UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Tanel Tähepõld

Context-aware Mobile Games using Android, Arduino and HTML5

Bachelor Thesis (6 ECTS)

Supervisor: Satish Narayana Srirama, PhD
Co-supervisor: Huber Flores, MSc

Author:..... ".....” May 2012

Supervisor:..... ".....” May 2012

Co-supervisor:..... ".....” May 2012

Professor:..... ".....” May 2012

Tartu 2012

Abstract

Latest technological achievements in mobile and open-source electronics platforms made it possible to develop pervasive applications that use environmental information to enhance software usability aspects in real-time, like in the case of context-aware mobile games. However, the development of this kind of pervasive applications is tied to specific aspects owned by each mobile platform (e.g. programming language, SDK and tools, etc.). Moreover, a considerable effort and knowledge in low-level programming techniques is required for porting the applications between platforms, and thus in general most of the solutions are targeted at particular platform. In order to investigate the possibility of creating portable pervasive applications that combine sensor information from the multiple micromechanical artefacts embedded within the smartphones, we used contextual sensor data provided by Arduino Microcontroller. The current thesis proposed extending the existing implementation of PhoneGap to create hybrid mobile applications based on HTML5 that are easy to port, maintain and reuse.

Contents

List of Figures

1

Introduction

1.1 Introduction

The rapid proliferation of smartphones is fostering the development of mobile applications, most of them focus on games (e.g. the Android (1) Market changed to the Play store). Mobile game applications are becoming more sophisticated as they are starting to consider environmental information. For example they are adapting the game to the situation, location and context of the user (aka context-aware games). Thus, combining virtual reality with real environmental data for improving the perception of the player during playing.

Smartphones can sense the environment as they are equipped with a variety of sensors such as the accelerometer, magnetic field, global positioning system (GPS) (2), etc. The information gathered through the sensors can be accessed and embedded within a native mobile application, as it is developed and executed on the top of mobile resources.

On the other hand, Arduino (3) is an open source electronic project that is looking towards providing a platform that can be used for extending the sensing capabilities of smartphones with context information. Consequently, Arduino has released a microprocessor called Arduino Mega ADK which is specific for the Android platform and allows mobile applications to request data from artefacts such as sensors, motors, etc., located in the environment so that the information can be used to improve some usability aspects in mobile applications (e.g. adapting screen resolution, brightness, etc.).

However, with the introduction of new Web technologies such as HTML5 (4) for developing cross-browser mobile applications, it becomes more difficult to access the physical resources of the handset and even more complex to use data from external resources such as Arduino Microcontroller, because the HTML5 application is executed within browser component and it cannot access the device native APIs.

This thesis aims to explore the facilities of creating a hybrid mobile application (5) that uses Web technologies and native properties of the Android mobile platform.

1.1.1 Motivation

The provisioning of context-aware services for mobile game applications is not a new idea and multiple researches have been conducted. For instance, Real Tournament (6) is a context-aware mobile game that combines real world and virtual reality components as it uses GPS and electronic compass data from the players hand held. Similarly, Human Pacman (7), is a mobile wide-area game, where the players adapt the role of the Pacman and the Ghosts characters (using wearable computers and hand held devices). However, with the introduction of new technological achievements in mobile technologies (e.g. better transmission achieved with 3G/4G, Microprocessors adapted to the mobile specifications etc.), it is more easy to narrow the gap between the virtual games and real world. Furthermore, current technologies such as HTML5 for developing cross-browser mobile applications enable to create pervasive games that can be deployed on multiple mobile platforms. Thus, mobile games that use context information are not tied to specific designs or ad-hoc approaches such as the already mentioned ones.

In this work, our goal is to create a pervasive game that uses sensor data (8) from local and external resources and requires a low effort in portability (it can be deployed in multiple mobile platforms).

1.1.2 Contributions

Hybrid application development for mobile platforms is emerging as a prominent trend in the mobile domain, because it reduces the effort of porting applications between different mobile platforms. For instance, by using cross-platform tools such as PhoneGap (9), the developer can create a core application based on Web technologies and then wrap it to a specific mobile vendor (e.g. iOS, Android, Windows Phone, etc.).

The current version of PhoneGap supports accession of multiple sensors embedded within the mobile resources. However, it does not support consumption of data from external sensor sources such as Arduino.

To counter this problem, we have extended PhoneGap library with a module that can be used to access data sent by the Arduino Microcontroller (specific for each mobile platform). Moreover, as a case study, a context-aware mobile game is presented. Among others the application uses information from embedded sensors, such as accelerometer, touch screen etc. Furthermore, it gathers sensor information from the thermistor sensor of Arduino in order to change the background of the game according to the temperature of the environment to which the user is exposed. Thus, enriching the experience of the player.

The application is analysed in order to show the potential of creating HTML5 pervasive applications that can be ported between different mobile platforms. The results are discussed in further sections. The rest of the thesis is structured as follows.

1.1.3 Outline

Chapter 1: Introduces a general overview of the current mobile technologies and its embedded micro-mechanical artefacts. It presents the current state of pervasive game applications and it emphasises the use of two technologies - Android and Arduino.
Chapter 2: Describes the main research question that is tackled by this thesis.
Chapter 3: Presents the proposed solution along with implementation and technical details.
Chapter 4: Introduces the context-aware game Fuzed as a case study and discusses the evaluation results of the game.

2

State of the Art

Latest technological achievements in mobile and open-source electronics platforms, it makes possible the development of pervasive applications that uses contextual information for enhancing software usability aspects in real-time. In this chapter, we present these advances and pointed out several drawbacks still unsolved in the designing and the development of context-aware games. Finally, we described multiple works that have been proposed by many authors in order to counter different issues in the field.

2.1 Mobile Devices

A mobile device is a computational unit that fits into the palm of the user and it allows performing a variety of operations such as store contacts, call, etc. Generally, a mobile device has a touch input and also may have a miniature keyboard to introduce commands for managing its resources. Mobile devices are usually used in situations where normal size laptops and notebooks are uncomfortable and impractical to carry. Furthermore, mobile devices can extend certain functionality (e.g. positioning, etc.) by integrating elements such as such as RFID, smart card and barcode.

2.1.1 Smartphones

A smartphone is a device that extends the capabilities of mobile phone by adapting a higher application layer that enables managing any artefact attached to the mobile resources. Modern smartphones usually have high-end touch screens and high-speed data access via Wi-Fi or mobile broadband (e.g. 3G/4G). Furthermore, smartphones are

equipped with embedded cameras and micro-mechanical artefacts such as accelerometer sensor, proximity sensor, gyroscope sensor, compass and global positioning system among others. The number and type of sensors may vary depending on the manufacturer and the model of the device itself. For example, the gyroscope is a fairly new addition that is only available for a few handsets such as Samsung Galaxy S2 and Apple's iPhone 4S.

Smartphones have specially tailored operation systems and current market is divided between five main mobile operating system vendors, Apple's iOS, Google's Android, Microsoft's Windows Phone, Nokia's Symbian, RIM's BlackBerry OS, and embedded Linux distributions such as Maemo and MeeGo. Proprietary vendors such as Apple, Windows, etc. are technological close, and thus they developed mobile software just for their products.

On the other hand Googles Android is an open-source platform that it is used in devices from different manufacturers such as HTC, Samsung and Sony Ericsson. Furthermore, the development of mobiles application for smartphones is tied to the mobile platform (e.g. programming language) and the use of certain tools (e.g. SDKs, plug-ins, etc.) provided by the mobile vendor. Table 2.1 shows a general overview of the requirements for developing applications for different mobile platforms.

Operating system	**Programming language**	**Tools**
Apple iOS	C, Objective C	Xcode
Google Android	Java (Harmony, Dalvik VM)	Android SDK
RIM BlackBerry	Java (J2ME)	BB Java Eclipse Plug-in
Symbian	C, C++, Python, HTML/CSS/JS	Nokia Qt SDK
Windows Phone 7	C, VB, .NET	Windows Phone Dev Tools
HP Palm webOS	HTML/CSS/JS	HP webOS SDK
MeeGo	C, C++, HTML/CSS/JS	MeeGo SDK
Samsung Bada	C++	Bada SDK and IDE

Table 2.1: Tools and programming languages needed for developing applications for multiple mobile platforms

2.1.1.1 Accelerometer

An accelerometer is a sensing element that measures physical acceleration that is associated to the weight in which it has been embedded (e.g. mobile). The accelerometer measures the acceleration of the device in three different axes: X, Y, and Z. Figure 2.1 shows the different accelerometer readings when person is standing or walking.

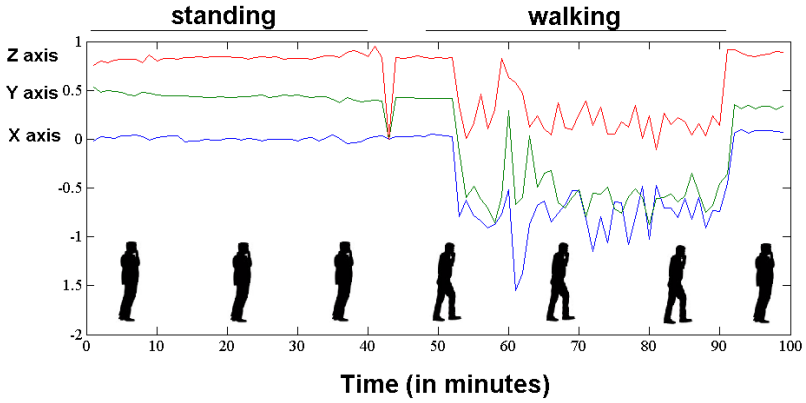

Figure 2.1: Accelerometer Data

2.1.1.2 Gyroscope

A gyroscope is an artefact primarily used for measuring orientation and direction. Gyroscope is based on the principles of angular momentum conservation and is used in navigation systems for ships and airplanes, in wireless computer pointing devices and in gesture recognition systems of the smartphones. The gyroscope structure can be described as a mounted rotor that consists of three freedom degrees(spinning, perpendicular and tilting). These degrees of freedom are obtained by mounting the rotor in two concentrically pivoted rings (inner and outer). The whole assembly is known as the gimbals.

Microelectromechanical gyroscopes are used within the smartphones (e.g. iOS) in combination with the accelerometer in order to sense six axes (left, right, up, down, forward and backward) which help in the process of video stabilization and face detection.

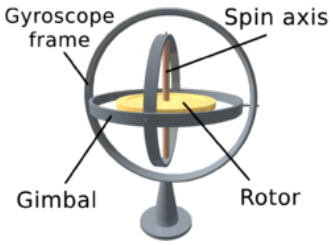

Figure 2.2: Gyroscope
(10)

2.1.1.3 Global Positioning System

Global Positioning System (GPS) is a utility that provides user with positioning, navigation and timing. The system consists of three segments: the space segment, the control segment, and the user segment. The system is maintained by the United States government.

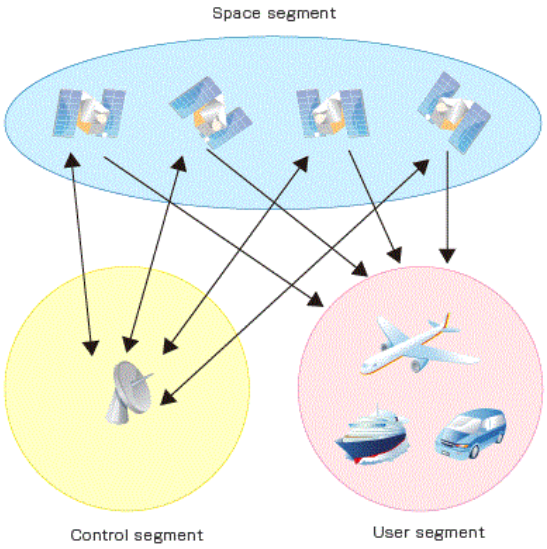

Figure 2.3: GPS Segments
(11)

The GPS space segment consists of a constellation of satellites flying in medium Earth orbit at an altitude of approximately 20,200 km and constantly transmitting radio signals to user. Control segment consists of ground antennas and monitor stations that track flight paths of satellites. It regularly synchronizes the atomic clocks on board

of the satellites to within a few nanoseconds of each other and adjusts the ephemeris of each satellite's internal orbital model. The user segment consists of different devices used by military, civil, commercial and scientific users. Furthermore, GPS is the most common sensor embedded within any smartphone. Although, GPS QoS is dependent of the GPS radio hardware which its use. The QoS can vary due many factors such as clothing, backpacks or car glove box that can interfere with the signal. Furthermore, battery life is also an issue as GPS radios uses a lot of power for establishing the positioning of the device.

2.1.2 Mobile Platforms

Nowadays, mobile platforms are evolving rapidly and opening a new perspective of possibilities for creating rich mobile applications based on sensor information, Web and cloud services. The most two popular mobile platforms are iOS and Android. Applications which are developed for these two platforms follow the distribution model that consists of an online store (App store and Play store) in which the user can search, download and install mobile applications on the fly from the handset.

2.1.2.1 iOS

iOS (12) is a mobile operating system created by Apple Inc. The first version was released in 2005 for iPhone and iPod Touch. Unlike some of the other mobile operating systems, iOS is available only for devices manufactured by Apple.. Furthermore, the applications are allowed to be installed only through the App Store or iTunes. A major disadvantage is that the application development environment is available only through the proprietary operating systems. The iOS is based on Mac OS X, and thus is similar to Unix. iOS provides four abstraction layers for developing mobile applications (Core OS layer, Core Services layer, Media layer, and Cocoa Touch layer).

Leaving aside the complexity involved in the development of iOS applications, Apple have pioneered the modern smartphones, and has a very large contribution to the improvement of mobile technologies.

2.1.2.2 Android

Android is operating system for mobile devices and originally it was developed by Android Inc., but in 2005 it was acquired by Google. While Android is mainly designed

for smartphones and tablets, it is also used effectively for netbooks and e-book readers like Amazons Kindle Fire. The first phone running on Android was the HTC Dream and it was released in 2008. Since then, many mobile phone companies started making phones running on Android, including Samsung, Sony Ericsson and many others. According to Canalys [22] Android became the worlds leading smartphone platform in fourth quarter of 2010 as their market share exceeded 100 million units. Although Android was acquired by Google it stayed open-source project and it is very appealing for developers because of its customizable nature, rich documentation and powerful APIs.

Android core is based on Linux kernel, the application programming interfaces are written in C and it uses Dalvik Virtual Machine for runtime. Android applications are written in Java and package as apk files. The Android application architecture usually contains one or more user screens that are called Activities and these are used for creating the user interactive interface.

2.1.2.3 Android Accessory Mode

The USB accessory mode allows connecting host hardware using USB connection, but the limitation is that the hardware must be specifically designed (compatible) for Android-powered devices. In this process, Android turns normal USB relationship upside-down with the accessory module so that the hardware acts as host and the Android device itself becomes the USB Device. The USB accessory APIs were introduced to the in Android version 3.1, however it is available in Android 2.3.4 using the Google APIs add-on library.

It is possible to establish connection between Arduino microcontroller and Android powered device when using Android Open Accessory Development Kit or other USB Host Shield compatible with Arduino. The most suitable microcontroller is Arduino Mega ADK that is specially developed for working side by side with Android devices.

2.1.2.4 Native Mobile Application

A native mobile application is piece of software that performs a specific task, such as game, calendar, clock etc., and is developed using vendor specific programming languages and tools, such as Objective-C for iOS applications or Java/Dalvik dialect for Android applications. Furthermore, each native application is created for a specific operating system, firmware version, etc.

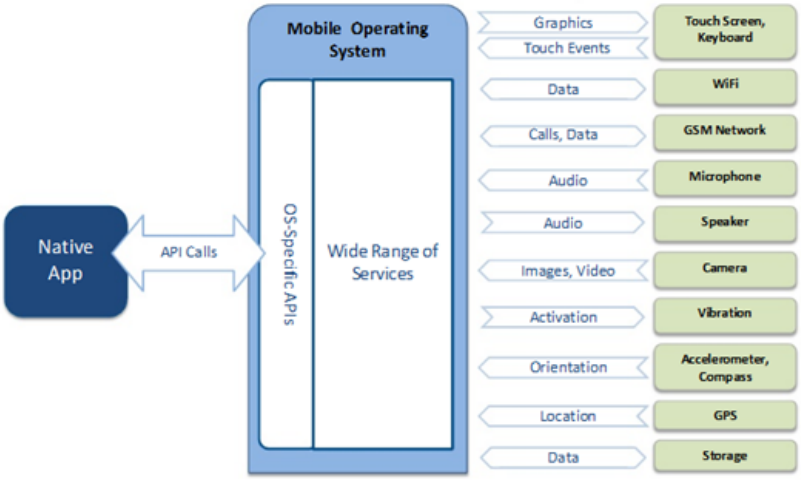

Figure 2.4: Anatomy of Native Application
(13)

2.1.2.5 Hybrid Mobile application

The increasing proliferation of new mobile platforms is increasing the complexity in the mobile development life cycle as each vendor has its specific programming language and tools to develop mobile applications. Thus, making the mobile application development process ad-hoc specific.

Moreover, migrating application between platforms requires a considerable effort and specialized knowledge on multiple programming languages and techniques. However, the use of Web technologies may alleviate the creation of mobile application by offering a common platform that is supported by all the different devices and mobile operating systems.

However, native applications usually lead to a better user experience than Web applications (as shown on figure 2.6), since a native application enables the access to native features like geolocation, camera, and user information (e.g. contact list, etc.)

In contrast, Web applications that run in the mobile browser have certain advantages over the native ones as Web application are cross-platform. In other words, the same code base can be used for creating applications for different mobile platforms. However, applications that run in the browser component are limited and do not have access to vendor specific APIs.

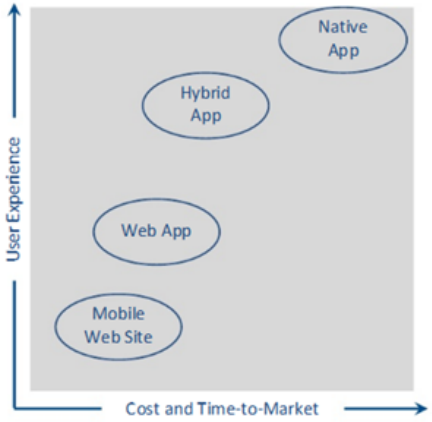

Figure 2.5: User Experience vs. Cost and Time-to-Market (14)

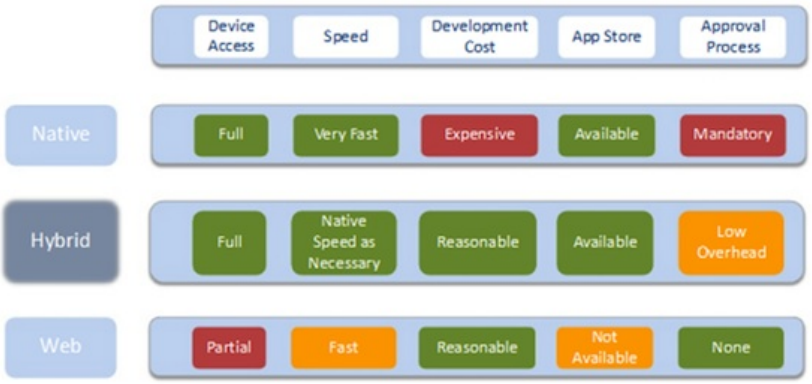

Figure 2.6: Comparison of Native, Hybrid and Web Mobile Applications (15)

On the other hand, an hybrid mobile application combine the different features of both native and Web applications, Thus achieving the reusable code base and the rich user experience properties in the development process. A hybrid application is a part native, part Web application. From the user perspective, a hybrid application is not different from a native one as it is downloaded from the phones marketplace and it is launched just like any other native application in device.

From the developer point of view, the difference is quite obvious as instead of rewriting the application from the scratch, the application may be easily ported with low-effort as HTML5 enables to keep the interfaces, the application logic and other properties (CSS and JavaScript) reusable between different platforms. Furthermore

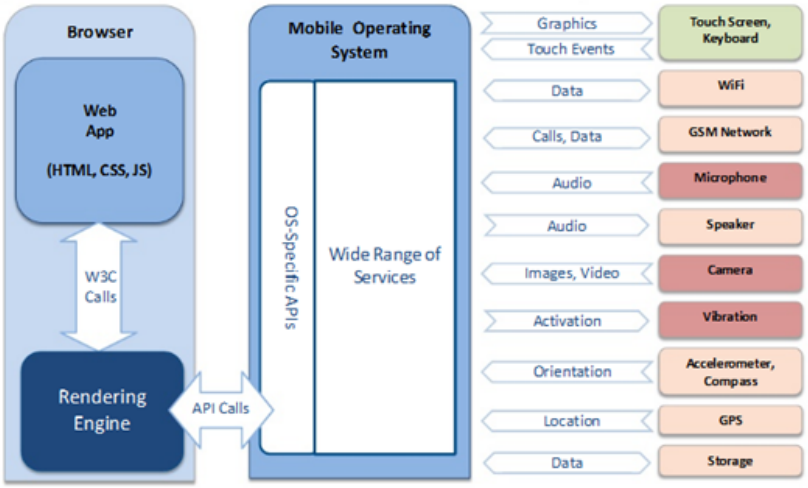

Figure 2.7: Anatomy of a Hybrid Mobile Application (13)

hybrid applications have access to native APIs and other frameworks such as PhoneGap, Worklight (16) and Titanium (17).

The increasing effort of establishing HTML5 as common mobile platform, it can be observed with the release of new frameworks such as Sencha Touch (18) and jQuery Mobile (19) which aims the creation of hybrid applications that are very similar (in properties and in performance) to a native one.

Furthermore, in 2011 Mozilla introduced Boot to Gecko project that will fully enable Web as platform for mobile devices. The goal of this project is to eliminate the need for platform specific development as developers can write everything using Open Web standards, such as HTML5, CSS and JavaScript. Entire project is based on open standards and the source code is accessible for everyone. According to Mozilla they believe that the web can displace proprietary, single-vendor stacks for application development.

2.1.2.6 HTML5

HTML5 is the next generation of HTML and it provides new features that are necessary for modern web applications. It is designed to be cross-platform and it does not need particular operating system rather than a Web browser. One of the main goal of HTML5 is to replace proprietary multimedia plug-ins with open standards that allows

web applications to behave more like native applications. Although adaptation on desktop browsers such as Internet Explorer is slow, the mobile devices are supporting HTML5 standards in record numbers, nearly every smartphone and tablet device sold today supports it and those numbers are growing. HTML5 is designed to be backward compatible with existing web browsers.

In addition to specifying mark-up, HTML5 specifies scripting application programming interfaces, such as canvas element and timed media callback that allow developers to build web-based applications that integrate seamlessly with a users computer. For example the timed media callback allows developers to grab a video or audio element using JavaScript and then change the URL property and effectively load a new media, it is even possible to control when the video loads and mute and unmute the audio. Still there's nothing new in JavaScript enabling this access, it's just the addition of semantically meaningful elements in HTML5 that gives us easier medium to work with and this makes JavaScript a little more powerful. More interesting achievements are the APIs for Geolocation and Web Storage. The Geolocation API introduces an interface to retrieve the geographical location information for a client-side device and is ideal for mobile devices as it allows to create applications with positioning features. The Geolocation API uses different sources for retrieving location information, the most common of them are IP address, Wi-Fi/Bluetooth MAC addresses and radio-frequency identification signals among others. The location is returned with a given accuracy depending on the best location information source available.

Similarly, the Web storage API (aka Local Storage or DOM Storage) allows to store named key/value pairs locally and is implemented natively in web browsers. The Web storage API enables to save data of sessions, temporal transactions, etc. This information is available even after you navigate away from the web site, close your browser tab or exit your browser. This is particularly useful for storing the game data and playing offline. To sum up, HTML5 offers several new and interesting features it will take time before these features are universally supported. Features, such as native audio and video, a programmable canvas area, and geolocation, will lead to the development of exciting new web applications and tools. And the enhanced multimedia and scripting support will allow designers to create a richer web experience, especially for mobile users.

2.1.2.7 Appcelerator Titanium

Titanium is a software development platform that allows creating native applications using web technologies. It analyses, pre-processes, pre-compiles and then compiles the Web code into the native language of the mobile platform that is selected. Titanium produces applications faster and compact than a hybrid application code from the scratch. Titanium was first introduced in 2008 by Appcelerator Inc., and currently latest released version is 2.0.

The Titanium developer community has pointed out that although Titanium is supposed to take care of memory management for you, it sometimes fails and this brings anomalies that result in unexpected application failures and crashes.

2.1.2.8 Worklight

Worklight is mobile application platform that allows to write mobile applications using web technologies, native technologies, or combine both types of technologies in a single app. Worklight is owned by IBM and besides the mobile application platform they also provide entire mobile ecosystem, such as mobile middlewares and management.

Since Worklight is directed more towards enterprise users and clients, then it is available only under commercial licence.

2.1.2.9 PhoneGap

PhoneGap is a HTML5 application platform that allows to use native APIs inside client-side code and it uses standards-based web technologies to bridge web applications and mobile devices. You can easily create cross-platform mobile apps with HTML, CSS and JavaScript for iPhone/iPad, Google Android, Palm, Symbian, BlackBerry, Windows Mobile and more as you write once and deploy same code for all platforms.

The main idea behind PhoneGap consists of wrapping a native container around the web application so that enables to use native API functionality through JavaScript. As seen on figure 2.8 PhoneGap creates additional layer between client-code and native application, since the client-side application is really running inside an embedded browser component it can be used for every platform supported by PhoneGap.

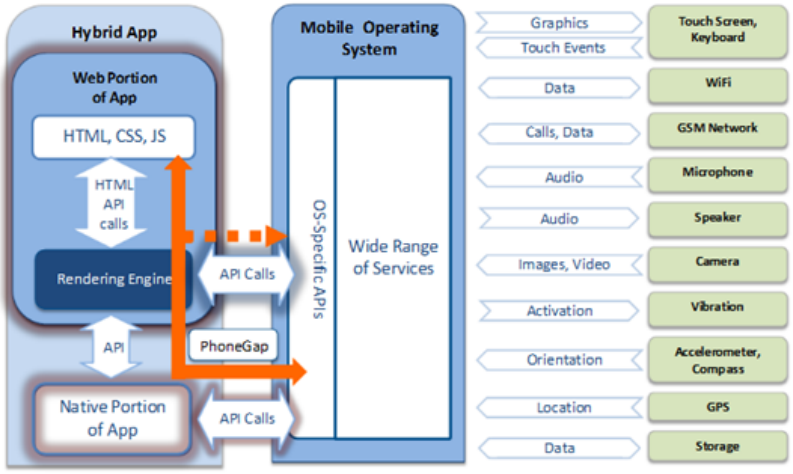

Figure 2.8: Architecture of Hybrid Mobile Application Using PhoneGap Framework (13)

	iPhone	**iPhone 3GS and newer**	Android	Windows Phone 7
Accelerometer	+	+	+	+
Camera	+	+	+	+
Compass	-	+	+	+
Contacts	+	+	+	+
File	+	+	+	+
Geolocation	+	+	+	+
Media	+	+	+	+
Network	+	+	+	+
Notifications	+	+	+	+
Storage	+	+	+	+

Table 2.2: Native Features Supported by PhoneGap Framework (9)

As shown in table 2.2, PhoneGap has out of the box implementation for all of the main native API components of a mobile platform and it allows to create own implementations to bridge native APIs functionality to client-side code.

2.1.2.10 Mobile Applications Using Sensors

At the moment there are many games implemented on Android platform that use device's geographical location for game play. For example, foursquare.com is social-driven location web application that uses GPS for determining user's location. The main idea behind Foursquare is that users can "check in" to places which are visiting in real-time and can share their current location with friends using different social network integrations. When doing a check-in, user's locations is examined and a list of nearby places is shown like a recommendation system. User can select existing place or create a new one. The game aspect of Foursquare is that virtual rewards are offered for check-ins.

On the other hand, several framework have been created that facilities the creation of context-aware games. For instance, FRAP (20) is a framework for constructing games is able to enrich its functionality from the environment. This framework helps to reduce the development time of context-aware game as implements features for tracking location and provisioning of context-aware game tasks. Clients location is tracked using smart phones GPS. The framework assumes that clients have a permanent internet access, otherwise the game fails. FRAP uses cloud based communication server to reduce the communication cost and preserve battery of mobile device based on push-oriented technologies. This means that clients never polls for information form server, but they will receive it whenever they are available.

As another example application from this domain we can present The ARCompass Application (21), that uses devices camera, orientation sensor and HTML5 canvas element to draw compass over captured camera view. The ARCompass Application is a hybrid application and uses native orientation sensor to calculate right direction for compass needle and client-side HTML5 canvas element to draw correct image over captured camera view. The ARCompass Application is currently implemented on Android platform, but as it is hybrid application, it takes less time to port it to another platform, than to rewrite entire application using another vendor specific programming language and development environment.

2.2 Arduino

Arduino is open-source electronics prototyping platform that can sense environment by receiving input from a various sensors and it affects its surroundings by managing accessories and other kind of sophisticated artefacts (e.g. motors, leds, etc.). According to the manufacturer it has easy to use hardware and software as it is intended for anyone interested in creating interactive objects or smart environments. The boards can be purchased pre-assembled or built by hand as Arduino uses open-source licence and hardware reference designs are available for free. The microcontroller on the board is programmed using Arduino programming language which is based on C/C++. Arduino board functionality can be extended using shields, which are printed circuit boards that are placed on top of an Arduino and plugged into the normally supplied pin-headers. There are a lot of different third party shields that extend and give new functions, a list of Arduino-compatible shields can be found at shieldlist.org.

The Arduino programs are called sketches and they consists of two main parts, the setup function, that is run once at start and the loop function, which acts as main loop of entire program. The created sketch is compiled and uploaded to the microcontroller board where it is running until board is powered off. The Arduino boards come in many shapes and sizes. For example Arduino has microcontroller that is designed for wearables and e-textiles and it is called the LilyPad Arduino 2.9. On the other hand Mega ADK board 2.10 has different designs than LilyPad, making it more rich for extending functionality. Mega ADK is a microcontroller board based on the ATmega2560, and it has a USB host interface to connect with Android based devices. It has 54 digital input/output pins, 16 analogue inputs, 4 hardware serial ports and a USB connection for connecting different sensors and devices via Accessory Mode. Also it can be extended with different shields, such as WiFly Shield 2.11 that equips the microcontroller with the ability to connect to 802.11b/g wireless networks. Since the Arduino Mega ADK is still relatively new, there is not much material about usage of Arduino microcontrollers in context-aware gaming domain, but it has been used for designing robots and smart-homes.

Since the Arduino Mega ADK microcontrollers can be easily adapted to communicate with smartphones running on Android using accessory mode or over Wi-Fi, it is

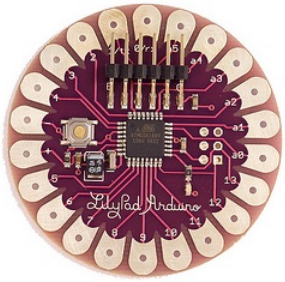

Figure 2.9: Arduino LilyPad
(1)

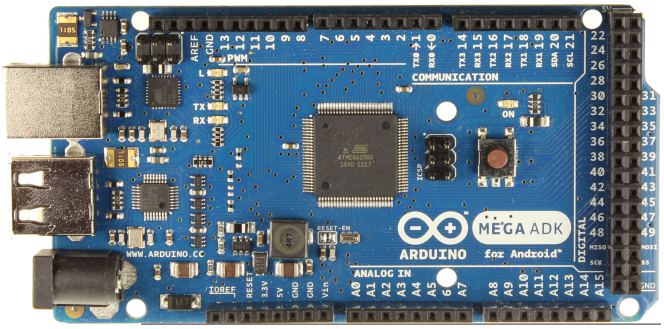

Figure 2.10: Arduino Mega ADK
(1)

well suited for the development and prototyping of context-aware games as the Arduino sensors can be used for external input of environmental information.

Furthermore, there already exist toolkits such as Amarino (22), that is toolkit to connect devices running on Android with Arduino microcontrollers using Bluetooth protocol. The toolkit provides access to devices internal events which can be processed by connected Arduino microcontroller. For example, if users phone rings then Arduino microcontroller can flash connected LED-lights or when text message is received it can be displayed on screen connected with Arduino. Since Amarino toolkit has very low entry level you can build custom interfaces almost without any programming experience.

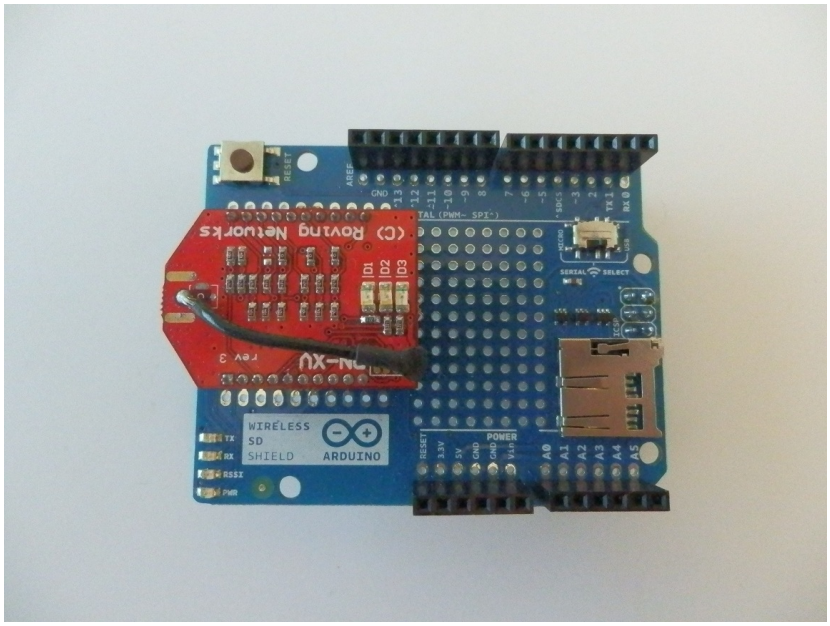

Figure 2.11: Arduino Wireless Shield

2.3 Context-aware Games

Context-aware games (aka pervasive games or augmented reality games) are the bridge between the real and the virtual gaming world. Context-aware games extend the gaming experience with real world data that is captured by many actuators which are embedded within the devices or located across the environment. There are several types of games that can be considered as context-aware depending on its contextual and interaction features. Several genders of context-aware games are introduced by Magerkurth et al. (23), such as smart toys, affective gaming, augmented tabletop or real world games, and location-aware games.

The Augmented Knights Castle (24) is modified Playmobil Knights Empire Castle play set introduced by Lampe et al. that extends default play set with background music, sound effects and verbal commentary of toys that react to the children's play. It offers interactive learning for children as the toy figures such as the king, a knight or a farmer teach children about the life in the Middle Ages from their perspective.

Many games use player physical location as main attribute for game. For instance Real Tournament is a context-aware mobile game that uses GPS and electronic compass data from players hand held and injects them into the game in order to extend the

gameplay. Similarly, games like Uncle Roy All Around You (25) and Capture the Flag (26) use smart phones with 3G access as main interfaces that makes them playable for everyday user as special devices are not needed. Uncle Roy All Around You mixes experience of online and real players walking on a street journey through the city. The real players have to follow clues on a hand held devices while looking for the elusive Uncle Roy and the online players follow their progress in a parallel virtual 3D world. Capture the Flag players have Symbian-based smartphones with GPS receiver and they have to locate the flag what is represented by a real wooden box with a Bluetooth device inside. The player has to physically pick it up and connect it with his smartphone to capture the flag.

The KinghtMage (27) is an augmented tabletop game that is built on top of S.T.A.R.S (28) platform. S.T.A.R.S is a framework for building augmented tabletop games and it integrates different input and output devices, such as touchscreen game tables, wall displays, personal digital assistant and audio devices. The KinghtMage mixes social role interaction within the game in order to move their characters on the screen.

2.4 Summary

Context-aware games combine the real world and the virtual gaming for providing a rich experience during the playtime. Most of the context-aware games that have been developed for mobile devices, they combine the sensor information of the handset with the sensor information retrieved from external sensor systems (e.g. microcontrollers, etc.). Several context-aware applications have been developed during the last years. Moreover, there are multiple frameworks that ease the development of context-aware applications within a mobile platform. However, most of these solution are in general ad-hoc and difficult to port, to reuse and to maintain. However, the latest technological achievements in hybrid mobile applications and open source electronic platforms enable the creation of context-aware applications can be developed for multiple platforms with considerable ease and low-effort.

3

Problem Statement

Even though, lot of context-aware games have been developed using specialized frameworks and technologies (discussed in previous section). Most of these solutions do not consider the effort required for porting the game between multiple mobile platforms. Generally, context aware games required to be rewritten for each specific platform and some customization (e.g. set an IP/port, etc.) is needed within the device in order to work with the external hardware. This chapter presents the possible drawbacks that emerge when porting a context-aware application between multiple mobile platforms.

3.1 Developing Portable Context-Aware Game Applications for Mobiles

When starting the development of a mobile application, often the first concern is what platform to choose? (e.g. Android, iOS, Windows 7, etc.). The answer to this question is mainly based on the purpose of the application (e.g. mobile game, social mobile application, etc.). However, developing a mobile application for a specific platform limits its scope, in terms of distribution, commercialization, etc. Thus, making the application perspective narrow to one single mobile vendor. In the context of mobile game development, applications are even more tied to some specific aspects of the mobile platform (e.g. proprietary graphic technologies, etc.), and thus making complex the process of creating a game that can be deployed in multiple mobile operating systems. Moreover, when a mobile game includes contextual information like in the case of context-aware applications, the game becomes more specific as the logic for managing

the network communication may change dramatically to fit specific requirements of the mobile vendor such as programming language, background communication (Android case), etc..

On the other hand, HTML5 is emerging as a technology that solves cross platform compatibility issues, supports the development of offline-based Web applications and allows the use of some native mobile features (sensors, touchscreen, etc.) to create hybrid mobile applications. However, it is necessary to conduct an extensive analysis in this technology as multiple issues can arise such as decreasing the mobile application performance, accessing the mobile resources (e.g. camera among others), etc. For instance, the functionality of an HTML5 application could be limited by the fixed memory assigned to the browser by the mobile operating system. Mobile game applications may benefit from HTML5 for the development of portable game applications that can be executed in the top of a mobile browser. However, current HTML5 technologies for managing the underlying mobile resources are limited (accessing certain embedded artefacts) and are just target for a few mobile platforms. Moreover, such solutions also do not consider the fact of using contextual sensor information within the HTML layer, making the game not suitable for context interaction.

3.2 Research Question

In order to investigate the possibility of creating portable pervasive applications on the top of the mobile browser using HTML5, the current thesis proposed extending the current implementation of PhoneGap for supporting contextual information sensed by Arduino Microcontroller.

PhoneGap allows us to get access to native APIs using Web technologies as it wraps native container around the Web application. The main idea consists of creating a module (specific per each platform) that enables connecting to the Microcontroller so that the raw sensor data can be processed locally (e.g. combined with other sensors, etc.) by the mobile resources and then passed to the HTML5 layer in order to be used within the game.

3.3 Summary

HTML5 has arisen as a prominent technology for creating hybrid mobile applications. Furthermore, it enables to execute a mobile application in the top of the mobile browser, and thus decreasing significantly the introduction of native code in the development process. However, HTML5 still is in its infancy and a lot of researches have to be performed in order to tackle the drawbacks such as access to contextual sensor information, cameras and other actuators. This thesis proposes extending the current version of PhoneGap with a module that enables supporting sensor information that is sensed by Arduino Microcontroller.

4

Towards a Portable Pervasive Game in HTML5

In this chapter, the thesis describes how to address the multiple drawbacks that emerged when creating a context-aware mobile game. Fuzed game (HTML5 application) is introduced as case of study along with its evaluation in multiple mobile platforms including Android and iOS..

4.1 General Description and Implementation Details

The main idea behind the solution is that Arduino microcontroller broadcasts collected environmental data over UDP protocol and Android device listens to assigned port and stores the received data. While data is sent over Wi-Fi, both Arduino microcontroller and Android device must be in the same network. The Wi-Fi connection is used for sending data instead of implementing the accessory mode, because the implementation using the Wi-Fi connection is more universal, as the accessory mode is available only for Android versions 3.1 and up.

The initial solution can be divided into three separate parts: the Arduino sketch containing the microcontroller configuration, the UDP packet collector for receiving data from Android and the module implementation for exchanging collected data between client-code (system component using PhoneGap framework).

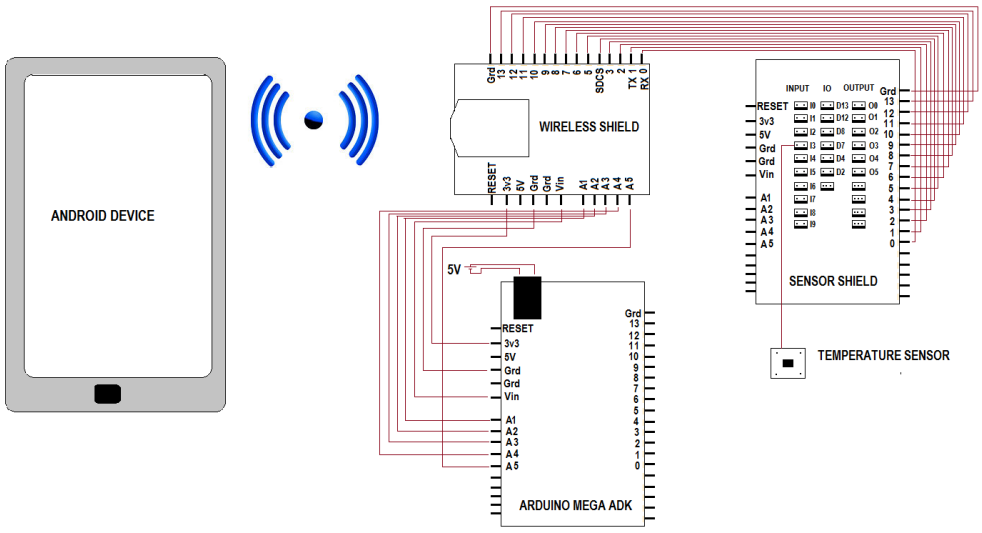

Figure 4.1: Architecture of the Proposed Solution

4.1.1 Setting Up the Arduino Mega ADK

Since data is sent over wireless network we extend our board with WiFly Shield that equips the microcontroller with the ability to connect to 802.11b/g wireless networks. First thing is to configure the board, so it can access needed Wi-Fi network. A complete overview of the steps to be done to set up the Arduino Mega ADK board with WiFly Shield is presented at the end of this section.

Although TCP protocol offers error correction and flow control to guarantee packet delivery, we selected UDP protocol for implementing presented solution as it is faster than TCP. The goal is not to receive every packet sent by microcontroller as the data is not so important that we cannot allow the loss of the package. Rather, the aim is the immediate detection of changes in the environment and for that data is sent over relatively small intervals. The sketch for board is rather simple as it currently implements only capturing and converting the temperature from analogue input. When temperature is converted from input to human readable decimal we will pass it to serial output and from there the WiFly Shield broadcast the packet to configured host over UDP protocol using the wireless network. The sketch must be compiled and uploaded to board using Arduino Toolkit that can be downloaded from Arduino homepage.

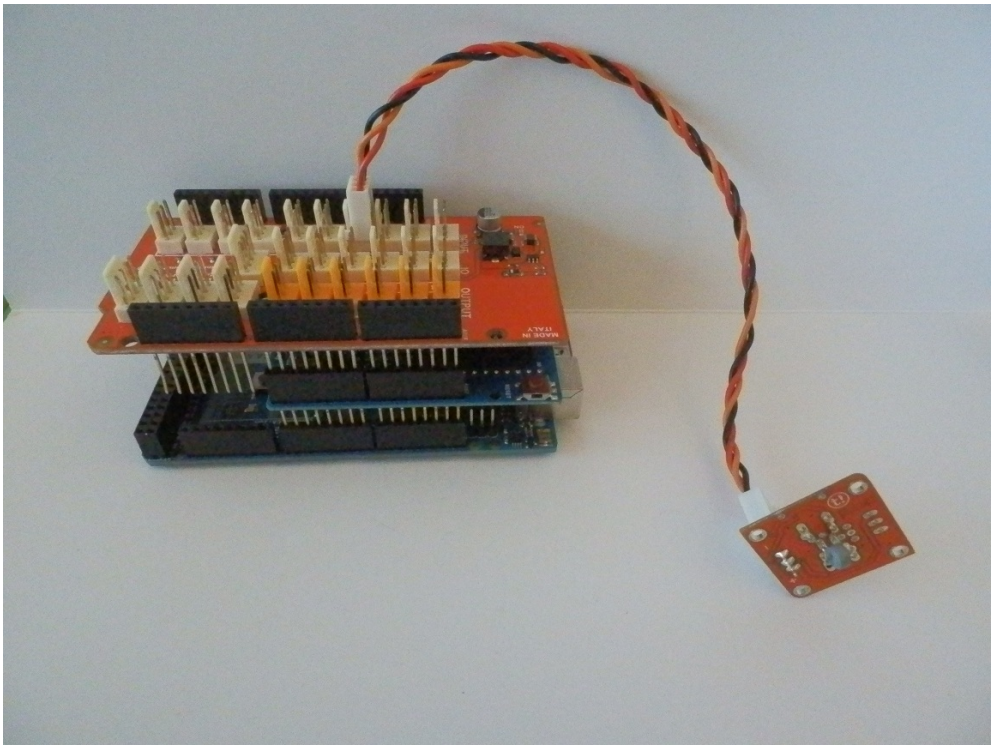

Figure 4.2: From Bottom: Arduino Mega ADK, Wireless Shield, Sensor Shield and Temperature Sensor.

4.1.1.1 Configuring the Arduino Mega ADK

To successfully connect with Android device the WiFly shield must be configured correctly. Before accessing the board make sure that the USB cable is connected and the Arduino microcontroller is connected over serial port and the WiFly Shield is switched to USB mode. Since we use Windows machine for development, we can use HTerm or Putty 4.3 to access the Arduino microcontroller. Following steps must be completed to configure the board correctly.

1. Switch the WiFly Shield is to USB mode.
2. Upload an empty sketch to your board (BareMinimum) using the Arduino Toolkit
3. Open HTerm or Putty and configure following connection parameters
   (a) Port - COM port where Arduino is connected
   (b) Baud - 115200

(c) Data - 8

(d) Stop - 1

(e) Parity - None

(f) Flow control - None

4. Connect

5. Type $$$ to access the Arduino command prompt.

6. Firstly we configure Arduino to use UDP connection. Insert following commands:

   (a) set ip proto 1

   (b) set comm timer 1000

   (c) set ip remote 2000

   (d) set ip local 2000

   (e) set ip host ¡android device ip¿ (sends data here)

7. Type scan to discover available Wi-Fi networks

8. To maintain Wi-Fi connection insert following commands

   (a) set wlan ssid ¡string¿

   (b) set wlan pass ¡string¿ (if protected)

9. To save and reboot the board insert commands:

   (a) save

   (b) reboot

10. Upload the correct sketch

11. Switch WiFly Shield back to Micro mode.

    The Arduino microcontrollers in now ready to communicate with Android device configured as the host. If Wi-Fi network is changed then you must reconfigure the board, for this you can repeat the steps 1 to 5 and 7 to 11. However if you want to connect to different Android device, then you have to reconfigure the host parameter, for this you must follow the steps 1 to 5, 6e and 9 to 11.

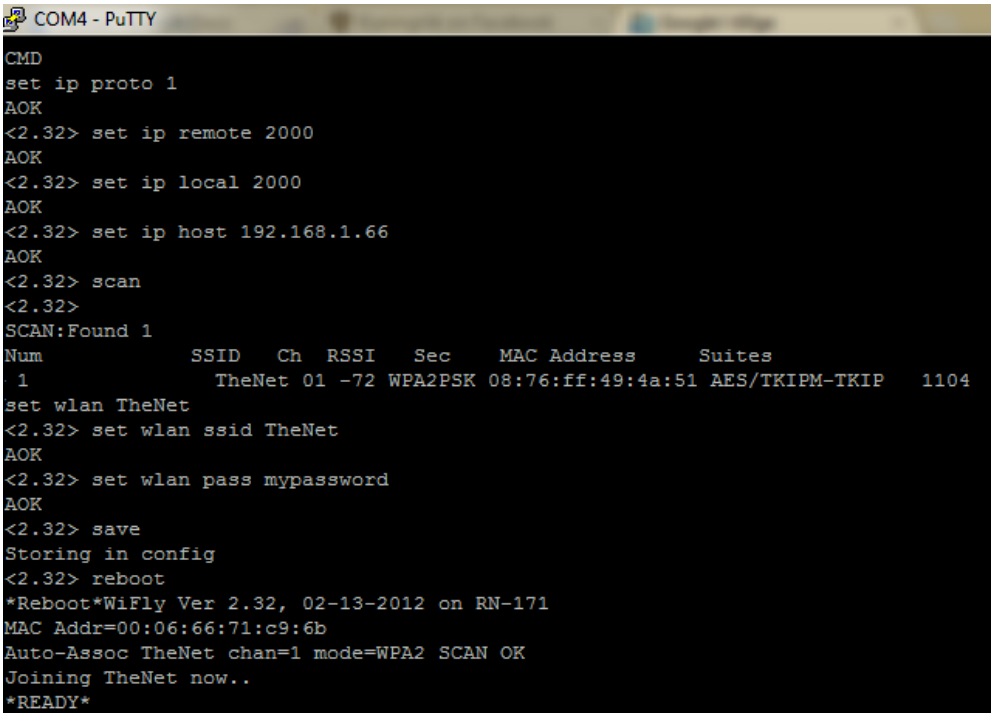

Figure 4.3: Arduino Configuration for UDP Connection and Wi-Fi

4.1.2 The Packet Collector and PhoneGap Plug-in

While packet collector is part of the Android application it is also written in Java programming language. Furthermore, as Java is very mature language it provides necessary tools to create UDP communications link between Android device and Arduino microcontroller. As seen on appendix A, the packet collector listens to assigned port and stores received data and as every collector instance is running in separate thread we can receive data from more than one microcontroller simultaneously. Furthermore, we can assign different port to every collector to ensure that packets sent from different devices do not collide.

The creators of PhoneGap have been prudent and have left an opportunity for developers to create their own plug-ins and modules that provide access to the system components from client-side code. The developer is responsible to implement the functionality that is executed when the client-side code uses the plug-in. All the data flows between system components and client-side code are handled by PhoneGap plug-in manager. On appendix B we can see how easy it is to im-

plement PhoneGap plug-ins on client side. Although, the message can only be a string, we can use the JavaScript Object Notation (JSON) to send more complex data from system component to client-side application.

The PhoneGap framework makes most of the hard work and developer has to concern about calling right function from client-side and send valid response from system component. From client-side implementation we have to create the JavaScript function that sends the request with callback information, parameters and action name, to right plug-in. Also we have to register the created function to PhoneGap plug-in manager to access it later.

4.2 Summary

As we see the implementation of the presented module is fairly simple as the PhoneGap framework enables the data flow between client-code and system component. However, one of the downsides of current solutions is that we have only one way communications, this means that Android device cannot send commands to microcontroller. Another thing is that Arduino microcontroller must be reconfigured for every Wi-Fi network and host device. At the moment the provided solution is not production ready and cannot be used out of the box solution for context-aware game development. However, these problems can be overcome and missing parts can be implemented in the future.

5

Case Study

As case of study we introduce the context-aware game Fuzed, to analyse the development aspects and performance of cross-browser pervasive games. Presented game is classical platform game, where player must fight with small enemies, collect awards and advance to next platform to complete the stage. Furthermore the player has to avoid the mines and enemies that decrease player's vitality points.

Figure 5.1: The Game View

The game consists of five main objects, which have a different functionality and purpose. On figure 5.1 we see blue character, which is the player, green characters that are the enemies, portals that are used to teleport between platforms, on the ground we can see the mines and collectable coins hanging on air. To destroy the

enemies, player can use the fused bombs that can be thrown. On figure 5.2 we can see the bombs in action.

Figure 5.2: One of the Enemies is Destroyed Using the Fused Bomb

The game is developed using hybrid mobile application approach, and it means that entire game logic is written using web technologies and exists as web applications. Furthermore the core game is playable in any modern web browser without need for any plug-ins as it only uses HTML, CSS and JavaScript technologies. However, if such web application is placed in the native wrapper, then the web application can also access to native API and use its functionality. In our case, if game is played on mobile, inside native wrapper then character is controlled by accelerometer, but when played in browser, the arrow keys are used for moving the character.

Several sensors are used to make the game context-aware. To control the movement of the main character the embedded accelerometer is used, thus it allows to change the movement direction by tilting the phone. Furthermore the environment temperature is used to extend the gameplay. When the temperature of the environment drops below 23 degrees, the grass in the game changes to snow and when the temperature raises the snow disappears and grass becomes visible once again. Such high number of temperature is just used for demonstrating the functionality and different values can be assigned. The changes in gameplay are demonstrated on figure 5.4.

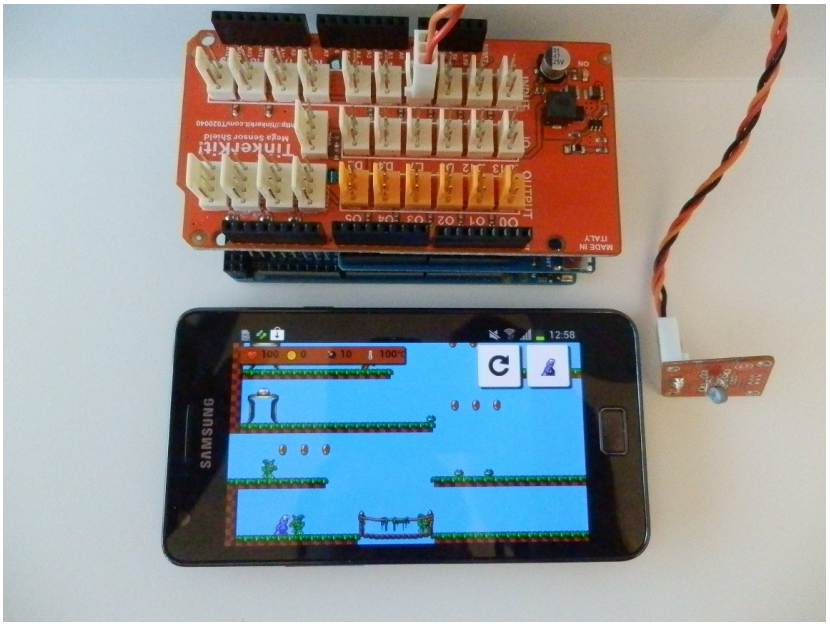

Figure 5.3: The Game Setup: Arduino Mega ADK with Extended Shields and Samsung Galaxy S2

To reduce development time, a JavaScript game engine Crafty is used for game core development, as it provides common set of functionality, such as scene loading, sprite animation and collision detection. Crafty is lightweight, entity based JavaScript game engine created by Louis Stowasser, and it already has very big community that helps to develop the engine further.

For bridging web application and mobile device we use PhoneGap that allows web technologies to get access to vendor specific APIs through native application and to communicate with the Arduino microcontroller we use the PhoneGap plug-in that is described in the previous chapter. Furthermore with the help of the PhoneGap framework it is possible to listen the button press events on client-side code. This functionality is used to implemented pause (Fig. 5.6) and about screen (Fig. 5.5), which are triggered when back or menu button is pressed. Thus giving the feel and sense of native application as in the usual web application, pressing back button would trigger back action inside the browser component. Furthermore, when user navigates away from the game, the game would also pause and player can return to the game whenever he wants.

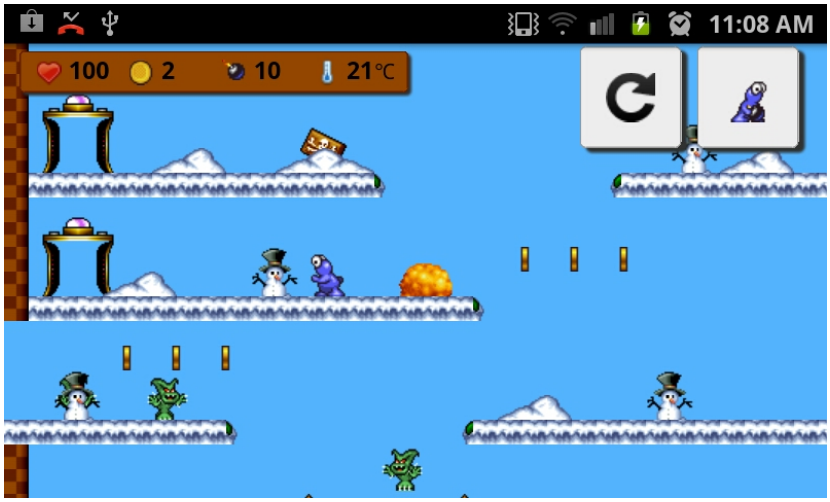

Figure 5.4: Changes in the Gameplay

Figure 5.5: About Screen

While the game has many animations, such as main character and enemy movement, explosions and coins, a lot of computing power goes to animating different parts of the game. However, the game implements the CSS3 keyframe for animations instead of widely used JavaScript approach (Appendix C), as the browser implements the CSS3 keyframe animations completely hidden from the developer and the browser itself is able to apply tricks such as GPU acceleration to achieve better performance and lower power consumption.

Figure 5.6: Pause Screen

5.1 Porting the Game for Other Mobile Operating Systems

As mentioned before, this game can be easily ported for other mobile operating systems. To prove this claim, we ported this game for iOS, but due to scope of this thesis, the PhoneGap plug-in for communicating with Arduino microcontrollers is not implemented for iOS. Although, we had no experience using the PhoneGap framework with iOS, we managed to port the game with only four hours and writing only few lines of vendors specific code. Since the browsers in iOS and Android devices are based on the WebKit Open Source Project and they interpret the CSS and JavaScript the exact same way, the client-side code behind the game is exactly the same as it is for Android implementation.

However, if we would like to port this game for Windows Phone 7, we have to make bigger changes, because the Windows Phone 7 uses the Internet Explorer 9 (IE9) as the browser component and it doesn't support all of the necessary functionality and features to run the game properly. Although the porting process is more complicated than it was for iOS, it is still possible to port this game for the Windows Phone 7 platform. Since the IE9 doesn't support the CSS3 keyframes, we have to rewrite the animation logic and implement them using JavaScript approach, which will greatly decrease the application performance as

Figure 5.7: From Top: Samsung Galaxy S2 (Android), HTC Desire Z (Android), iPhone 4 (iOS)

more computing power is used for animations. Although the estimated time for rewriting the animation logic would be a few days, then it is still faster that to write independent game for Windows Phone 7 platform.

5.2 Visual Analysis of Game Performance

To analyse the game performance on different devices we use MOS (Mean Opinion Score) and visually evaluate the game performance while changing the game's frames per second (FPS) rate. The higher the frames per second rate is, the smoother are animations and character movement, however if the device cannot handle the given FPS the game will be slow and eventually freezes or crashes.

Phone	**Operating system**	**Memory**	**CPU**
HTC Desire Z	Android 2.3.3	512MB	800MHz
Samsung Galaxy S2	Android 4.0.3	1GB	Dual-Core 1.2GHz
iPhone 4	iOS 5.0.1	512MB	1GHz

Table 5.1: Tested Smartphone Specifications

MOS	**Performance**	
5	Excellent	There are no visible delays in the movement and animations
4	Good	Minimal visible delays in the movement and animations
3	Fair	Movement and animations are rough and not smooth
2	Poor	Movement and animations are discontinuous
1	Bad	Game is unplayable

Table 5.2: Mean Opinion Score

Device	**20 FPS**	**30 FPS**	**40 FPS**	**60 FPS**
HTC Desire Z	2	3	2	1
Samsung Galaxy S2	2	4	5	5
iPhone 4	2	3	2	1

Table 5.3: Visual analysis of game performance using MOS approach

To analyse the performance of different devices and platforms we configured the game with different FPS rates and visually evaluated the smoothness of the animations and movement. Although lower frame rate decreases needed computing power, it is obvious that if the frame rate is too low it decreases the overall smoothness of the entire game, because the game frame is updated too rarely. As show in table 5.3 Samsung Galaxy S2 achieved the best graphical performance.

The results are not surprising, since the Samsung Galaxy S2 has more powerful hardware compared to other devices used in the analysis. Although, the iPhone 4 has slightly better central processing unit, it performed similar to HTC Desire Z that has the lowest specs considering the memory and the CPU. The analysis shows that the game is playable on every device, but we have to establish a suitable FPS for the game that does not distract the user and creates a sense that there is no need for better performance.

5.3 Scalability Analysis

Unfortunately, Arduino microcontroller is not scalable enough to provide concurrent services to multiple mobile users. By using the Wifly module the solution is limited to one particular user at the time. Alternatively, to alleviate this problem, Arduino may send its sensor information via XMPP to the cloud in order to be accessible by any amount of users, and thus scaling the sensor data on demand. However, this is out of the scope of the thesis and it can be considered as a future research direction.

5.4 Summary

Presented game shows that hybrid mobile applications and open-source electronics kits, such as Arduino, can be successfully used for developing context-aware games. Furthermore, when computing power of smartphones and mobile browser components increases, we can create mobile games with complex animations and graphics using web technologies and give the native feel and sense, since the web application can be packed inside the native container. Although, current status of smartphones and web technologies allows us to create complex games, most of the smartphones on market are not able to run such games as smoothly as they do with native applications. However, the web and hybrid application approach can be used to create games that do not need animations and huge computing power.

6

Related Work

The Arduino Mega ADK is relatively new in context-aware gaming domain as it was released in 2011. Furthermore, most of the related work in context-aware gaming domain requires different modified devices and the smartphones are not much used, however there are some exceptions. For example, FRAP (20) is a framework for constructing context aware games and it reduces the development time of context aware game as it implements primary location tracking and provision of context aware game tasks. Clients location is tracked using smart phones GPS and framework assumes that clients have a permanent internet access, but loss of connection due to missing network coverage will be taken into account and disconnection of any client will not have fatal consequences for the game play. As example application, context aware game King Of Location is shipped with the framework. Furthermore, many games use player physical location as main attribute for game. For instance Real Tournament is context aware mobile game that uses GPS and electronic compass data from players hand held and injects them into game to extend gameplay. For this game specially modified personal digital assistant were used. On another hand games like Uncle Roy All Around You (25) and Capture the Flag (26) use smart phones with 3G access as main interfaces that makes them playable for everyday user as special devices are not needed. Uncle Roy All Around You mixes experience of online and street players as players on street journey through a city following clues on a handheld device in search of elusive Uncle Roy and online players follow their progress in parallel

virtual 3D world. In Capture the Flag players have Symbian-based smartphones with GPS receiver and they have to locate the flag what is represented by a real wooden box with a Bluetooth device inside and the player has to physically pick it up and connect it with his smartphone to capture the flag. Although, many mobile games use players physical location as game attribute, there is very few or even no games that also use other environmental data to extend the gameplay. As shown in presented game Fuzed, we can successfully use the environment temperature as one of the game attributes and thereby enrich the gameplay. Another advantage over the existing games in context-aware gaming domain, is that the introduced games is hybrid application and can be ported to other platforms with very low cost. The context-aware game Fuzed that is presented as case of study is relatively unique in current context-aware gaming domain as it uses the Arduino Mega ADK microcontroller for external sensors. Furthermore, it uses common smartphone as main device, when most of the context-aware games require special hardware or when smartphones are used, then only built-in sensors are selected to extend the gameplay.

7

Conclusions and Future Research Directions

Mobile technologies are evolving very rapidly, and the latest achievements in the mobile domain offer the tools to develop cross-browser applications that can be deployed on any mobile platform. Thus, reducing the effort to port, to maintain and to reuse mobile applications. Furthermore, the embedded sensors (e.g. accelerometer, GPS, gyroscope, etc.) are a must for today's smartphones as they can be used to extend the applications and increase the user experience. Moreover, the presented game demonstrates that Web technologies can be used for creating pervasive games that look and feel as native applications. Furthermore, the plug-in for PhoneGap framework implements the communication channel between Arduino Microcontrollers and Android powered devices, thus allowing THE gathering OF environmental data by using client-side resources.

In previous sections we stated that the presented game is a cross-platform and to prove this claim, we ported the game for iOS. However due to the scope of the thesis, the PhoneGap plug-in for communication channel between Arduino Microcontrollers and mobile devices is not implemented for iOS. Although, we had no experience using the PhoneGap framework with iOS, we managed to port the game within four hours and wrote a few lines of vendor's specific code. While the browsers in iOS and Android devices are both based on the WebKit Open Source Project, they interpret the CSS and JavaScript exactly the same way.

Thus, no changes to the client-side code were made and the implementation is same as it is for Android.

Although, the context-aware gaming domain is not a new discovery, it still is rather unexplored territory, since our researches show that most of the publications found, were written more than five years ago. Thus, the given papers do not reflect the current situation of the context-aware gaming domain and as outlined in some of the papers, there are very few guidelines for context-aware game developers.

Our goal was to create a pervasive game that uses sensor data from local and external resources and would be portable between multiple mobile platforms (e.g. Android, iOS, Windows Phone 7, etc.). Although, we accomplished all the set objectives, new questions arose which require detailed analysis, that is why the knowledge about the cross-platform pervasive gaming domain must be extended. Furthermore, the provided solution only implements the thermistor and one way communication. This means that the Android device can only receive data as the Arduino Microcontroller broadcasts collected information only over a certain interval. As a result, one of the future research directions could be the implementation of two way communication between Android devices and Arduino Microcontroller as it would allow us to ask data from the Microcontroller instead of waiting until the Microcontroller decides to send it. Furthermore, the current sketch for Android Microcontroller should be extended to provide implementation for other sensors and artefacts, to provides full functionality of the Arduino Microcontroller. Although, the PhoneGap plug-in is currently implemented only for Android powered devices, it can be implemented for other platforms (e.g. iOS, Windows Mobile 7, etc.) using similar methodologies.

8

Sisukokkuvõte

Keskkonnateadliku mobiilimängu programmeermine kasutades Android ja HTML5 tehnoloogiaid
Context-aware mobile games using Android and HTML5

Bakalaurusetöö

Tanel Tähepõld

Resümee

Mobiilitehnoloogiate areng ja nutitelefonide kiire levik loob uusi võimalusi mobiilirakenduste arendamiseks ning palju rõhku pannakse mängudele. Loodavad mängud muutuvad aga keerulisemaks, sest üha enam kasutatakse keskkonnast tulevat informatsiooni, et kohandada mängu vastavalt kasutaja asukohale ja kontekstile. Selleks kogutakse andmeid erinevatest sensoritest, nagu näiteks GPS, kiirendusmõõtur, kompass ning güroskoop, mis annavad informatsiooni kasutaja asukoha ja liikumise kohta. Sellistel põhimõtetel loodud mängu nimetatakse keskkonnateadlikuks mänguks ning ned seovad mängija füüsilise asukoha ja oleku virtuaalmaailmaga. Näitena võib tuua mängu Real Tournament (6), mis kasutab GPS tehnoloogiat, et määrata kasutaja reaalne asukoht ja liikumine ning seejärel kasutatakse saadud infot mängija asukoha määramiseks virtuaalmaailmas.

Üha enam luuakse rakendusi veebiplatvormile, mitte kindlatele operatsioonisüsteemidele, sest veebitehnoloogiad HTML5, CSS ja JavaScript võimaldavad luua võrdväärse funktsionaalsuse ning kasutajamugavusega rakendusi. Veebipõhiste rakenduste suurimaks

eeliseks on see, et nende tööks on vaja ainult veebibrauserit ning igale operastioonisüsteemile ei pea kirjutama eraldi programmi. Antud tehnoloogiad on jõudnud ka mobiilimaailma, kus iga operatsioonisüsteemi jaoks peab arendama eraldiseisva rakenduse, sest iga tootja süsteem (iOS, Android, WebOS jne) on teistest erinev. Samas uusimaid standardeid toetav veebibrauser on olemas igas nutitelefonis, seega saame arenduskulusid kokku hoida ning erinevatel platvormidel kasutada ühesugust lahendust.

Suureks miinuseks selliste rakenduste puhul on ligipääsu puudumine tootja spetsiifilisele liidesele, mis võimaldab suhelda telefoniga ning kasutada platvormi poolt pakutavat funktsionaalsust, nagu näiteks kaamera ja seadmes paiknevate sensorite andmed. Selle probleemi lahendamiseks on väljatöötatud raamistikud, mis võimaldavad brauseris käitatavale rakendusele ligipääsu tootja spetsiifilisele liidesele. Nii saab rakenduse pakkida konteinerisse ja seda kasutajale esitada tootja septsiifilise rakendusena ning seda minimaalse arenduskuluga. Aplikatsioonid, mis kasutavad koos veebitehnoloogiaid ja platvormi spetsiifilist liidest, nimetatakse hübriidrakendusteks ning hetkel arvatakse, et just selline lahendus on kõige jätkusuutlikum.

Antud bakalaurusetöö eesmärgiks on luua alternatiivne mobiilimäng, mis erineb traditsioonilistest arvutimängudest selle poolest, et mängu juhtimiseks kasutatakse väliskeskkonnast tulenevaid andmeid. Tänu sellele saab vastavalt väliskeskkonnast saabunud infole mängu rikastada erinevate elementidega. Nii võib temperatuuri langus võib vallandada lumesaju või määratud müirataseme ületamine põhjustada mängumaailmas maavärina. Tehnilise lahendusena kasutatakse eelpool kirjeldatud hübriidtehnoloogiat Android platvormil. Hübriidrakenduse loomiseks kasutame PhoneGap nimelist raamistikku ning loodava mängu teeb eriliseks see, et lisaks nutitelefonis olevatele sensoritele, nagu näiteks kiirendusmõõtur ja GPS, kasutab rakendus väliskeskkonnast andmete kogumiseks Arduino mikrokontrollereid, mis vahetavad mobiiltelefoniga andmeid kasutades Wi-Fi liidest. Loodava mängu edasiarendusena võib näha sotsiaalseid mitme kasutaja mänge, kus reaalse maailma tegevustega saab mõjutada teiste mängijate tulemusi. Nii saab luua täiesti uusi mänge, mis ühendavad traditsioonilise seltskonnamängu sotsiaalsed aspektid ning arvutimängudest tuttava virtuaalse keskkonna.

Antud bakalaurusetöö suurimaks panuseks on PhoneGap raamistikule loodud moodul, mis võimaldab pärida Arduino mikrokontrolleri saadetud sensorite informatsiooni kliendipoolsest koodist, mis tavaliste lahendustega pole võimalik. Loodud moodulit kasutatakse ka

näitena toodud rakenduses, milleks on keskkonnateadlik mäng Fuzed, kus mängija peab võitmiseks jõudma mänguväljaku kõige kõrgemale astmele. Mängija teel aga on takistuseks vaenalsed, kes toituvad tema eluenergiast ning miinid, mis kokkupuutel mängijaga plahvatavad. Mäng kasutab telefonis olevat kiirendusmõõturit, mis laseb meil peategelast juhtida telefoni kallutades (tegelane liigub vasakule ja paremale vastavale poolele telefoni kallutades). Kuna nutitelefonid on tavaliselt puutetundliku ekraaniga, siis tegelase paneb hüppama ekraani puudutamine. Selliste funktsioonide olemasolu teeb tegelase juhtimise palju sujuvamaks. Keskkonnateadliku elemendi lisab Arduino mikrokontrolleriga ühendatud termomeeter ning kui antud sensor registreerib temepratuuriks 21 kraadi või vähem, siis kattub mänguväljak lumega, kui temperatuur taas tõuseb ;; siis lumi sulab.

Esitletud mäng Fuzed näitab, et veebitehnoloogiad ning avatud lähtekoodiga elektroonikapakette, nagu näiteks Arduino, saab edukalt kasutada keskkonnateadlike mängude arendamiseks. Kuigi hetkel võimaldavad veebitehnoloogiad luua keerulisi mänge, siis enamus turul olevaid nutitelefone ei suuda neid nii sujuvalt jooksutada, kui kasutajad sooviksid. Nutitelefonide arvutusvõimsus kasvades, saame luua keeruliste animatsioonidega mobiilimänge, mis on võrdväärsed hetkel süsteemispetsiifilises keeles kirjutatud mängudega.

9

Appendices

9.1 Appendix A

Code snippet from Android implementation for PhoneGap plug-in that shows data packets sent by Arduino Microcontroller are collected by the Android device. When packet is received, it is parsed and stored to memory.

```
public static final int BCAST_PORT = 2000;
DatagramSocket mSocket;
InetAddress myLocalIP;

public ComThread() {
  try {
    // Get local ip
    this.myLocalIP = getLocalAddress();
    // Create socket
    this.mSocket = new DatagramSocket(ComThread.BCAST_PORT);
    this.mSocket.setBroadcast(true);
  } catch (IOException e) {
    Log.e(TAG, "Could not make socket", e);
  }
}
public void run() {
  try {
    byte[] buf = new byte[1024];
    //Listen on socket to receive messages
    while (true) {
      DatagramPacket packet = new DatagramPacket(buf, buf.length);
      this.mSocket.receive(packet);

      InetAddress remoteIP = packet.getAddress();
      if(remoteIP.equals(this.myLocalIP))
        continue;

      String s = new String(packet.getData(), 0, packet.getLength());
      // Send the obtained bytes to the UI Activity
      mHandler.obtainMessage(Arduino.MESSAGE_READ,-1,-1, s).sendToTarget();
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

9.2 Appendix B

Registration of client-side PhoneGap plug-in that creates creates bridge between the Android's native API and clinet-side implementation.

```
var Arduino = function() {
};

/**
 *  Returns temperature registered in Arduino controller
 *
  * @param successCallback The callback which will be called when directory listing is successful
  * @param failureCallback The callback which will be called when directory listing encouters an error
  */
Arduino.prototype.getTemperature = function(successCallback, failureCallback) {
  return PhoneGap.exec(
    successCallback,
    failureCallback,
    'Arduino',
    'getTemperature',
    []
  );
};

PhoneGap.addConstructor(function() {
   PhoneGap.addPlugin("arduino", new Arduino());
});
```

9.3 Appendix C

Code snippet that shows how to animate sprite with eight frames using JavaScript approach. The main point of this approach is to change the background attribute of html element over given interval.

```
var el = document.getElementById("coin_gold");
var x = 0;
setInterval(function() {
  if (x <= 112) {
    x = 0;
  }
  el.style["background-position"] =  "-"+x+"px 0px";
  x += 16;
}, 1000/60);
```

Bibliography

[1] Android Inc., Android, http://www.android.com (2012). 1, 20, 21

[2] The U.S. government, Gps, http://www.gps.gov/ (2012). 1

[3] Arduino Inc., Arduino, http://arduino.cc (2012). 1

[4] M. Pilgrim, HTML5: up and running, OReilly & Associates Inc, 2010. 2

[5] A. Charland, B. Leroux, Mobile application development: web vs. native, Communications of the ACM 54 (5) (2011) 49–53. 2

[6] K. Mitchell, D. McCaffery, G. Metaxas, J. Finney, S. Schmid, A. Scott, Six in the city: introducing real tournament-a mobile ipv6 based context-aware multiplayer game, in: Proceedings of the 2nd workshop on Network and system support for games, ACM, 2003, pp. 91–100. 2, 51

[7] A. Cheok, K. Goh, W. Liu, F. Farbiz, S. Fong, S. Teo, Y. Li, X. Yang, Human pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing, Personal and Ubiquitous Computing 8 (2) (2004) 71–81. 2

[8] S. Srirama, H. Flores, C. Paniagua, Zompopo: Mobile calendar prediction based on human activities recognition using the accelerometer and cloud services, in: Next Generation Mobile Applications, Services and Technologies (NGMAST), 2011 5th International Conference on, IEEE, 2011, pp. 63–69. 2

[9] PhoneGap Inc., HTML5 application platform, http://phonegap.com (2012). 2, 18

[10] Wikimedia Inc, Picture of a Gyroscope, http://upload.wikimedia.org (2012). 8

[11] Furono Electric CO., LTD, Picture of GPS segments, http://www.furuno.com (2012). 9

[12] Apple Inc, Apple, http://www.apple.com/ (2012). 9

[13] GeoSpatial Training Services, LLC, Anatomy of a Hybrid Mobile GIS Application, http://www.geospatialtraining.com/blog/?p=1897 (2012). 11, 14, 17

[14] E-consultancy.com Limited, The fight gets technical: mobile apps vs. mobile sites, http://econsultancy.com/uk/blog/7832-the-fight-gets-technical-mobile-apps-vs-mobile-sites (2012). 12

[15] Forbes.com LLC, Comparison of Native, Hybrid and Web mobile applications, http://www.forbes.com/sites/fredcavazza/2011/09/27/mobile-web-app-vs-native-app-its-complicated/ (09 2012). 13

[16] Worklight, an IBM company, Mobile Application Platform and Tools, http://worklight.com (2012). 13

[17] Appcelerator Inc, Titanium, the leading mobile platform of choice for thousands of companies seizing the mobile opportunity, http://www.appcelerator.com (2012). 13

[18] Sencha Inc, HTML5 Framework for Desktop and Mobile Devices, http://www.sencha.com (2012). 13

[19] The jQuery Foundation, jQuery Mobile, http://jquerymobile.com/ (2012). 14

[20] J. Tutzschke, O. Zukunft, Frap: a framework for pervasive games, in: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, ACM, 2009, pp. 133–142. 18, 47

[21] R. Rogers, Augmented reality with html5, Linux Journal 2011 (203) (2011) 3. 18

[22] B. Kaufmann, L. Buechley, Amarino: a toolkit for the rapid prototyping of mobile ubiquitous computing, in: Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, ACM, 2010, pp. 291–298. 20

[23] C. Magerkurth, A. Cheok, R. Mandryk, T. Nilsen, Pervasive games: bringing computer entertainment back to the real world, Computers in Entertainment (CIE) 3 (3) (2005) 4–4. 21

[24] M. Lampe, S. Hinske, Integrating interactive learning experiences into augmented toy environments, in: Pervasive Learning Workshop at Pervasive, Citeseer, 2007. 21

[25] S. Benford, M. Flintham, A. Drozd, R. Anastasi, D. Rowland, N. Tandavanitj, M. Adams, J. Row-Farr, A. Oldroyd, J. Sutton, Uncle roy all around you: Implicating the city in a location-based performance, Proc. Advances in Computer Entertainment (ACE 2004). 21, 47

[26] A. Cheok, A. Sreekumar, C. Lei, L. Thang, Capture the flag: mixed-reality social gaming with smart phones, Pervasive Computing, IEEE 5 (2) (2006) 62–69. 21, 47

[27] S. Benford, C. Magerkurth, P. Ljungstrand, Bridging the physical and digital in pervasive gaming, Communications of the ACM 48 (3) (2005) 54–57. 22

[28] C. Magerkurth, R. Stenzel, T. Prante, Stars-a ubiquitous computing platform for computer augmented tabletop games, Proceedings of Video Track of Ubiquitous Computing (UBICOMP03). 22