TECHNOLOGICAL MASHUPS - building HiFi wearables

Cuartielles D., Göransson A.., Olsson T.
K3 – Malmo University
SE – 20506 MALMO, Sweden
david.cuartielles@mah.se
andreas.goransson@mah.se
tony.olsson@mah.se

Stenslie S.
School of Architecture and Design
Maridalsveien 29, 0175 Oslo – N
stalsten@aho.no

Sjunnesson D.
1scale1 HB
Stora Varvsgatan 13
SE – 21119 MALMO, Sweden
d.sjunnesson@1scale1.com

ABSTRACT
The state of the art in digital technologies allows for tools to help prototyping interactive artifacts much faster than ever before. Even if many of those might not be ready for entering the everyday life, they become relevant pieces within the art and design fields. This paper explores the creation of wearable artifacts including digital intelligence with the ability of getting/serving information feeds from/to the internet and bringing them to live as haptic feedback patterns on wearables.

We hereby present a way to quickly deploy wearable sensor networks that will either give physical feedback to the user or broadcast that information to a remote location. We will focus in where to host the intelligence of the system, and how to implement the communication between the different devices in our suggested design solution.

This technological mash-up of several hardware and software parts, can be used to create everything from art pieces to medical devices. The systems should be able of operating by themselves but also give control to external flows of commands.

Topic and Subject Descriptors
D.3.3 [**Wearable HiFi Prototypes**]: Sketching Interactive Systems – Prototyping Techniques, Interactivity Design: Software and Hardware Tools, Embodiment, Open Source Tools.

Keywords
HiFi prototyping, Open Source Software, Open Source Hardware.

1. INTRODUCTION
During the last years our research has focused on the creation of high-fidelity prototypes. The artifacts we create range from wearable sculptures to solar powered handbags. The aim behind these objects is not as much to emulate the real functionality of a potential everyday life device as analyzing the experience of having this object. In other words, we research how to prototype the user experience by means of interactive objects that resemble real life ones, or that could eventually become everyday objects.

We have been studying existing prototyping platforms and we have concluded that the best alternative to create wearble interactive objects consists in a mash-up of a series of open source tools: an Android mobile platform, an Arduino board, and a piece of Java software. We will release all of our tools in the public domain, and we want this paper to be an explanation of the system, the needs that triggered its design and the decisions we took.

1.1 Definition: Low-Fidelity and High-Fidelity Prototypes
Low-Fidelity (LoFi) prototypes are normally understood as limited in their functionality and interaction possibilities. They are constructed to illustrate concepts and often designed or laid out with inexpensive materials and within strict time constraints. LoFi prototypes are not intended to show how an artifact is intended to operate in detail [1].

In contrast High Fidelity (HiFi) prototypes have a high degree of finishing. They present a lot of the intended functionality and overall look-and-feel. Unlike LoFi these prototypes often require additional skills and materials [2]. The advantage of HiFi prototypes is that they can give an idea of how the final artifact would be in a more realistic manner in terms of design and interaction patterns.

2. OUR VIEW ON WEARABLES
Historically most wearable computing systems have been HiFi prototypes. There is a long list of technically advanced devices offering novel ways to interact with digital machines. For many, the first true wearable computer was a shoe-based system developed by researchers and scientists in California in the 1970s [3].

The purpose of this shoe was to aid people in gambling on roulette at casinos. This system was fully functional but it was also entirely embedded inside a normal shoe.

2.1 Trying means embracing
Steve Mann, known for his work both in the arts and the engineering fields, thinks that the ubiquitousness of wearable system has played a minor role in his definition of wearable computing (or WearComp) [4]. Back in the early 1980s he developed his first wearable computer. He has been wearing it, and its subsequent versions, ever since.

From the first design iterations he worked with HiFi prototypes in order to achieve high degrees of functionality. One of his initial ideas was based on an issue he found as photographer. He was constantly missing the moment of a "good picture" because of

having to bring the camera out of its bag. Once he was ready to take the picture the moment had already passed. To test his idea that a wearable camera could solve this situation he had to implement the functionality of a camera in his wearable system. The device was constantly taking pictures in every direction he was looking to and storing them [4].

2.2 Reshaping the form factor

The need for HiFi prototyping of wearables doesn't come from its potential technological novelty. On the contrary most wearable systems are a combination of old technologies in a new form factor. But by making technology wearable we can provide the wearer with information that was previously unavailable [5].

In later years wearable computing has become an extension of ubiquitous computing. This post-desktop user-centric paradigm of

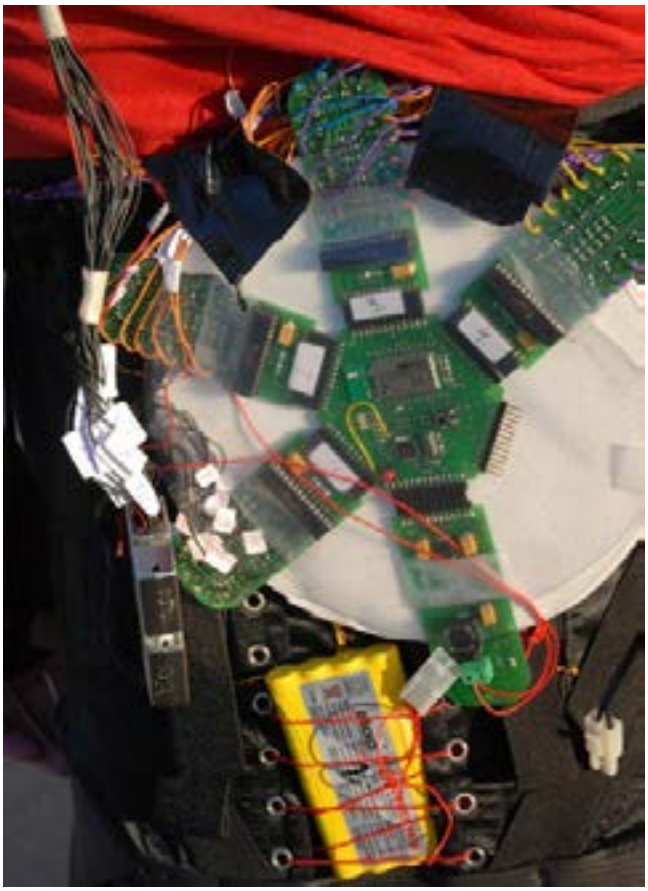

Illustration 1: HiFi Prototype for the Psychoplastic Project

human computer interaction focuses on embedding computational power seamlessly in every day objects [6]. York also refers to it as machine fitting into the human environment [7].

For wearable computer prototypes this forces a higher degree of finishing. For Mann this was imposed by a social factor. While wearing earlier versions of his system other people would treat him differently and he could not conduct his life as he normally would [4]. So he had to embed his system into everyday objects that people would normally wear.

The development of wearable systems has the potential to provide new intimate forms of interactions when they exist within a wearers space but the only way to test the interaction is to provide the wearer with a HiFi prototype.

An example of this can be seen at the Psychoplastic Project shown in Illustration 1. It is a vest that provides users with auditive and physical feedback in the form of 3-D sound and vibration patterns. It carries 64 motors and is controlled by an out-of-the-shelf mobile phone. As mentioned on the author's project report [8]:

The suit imprints stories about corporal ecstasy. The touch based bodysuit renders the stories physical. So the experience becomes a real, personal and intimate play with ones' own body and identity.

As seen in the image, the object is at a prototype stage, and it was used at several art venues, but it required a high degree of finalization and craftmanship. The technology involved in that object was not innovative in technological terms, it just required to be reshaped to fit the purpose. It inspired us to think about which are the main component blocks needed to be part of a prototyping platform for high-fidelity wearables.

3. A HiFi PROTOTYPING PLATFORM

In this document we suggest a platform to quickly prototype wearable computing devices. This object is made out of the combination of different existing technologies, it is a hybrid or, how we like to call it, a mash-up.

It is our aim to provide designers and artists interested in creating pieces that require embodied interaction with a toolbox to do so. We understand wearable as a combination of hardware and software. When prototyping with digital technology, there is always the question about where the intelligence of the system should be located.

3.1 The System

Our system is made of a series of blocks. The wearable itself is made of two hardware pieces: an Android smartphone[1] and an Arduino microcontroller board [9]. The board runs as a peripheral to the phone using a communication protocol we have created for the purpose.

The phone carries all the system's intelligence, it can use its internal sensors (like accelerometer, touch interface, compass, buttons, GPS[2]...) or the ones attached to the microcontroller as a way to trigger events. It will make decisions based on thresholds, thus programmed comparison values.

The microcontroller board has a double function. It provides an input for sensors not present on the phone like a distance sensor, a potentiometer or a gyroscope. It also allows controlling outputs like e.g. motors.

One issue is how to map the inputs (events) to the outputs. That is made on the phone's user interface. Yet another issue is how to program sequences of outputs, what we call patterns. In order to create output patterns we have created a Java tool that programs them in a computer and sends them straight to the phone (the wearable).

In this way, a designer willing to create a prototype, like e.g. a jacket that will guide people to a location when a button is pressed, can use a computer to plan the patterns and the location

[1] Smartphone: mobile phone with extended capabilities

[2] GPS: Global Positioning System

based feedback, and upload the patterns, maps and thresholds to the phone. From then and on, the wearable will be ready to run the experience.

It is pretty obvious that a device aimed to a general audience would be made in a different way, looking into ways to optimize cost, integrating everything into a single unit, etc. The aim of this toolkit is to prototype experiences using wearable technology as the mediator.

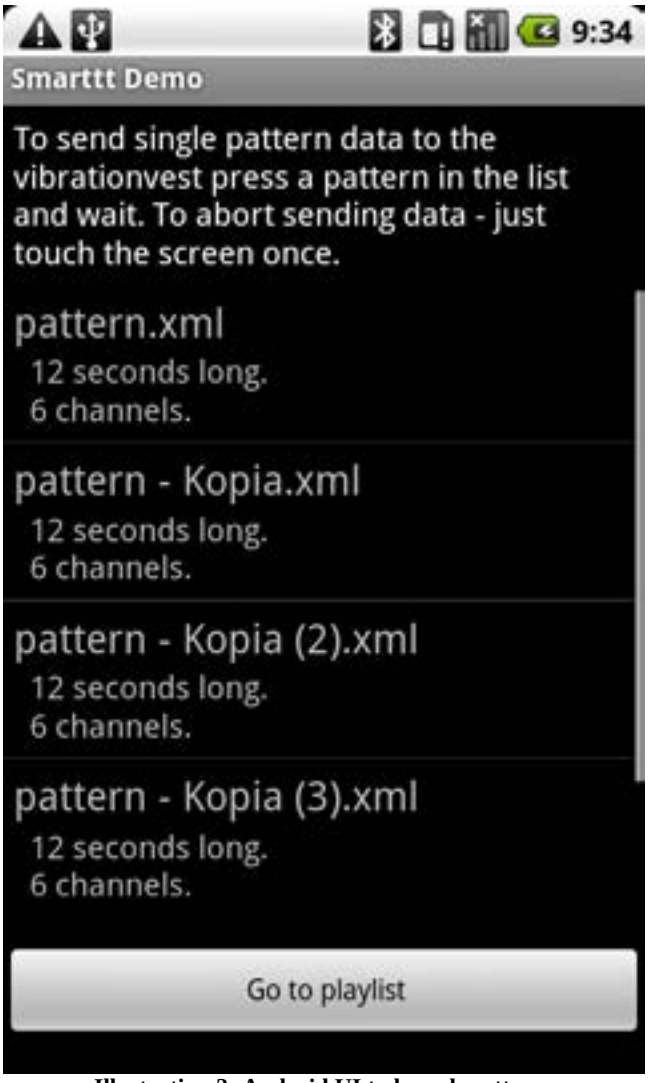

Illustration 2: Android UI to launch patterns

3.2 Why Android

Currently there are five main mobile platforms, understood as the operating systems (OS) running the devices: Symbian from Nokia, iOS from Apple, Android from the Open Handset Alliance, RIM from Research In Motion, and Windows 7 Mobile from Microsoft.

At the time of writing this paper, Windows 7 Mobile was too young for us to have tried it out. It isn't easy to have an opinion on a platform you haven't tested.

Symbian, on the other hand, happens to be the OS that is present on the biggest amount of mobile devices, however the diversity of the hardware makes it hard to port the technology between devices and its development tools' future is uncertain [10].

RIM's OS is not very friendly when it comes to developing applications, besides the ecology of devices offered by it is not very wide, which limits the possibility of using it in prototyping.

Therefore the discussion for us, like for many other designers, went into which was the best development platform: the one provided by Apple [11] (creator of the iOS, the OS for the iPhone, iPod, and iPad devices) or the one provided the Open Handset Alliance [12] and the Java Community (Android is an adaptation of the open source technology made by the Java Community to the mobile world).

We decided on Android because of the portability between devices. This portability means that we could create an application for a certain phone and make it run on a different one without too much trouble. But also that it should be easy to port it between different screen form factors: like taking an application made for a phone and make it work in a tablet format.

On the other hand, the limitation in terms of installation of the software made for the iDevices makes it really hard to consider Apple's OS as a development platform, mostly because of the way applications are distributed to final users [13].

In our system we are using an Android phone as a way to connect our wearable piece to the network, but also to be able of gathering data from sensor technology already available on that specific device like acceleration, location, the arrival of an SMS, images, etc. The phone can be used to collect and send data, but also to act as the brain of our wearable device.

We have created a software package that allows activating a series of patterns on the actuators of our wearable platform. The program offers a user interface where to enable certain output patterns based on the arrival of a certain event. The event is triggered either by the readings of a sensor within the wearable, or by the readings of the sensors on the phone. Again, Android is extremely designer friendly to this extent, since other systems won't allow triggering events on e.g. a phone call or any other phone-related operations. iOS' development documents are very specific when it comes to the way designers can make use of the core-functions of the device [13].

The first version of the software is looking at two types of events; the location based ones and the sensor based ones. An example of a sensor triggered event consists in detecting a button press or whether a knob's reading has reached a certain value. An example of a location triggered event consists in looking at the phone's GPS information and detecting a location previously configured as a target. This is how we solved the Psychoplastics Project prototype technically[8].

Future iterations of the software will be including more of the different options available in different devices: arrival of calls or SMS, use of the accelerometer, detection of light levels, use of the multitouch screen, etc. It is possible to modify it to be triggered by potentially anything, however we focus in the ease of use and configurability of the software as a way to make prototypes, therefore we have decided to compromise on some of the potential features.

3.3 Why an Arduino compatible I/O board

On the hardware side of our wearable platform, we have decided to use an Arduino [9] compatible microcontroller board which is a

derivative of the Arduino Bluetooth board[3]. The license agreement [14] on the Arduino boards allows doing this. It makes it very easy to test our sensors and actuators on a normal Arduino[4] board and later migrate all the code to its bluetooth equivalent that will wirelessly communicate with the phone.

Illustration 3: HiFi Toolkit version 1

We should mention that our initial design looked at a feature that Android phones offer of using the USB data port as a serial port. However there is no standard in the way this can be made, nor it is available at all devices without having to either root[5] the phone or make a complex update to the phone's firmware. We realized that most of the Android devices are equipped with a bluetooth communications port and that the communication through it to and from an Arduino BT board was easy to implement.

We created a prototype on this platform in the form of a so-called Arduino shield, this is a board plugged on top of an Arduino, that allows to easily plug in/out wires to -in this case- motors to give force feedback to users. We wrote a piece of code for the Arduino board (firmware) that gets it to read data from the phone over a bluetooth connection at very high speeds. We reach up to 5 updates of the actuators per second, but the theoretical limit is higher. This allows for very accurate control of the motors, but also any other devices we could be interested in using like LEDs, peltier elements, etc.

Our second prototype is a self-made circuit board integrating both the Arduino BT board and the shield in one single piece. This platform counts with 6 inputs for analog sensors and 6 outputs able of sending out analog signals to devices. This is an Arduino compatible board[6] that communicates with the phone in the same way the previous prototype did, it is just smaller and easier to embed in the wearable piece. It gives away pins available in the microcontroller platform, but trades them for pins that are fully compatible with each other at a functional level.

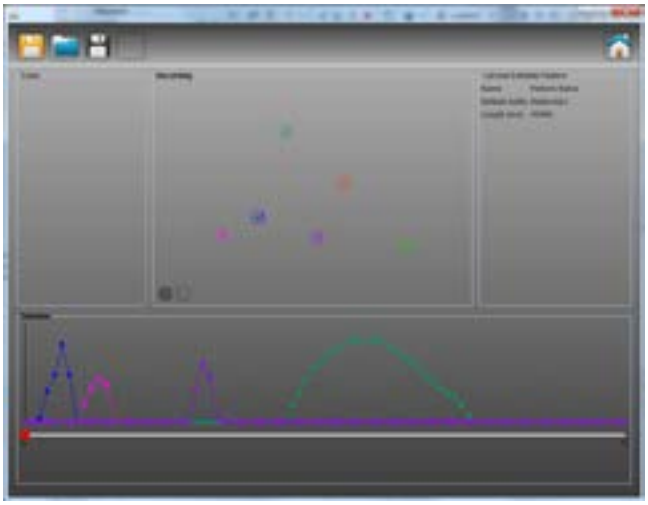

Illustration 4: GUI to the pattern editor on a computer

3.4 The Glue: a Protocol

The key idea in this prototyping tool is that designers, our users, will look at the combination phone+microcontroller as a single piece of hardware. For real it is a distributed computer with two processors: the phone has its own, and the microcontroller platform also has its own.

The glue between both parts is a communication protocol to exchange information between them at high speeds. There is no embedded intelligence inside the microcontroller board beyond the exchange of data with the phone and the ability to turn off all the outputs if there was no signal coming from the phone for some time (as a safety measure).

There are other protocols to reach a similar functionality to this one like Firmata [15], a protocol to exchange data between an Arduino board and a computer application, and Amarino [16], a whole platform to link an Arduino BT board with an Android device over a bluetooth connection. We tried both and none of them were giving us either the speed, or the safety we needed. Therefore we decided to implement our own protocol focusing on those two aspects.

Reaching high data transfers is important in our case since we understand that an event can trigger a whole series of actions on the actuators that can last for a certain amount of time. Since the microcontroller platform of our choice doesn't count with very much memory for data, we decided that it should not be storing any data besides the current state of all the inputs and outputs. Therefore any changes to be made on an output must come over the bluetooth connection instantly.

When it comes to the safety, the best is to explain this concept through an example. Imagine we connected a motor to give force feedback to a user and that the phone -for whatever reason- broke or ran out of batteries after some time while the motor was running. There would be no way to stop the motor besides

[3] Arduino Bluetooth or Arduino BT: microcontroller board from the Arduino brand that communicates over a bluetooth port to other devices in a wireless fashion.

[4] A “normal” Arduino board is a microcontroller board that communicates over USB to a computer. Since current computers have USB ports as a standard communication method, the Arduino USB board can literally be plugged to any computer for testing and prototyping.

[5] To root a phone: technical term that refers to the technique of acquiring the ability to literally install any software package on the device. Usually this feature is disabled by the manufacturer or the carrier selling the device. Rooting a phone implies voiding the warranty and this is something we wanted to avoid.

[6] Arduino compatible board: a microcontroller board created as a derivative design departing from the design file of an Arduino branded board.

pressing the reset button on our wearable device. But imagine the user is carrying the phone in a pocket and is not looking at it. There is no way he/she can know the phone is not working. This is why we went for implementing a watchdog[7] safety mechanism to stop the device without provoking any harm or a failure.

Our communication protocol implements all sort of redundancy and error checking mechanisms to make it safe without compromising the speed. All of these are made in a transparent way to the user. He/she will just see a wearable device that will react as programmed and if it stopped working it will be as a whole, or at least react in a way that could easily be understood by the user.

3.5 The Java Software

As explained earlier, we foresee that many experienced designers will be interested in the creation of feedback patterns in the form of vibrations, light shows, or temperature changes. We have therefore created a piece of software in Java to interactively create patterns for the different actuators. As for version 1 of this software, it lets the users program the 6 outputs available on our shield and our board and to export them directly to the phone acting as controller.

The patterns are stored as XML[8] files inside the phone's application, which means that it should be possible to edit them with a text editor as well or to create whatever other application to fulfill the same purpose as ours. Also the phone allows selecting which pattern should be activated for each event.

At this point, this part of the programming is still handled by the phone's software, it could of course be made on the computer side. Our vision is that we will integrate all of the tools on the phone in order to be able of doing the programming of the interaction while carrying the wearable on site. Because of Android being a Java based language, porting the application from the computer to the phone is theoretically easy.

4. CONCLUSION

This paper defends the need to create HiFi prototypes in fields where embodied interaction will be put to test. There are some fields within research where their novelty makes it hard for users to imagine the functionality and therefore traditional interaction design techniques like paper prototyping make it hard to understand the implications behind using the device being created.

Wearable computing, the field where the computer gets dismantled and attached to the body, is a field that invites using HiFi prototyping as a tool to illustrate concepts.

The hereby proposed toolkit allows for quickly creating and trying out wearable artifacts that can respond with patterns to events triggered by any kind of sensor. In order to keep the complexity of programming the wearable as low as possible, we present a series of tools that will allow going from idea to HiFi prototype in a very short time. Also many designs can be created without any knowledge in electronics or software, just by dealing with interaction design concepts like patterns (event sequencing), or sensor mapping via a simplified graphical user interface.

We are also strong believers in open source, and therefore besides the scope of this publication, we offer all the source code and schematics needed to replicate our experiments.

5. ACKNOWLEDGMENTS

Our thanks to G. Martino for his kind contribution in the form of materials and to the whole Arduino community for their efforts making open source tools.

6. REFERENCES

[1] S. Isensee, J. Rud, K. Stern 1996. Low vs High Fidelity prototyping debate, Interactions, Volume 3 Issue 1, Jan. 1996

[2] J. Lawson, M. Coterot, C. Carincotte, B. Macq 2010. Component-Based High Fidelity Interactive Prototyping of Post-WIMP Interactions, Proceeding ICMI-MLMI '10 International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction.

[3] S. Mann 1996. eSmart Clothing: The Shift to Wearable Computing, Communications of the ACM, August Vol. 39, No. 8.

[4] S. Mann, H. Niedzviecki 2001. CYBORG, Destiny and Human Possibility in the Age of the Wearable Computer, Doubleday Canada.

[5] A. Marion, E. Heinsen, R. Chin, B. Helmso 1997. Wrist instrument opens new dimension in personal information, Hewlett-Packard Journal, December 1977

[6] M. Weiser 1991. The Computer for the 21st Century, Scientific American, September 1991.

[7] York, J., Pendharkar, P.C., 2004. Human-computer interaction issues for mobile computing in a variable work context. International Journal of Human-Computer Studies 60, 771-797.

[8] S. Stenslie 2011. Psychoplastics project, as documented on 2011-01-07, http://psychoplastics.wordpress.com .

[9] Arduino Project 2011. An Open Source Hardware Platform for Prototyping, as seen on 2011-01-07, http://www.arduino.cc .

[10] Symbian Foundation 2010. Foundation's website report on the future of the Symbian platform, as seen 2011-01-07, http://blog.symbian.org/2010/12/17/symbian-foundation-is-completing-its-transition-to-a-licensing-body .

[11] Apple Inc. 2011. Apple iOS Developmer Program, as seen 2011-01-07, http://developer.apple.com/programs/ios .

[12] Open Handset Alliance 2011. Android Software Development Kit, as seen 2011-01-07, http://www.openhandsetalliance.com .

[13] Electronic Frontier Foundation 2010. iPhone Developer Program License Agreement, as seen 2011-01-07, http://www.eff.org/files/20100127_iphone_dev_agr.pdf .

[14] Arduino Project 2011. Explanation on the License Agreement, as seen on 2011-01-07, http://www.arduino.cc/en/Main/FAQ .

[15] Firmata Protocol Project 2011. Main website to the poject, as seen on 2011-01-07, http://www.firmata.org .

[16] Amarino Toolkit Project 2011. Main website to the project, as seen on 2011-01-07, http://www.amarino-toolkit.net .

[7] Watchdog: technical term for a time based monitoring device.

[8] XML: eXtended Markup Language, text- and tag-based language used to sort data inside documents.