Annotation Framework for the Physical Activity Evaluation in Real-Time

Miroslav Mužný[1], **Jan Mužík**[1]

[1] Spin-off Application Centre, First Medical Faculty, Charles University in Prague, Czech Republic

Abstract

Objectives: Work aims to create a portable tool with a decision support, providing relevant methods for purposes of physical activity evaluation in real-time.
Methods: We have utilized accelerometer equipped ez430Chronos watch in conjunction with a preconfigured RaspberryPi-based setup. Wireless transmission of accelerometer data into a running web application instance, which served as a user frontend, is provided through the WebSocket protocol. Decision support is based on a Weka classifier.
Results: The proposed framework is ready to be used for the annotation and basic evaluation of physical activity data in a Wi-Fi covered areas. Minor issues are related to the occasional instability of data transmission, which has to be handled consequently.
Conclusions: We found the overall framework architecture robust enough to serve its purpose. Next steps in the development will lead to an expansion of outlined functionality.

Keywords

Human Activity recognition, WebSockets, Activity counts, ez430Chronos, Annotation Framework

Correspondence to:

Miroslav Mužný
Spin-off Application Centre, First Medical Faculty,
Charles University in Prague
Address: Studnickova 7, Prague 2, 120 00, Czech Republic
E–mail: miroslav.muzny@lf1.cuni.cz

EJBI 2013; 9(3):32–37
recieved: July 31, 2013
accepted: November 2, 2013
published: November 20, 2013

1 Introduction

Human activity recognition (HAR) is an extensive area of research for a long time. In this respect, variety of different solutions relying on an accelerometer sensor have been proposed. It is evident that recently, many of these solutions started to utilize the pervasiveness of the internet to enrich the knowledge, when operating in a real-life environment. We show, that advances in technology allow to make a tool for acquisition and evaluation of physical activity data in real-time. The usage of the this tool is not bounded by borders of testing under laboratory conditions, but it is utilizable in real-life experiments. The project we are going to present here, takes an advantage of commonly available hardware and software instruments. Therefore, it is particularly easy to reproduce it.

As far we have been using ez430Chronos watch [1] for a purpose of collection of accelerometer data. The watch features wireless transmission of raw accelerometer data. However, the utilization of the watch itself for purpose of analysis of physical activity data in real-time is limited in many ways. One of the shortcomings is directly given by the reasonably small range of the Texas Instruments USB dongle, which is supplied with the watch. This fact contributes to the difficulty of recording accelerometer data in some environments. Another shortcoming is a lack of annotation tool. Therefore, we have decided to develop a portable solution for the collection and annotation of user's physical activity data. The JavaScript enhanced web application served as a user frontend for our more complex framework behind the scene.

Similar approach, introduced as a smart phone application, was proposed by Lara et. al. [2]. The relevant part of the complex real-time HAR system was able to fetch a set of recognizable activities from the remote Application server using a HTTP protocol. The recognition itself took a place on a smart phone and only aggregated feedback data were passed to the remote server.

Another similar approach was proposed by Riboni et. al. [3]. Their COSAR context-aware activity recognition tool based on ontological reasoning and statistical inferencing aimed to recognize more complex activities on a basis of Android device's integrated sensors. Similarly as in the previous case, merged sensor's data were used to form feature vectors, which were consequently evaluated by a remote reasoner.

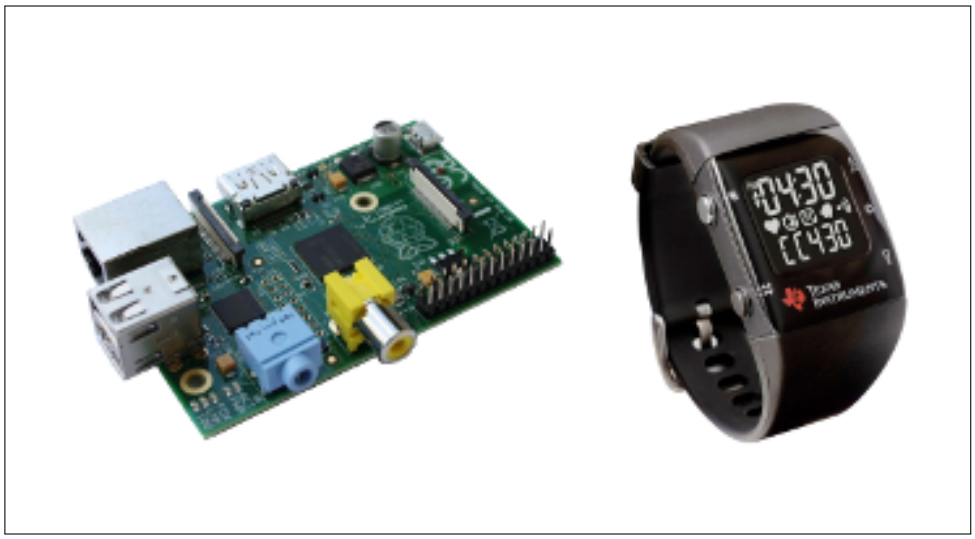

Figure 1: RaspberryPi-based portable setup.

Finally, recently published study about a multi-layer mHealth monitoring system introduced a platform featuring a real-time data transmission based on WebSockets [4]. The platform gathered data from various wirelessly connected sensors in a body area network (BAN) and sent them to a remote server. However, contrary to previously referred projects, this platform served a monitoring purpose and therefore lacked the decision support.

In this work, we deal with a design of annotation framework enhanced by a basic decision support and usable in a real-life environments. We are neither targetting to create a complex BAN, nor the specific smart phone application for an activity recognition. Our goal is an easy to setup, robust tool with a real-time data transmission based on a WebSockets technology, which can be used for a further analysis of physical activity data in real-time.

2 Methods

It is important to mention, that in this stage of development we are not focusing on particular aspects of activity recognition process. The robustness and overall stability were primary goals though the real-time nature of the whole system. However, to enable the full data flow through the system, we have configured the crucial elements contributing to the recognition of physical activity according to results in other HAR related works [2, 5].

Modern web browsers provide a wide range of features related to the real time data processing. The WebSocket protocol [6] is one of the technologies, which makes possible to conveniently develop such applications, while retaining odds of web based user interface.

Therefore, our early aim in this project was to ensure that the capabilities of the WebSockets protocol are enough to handle the real-time transmission of accelerometer data. Furthermore, we have experienced a strong demand on the convenient data annotation and visualization. The preservation of the portability of whole setup was our intention as well.

2.1 Instrumentation

The portable setup, intended to be worn at user's waist or in a backpack, consisted of ARM-based computer, RaspberryPi [7], along with a connected Texas Instruments USB dongle. Wireless internet connectivity ensured the low-powered USB Wi-Fi adapter [8]. All of the components were powered by an external rechargeable battery pack. That allowed us to keep the overall setup up and running at least for 3 hours. Figure 1 shows the RaspberryPi and the ez430Chronos watch, which served as a wirelessly connected source of raw accelerometer data.

2.2 Configuration

RaspberryPi is capable of running several different Linux distributions. We chose an ArchLinux distribution [9] because we have experienced a Wi-Fi connectivity issues when using other distributions. Upon the startup, the device connects to the nearby Wi-Fi network. At the same time, the reverse SSH tunnel with a preconfigured remote computer is automatically established. Tunnel itself solves multiple forthcoming issues. At first, it bypasses eventual NAT and Firewall restrictions of the network. Furthermore, the tunnel makes the annotation framework accessible from the public internet.

We have configured multiple wireless profiles in order to cover the most of the user's space of movement. In case when the device is suffering from a weak signal, the network is automatically switched to the one with a better coverage.

2.3 System Architecture

On a basis of our previous positive experience, we have managed to use the JRuby scripting language together with a Rails 4 technology, which already supports WebSockets protocol. Integration of the Weka machine learning framework [10] through its native Java bindings facilitated our development process in many aspects.

Figure 2 shows the system architecture from the viewpoint of basic functional blocks. The core Rails 4 application bundle is denoted by a dashed line. This bundle is actually deployed and runs on the RaspberryPi device.

Sensor-equipped Tester actor is a person who wears an ez430Chronos watch and the RapberryPi setup as well. Administrator actor represents a person who services the Web-based user interface. Administrator has an access to raw sensor data, some of preprocessed data and also to activity labels. Besides a passive participation, Administrator can initiate certain actions towards the framework. This feature is provided by WebSockets full-duplex connectivity [11]. Therefore, existence of Backward parameters setting paths on the Figure 2 should not be understood as a way of automatic adjustment of parameters in an autonomous system. They are dedicated to accomplish actions initiated from the Administrator's side and their specific purpose is further discussed.

The Data preprocessing component handles incoming data from the ez430Chronos watch. The watch's accelerometer sampling frequency is set to 30Hz. For now, we configured the component to extract median average deviation, root mean square and curtosis features from a half overlapping windows with 128 samples. These extracted features are consequently forwarded to a Classification component, which takes a further action.

In our proposed architecture scheme, the Data processing component also serves for the computation of features, which are not passed to the Classification component. These features are meant to be visualized in the Web-based user interface and that is why they are passed directly to the Communication component. We demonstrate this possibility on the example of activity counts, which are further examined in a recent study from the University of Portland [12]. Computation of activity counts commonly precedes the estimation of energy expenditure. We have integrated the estimation as well as the visualization of activity counts of 1 second epochs into our framework. While we are not taking benefit of the activity counts output yet, its estimation looks promising for the future calculation of energy expenditure.

The Classification component currently holds the implementation of C4.5 decision tree from the Weka machine learning framework. The C4.5 classifier was trained to distinguish, on a basis of time domain features from the Data preprocessing module, between several levels of walking and running activities.

We put the Weka activity classification model in a separate file, aside of the Classification component's processing logic. Besides the apparent design intention, there was another reason to make this step. Since the WebSockets protocol supports also transmission of binary data, the separate classification model can be easily replaced by another one from the Web-based user interface.

Event handling and Communication component deserve a special attention. As it acts as a mediator of nearly all communication, its flawless functionality is crucial for the whole system. We had to deal with multiple asynchronous inputs and therefore we have utilized a reactor pattern [13], which was the most likely pattern to meet design requirements. More specifically, we used the EventMachine library, which implements the reactor pattern in Ruby language. Figure 3 shows the Event handling and Communication component in a detailed view with all inputs to better illustrate the bindings to asynchronous events. As noticeable, events originating from the network are based on incoming WebSockets notifications in a JSON format. On the other side, events from the inside of application are fired by a Redis database channel. Redis database is an advanced key value store [14]. The publish/subscribe pattern was implemented in version 2.0. Therefore, it allows to specify a series of channels that a user can subscribe to. This was a basic presumption for an implementation of an event handler, which receives notifications from different parts of the accelerometer data processing pipeline.

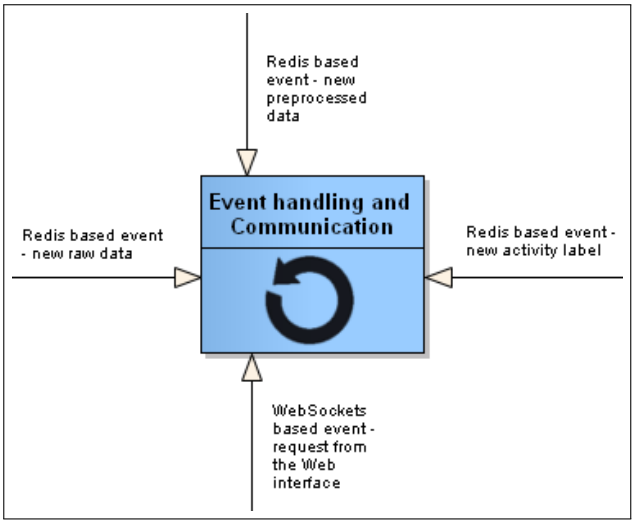

Figure 3: Event handler.

2.4 Data Visualization and Interpretation

This section covers the description of Visualization and Management component. Originally, we have been mostly interested in the real time visualization of the raw accelerometer data. For that purpose, we have used the FlotChart, JavaScript graph component [15]. The FlotChart component contains additional set of plugins, which makes easier to consecutively inspect and annotate captured data. From this perspective, our framework currently supports zooming, selection, continuous recording,

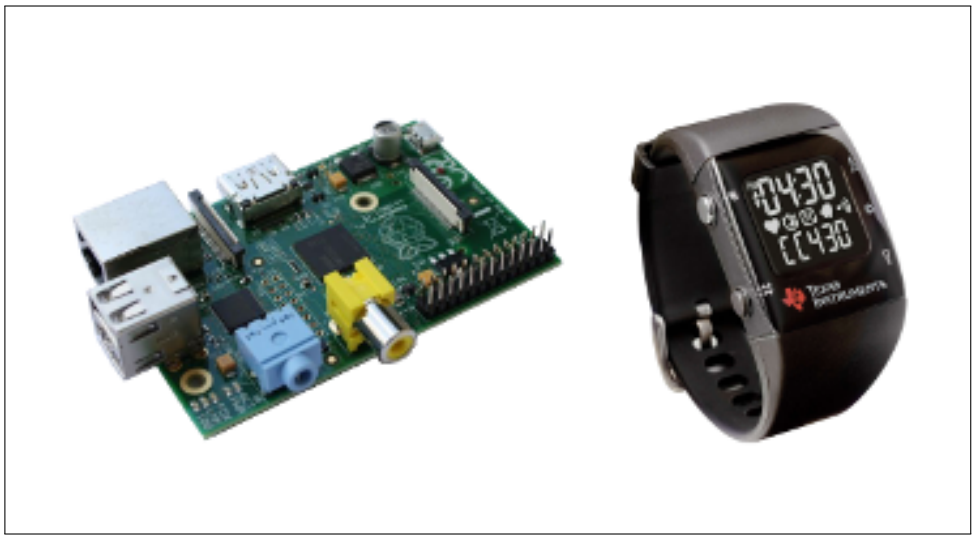

Figure 1: RaspberryPi-based portable setup.

Finally, recently published study about a multi-layer mHealth monitoring system introduced a platform featuring a real-time data transmission based on WebSockets [4]. The platform gathered data from various wirelessly connected sensors in a body area network (BAN) and sent them to a remote server. However, contrary to previously referred projects, this platform served a monitoring purpose and therefore lacked the decision support.

In this work, we deal with a design of annotation framework enhanced by a basic decision support and usable in a real-life environments. We are neither targetting to create a complex BAN, nor the specific smart phone application for an activity recognition. Our goal is an easy to setup, robust tool with a real-time data transmission based on a WebSockets technology, which can be used for a further analysis of physical activity data in real-time.

2 Methods

It is important to mention, that in this stage of development we are not focusing on particular aspects of activity recognition process. The robustness and overall stability were primary goals though the real-time nature of the whole system. However, to enable the full data flow through the system, we have configured the crucial elements contributing to the recognition of physical activity according to results in other HAR related works [2, 5].

Modern web browsers provide a wide range of features related to the real time data processing. The WebSocket protocol [6] is one of the technologies, which makes possible to conveniently develop such applications, while retaining odds of web based user interface.

Therefore, our early aim in this project was to ensure that the capabilities of the WebSockets protocol are enough to handle the real-time transmission of accelerometer data. Furthermore, we have experienced a strong demand on the convenient data annotation and visualization. The preservation of the portability of whole setup was our intention as well.

2.1 Instrumentation

The portable setup, intended to be worn at user's waist or in a backpack, consisted of ARM-based computer, RaspberryPi [7], along with a connected Texas Instruments USB dongle. Wireless internet connectivity ensured the low-powered USB Wi-Fi adapter [8]. All of the components were powered by an external rechargeable battery pack. That allowed us to keep the overall setup up and running at least for 3 hours. Figure 1 shows the RaspberryPi and the ez430Chronos watch, which served as a wirelessly connected source of raw accelerometer data.

2.2 Configuration

RaspberryPi is capable of running several different Linux distributions. We chose an ArchLinux distribution [9] because we have experienced a Wi-Fi connectivity issues when using other distributions. Upon the startup, the device connects to the nearby Wi-Fi network. At the same time, the reverse SSH tunnel with a preconfigured remote computer is automatically established. Tunnel itself solves multiple forthcoming issues. At first, it bypasses eventual NAT and Firewall restrictions of the network. Furthermore, the tunnel makes the annotation framework accessible from the public internet.

We have configured multiple wireless profiles in order to cover the most of the user's space of movement. In case when the device is suffering from a weak signal, the network is automatically switched to the one with a better coverage.

2.3 System Architecture

On a basis of our previous positive experience, we have managed to use the JRuby scripting language together with a Rails 4 technology, which already supports WebSockets protocol. Integration of the Weka machine learning framework [10] through its native Java bindings facilitated our development process in many aspects.

Figure 2 shows the system architecture from the viewpoint of basic functional blocks. The core Rails 4 application bundle is denoted by a dashed line. This bundle is actually deployed and runs on the RaspberryPi device.

Sensor-equipped Tester actor is a person who wears an ez430Chronos watch and the RapberryPi setup as well. Administrator actor represents a person who services the Web-based user interface. Administrator has an access to raw sensor data, some of preprocessed data and also to activity labels. Besides a passive participation, Administrator can initiate certain actions towards the framework. This feature is provided by WebSockets full-duplex connectivity [11]. Therefore, existence of Backward parameters setting paths on the Figure 2 should not be understood as a way of automatic adjustment of parameters in an autonomous system. They are dedicated to accomplish actions initiated from the Administrator's side and their specific purpose is further discussed.

The Data preprocessing component handles incoming data from the ez430Chronos watch. The watch's accelerometer sampling frequency is set to 30Hz. For now, we configured the component to extract median average deviation, root mean square and curtosis features from a half overlapping windows with 128 samples. These extracted features are consequently forwarded to a Classification component, which takes a further action.

In our proposed architecture scheme, the Data processing component also serves for the computation of features, which are not passed to the Classification component. These features are meant to be visualized in the Web-based user interface and that is why they are passed directly to the Communication component. We demonstrate this possibility on the example of activity counts, which are further examined in a recent study from the University of Portland [12]. Computation of activity counts commonly precedes the estimation of energy expenditure. We have integrated the estimation as well as the visualization of activity counts of 1 second epochs into our framework. While we are not taking benefit of the activity counts output yet, its estimation looks promising for the future calculation of energy expenditure.

The Classification component currently holds the implementation of C4.5 decision tree from the Weka machine learning framework. The C4.5 classifier was trained to distinguish, on a basis of time domain features from the Data preprocessing module, between several levels of walking and running activities.

We put the Weka activity classification model in a separate file, aside of the Classification component's processing logic. Besides the apparent design intention, there was another reason to make this step. Since the WebSockets protocol supports also transmission of binary data, the separate classification model can be easily replaced by another one from the Web-based user interface.

Event handling and Communication component deserve a special attention. As it acts as a mediator of nearly all communication, its flawless functionality is crucial for the whole system. We had to deal with multiple asynchronous inputs and therefore we have utilized a reactor pattern [13], which was the most likely pattern to meet design requirements. More specifically, we used the EventMachine library, which implements the reactor pattern in Ruby language. Figure 3 shows the Event handling and Communication component in a detailed view with all inputs to better illustrate the bindings to asynchronous events. As noticeable, events originating from the network are based on incoming WebSockets notifications in a JSON format. On the other side, events from the inside of application are fired by a Redis database channel. Redis database is an advanced key value store [14]. The publish/subscribe pattern was implemented in version 2.0. Therefore, it allows to specify a series of channels that a user can subscribe to. This was a basic presumption for an implementation of an event handler, which receives notifications from different parts of the accelerometer data processing pipeline.

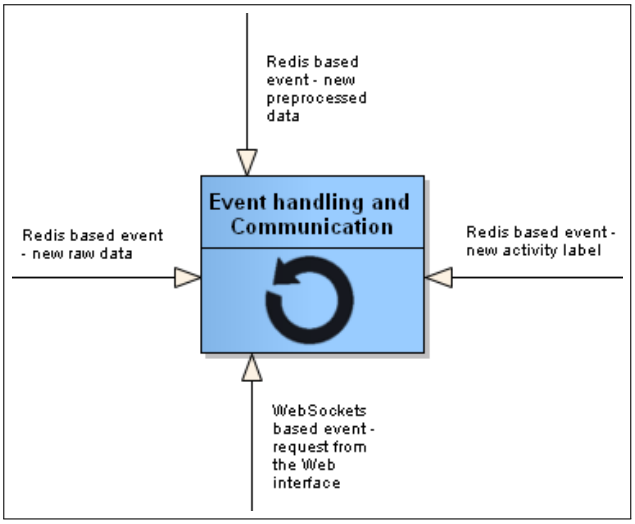

Figure 3: Event handler.

2.4 Data Visualization and Interpretation

This section covers the description of Visualization and Management component. Originally, we have been mostly interested in the real time visualization of the raw accelerometer data. For that purpose, we have used the FlotChart, JavaScript graph component [15]. The FlotChart component contains additional set of plugins, which makes easier to consecutively inspect and annotate captured data. From this perspective, our framework currently supports zooming, selection, continuous recording,

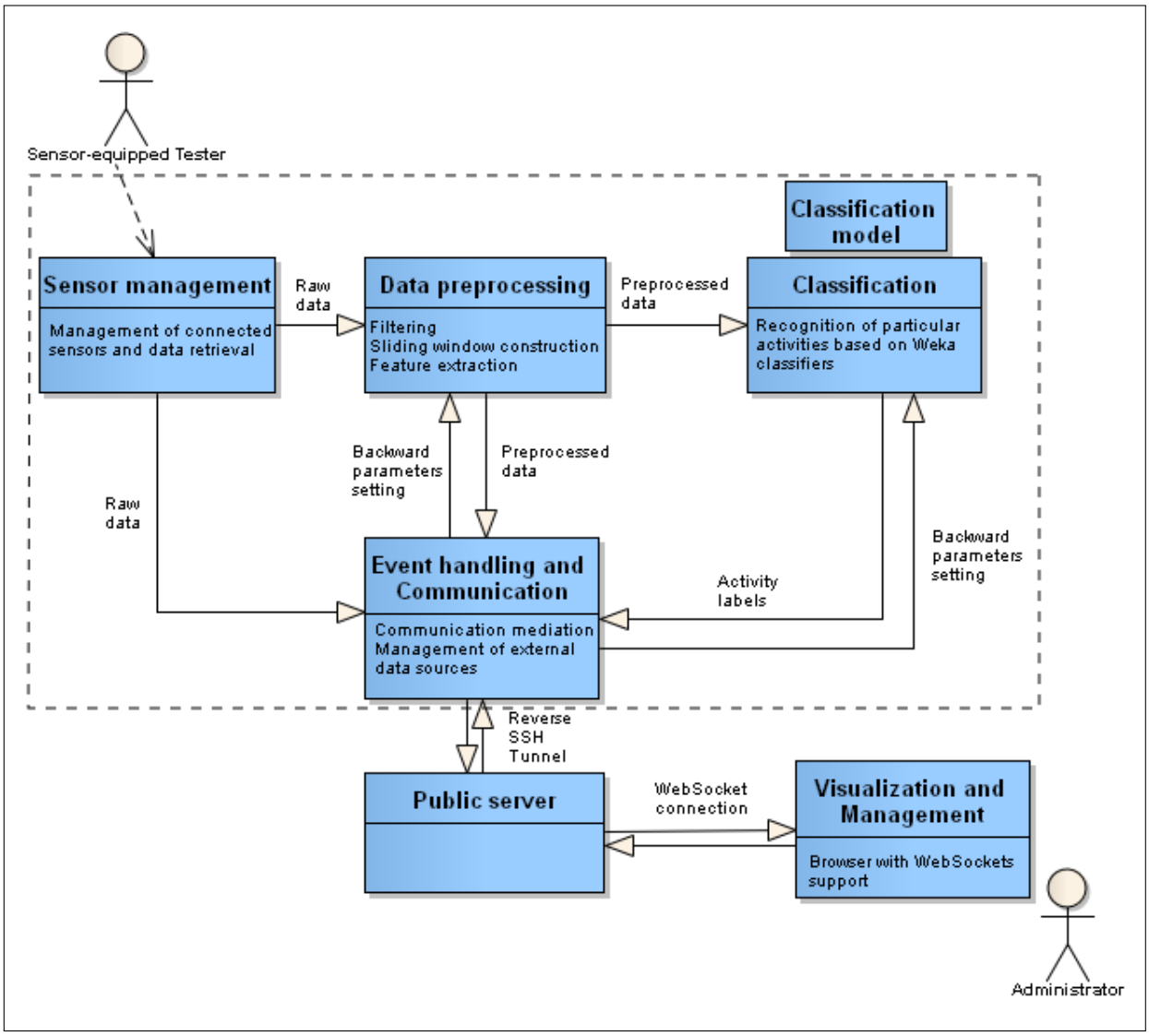

Figure 2: System architecture.

displaying of received annotation flags and insertion of own annotation flags. Besides that, the Administrator is allowed to select accelerometer axes, which has to be displayed. Among implemented functionality counts also an export of recorded data with annotation flags into a file in a CSV format.

The Web-based user interface also serves as a remote management interface. As stated earlier, the Administrator can remotely update the classification model and also adjust settings of Data preprocessing component. At the moment, the only possible adjustment is the change of the time epoch length for the computation of activity counts.

Figure 4 shows the plotting part of the Web-based user interface. The red, blue and yellow series represent incoming data from the 3-axis accelerometer. The green serie in the bottom of figure represents estimated activity counts. On the Figure 4 are also visible annotation flags as the output of the Classification component.

3 Results

We have prepared a basic architecture ready to serve for purposes of real-time physical activity evaluation. Fully automatic startup simplified the operation steps on the plugging of the power cable into a battery pack. Relatively small and lightweight package can be worn on a belt around the waist. Also, 3 hours of operation looks promising for most of planned testing scenarios. However, the prototype is not suitable for a daily activity tracking purposes in a comparison with other activity trackers on the market.

Occasionally, we have been dealing with a non-fluent data transmission. As its consequence, the drawing of the chart got stuck until fully re-buffered the data packets. Therefore, we have analyzed the average delay between receiving the pair of incoming packets for the last second. On the basis of its value, we have eventually stopped redrawing the raw accelerometer data, so the transmission

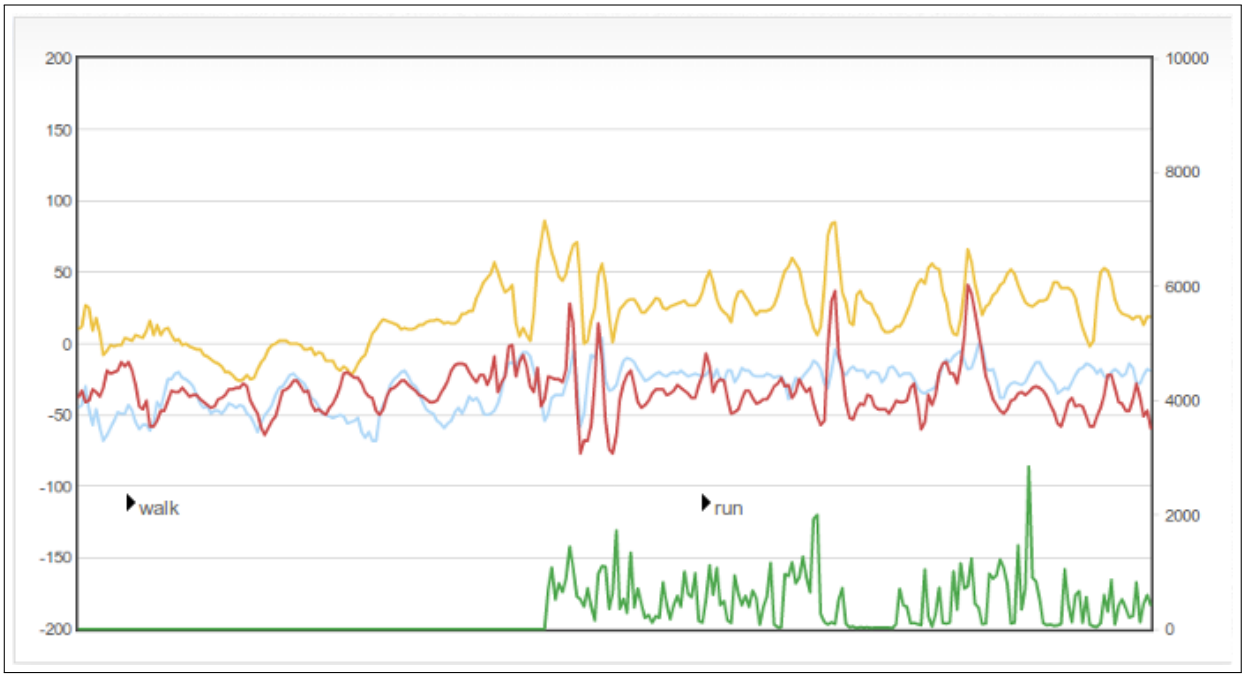

Figure 4: Plotting part of the Web-based user interface.

was cut down to an activity counts data together with output of the Classification component. This feedback-initiated fallback mode is supported by an automatic reconnection process implemented in the WebSockets protocol.

Besides the data transmission related issues, the framework has several minor flaws. For the proper over the internet functionality the framework requires preconfigured computer with a public IP address. With a slight modification, the framework can however be used completely without a network connection. This modification resides in locally storing all data for the future offline analysis. The framework is also dependent on a linux environment for many bindings to native libraries. Therefore, it is not possible to implement it as a smart phone application yet.

4 Discussion

We are about to extend the existing framework functionality. Besides its current purpose, the solution could serve as a more complex framework for the collection of data from various sources. That involves also low-powered Bluetooth equipped sensors, which spread out recently. Next steps in the development process will also lead to the expansion of features of the Web-based interface, as most of them currently serve as testimonials of the system architecture.

Results of the framework related research will not be limited to a single device used in our experiments but will affect a wide group of devices. That includes particularly smart phones, which are available to a public audience. Taking advantage of a daily used device with a sufficient computation power could open a range of possibilities regarding the real-time processing of activity data.

After all, we think that our framework holds a potential for a future usage and that we made a step towards achieving a platform, which can be utilizable for the physical activity evaluation in a wide scale.

Acknowledgements

The paper has been supported by the SVV-2013-266 517 project of Charles University in Prague.

References

[1] Ez430 - Chronos. [Internet]. [cited 2013 Oct 31]; Available from: http://processors.wiki.ti.com/index.php/EZ430-Chronos

[2] Lara Oscar D, Labrador Miguel A. A mobile platform for real-time human activity recognition. In Consumer Communications and Networking Conference (CCNC), IEEE, 2012, 667-671.

[3] Riboni D, Bettini C. COSAR: hybrid reasoning for context-aware activity recognition. In Personal and Ubiquitous Computing, 2011, 15(3), 271-289.

[4] Zhang W, Stoll R, Stoll N, Thurow K. An mHealth Monitoring System for Telemedicine Based on WebSocket Wireless Communication. Journal of Networks, 2013, 8(4), 955-962.

[5] Chernbumroong S, Atkins A S, Yu H. Activity classification using a single wrist-worn accelerometer. In Software, Knowledge Information, Industrial Management and Applications (SKIMA), 2011, 5th International Conference on (pp. 1-6). IEEE.

[6] The WebSocket Protocol. [Internet]. 2011 [cited 2013 Oct 31]; Available from: http://tools.ietf.org/html/rfc6455

[7] Raspberry Pi. RASPBERRY PI FOUNDATION. [Internet]. [cited 2013 Oct 31]; Available from: http://www.raspberrypi.org/

[8] Edimax EW-7811Un. EDIMAX. [Internet]. [cited 2013 Oct 31]; Available from: http://www.edimax.com/en/produce _detail.php?pd_id=347pl1_id=1

[9] Arch Linux Arm. [Internet]. 2009 [cited 2013 Oct 31]; Available from: http://archlinuxarm.org/platforms/armv6/raspberry-pi

[10] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1), 10-18.

[11] Furukawa Y. Web-based Control Application Using Websocket, ICALEPCS2011, 673-675.

[12] Bernstein D, Coleman E, Lambert A, Peach D, Shindler J, Callender H, Niederhausen N. Comparison of Physical Activity Assessment Tools. [Internet]. 2012 [cited 2013 Oct 31]; Available from: http://reu-ret.hosted.willamette.edu/reuret2012/2012results/UP_REU_Final_Presentation.pdf

[13] Schmidt D. C. Using design patterns to develop reusable object-oriented communication software. In Communications of the ACM, 1995, 38(10), 65-74.

[14] Sanfilippo S, Noordhuis P. Redis. [Internet]. [cited 2013 Oct 31]; Available from: http://redis. io.

[15] Flot: Attractive JavaScript plotting for JQuery. [Internet]. [cited 2013 Oct 31]; Available from: http://www.flotcharts.org/

