone detection daemon on PC	100 %
08/06	Technical Webinar	100 %
10/06 - 20/06	Ubilinux installation and setup on Galileo	100 %
20/06	Setup subnet and wireless router to test the daemon on Galileo	100 %
21/06	Primary tests for the smartphone detection in Galileo (Proof of Concept)	100 %
22/06 - 30/06	First tests with Galileo and EPOSMoteIII (attempting to control LISHA's Smart Room lights)	100 %
01/07 - 10/07	Evaluation of desired sensors to attach to Galileo and EPOSMoteIII	100 %
10/07 - 12/07	Installation and tests with wiringx86 python module for Galileo Gen2	100 %
13/07 - 25/07	Sensor Tests on Galileo Board with wiringx86	80 %
30/07	Start development of context-aware algorithm	30 %
01/08 - 08/08	Communicate hybrid application with python in Galileo through Flask API	80%
08/08 - 12/08	Performance evaluation for the user detection daemon	100 %
24/08	Technical Webinar	100 %
01/09 - 10/09	Construction of Galileo and EPOSMoteIII Case	90 %
11/09	Galileo and EPOSMoteIII installation on Case	100 %
12/09	Design printed circuit to handle the sensors and other peripherals	20 %
15/09	Start construction of Smart Room Model for the Demo at SBESC	15 %

20/09	Partial report submission	100 %
20/09	Reinstall AC control with smart device	-
21/09 - 28/09	Finish hybrid application development	-
29/09 - 07/10	Train and test algorithm with real data	-
08/10 - 10/10	Finish refining machine learning algorithm	-
11/10	Validate results and overall implementation	-
11/10	Record video for presentation	-
12/10 - 19/10	Finish demo model of the smart room	-
20/10	Prepare for presentation	-
20/10 - 28/10	Finish documentation	-
30/10	Submit final report	-

Difficulties and Workarounds

Situation 1

Portability is a very important issue in Internet of Things based projects. A user must be able to have access to what he wants in the closest platform. If a user is surfing the Internet on his computer, it is easier to open another tab in the browser than to find the smartphone and open an app to control a device or check the current consumption of an appliance. Therefore, cross-platform implementations of the system are crucial. The implementation of such, however, is not trivial. Android applications are implemented in Java while web apps are in HTML, JavaScript, and CSS.

Solution 1

To deal with this portability issue, we chose the Apache Cordova, an open-source mobile development framework, that can generate cross-platform applications from HTML5, CSS3, and JavaScript. This way it is possible to extend an application across more than one platform, without having to re-implement it with each platform's language and toolset.

Situation 2

Our project complies a vast range of functionalities, from controlling GPIO pins, scanning the network to search for a specified MAC address and interfacing applications through the internet. This requires many different packages and functionalities from Linux. With the yocto many times it was needed to create a specific image with the package we wanted to install. Besides, to our application, it is much harder to develop using the Arduino interface to make system calls to run Linux scripts. It is much easier to develop the whole system in a single environment.

Solution 2

Instead of using the Linux image available on many of the Galileo Gen. 2 tutorials, we chose to use an alternative embedded Linux distribution based on Debian, called ubilinux. Ubilinux is an embedded Linux distribution from Emutex, based on Debian Jessie. It is targeted at embedded devices that have limited memory and storage capabilities. This Linux distribution uses the same package manager as many other Debian based distributions, the apt-get. Therefore this distribution has access to a wide range of packages available for Linux. Among them, the Nmap tool, which is used to detect a smartphone's MAC address in the network. Also, there is a python module developed by Emutex that provides a straightforward and unified API to talk to the GPIO pins on the Intel Arduino capable boards, such as Galileo Gen. 2. Up to now, this module supports, among many others, writing to and reading from both digital and analog inputs/outputs from the Galileo GPIO, which is enough to satisfy all the desired functionalities of our project.

Situation 3

The smart room has a range of devices that are controllable through an IEEE 802.15.4 network. These devices are connected to EPOSMoteIII, wireless, real-time microcontrollers. Since our Galileo board does not have any native/integrated wireless capability, we extended it with another EPOSMoteIII. This way the Galileo board can control all the appliances forwarding commands and receive data through the EPOSMoteIII. To interface the mote with the board, we chose to use the Python Programming Language. Python has a module that encapsulates the access for the serial port making it easy to communicate the mote with the Galileo board through serial communication. The choice of Python as the default programming language, however, rendered necessary a framework enable the communication between the python scripts and the hybrid application. This framework should also be lightweight to run in the Galileo without causing any major overheads and be able to respond to requests made by the hybrid application.

Solution 3

Flask is a "microframework" primarily aimed at small applications with simpler requirements. It provides support for quickly building REST APIs. With this we were able to develop routines in Python that handle incoming requests from the hybrid application to the Galileo Board. As for the hybrid application side, it issues JQuery Ajax REST requests, enabling communication between both.

Situation 4

On our first try to detect a given smartphone's MAC address connected to the network we implemented a BASH script on a PC that used the Nmap tool to verify if it was connected or not to the network. In the PC we did not encounter any performance related problems. However, a PC processor is many times faster than an embedded such as the one used on the Galileo Board. Since user detection is a process that must be running with high frequency to detect if the user is present or not in the network, when running the script on the Galileo it made the processor busy for a long time, reducing the overall performance of the system. This performance issue is especially a problem when the user is present in the room because the system needs to keep constantly adjusting the devices to the user preferences. On the other hand, when there are no users connected to the network, the system can use the Nmap tool to discover a possible reconnection without the need to worry about the processing time overhead.

Solution 4

To deal with this performance issue, we noticed that instead of always checking the entire network for the given MAC address we could ping the respective IP address once the Galileo discovers the IP to which the device is connected. In our tests we found out that by using the ping command we were able to considerably reduce the processing overhead, reducing the time needed to check if a user is still connected to the network or disconnected from it. If the user disconnects from the network, we can resume using Nmap to search for any other user connection.

Situation 5

The choice of the machine learning algorithm used in this project is considered a milestone because it affects directly the performance, prediction accuracy, and development of the project. To find the most suitable one for our case, we consulted professors familiar with such techniques as well as books and other resources available on the Internet.

Solution 5

We started by defining our problem by laying down its tasks, performance measure and training experience, and ended up classifying it as a problem of supervised regression. From there we searched and compared different algorithms that are commonly used to address this type of problem to finally choose Artificial Neural Networks, mainly because it's robust to errors in the training input and has been successfully applied to problems that involve sensor data. Since we had previously decided to work with the Python Programming Language, we ended up using Keras, a highly modular neural networks library, on top of the Theano Python Library.