Home Automation and Security for Mobile Devices

Somak R. Das*, Silvia Chita†, Nina Peterson‡, Behrooz A. Shirazi†, and Medha Bhadkamkar†
* Massachusetts Institute of Technology, † Washington State University, ‡ Lewis–Clark State College
* das@mit.edu, † {schita, shirazi, medha}@eecs.wsu.edu, ‡ nmpeterson@lcsc.edu

***Abstract*—As mobile devices continue to grow in popularity and functionality, the demand for advanced ubiquitous mobile applications in our daily lives also increases. This paper deals with the design and implementation of HASec, a Home Automation and Security system for mobile devices, that leverages mobile technology to provide essential security to our homes and associated control operations. In particular, with the help of mobile devices, HASec operates and controls motion detectors and video cameras for remote sensing and surveillance, streams live video and records it for future playback, and finally manages operations on home appliances, such as turning ON/OFF a television or microwave or altering the intensity of lighting around the house. The proposed home security solution hinges on our novel integration of cameras and motion detectors into a mobile application. For instance, when motion is detected, the cameras automatically initiate recording and the iOS device alerts the homeowner of the possible intrusion. HASec has two main components interacting with each other: the iOS application that executes on the mobile device and server-side scripts that run in a cloud. Although HASec is implemented for Apple's iOS devices such as iPhone, iPod Touch, and iPad, it can be easily ported to other mobile platforms. Furthermore, our application is not only limited to smartphones but also can be used by feature phones through their browsers.**

Keywords—smartphone application; iOS application; home security; home automation; mobile cloud computing; sensors

I. INTRODUCTION

Mobile phones have become ubiquitous in our society. Smartphones have wide applications in various domains such as education, fitness and healthcare, entertainment, travel, news, and business, due to their easy access through BlackBerry App World, Apple's App Store, Google's Android Market, or Windows Marketplace for Mobile. However, to the best of our knowledge, there is no application available yet for smartphones that combines home automation and security.

Leveraging the capabilities of smartphones, we designed, developed, and implemented an iOS application, called HASec (shown in Figure 1), that integrates Home Automation and Security capability into mobile devices. Our application is also accessible through a web browser. The advantage of this approach is twofold: (i) it makes the application independent of the mobile operating system, vendors, and carriers, and (ii) it can be extended to feature phones that do not have all the capabilities of smartphones but have built-in browsers.

Specifically, HASec makes the following contributions: (a) it operates and controls motion detectors and video cameras for remote surveillance through mobile devices; (b) it enables streaming live video from the remote cameras to mobile devices; (c) it enables recording videos on a server for future playback; and (d) it controls and manages home appliances through mobile devices.

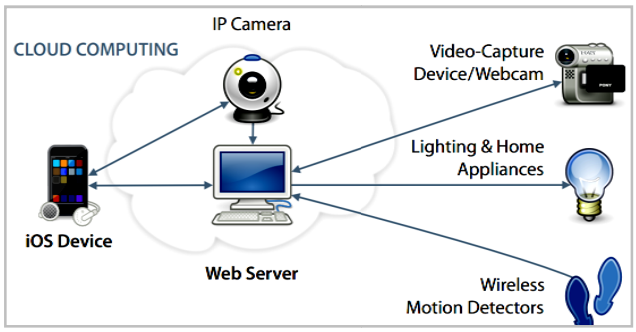

Figure 1. A system overview of HASec.

The proposed system HASec manipulates and controls the actions and movements (RECORD/STOP and PAN/TILT) of up to two cameras in real time. It can playback video recordings listed in the order of their timestamps, turn ON/OFF small appliances (e.g., television, stereo, computer monitor, toaster, and coffee pot), and control the intensity of lights around the house. HASec can also detect motion through sensors. Once motion is detected, it begins to automatically record and then sends an alert to the user's mobile device. The user can view a live video feed from the cameras and remotely control any device that is integrated with HASec. For instance, when an intruder is detected, the user can decide to record several minutes of video from a camera for identification of the intruder and thus turn on lights and stereo to deter him. HASec's goal is to provide a comprehensive integration of cheap off-the-shelf, widely available electronics to provide home automation and security with user-friendly installation and interface, leading to greater user adoption and retention.

HASec incorporates several devices, user interfaces, the cloud, and mobile software. It uses sensors for motion detection, cameras for remote viewing and recording, a server for data storage and intermediary interface, and mobile devices (in our implementation an iPhone, iPod Touch, or iPad) executing the iOS application as the human interface. HASec is implemented as a client–server model, in which the iOS device acts as the client using our application while the server is located in the cloud. All external devices, such as cameras, lights, and motion sensors, are connected to the server directly or through the X10 control protocol [8].

This paper is organized as follows. Section II provides the system overview of HASec. Sections III and IV respectively

This work is partially supported by NSF-REU grant IIS-0647705.

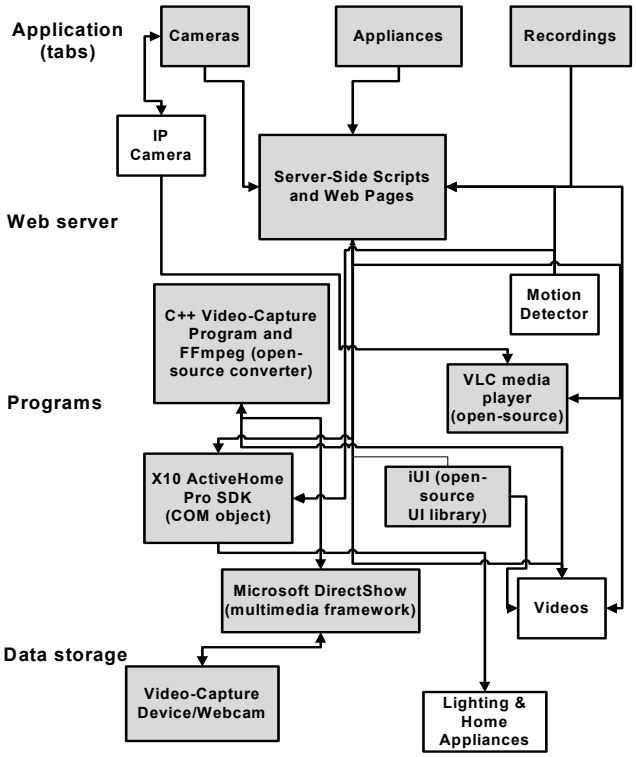

Figure 2. Software architecture of HASec.

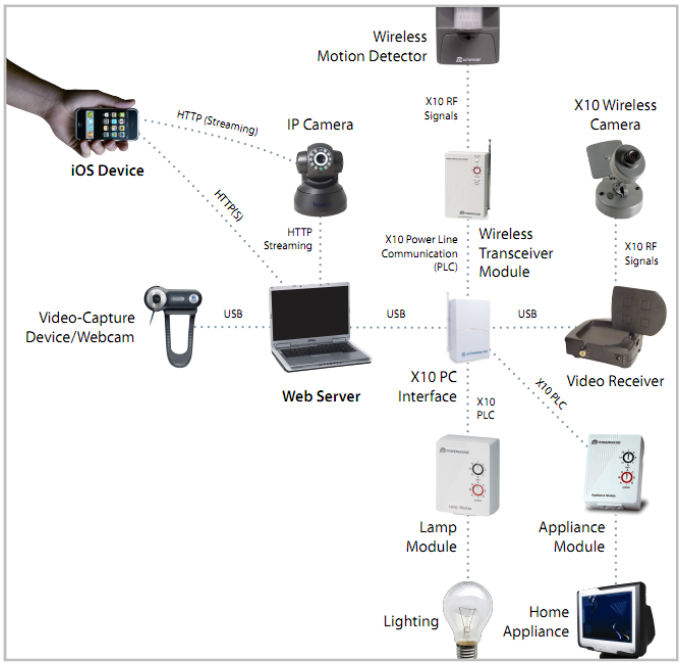

Figure 3. Communication between hardware components.

describe the server and client components, their specific functionalities, related challenges, and our proposed solutions. The performance evaluation of our system is presented in Section V, followed by a discussion of related works in Section VI. Concluding remarks are offered in Section VII.

II. System Overview

For home automation, we chose to use X10 power line communication (PLC) [1] technology due to its popularity and also ease of installation since it requires no additional wiring. Controllers and controlled modules (lighting and appliances) utilize the existing power line and thus are simply plugged into the existing power outlets.

HASec is implemented with Apple's mobile devices due to their increasing popularity among consumers [7]. However, since we used web technologies to implement the application server, our proposed solution can be easily ported to any smartphone from other vendors, such as Android or Windows Phone. Furthermore, HASec is not only limited to smartphones but also can be accessed via built-in browsers used by any other mobile devices.

Figure 2 describes HASec's software architecture, while Figure 3 displays the hardware components and implementation, including the communication between the devices, the server, and the client. As shown in Figure 2, we divide the software architecture into the four layers: application, web server, programs, and data storage. Each of them is further broken down into smaller entities with unidirectional or bidirectional communication links.

We set up an Apache HTTP Server [2] with open-source XAMPP [9] web-server package. Although our application uses web technology, we chose to implement it as a native iOS application rather than a web application. We plan to add context-aware features to HASec, and only a native application can access the iOS device's own sensors, like GPS.

HASec follows the client–server architecture model. Video is streamed from the server to the client, while commands and controls are sent from the client to the server. When necessary, alerts are sent from the server to the client.

III. Cloud and Server-Side

A. Control of Cameras

Since HASec provides the user the flexibility to choose any type of video-capture devices (e.g., webcams linked either internally or externally to the server, or IP cameras in the cloud), we had to account for each of their distinct design details and adapt a solution that would seamlessly integrate them into one cohesive application. Of the two types, generic webcams were more difficult to implement since they presented a unique challenge to allow the user to simultaneously view and record live video. We coded a video-capture program in C++ that responds to RECORD/STOP commands using DirectShow, Microsoft's multimedia framework and API [6]. It records video to an AVI file, which is then converted to a playable MP4 file with FFmpeg's open-source multimedia converter.

If the user is only allowed to view live video without the option to also record, as done by other applications, the implementation would have been relatively simple. However, our goal was to provide the user with the flexibility to record the stream while viewing it on the mobile device. The problem we encountered was that the stream could not be viewed until the

recording had stopped. We overcame this challenge by invoking the open-source VLC media player (from the script for controlling cameras) to access the input stream via DirectShow, duplicate it, and process it in the following two ways:

- Transcode and record to a MP4 file,
- Transcode to Motion JPEG (so mobile web browsers can natively view the live stream – without any additional plug-ins) and stream to the web server.

This also allowed us to avoid splitting or sharing the input stream.

The IP camera, on the other hand, provides a Motion JPEG stream, so we employed VLC media player to transcode and record the input stream to a MP4 file. To ensure that the PHP script continues to execute before the C++ and VLC programs exit (in other words, without waiting for those applications to terminate), it is executed as a background process.

Data Storage and Retrieval: Recorded videos are stored on the web server with timestamps as the filenames (e.g., 2010-07-21_12,13,18.mp4). To display the list of recordings in our application, the PHP script for recordings reads the contents of the video directory in reverse chronological order, parses the names of MP4 files (e.g., 2010-07-21 12:13:18), and places them into a hyperlinked list. Next, using the open-source iUI library, the list is formatted to match the iPhone Human Interface Guidelines, and the resulting web page is embedded into our application. The user simply has to tap on a timestamp on the iPhone, as shown in Figure 4 (c), to launch the corresponding video.

B. *Control of Home Appliances*

As discussed in Section IV, based on user input, the iOS application sends instructions to the web server – for example, brighten "Desk Lamp" to 100%. The PHP script for controlling home appliances converts the human-readable device name to the matching device codes (that is, house and unit codes) and then sends the command to the X10 PC Interface via the ActiveHome Pro SDK. These codes are set on the Lamp/Appliance Module and used by the X10 interface for power line communication with the device.

C. *Home Security*

We chose to combine home automation and home security into one application because of the user friendliness and ease of use that it enables. Additionally, this integration allows for the customization of the final product as both components strengthen each other. For example, cameras integrate with X10, particularly motion detectors, to allow for better security. If we only had cameras, we would not be able to determine the occurrence of an intrusion. Likewise, if we had only implemented the home-security component (e.g., motion sensors) then we would not be able to send the user a live feed of the video to see who/what was intruding if motion was detected.

Our system's default response to motion detection can be modified to fit the needs of the user. The user's devices (motion sensors, cameras, and home appliances) act as the home-security system that can be enabled or disabled in the HASec application.

IV. CLIENT-SIDE

The iOS device, executing our application, acts as a thin client that provides the user with a graphical user interface to HASec. We chose the thin client approach because of the limited computational resources of mobile devices. Since we must have a home computer anyway to communicate with the device controllers (because of X10's closed-source SDK), this approach was best suited for the task at hand.

Figure 4 (a–d). *Cameras*, *Appliances*, *Recordings*, and *Options* tabs.

Based on user input, our application is built to communicate with the web server by sending commands to control and manage devices located around the home. The application is organized into four tabs: (1) *Cameras*, (2) *Appliances*, (3) *Recordings*, and (4) *Options*.

The buttons on each page are associated with a Uniform Resource Locator (e.g., a M-JPEG stream, MP4 video, or a server-side script in PHP), and accessed via HTTP. When the user presses a button, a connection to load the corresponding URL is sent [4], in the form of http://(server's IP address)/(path to script)?(command). UIWebView embeds web

content [5], such as live streams in the *Cameras* tab and a hyperlinked list of captured videos in the *Recordings* tab.

- The *Cameras* tab's preview page is shown in Figure 4 (a), where the user has the option of viewing one or two cameras. The live streams are loaded from the cloud, either directly from an IP camera or from the server, and are shown split-screen. The user can then view the cameras individually with the zoom feature and PAN/TILT them if supported. The user can start or stop video recordings (from one or both cameras) in the preview and individual pages.
- The *Appliances* tab, shown in Figure 4 (b), displays an example of appliances integrated with HASec. They can include lamps and simple (ON/OFF) appliances such as television, computer monitor, alarm clock, and so on. X10's Lamp Module allows for changes in light intensity and we provided both a slider and a text field to control the percentage of brightness. The user can specify the intensity using either one. Implementing the slider presented a challenge: typically, when a slider is pressed, it automatically (and continuously) sends its value to the server until it is released. However, we observed that this results in many unnecessary commands that can burden the X10 PC Interface and add excessive delay. Hence, we modified the slider so as to only send the final value.
- After loading the corresponding web page from the server, the *Recordings* tab shown in Figure 4 (c) displays a list of videos (taken by the user or automatically by HASec) sorted in chronological order of timestamps, with the most recent video at the top of the list. When the user selects the recording to watch (by tapping on the timestamp), it is played in an appropriate format for the iOS device. Since each video has a URL, it can be viewed on a web browser as well.
- The *Options* tab, shown in Figure 4 (d), gives the user the ability to change settings, including security, the server's IP address, camera types, and timers for any device. When motion is detected and the security feature is enabled, video is automatically recorded and an alert is sent via email, which the homeowner receives along with sound and vibration (instantly for push email).

V. PERFORMANCE EVALUATION

We conducted experiments to first measure the latency of streaming video from a remote location for two cases (average network congestion and high network congestion) and then expanded the scope of our evaluation to include other tests as well. These results were recorded for 6 successive latency measurements 20 seconds apart (totaling 2 minutes) and then averaged them. We repeated each experiment 10 times.

For the first scenario, depicted in Figure 5, we used high resolution of 640×480 pixels and the highest allowable frame rate allowable by iOS device, i.e., 7.5 frames per second (fps). We tested both an IP camera and a webcam separately as well as together. As expected, the results for the individual streams (approximately 2–5 seconds) are lower than those for the simultaneous streams (approximately 3–7 seconds). Even the high-quality video performed well, especially with simultaneous streams.

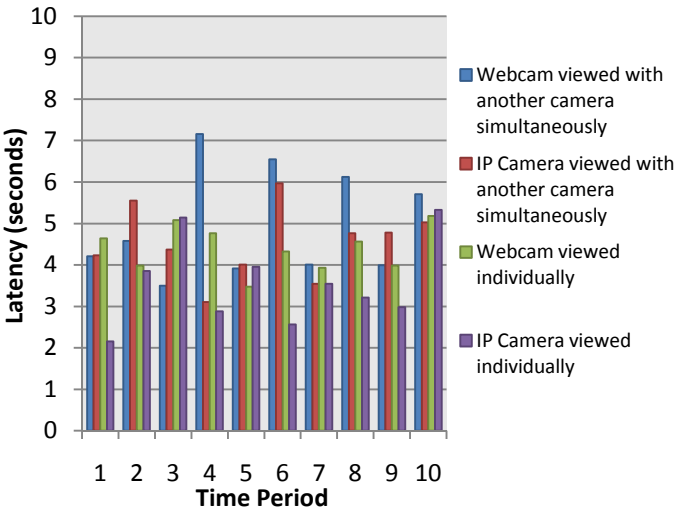

Figure 5. Latency of high-quality streaming in network with average traffic.

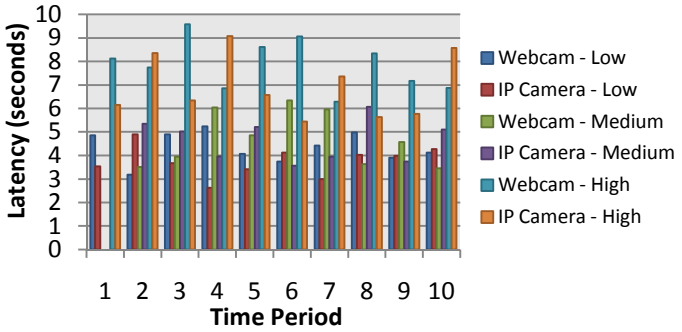

Figure 6. Latency of variable-quality streaming in network with high traffic.

The second scenario, shown in Figure 6, exhibited a worst case or stress test. In this scenario, we used a high traffic network (performed during busy hour) while continually using simultaneous streams. We did this for three resolutions: high (640×480 pixels, 7.5 fps), medium (320×240, 5 fps), and low (160×120, 2.5 fps). As expected, the reduction in delay is proportional to the quality of the video streams. However, even in the worst case scenario (of this stress test) the delay is never greater than 10 seconds. We experimented with different streaming qualities. We observed that if a HASec user is using a poor quality network (due to a slow connection or high network traffic), then we can adaptively reduce the quality of the video stream and thus continue to achieve good performance (2–6-second delays in the worst case).

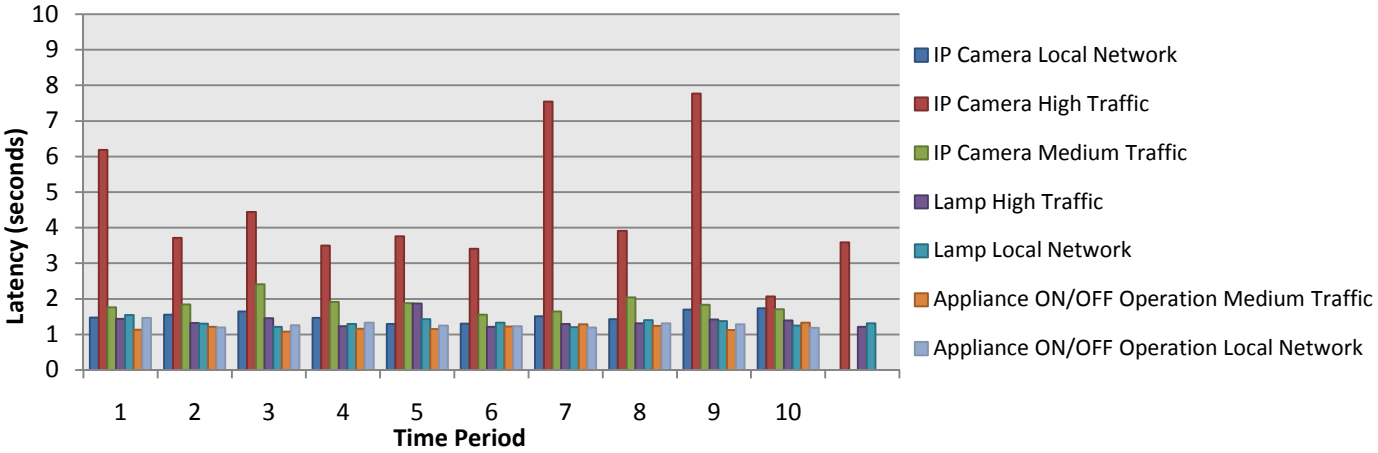

Figure 7. Latency of control operations in networks with variable traffic.

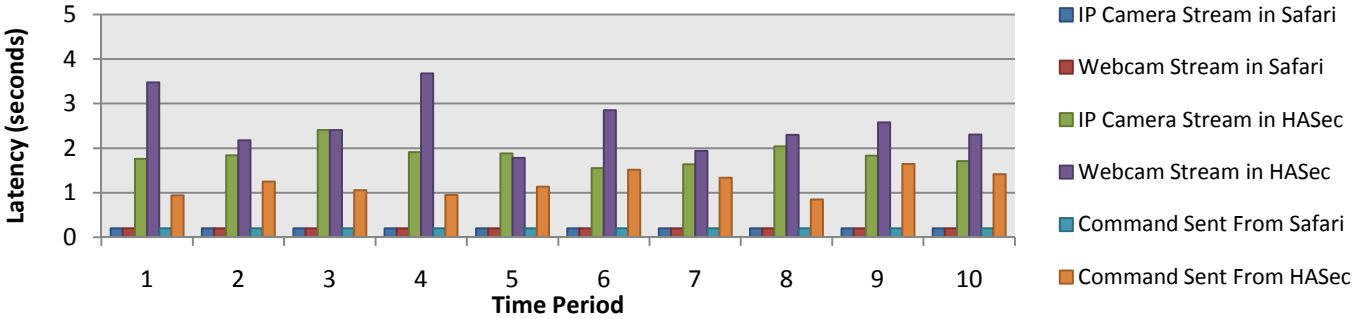

Figure 8. Latency for mobile vs. non-mobile devices.

For the third scenario, tests were run using different networks that represent a user being at home and viewing the video streams from inside a local network, a user viewing the stream from a medium-traffic network outside the home, and a user viewing the stream from a high-traffic outside network. As expected, the high traffic network experienced significant delays when compared to the others. We also tested latencies for home automation processes such as turning ON/OFF appliances, lamps, and TVs and dimming lamps. The test results depicted in Figure 7 demonstrate that under varying network conditions, home automation actions are carried out in a consistent 1–2.5-second range.

For the fourth scenario, tests were conducted to display the difference between running streams or commands on the iOS device as opposed to using the same browser on a non-mobile device. For the purposes of this test, the browser used was Safari (note that the iOS uses Safari as the built-in browser, the UIWebView has the same functionality as Safari, and Safari is available for use on a desktop or laptop) on a MacBook using the same Wi-Fi network as the mobile device, with average network traffic. The results in Figure 8 demonstrate that the browser used on a non-mobile device is consistently faster, with nearly three times less delay. The browser used in Safari and the UIWebView used in the HASec show significantly slower results. This shows that the longest delay comes natively from the mobile OS itself, instead of the network conditions or HASec implementation.

VI. Related Works

Existing iOS applications support the X10 standard and video cameras, with limitations [3]. X10 Commander controls X10-compatible lights and appliances, but it does not support any cameras or motion detectors. Like X10 Commander, MobiLinc Pro/Lite Insteon, and X10 Controller do not support any cameras. PixieLinc, HomeControl, and Indigo Touch do not support the X10 PC Interface but rather requires the user to purchase and install a Smarthome SmartLinc server. None of them supports any cameras.

Live Cams is an application that allows IP cameras to be viewed and also controls pan, tilt, and zoom. However, it does not natively support viewing webcams or recording from any type of camera. Both HSTouch and HomeWatch stream videos; however, they only support two specific Axis IP cameras. Additionally, they require that the user purchase additional hardware. Some applications, such as CMS Mobile allow viewing and recording video streams, but only after the user purchases and installs a server by one of their distributors. Others, such as Mobiscope Home Surveillance, iCam, Air Cam Live Video, and AtHome Camera only allow viewing and/or recording video.

While the above applications provide either camera or some home security feature, none of them provides video recording as well as home security (motion detection, lights, and appliances) and cannot be easily extended to provide both features. As our lives become more complicated, the simplifica-

tion and integration of sophisticated systems is ideal for greater user adoption. Thus, our application is a unique integration of these two features into a single cohesive package.

VII. Conclusions and Future Works

We developed a comprehensive solution called HASec that provides a user-friendly home automation and security application. We accomplished this through the integration of cheap, off-the-shelf, widely available devices, interfaces, and software coupled with a user-friendly interface. HASec provides users with an easy-to-use mobile application from which they can remotely access and control their home appliances and security.

The proposed architecture of HASec operates under the client–server model, with a mobile application as the client and server in the cloud connected to external devices. HASec can be accessed by any web-enabled device through browsers. Our flexible architecture allows extending HASec to other mobile devices as well, such as BlackBerry and Android, as well as integrating other security systems.

In the future we intend to add temperature and power-usage readings, thus leading to manual and automated energy-efficiency solutions. Additionally, we are looking into how to make our application context- or location-aware. More specifically, if the user opens our application and has forgotten about the security ON/OFF feature, then our application should alert him to enable security if he is more than 5 miles away from home, for example.

References

[1] M. Kovatsch, M. Weiss, and D. Guinard. Embedding Internet technology for home automation. In *Proceedings of IEEE Conference on Emerging Technologies and Factory Automation (ETFA'10)*, 2010.

[2] Apache HTTP Server Project, httpd.apache.org

[3] See Apple App Store, apple.com/iphone/apps-for-iphone

[4] URL Loading System Programming Guide, developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/URLLoadingSystem

[5] UIWebView, developer.apple.com/iphone/library/documentation/uikit/reference/UIWebView_Class

[6] Microsoft DirectShow, msdn.microsoft.com/en-us/library/dd375454

[7] T. Spierings. Mobile everywhere. IN10 Communications, 2010.

[8] X10, x10.com/support/basicx10.htm

[9] XAMPP, apachefriends.org/en/xampp.html