Smart watches as a web technology: Android Wear

Lise Stork
MSc student
lise.stork@gmail.com

Sander de Bont
MSc student
a.p.v.de.bont@umail.leidenuniv.nl

Matei Szabo
MSc student
mateiszabo@hotmail.com

ABSTRACT In the following paper we will discuss Google's new operating system for smart watches: Android Wear, and how it fits into the internet architecture as a web technology. We will start by giving a concise overview of its history and purpose after which we will provide an overview of its technology. The rest of the paper will discuss its strengths and weaknesses, its intended and unintended applications, and a short how to, to let you start working with the technology.

1. PURPOSE, CONTEXT, HISTORY

Before wireless communication was possible, computing devices could only send and receive data via physical copper or glass fibre wires. With the coming of wireless technologies it became possible to also connect devices to networks by use of electromagnetic radiation[1]. Because of these new technologies the demand for different "smart" objects started growing. By use of protocols to facilitate the communication between different devices, the internet is evolving to an "Internet of Things" (IoT) where computers and smart objects both form a part of the internet infrastructure [2].

Smart watches, although still in their infancy, are to contribute to this IoT. Ideas about smart watches arose over fifty years ago, but technology was not advanced enough to construct them then [3]. Now that processor sizes have decreased, prices have dropped and battery life has extended, manufacturing smart watches became viable. By enabling them to talk to other devices and the internet, the integration of smart watches in society will possibly provide new ways of communicating and change the way we interact.

Smart watches are small wearable computing devices that are worn around the wrist and, apart from telling the time, can connect to networks via bluetooth to send and receive information. As they make use of a variety of sensors that can be accessed for application development, sensory information about e.g. physiology or GPS coordinates can be send over a network to alert or talk to other devices [3]. Since recently, smart watches can also connect via Wi-Fi thus expanding their connection range[4]. For now however, Android Wear (AW), Google's operating system for wearables, still has quite some limitations for the use of the smart watch as an autonomous agent. The smart watch uses a handheld device as its host. Because of this, a wearable can still only connect to the internet indirectly by use of a handheld device as its gateway.

The User Interface (UI) of the Android Wear Operating System (OS) typically works with messages that show information that is retrieved from the handheld device. These messages appear on the screen of the watch via a stream of cards and include information that should be interesting for the user at that moment. Examples are text messages, emails or personalised information. Data can also be send back to the handheld device when simple controls are performed. Android Wear integrates Google's voice commands, Google Now, which can be used to trigger controls such as sending a message, open applications or take notes [5].

In March 2015 Google launched the Android Wear Operating System [5], but the history of smart watches goes back to the 1927's with the first navigational apparatus that could be worn around ones wrists. Other devices have been developed over the years but pebble's successful smart watch kick-starter campaign opened the eyes of other companies. Underneath some technologies that lead to the development of the AW OS.

- 1927: Plus Four Wristlet Route Indicator, the first navigational device to be worn around the wrist [6].
- 1970's: the first LED and LCD watches such as Pulsar NL C81 [3].
- 1994: Jaap Haartsen invented bluetooth at Ericsson [7].
- 1995: Seiko MessageWatch could already display caller ID's (by use of FM sideband frequencies) and other data as stock prices, weather forecasts and sports data [6]
- 1995: Breitling Emergency Watch could send out signals in case of emergency [7].
- 1997: First version of the 802.11 protocol - wireless network standard - was released.
- 1998: the Linux Wristwatch was already contributing to the IoT as it could connect wirelessly to other devices [7].
- 2012: the Pebble watch had an incredibly successful kick-starter campaign, showing other companies the demand for smart watches [3].
- 18-03-2014: Launch of Android Wear platform [5].
- 10-12-2014: Release update that featured the watch face API and change of software to be based on Android's 5.0 "Lollipop" [5].
- 10-03-2015: Google announces a new update for AW which allows Wi-Fi connection and hand gestures [4].

2. OPERATING PRINCIPLES

Underneath we will shortly provide a summation of the key concepts of AW's Technology, starting with its topology followed by the basics of the OS and UI, to

finally conclude with the Wearable Data Layer Application Programmable Interface (API)[1] for application development and device inter-communication.

2.1 Android Wear network topology

With the AW operating system, a wearable connects by means of tethering[2] to other devices. It can connect to a handheld device over a 6-digit encrypted bluetooth 4.0 connection[8] that is used as communication channel between devices. Since recently a wearable is also capable of connecting over Wi-Fi to the cloud node, the Google Servers, so that data can be synced between devices from a greater range. A wearable can however still not communicate to the internet directly, see figure 1.

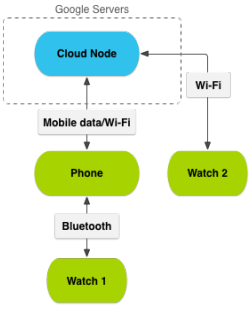

Figure 1. AW data communication [9]

Reasons for this could be power and size constraints. This means that for a wearable to connect to a web browser it needs to connect with its host first which will establish the connection. In this way the handheld device serves as a proxy server to which the wearable client can connect to enter the internet. Because the wearable can not connect to the internet directly[10], the handheld device handles the Hypertext Transfer Protocol (HTTP)[3] or Hypertext Transfer Protocol Secure (HTTPS) requests to the web server and receives its responses [11][12].

2.2 Android Wear OS and UI

The Android Wear OS is a reduced version of the latest Android OS. Because the smart watch is still an extension of a handheld device, developing for AW typically means developing two Android Package-files (APK); one for the wearable and one for the handheld device, see figure 2. This Wear APK has to be packaged inside the handheld APK, because a wearable can not yet browse on the internet to download and install an APK autonomously. This way, the package is downloaded on the handheld device which pushes the Wear APK to the we wearable via the bluetooth connection [13].

Although the operating systems for both devices are similar, the UI had to be re-developed to fit the screen of a watch without worsening usability. To make the use of a small screen easier, AW's UI is based on what Google calls the Context Stream and the Cue Card. The Context Stream exists of a continuous stream of cards that show potentially interesting information to the user. This information physically exists on the host, the handheld device, and is constantly synced between devices so that information is up-to-date. The Cue Card merely reacts to the voice commands of Google Now and executes them by sending them over the connection to the host device [5].

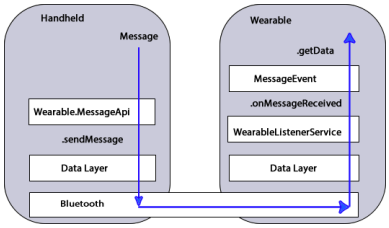

Figure 2. Android Wear Protocol [14].

Fortunately, although still in an early stage, it is also possible to develop fullscreen applications that do not use this restrictive card-based system. For fullscreen applications Google also developed API's to simplify control on a smart watch, developers can however choose not to use these and write applications from scratch.

2.3 The Data Layer API

Developers can use the Java programming language to develop applications. The Google Play services application runs in the background of the Android operating system and creates a channel for communication between two devices. To access this communication channel for application development, programmers can implement the Wearable Data Layer API in their code. This protocol stack consists of three sub-API's: a Data API, a Message API and a Node API. Messages and data that are send from one device to another move down this protocol stack on one side, over the bluetooth connection and up again on the other [14]. See figure 2 & 3.

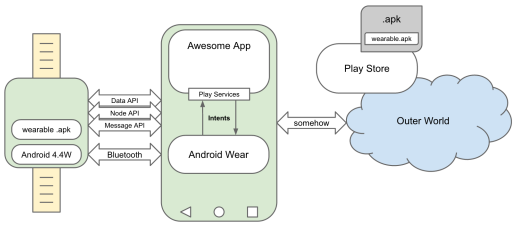

Figure 3. The Wearable APK and data layer API [15]

The Node API holds track of the connection between nodes. A node in this architecture could be the cloud

---

[1] An Application Programmable Interface (API) is a set of routines, protocols, and tools for building applications.

[2] Tethering: connecting one device to another via wireless LAN (Local Area Network) (Wi-Fi), Bluetooth or via a physical connection

[3] The Hypertext Transfer Protocol (HTTP) or HTTP secure (HTTPS) is the protocol that arranges the communication between a web client and a web server.

node over Wi-Fi, a handheld device or another smart watch that it could connect to. It notifies the user when such a node is within reach and thus when it can connect. It can also be used to inform the user when and if it is connected to a node. The message API creates a possibility to send byte arrays between both devices, allowing for simple controls to be send between devices called Remote Procedure Calls (RPC). Finally, the data API is the biggest API, as it stores and interchanges data as for example sensory data and larger data such as images. All data items are synchronised across all the devices of a wearable network, allowing all the devices to check and manipulate the data [16]. Each data item is identified by a unique Uniform Resource Identifier (URI) defined by the source node (the host) and a custom path [17].

3. STRENGTHS AND WEAKNESSES

Although Android Wear has only recently been announced and is still developing, the operating system definitely shows some strong features already and its future seems promising. Underneath we will discuss the strengths and weaknesses of Android Wear OS and the the Android watch.

3.1 Strengths

One of the intended purposes for the smart watch was to change the way we communicate via our handheld devices. Constantly checking to see if for instance a message is received can cause people to lose focus and waist time. Wearables should change this by merely displaying important messages thus taking away the need to check updates on a handheld device that is stored away in a pocket or bag. For professionals, this advantage could help improve task efficiency[18].

Second, AW provides a link to a set of useful sensors and communication technologies that are provided on wearables and because of its fixed bodily position, Android Wear can easily retrieve and send bodily data through these sensors.[19].

1. Data acquired from the heart rate sensor can be used for fitness and health monitoring applications.
2. Integrated voice commands can be triggered by the wearable's microphone to send controls or messages to the handheld device. With this, phone calls can be triggered from the wearable.
3. A GPS tracking unit can be used to record geological data from the user.
4. A bluetooth connection enables pairing with other devices to interchange data.
5. The Wi-Fi update allows users to still access their phone and receive and manipulate important data over long distance [4].

Further, because AW developers comply with the Android compatibility program that stipulates exactly how an application must be made in order to be compatible with others, AW applications can easily communicate with existing technologies based on this protocol, hereby simplifying connectivity for Android users but also the developing process for developers [20]. Applications for smart watches can be realised within the Android Developer environment, so anyone looking to create applications has a large library at hand. Because of the compact interface of the watch, developers are encouraged to create applications which require very little navigation and take little time to use. As a result, the apps are easy and quick in their usage. This however also is a weakness as the options for developers are limited and the typical applications remain rudimentary.

Lastly, through fast data connections such as Wi-Fi or Bluetooth, fast data transmission is secured. This could prove useful in institutions such as hospitals where fast data communication within a Local Area Network (LAN) is essential. Because of these fast data connections and its fixed bodily position, AW is a useful contribution to the IoT.

3.2 Weaknesses

As of yet the biggest deficiency of smart watches is their lack of autonomy as they are still an addition to smartphones rather than fully functional standalone units. They do not yet seem to be able to compete with the capabilities of Smartphones, e.g. making calls and establishing a mobile internet connection without the use of other devices.

Size limitations are obvious with limited space for batteries, storage and processors. Given these limitations, applications that would fit the demands within such a device such as for example Video Chat, gaming and calling, can not yet be realised on stand alone smart watches. Because the user interface is also limited in size, applications that use more complex interfaces such as Virtual Keyboard must either be modified or can not work on such a device, leaving developers with a limited set of affordances [21].

Due to its restrictive web environment, AW does not provide a system WebView, a component that allows applications to display web content [22]. Since recently however, a Wear application called the Wear Internet Browser can be downloaded from the Google play store that implements its own WebView for AW devices and supports HTTP and HTTPS [23]. A small keyboard is provided in the application but also Google Now can be used to search the web by use of voice commands. The browser is however slow, can not load video's, and does not work without the connection with a handheld device.

Web application development for Android Wear proves little possibilities, but there are some such as HTMLView which integrates HTML 4[4] and CSS 2.1[5] so that it can be used to develop applications that show HTML code on the wearable, see figure 4. A framework that also shows some potential is called CocoonJS and should be the first platform that allows programmers to write APK's that use the new IOS 8 WKWebView. This means that it is possible to have HTML5 running on a wearable. CocoonJS implements WebGL, which is an

---

[4] HyperText Markup Language (HTML) is a markup language for the specification of documents, typically on the World Wide Web.

[5] Cascading Style Sheets (CSS) is a technique for web-page design.

open standard for specifying 3D computer graphics in web pages through OpenGL so that a plug-in for the web browser is not needed. With this platform it should be possible to develop and publish web-based games for Android Wear. Creators of CocoonJS however also warn that their project is still in its beginners phase and that it does not always function properly [22].

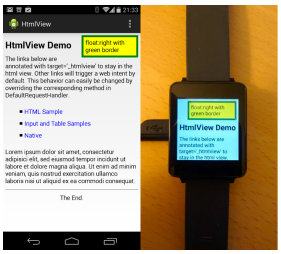

Figure 4. Android Wear HTMLView [23].

4.INTENDED APPLICATIONS

The current typical applications that are to be found on probably every watch with Android Wear are firstly those that are based on its bio-sensing capabilities. The variety of built-in sensors are often used to support Health applications. Google Fit for example has the ability to track a persons steps and heart rate, also more interesting data such as blood glucose [24]. These data are used in applications to for example keep track of burned calories or monitor what distance is covered. Applications even provide charts and fitness history. Other interesting applications are being developed at the moment that use these bio-sensing capabilities, such as a diabetes application that monitors the blood glucose of a patient continuously [25].

Another typical application is based on the system's built-in GPS such as Google Maps, which lets you access walking directions or lets you navigate your car from your the directions given on your watch. For cycling this could prove useful as navigating with handheld devices prove dangerous in busy traffic situations [26]

Smart watches from other companies such as Apple implement features that are not yet available on AW but use the devices to their full potential [27]:

1. Applications for accessing public transportation are currently available in China and only requires the user to wave their hand across the scanner by use of Near Field Communication[6].
2. Microsoft's FitBit app allows users to check into the Gym and may soon also allow for integration with smart fitness machinery to choose a custom workout difficulty. Presumably, more likewise applications are or will be under development.
3. Seeing as BMW is a partner developer on smart watches there is a big chance that the devices will be used as keys that can unlock the door. Also BMW has plans for locating users and summoning the cars to their position.
4. Samsung is working on integration of smartphones in their SmartThings home system, which allows users to unlock the door by proximity, turn lights on and off and control the thermostat.

5. SURPRISING APPLICATIONS

Besides Google's intentions for the implementations of smart watches in daily life, smart watches can and are used in different ways than allowed by Google's operating system.

5.1. Misuse of data resources

Seeing as a smart watch is presumably being worn all through the day, data gathered by it may be recorded and reveal things like heart rate, global position and (if the features are available and active) payment information about the user. As is with smartphone hacking this data can presumably be misused in a number of ways such as: monitoring heart rate with the intent of gathering information on the user's state of health and localising users such as it has been already done using smartphones. Because the obfuscation and securing of data of a smart watch only relies on a six number pin code, the code can be cracked simply by applying a brute force approach, leaving private data like text messages that are pushed to the watch to be read by others. This can only be done if the hacker is in close enough proximity to make a Bluetooth connexion with the victim's watch. A possible remedy is to implement Near Field Communication in order to secure data with a more effective pin code, although this would probably raise the price of the device[8].

5.3. Windows on AW smart watch.

Surprisingly enough, Windows 95 seems to run very well on an Android Wear wearable. Corbin Davenport has successfully installed the Microsoft operating system onto his smart watch that was before that running the Android Wear OS. In this environment, the touchscreen works perfectly to navigate between windows. Since Android Wear is very new, this shows possibilities for future unintended but interesting applications and its use as a standalone device. [28].

Figure 5. Windows on Android Wear.

Also, as Windows has its own system WebView, an Android wearable could technically surf the web without the use of an Android handheld device.

6 Near Field Communication (NFC) is a wireless communication technology that uses the ISM-frequencyband

6. GETTING STARTED

In this section we will explain to you the basics for setting up and building your first application for the AW operating system and creating a connection between your handheld device and your wearable. Setting up for debugging does not require a smart watch per se but definitely requires an Android handheld device. The underneath example will walk you through the basics of setting up a "hello world" [29] application for the handheld mobile device and a similar application for the AW smart watch, and shows you how to let them communicate and exchange this "hello world" message.

6.1 Requirements

A comprehensive collection of software packages is needed to program, compile and emulate this example. The necessary software packages are:

- Java Development Kit (JDK)
- Android Studio (AS)
- Android Software Development Kit (SDK) manager
- Android Virtual Device (AVD)

6.2 Setting up

6.2.1 Runtime and Development Kit

The Java Development Kit 7 [30] is needed to compile and run the Android applications. The version of JDK 7 is a minimum requirement. The setup on Mac OS requires extra Java software, namely Java Runtime Environment (JRE) 6 [31].

6.2.2 Integrated Development Environment

AS [32] is an Integrated Development Environment (IDE) for developing on the Android Platform. Alternatively, another IDE such as Eclipse can be used. Eclipse [33] was the de facto standard for Android application development with the Eclipse ADT (Android Developer Tools) plugin [34]. This changed after the release of AS in December 2014 [35]. AS is now the official IDE for Android. Google officially only supports AS since December 2014.

6.2.3 Software Management

Android SDK manager [36] is a tool to download and install all the libraries, drivers, tools and other packages developed by Google that are needed for Android development. For the example in this report the following packages are used:

In the tools section:

- Android SDK Tools
- Android SDK Platform-tools
- Android Build Tools

In the most recent android 5.1.1. (API 22):

- All packages

In the extra section:

- Google play services
- Google USB driver
- Hardware Accelerated Execution Manager (HAXM) installer

6.2.4. Starting a project in AS

Open AS to start a project. For this example you can download the example file [37] and open up the *settings.gradle* file with *open existing project* in AS. Normally, when creating a new project, you get the option to choose a module in the Target Android Devices window. By choosing "Wear" and "Phone & Tablet", two modules are created. As mentioned before, developing for AW requires two modules; one for the mobile device and one for the watch.

6.2.5 Setting up the wearable

A smart watch is not necessarily needed to build the first example application. Instead, the AW smart watch can be emulated with the Android Virtual Device manager [38]. The settings for the virtual device can be found in the figure below.

To get the best performance on the wearable emulator, make sure it runs on the x86 Central Processing Unit's (CPU) with the x86 atom system image. Further, the *use host Graphics Processing Unit (GPU)* is recommended. When running, a demo is shown to learn the basic controls of the smart watch. To obtain the necessary developer options, go to **settings > about** and tap the build number seven times. The developer options are now available in the settings menu to enable the Android Debug Bridge (ADB) debugging.

6.2.6 Setting up the handheld device

Even though a setup with both a smart phone emulator and a AW smart watch emulators is possible, it is not preferable due to problems with the data handling between the emulators. A more reliable setup can be accomplished with an Android smartphone version 4.3 or higher and the emulated smart watch as described in the previous section. The Android smartphone needs to be connected to the computer via USB. The phone should be recognised by computer thanks to the Google USB driver installed by the SDK manager. If not, download and install the vendors phone driver before proceeding. Like the watch, the phone also requires ADB debugging to be turned on. Go to **settings > about** and tap the build number seven times. The developer options are now available in the settings menu to enable the ADB debugging.

6.2.7 Device connection

All the data communication between the two devices is handled by the data layer as discussed in section 2. To pair the devices, download the AW application [39] in the Google Play store. Follow the steps in de AW application on the phone. To forward the AVD's communication port to the connected handheld device a terminal is needed. In the terminal the connected devices can be shown with the follow command:

```
ADB devices
```

Use the follow command to connect the devices:

```
ADB –d forward tcp:5601 tcp:5601
```

In some cases the above commands only work when the terminal prompt is connected to the platform-tools folder where the ADB executable can be found.

6.3 Building the application

Once we have set up our topology and example file we can look at the code. You can see that the two module

branches are situated in the left column, the wear and the mobile module. When you look at the wearable module java code, you can see that it contains two parts: the prior part is de message sender. This is a Java class that is part of the mobile module. This class uses the message API to send a message over the data layer. The ListenerService java class in the wear module functions as a listener on the data layer with the same message API.

6.4 App testing

Select the wear module in AS and choose run. Select the AW created watch in the menu. Then select the mobile module in AS and choose run again and choose the mobile running device. When the device connection is up and running, the AW smart watch should display the message send by the mobile application. In this case: *Hello World!*.

Further, a large amount of tutorials for setting up the connection, sending data and the use of different API's can be found on the internet to get used to developing for Android Wear.

8. FINAL THOUGHTS

Android Wear watches are a step froward for wearable computers offering a platform for easy usage and development of applications. As capabilities for implementing more hardware on the small devices grow so will the potential for their usefulness either as a add-on to handheld devices or, perhaps as standalone units. The funnelling of development through conventions such as the Android Interface both limits the freedom for developing out of the box applications and ensures a foundation for interconnectivity of software that will most certainly play a significant role in upcoming Internet of Things technologies. Smart watches are still in the infant stage of their existence and interest from developers has not yet peaked. In the future we can expect more applications for the devices, perhaps some that utilise their convenient nature and connectivity in new and interesting ways.

9. REFERENCES

[1] Tanenbaum, A. S. (2003). *Computer networks*, 4-th edition. ed: Prentice Hall.

[2] Bonetto, R., Bui, N., Lakkundi, V., Olivereau, A., Serbanati, A., & Rossi, M. (2012). Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a* (pp. 1-7). IEEE.

[3] Rawassizadeh, R., Price, B. A., & Petre, M. (2014). Wearables: has the age of smartwatches finally arrived?. *Communications of the ACM*, 58(1), 45-47.

[4] Raphael, J. (2015) http://www.computerworld.com/article/2919013/android/android-wear-on-wi-fi-using-a-smartwatch-without-a-phone-nearby.html

[5] Silva, V. (2012). *Pro Android Games*. Apress.

[6] Lamkin, P. (2015) "Smartwatch timeline: The devices that paved the way for the Apple Watch" http://www.wareable.com/smartwatches/smartwatch-timeline-history-watches

[7] Nos.nl (2015) http://nos.nl/artikel/2028508-nederlandse-bluetoothuitvinder-komt-in-hall-of-fame.html

[8] Lamkin, P. (2014)"Android Wear hack alert signalled by security expert" http://www.wareable.com/android-wear/android-wear-hack-alert-signalled-by-security-expert-581

[9] Android Website. http://developer.android.com/training/wearables/data-layer/index.html

[10] Pierce, D. (2015) http://www.wired.com/2015/04/android-wear-wifi-emoji/

[11] Lind, J. (2015) http://www.oddhill.se/blogg/lurch-and-android-wear

[12] StackOverflow http://stackoverflow.com/questions/24717538/does-android-wear-support-httpurlconnection-getting-eofexception

[13] Android Website https://developer.android.com/training/wearables/apps/packaging.html

[14] Hahn, M. (2015) http://android-wear-docs.readthedocs.org/en/latest/sync.html

[15] Andrusiv, O. (2014) http://elekslabs.com/2014/11/wearable-runtime-connectivity-architectures.html

[16] Android Website https://developer.android.com/reference/com/google/android/gms/wearable.html

[17] (2015) "Data Exchange and Sync" https://alchemiasoft.wordpress.com/2015/02/26/data-exchange-and-sync/

[18] Forrest, C. (2015) "Android Wear smart watches: The benefits for professionals" http://www.techrepublic.com/article/android-wear-smartwatches-the-benefits-for-professionals/

[19] Cunningham, A. (2014) "Android Wear hardware review: Sometimes promising, often frustrating" http://arstechnica.com/gadgets/2014/06/reviewing-android-wears-first-watches-sometimes-promising-often-frustrating/

[20] Android Website http://source.android.com/compatibility/

[21] Hamblen, M. (2014) "The 12 pros and cons of a cellular smartwatch" http://www.computerworld.com/article/2488893/mobile-wireless/the-12-pros-and-cons-of-a-cellular-smartwatch.html

[22] Jamardo, I. (2015) http://blog.ludei.com/cocoonjs-enables-android-wear-html5-app-development/

[23] Github "HtmlView" https://github.com/kobjects/htmlview

[24] Beavis, G. & Rogerson, J. (2015) “Android Wear: everything you need to know” http://www.techradar.com/news/wearables/google-android-wear-what-you-need-to-know-1235025/2

[25] Hall, S. (2015) “Developer builds a diabetes app for continuous blood glucose monitoring on Android Wear” http://9to5google.com/2015/05/05/diabetes-android-wear-blood-glucose-nightwatch-xdrip/

[26] Sage, S. (2014) “Google Maps updated for Android Wear” http://www.androidcentral.com/google-maps-updated-android-wear

[27] Sullivan, M. (2014) “These 5 smartwatch uses will make you stop loving your smartphone” http://venturebeat.com/2014/11/21/these-5-smartwatch-functions-will-start-the-fall-of-the-smartphone/

[28] Popa, B. (2014) “Android Wear buyer deploys Windows 95 on his device” http://news.softpedia.com/news/Windows-95-on-Android-Wear-Makes-Your-Wrist-Look-Retro-461087.shtml#sgal_0

[29] Wikipedia http://en.wikipedia.org/wiki/%22Hello,_World!%22_program

[30] http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

[31] Java Website https://java.com/en/download/

[32] Android Website https://developer.android.com/sdk/index.html

[33] Android Website http://developers.google.com/eclipse/docs/getting_started

[34] Android Website http://developer.android.com/tools/help/adt.html

[35] Android Website http://tools.android.com/recent/androidstudio1rc1_releasecandidate1released

[36] Android Website http://developer.android.com/sdk/index.html

[37] Github https://github.com/LarkspurCA/WearableMessage

[38] Android Website http://developer.android.com/tools/help/avd-manager.html

[39] Google Play Store https://play.google.com/store/apps/details?id=com.google.android.wearable.app