Portable Platform Independent Patient Monitoring

Rogier J Barendse[1], Teus B van Dam[1], Stefan P Nelwan[1]

[1]Erasmus MC, Rotterdam, The Netherlands

Abstract

Medical applications are becoming increasingly available on portable devices such as smart phones, tablets and small notebooks in healthcare organizations. However, application development for diagnostic and monitoring purposes on these devices is still limited, because of regulatory requirements and significant development time and complexity for the multiple mobile platforms in use. Our objective was to create a cross-platform patient monitor solution, with the main focus on tablets and test the feasibility of this platform for patient monitoring.

We have developed an HTML5-based portable patient monitoring application to be a cross-platform based solution for a broad range of tablets. The application shows the vital signs, curves and trend parameters as a monitor would display.

The usability of the portable patient monitoring application has been evaluated on the iPad 3, Samsung Tab 2 10.1 and the Windows 8 platform. We tested the main browser on each platform and also the latest version of additional popular (mobile) browsers.

In conclusion, platform independent patient monitoring is possible by using standard tablet operating systems and the HTML5 standard without requiring app-development.

1. Introduction

With the growing popularity of portable devices such as smart phones, tablets and small notebooks for consumers, these devices have become increasingly available in healthcare organizations. Many healthcare organizations are currently implementing mobile healthcare applications within their organizations. However, application development for diagnostic and monitoring purposes on these devices is still limited, because of regulatory requirements, significant development time and complexity for the multiple mobile operating system platforms in use. Mobile platforms may facilitate remote patient monitoring solutions inside and outside the hospital. For example, a thoracic surgeon may want to monitor his post-operation patient on the intensive care unit. Another case is the anesthetist on call during a surgeon to check the vital signs remotely.

In the recent years HTML5 has become a stable and mature platform that is implemented in all major browsers on almost every platform. With the introduction of web sockets and the canvas element, HTML5 could provide the necessary tools to display the patient monitor data in the same fashion as a patient monitor.

1.1. Background

In earlier work we demonstrated that the physiological information of a patient monitor could be displayed on a mobile devices as the iPAQ [1]. Since then many mobile platforms have appeared and disappeared which made it cumbersome to keep up-to-date with the then current devices. Each new platform had a specific set of libraries and programming languages, which would have required much effort into porting and redesign to support these platforms. Also since then, mobile devices have a much bigger penetration in the general population. The medical application library on these devices has grown rapidly.

To build a platform independent patient monitoring application a continuous socket connection is required (to receive continuous data) as well as a library that provides rapid screen updates (to draw continuously on the screen). Previously this was available in the platform specific libraries or in technologies like JAVA, Adobe Flash or Microsoft Silverlight.

With the arrival of the HTML5 [2] standard both of these requirements are fulfilled. HTML5 offers web sockets for the networking and the canvas object can be updated rapidly to do animations.

One other main advantage is that all major browser companies have adopted the HTML5 standard. HTML5 is available on all current mobile platforms that have an internet browser. The HTML5 standard should be able to eliminate the effort spent in supporting multiple platforms natively for patient monitoring.

1.2. Objective

Our objective was to create a cross-platform patient monitor solution, with the main focus on tablets by using HTML5 as platform and test the feasibility of this

platform for patient monitoring.

2. Methods

We developed an HTML5-based portable patient monitoring application designed to be a cross-platform based solution for a broad range of tablet operating systems (iOS, Android, Windows 8). The application shows the vital signs, curves and trend parameters as a monitor would display.

The backend, running on a web server (IIS7), captures the patient monitor data from the network and sends it to through a web service using the SignalR library [3]. SignalR is a socket library written in ASP.NET and JavaScript and enables real-time access web functionality to the web client application.

The web service extracts the information and sends it to the clients with SignalR. The JavaScript client code draws the signal and parameter data onto the HTML5 canvas object. The canvas implementation provides rapid screen updates needed to draw the curve signal seamlessly on the screen.

2.1. Monitoring side

Our hospital has a variety of patient monitors from Draeger Medical. As a result of earlier collaborative projects we had access to the description of the network protocol. With this knowledge we were able to receive and decode the packets from the monitor [1,4]. The network code used in our solution also works with the telemetry monitors on our medium care unit.

2.2. Client side

In order for the client side to be platform independent on current portable platforms we choose HTML5 and JavaScript as development language. The main part of the screen consists of the HTML5 canvas object. On the canvas object we draw the curves and display the parameters.

The SignalR JavaScript library handles the network connectivity. This library creates web sockets on which the data from the patient monitor is received from the webserver. The data is stored in several JavaScript buffers of the received JSON objects [5]. We implemented responsive web design with CSS 3.0 and relative scalable variables on the canvas object to respond adequately to the various resolutions on the devices.

The drawing routine obtains the data from these buffers continuously and removes used data from these buffers. The canvas object provides rapid screen updates to draw the curves seamlessly onto the screen.

2.3. Server side

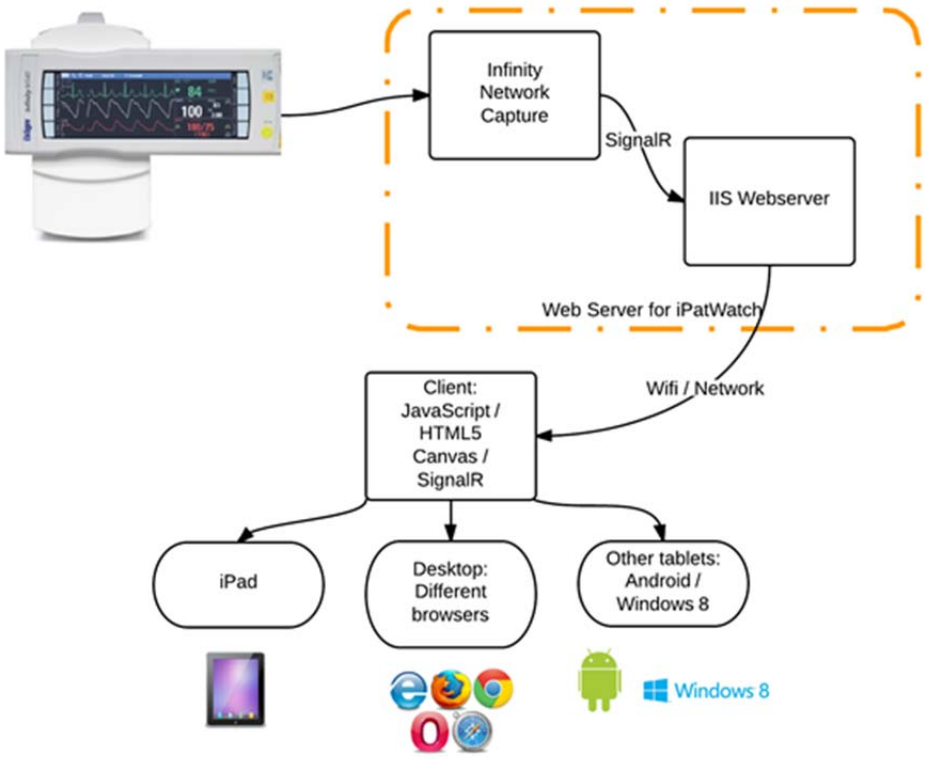

Figure 1. Complete overview of the setup.

Figure 2. Screenshot of the application on an iPad 3 showing the physiological curves and trends of a patient.

Figure 1 shows a system setup with the different components and interfaces. The server is connected to the patient data network of the patient monitors. The Infinity network component library receives the data from the patient monitors and sends it to the webserver component by SignalR. We built a library loosely based on our previous work [4] to assess the monitoring information. The library receives, decodes and parses the monitor data by subscribing on a multicast stream on which the patient monitor broadcasts its information.

The server platform is Windows 2008 with IIS 7.0 and .NET 4.0. Internally the SignalR library handles the connections to the clients and the re-encoding of the data using JSON.

3. Results

We visually compared the results on the screen with the current commercially available product Draeger PatientWatch [6]. This ActiveX component also provides near real-time display of patient monitoring data. We assessed the right colors from this application. When comparing the latency of the curves between both applications our solution seemed slightly faster in handling the data. This gap was hard to measure correctly and could not be assessed visibly using a stopwatch.

We evaluated the usability of the portable patient monitoring application on the iPad 3, Samsung Tab 2 10.1 and the Windows 8 platform. We tested the main browser on each platform and also the latest version of additional (mobile) browsers as Chrome, Firefox and Opera. All browsers were capable to display the monitor curves and trend parameters.

Figure 2 shows the application in action on an iPad 3; displaying near real-time (processing and network overhead) physiological curves and parameters from the monitor.

4. Discussion

This paper shows it is feasible to build a platform independent monitor application using existing open source technology. The application can display near real-time patient information including continuously updated waveforms and parameters. It is technically even possible to implement alarming into this application. However that will turn the device into a medical device, submitted to many regulations.

Tools for mobile patient monitoring on commercial of the shelf technologies, such as smart phones or tablets, have been made available by using specific software from medical device manufacturers. However, this software is specific. A clinician may require several applications for different care units. In addition, it is often difficult to use the software outside of the hospital intranet due to custom proprietary protocols.

The solution presented in this paper provides a single, consistent user experience which uses standard communication protocols (HTTP) and makes secure off-site communication possible using Web-VPN technology.

Commercial platform independent viewers, such as the App from AirStrip Technologies [7], provide a similar functionality to that of ours. Airstrip provides viewers for the two most widely available platforms (iPhone and

Android) at this moment and add additional functionality, such as laboratory results. These functionalities from Airstrip extend the smartphone to a mobile point of care data entry device.

The application should be secured to ensure the patient information is safe from malicious users. This can be achieved by having secured wireless network, VPN access for outside of hospital use and HTTPS authentication. HTTPS encryption and authentication is still something we are looking into to implement.

Another point of discussion is what the added value would be of an alarming monitoring application to be outside the ward and in theory outside of the hospital even. It can be helpful for a physician on standby to have a quick view of what is going on with the patient, but in that scenario alarming would not be needed. This is also the case when a thorax surgeon wants to monitor the status of his patient, which just has been in his operation room.

Another downside of implementing a good alarming solution is that it will probably need indeed platform specific app-development. An app can take more control of the hardware of the device, which is needed to guarantee a robust way of delivering the alarm, even if the device is used for something else at that moment.

5. Limitations

We created this solely for a feasibility test to determine if the standard techniques used in this program were capable of creating a monitoring display screen on a mobile device. In the current state this software should not be used in clinical practice. To achieve this more robustness and testing is needed. The application should be verified and extensively checked for correctness to the patient monitor.

In theory the system can also display and handle alarms, however in the initial phase we decided not to implement the alarms. The main reason for this is that we did not yet want to comply to the strict regulations that are required medical devices with paging capabilities.

New web standards and browser may cause limitations in the future but we hope that new browser will handle the backward compatibility with great care, however each new browser version should be carefully tested.

For now this solution is limited to the Draeger monitoring platform. We think that this solution could be extended to support other vendors.

6. Conclusion

Platform independent patient monitoring is possible by using standard tablet operating systems and the HTML5 standard without requiring platform specific app-development.

References

[1] Nelwan SP, Van Dam TB, Klootwijk P, Meij SH. Ubiquitous mobile access to real-time patient monitoring data. Computers in Cardiology 2002: 557–60.

[2] Hickson I, Hyatt D. HTML5: A vocabulary and associated APIs for HTML and XHTML. W3C Work Draft Ed. 2011;

[3] ASP.NET SignalR [Internet]. [cited 2013 Jul 31]. Available from: http://signalr.net/

[4] Nelwan S, Meij S, Fuchs K, Van Dam T. Ubiquitous access to real-time patient monitoring data. Computers in Cardiology 1997: 271–4.

[5] Crockford D. The application/json media type for javascript object notation (json). 2006;

[6] Infinity® Gateway Suite [Internet]. [cited 2013 Sep 4]. Available from: http://www.draeger.com/sites/enus_us/Pages/Hospital/Infinity-Gateway-Suite.aspx

[7] AirStrip ONE™ Patient Monitoring [Internet]. [cited 2013 Sep 4]. Available from: http://www.airstriptech.com/airstrip-one-patient-monitoring

Address for correspondence:

Rogier J. Barendse
‘s-Gravendijkwal 230
Room Ba-567
3015 CE Rotterdam
The Netherlands
r.barendse@erasmusmc.nl