etsinf

German Research Center for Artificial Intelligence GmbH

FINAL PROJECT THESIS

Development and Implementation of a Mobile AR-Based Assistance System on the Android-Platform for the *SmartFactory*^{KL}

Editor: Ricardo Campos García

Director: Prof. Martín Mellado

Tutor: Dipl.-Ing. Dominic Gorecky

Date: 15. September 2011

Degree: Computer Science

Declaration

I declare that I wrote this thesis autonomously and without any unauthorized outside help. Text sections or images, which are based on external sources contain references and are indicated in the bibliography.

Kaiserslautern, 15. September 2011

Content

1 Introduction 1

2 Task 3

3 State of the Art 4

3.1 Augmented Reality 5

3.1.1 Overview 5

3.1.2 History 6

3.1.3 Augmented Reality vs. Virtual Reality 8

3.1.4 Augmented Reality Application Domains 9

3.1.5 Augmented Reality in Manufacturing 9

3.1.6 Augmented Reality System Architecture 11

3.2 Android Operating System 14

3.2.1 Overview 14

3.2.2 Features 14

3.2.3 Architecture 15

3.2.4 Application Fundamentals 18

3.2.5 Application Components 19

3.3 AndAR Library 20

3.3.1 Overview 20

3.3.2 Structure 21

3.4 *SmartFactory*^{KL} 22

3.4.1 Introduction 22

3.4.2 Description 23

3.4.3 Architectures 23

3.4.4 Projects 24

3.4.5 Plant 24

4 Approach 26

5 System and Concept Description 27

5.1 Requirements Analysis 27

5.1.1 Overview 27

5.1.2 Use Cases 27
5.1.3 Functional Requirements 29
5.1.4 Environmental Requirements 30
5.2 Concept 30
5.3 Development 31
5.3.1 Components Used for the Development 31
5.3.2 Software Architecture 33
5.3.3 Database Tables 35
5.3.4 Highlighted Classes 36
5.3.5 Functional Specifications 39
5.4 Implementation and Testing 45
5.4.1 Acer Iconia Tab A500 45
5.4.2 Database Server 46
5.4.3 Fiduciary Markers 46
5.4.4 *SmartFactory*^{KL} 48
6 Evaluation and Outlook 53
7 Bibliography 55

1 Introduction

In November 2005, the International Telecommunications Union, an agency of the United Nations, presented its report about the ***Internet of Things***.

> *"The next step is to integrate things in a communication network. This is the vision of a truly ubiquitous network: anywhere, anytime, by anyone and with anything"*

In a chapter describing the future, they imagine the life of *"Rosa*, a Spanish student in year 2020" who wants to spend one weekend in the Alps. During her trip, she is supported by ubiquitous technologies: the sensors integrated in the tires of her car notifies a failure, a multimedia jacket with temperature settings, she has a videoconference with her boyfriend using video goggles and they meet on the way thanks to the integrated communication capabilities.

This idea illustrates where the future applications and technologies are moving towards: by providing real-time information about any object or device to, both the human and the machine, can act in the better and faster way [Del 07].

One of these new technologies that are recently emerging is **Augmented Reality (AR)**. *AR* enables a new way of seeing the real world where real elements are combined with virtual elements to create a mixed reality in real time. These virtual elements can be information overlays about the objects around us. In this way, AR allows to receive all relevant information at a glance and gives the ability to interact with them.

Augmented Reality, combined with a database access, allows the user to have real-time applications, and with that it is very easy to obtain updated information at every time. Such applications in an industrial environment are very useful because you can interact with the environment in the most effective and efficient way and as soon as possible.

An example of future production site is *SmartFactory*^{KL}, a manufacturer-independent research and demonstration plant. This plant is located in the *German Research Center for Artificial Intelligence* (DFKI) in Kaiserslautern (Germany).

With this idea, the ***Augmented.SmartFactory***^{KL} project was born, a tool to provide fast and concrete real-time information to technicians in a factory shop-floor. Thanks to this application they will know at a glance the status of every module or component inside the factory, and they will be able to localize any problem or incidence in a few seconds. This tool will be also useful as a communication system between workers due to the virtual

information board, where technicians will be able to write any comments concerning to the modules or components.

Augmented.SmartFactory[KL] is a further step to introduce last technologies in the world of manufacturing.

2 Task

This thesis aims to develop and implement an application-oriented concept for operator support in production environments on the basis of *Augmented Reality* (AR) and mobile personal systems.

Nowadays one of the main problems in the production chain is the delayed response of operators to potential failures and incidents that could happen in every process. New technologies can shorten that response in time or even make it disappear using real-time systems and in combination with *Augmented Reality*. Augmented Reality allows us, by overlaying information on the screen of a device, to view information located over any object you are pointing with the camera.

In this way, the operator who is in charge of repairing the damage could locate immediately the exact place and could also use this technology to repair it, as *Augmented Reality* allows designing applications that makes the task of fixing problems easier for technicians.

Therefore, the aim of this thesis is to design an application for the *Android Operating System* which is based on *Augmented Reality* technology and which is using real-time information to show all the data required for the maintenance of every modules of the *SmartFactory*^{KL} in an interactive and intuitive way. Each of those modules is identified with an unique marker and, when the application detects one of these markers, it displays on the device the current status and other useful information of the targeted module. At first glance it is possible to know the basic parameters of the module, and go ahead by clicking the appropriate button.

The application also allows the technicians to insert comments relating to each of the modules or components in the factory. In this way, the different technicians responsible for the maintenance of the *SmartFactory*^{KL} can communicate with each other typing messages which are stored in the database. These messages may have different priorities, but the messages with the high-priority are displayed on the screen of the application, showing a little star in the information panel. In these messages users provide feedback regarding the status or even give an order to a technician.

All the tests and demos of this application have been realized in the *SmartFactory*^{KL}, A short video was created to demonstrate the functionality of the application

3 State of the Art

The latest technologies have been used for the development of this project. One of them is the ***Augmented Reality* (AR),** which mixes both real and virtual world showing efficient and interactive real-time information.

Further uprising technologies are mobile and personal devices such as *SmartPhones* and *Tablet-PCs.* Here the ***Android Operating System*** is of central significance, since it was used in this project. It is the operating system of *Google* and, as shown in Figure 1, the one with the fastest growing from the last year to the next 5 years. During the *Q1 2011*, *Jeff Huber*, senior vice-president of commerce and local, announced that some 350,000 Android devices are being activated every day. That's almost 2.5 million a week [Nic 11].

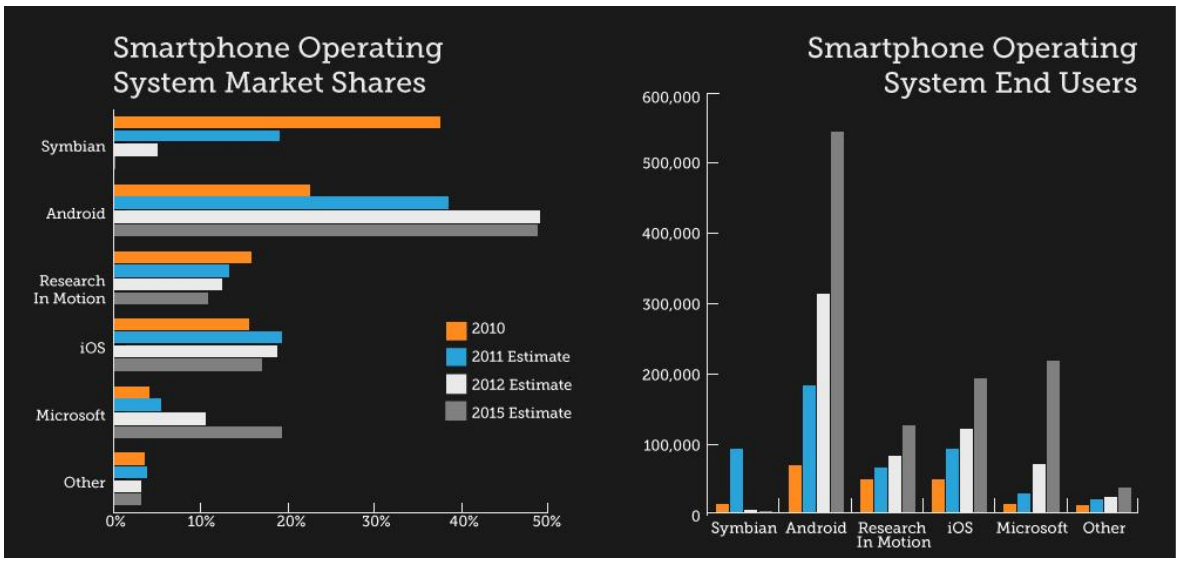

Figure 1: Mobile OS Battle by ***Lucid Agency*** *[Luc 11]*

Android OS is developed in ***Java*****,** a powerful programming language, and ***XML (Extensible Markup Language)*** for the configuration and the design of the user interfaces.

For implementing the *Augmented Reality* concept on the *Android OS* the library ***AndAR*** has been used. It was developed by *Tobias Domhan* in Stuttgart (Germany). It is a very new library, so some features have been added during the development of this project.

The application has been deployed on the ***Acer Iconia A500***, one of the newest *Tablet-PC* working with *Android 3.0*, which is especially dedicated to tablets. Finally the application has been implemented and tested in the *SmartFactory*^{KL}, a manufacturer-independent research and demonstration plant in Kaiserslautern (Germany).

In the next chapters all technologies and concepts which have been applied to this project are explained.

3.1 Augmented Reality

3.1.1 Overview

Augmented Reality (AR) is a variation of *Virtual Environments* (VE), or *Virtual Reality* as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therefore, AR supplements reality, rather than completely replacing it. Ideally, it would appear to the user that the virtual and real objects coexisted in the same space, similar to the effects achieved in the film "Who Framed Roger Rabbit?" Figure 2 shows an example of what this might look like. It shows a real desk with a real phone. Inside this room there are also a virtual lamp and two virtual chairs. Note that the objects are combined in 3-D, so that the virtual lamp covers the real table, and the real table covers parts of the two virtual chairs. AR can be thought of as the "middle ground" between VE (completely synthetic) and telepresence (completely real) [Mil 94a] [Mil 94b] [Azu 95].

Figure 2: Real desk with virtual lamp and two virtual chairs. (Courtesy ECRC)

3.1.2 History

In the following, a short overview about the history of the *Augmented Reality* development through the years [Jon 08]:

1957: *Morton Helig* began building a machine called the *Sensorama* (Figure 3). It was designed as a cinematic experience to take in all your senses and, shaped, rather like arcade machine from the 80s, it blew wind at you, vibrated the seat you sat on, played sounds to your eyes and projected a form of a stereoscopic 3D environment to the front and sides of your head.

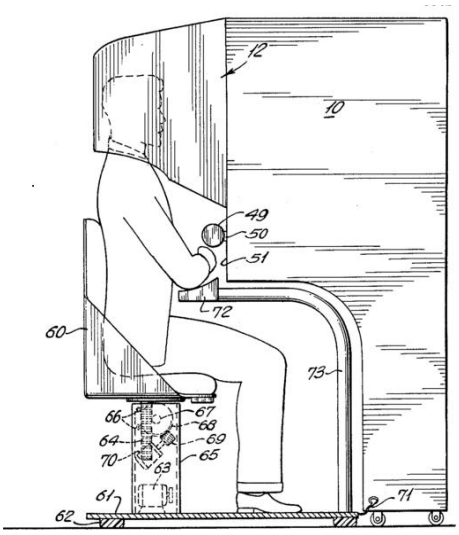

Figure 3: Sensorama

1966: *Ivan Sutherland* invents the head-mounted display (Figure 4) and positions it as a window into a virtual world.

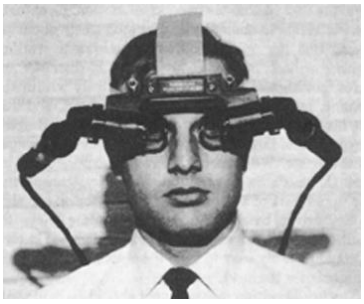

Figure 4: Sutherland Display

1975: *Myron Krueger* creates *Videoplace* to allow users to interact with virtual objects for the first time.

1989: *Jaron Lanier* coins the phrase *Virtual Reality* and creates the first commercial business around virtual worlds.

1990: *Tom Caudell* coins the phrase *Augmented Reality* while at *Boeing* helping workers assemble cables into aircraft.

1992: *L.B. Rosenberg* develops one of the first functioning AR systems, called *Virtual Fixtures*, at the *U.S. Air Force Research Laboratory*.

1993: *Loral WDL*, with sponsorship from *STRICOM*, performed the first demonstration combining live AR-equipped vehicles and manned simulators. Unpublished paper, J. Barrilleaux, "*Experiences and Observations in Applying Augmented Reality to Live Training*", 1999.

1994: *Julie Martin* creates first Augmented Reality Theatre Production, *Dancing In Cyberspace*, funded by the *Australia Council for the Arts*. The acrobats appeared immersed within the virtual object and environments. The installation used *Silicon Graphics* computers and *Polhemus* sensing system.

1999: *Hirokazu Kato* created *ARToolKit* at *HIT Lab*, where AR later was further developed by other *HIT Lab* scientists.

2000: *Bruce H. Thomas* develops *ARQuake*, shown in Figure 5, the first outdoor mobile AR game, demonstrating it in the *International Symposium on Wearable Computers*.

Figure 5: ARQuake

2008: *Wikitude AR Travel Guide* launches on Oct. 20, 2008 with the G1 *Android* phone.

2009: *ARToolkit* was ported to *Adobe Flash*, *FLARToolkit* by *Saqoosha*, bringing *Augmented Reality* to the web browser.

2010: Some *Augmented Reality* libraries are released for the new *Google* operating system, *Android*, like the *AndAR* library, implemented in *Java*.

3.1.3 Augmented Reality vs. Virtual Reality

Virtual Reality is a technology that encompasses a broad spectrum of ideas. It defines an umbrella under which many researchers and companies express their work. The phrase was originated by *Jaron Lanier* the founder of *VPL Research*, one of the original companies selling virtual reality systems. The term was defined as *"a computer generated, interactive, three-dimensional environment in which a person is immersed."* [Auk 92].

There are three key points in this definition:

First, this virtual environment is a computer generated three-dimensional scene which requires high performance computer graphics to provide an adequate level of realism.

Second, is that the virtual world is interactive. A user requires real-time response from the system to be able to interact with it in an effective manner.

Third, is that the user is immersed in this virtual environment. One of the identifying marks of a virtual reality system is the head mounted display worn by users. These displays block out the entire external world and present to the wearer a view that is under the complete control of the computer. The user is completely immersed in an artificial world and separated from the real environment. For this immersion to appear realistic the virtual reality system must accurately sense how the user is moving and determine what effect that will have on the scene being rendered in the head mounted display.

The discussion above highlights the similarities and differences between virtual reality and *Augmented Reality* systems. A very visible difference between these two types of systems is the immersiveness of the system.

Virtual Reality strives for a totally immersive environment. The visual, and in some systems aural and proprioceptive, senses are under control of the system.

In contrast, an *Augmented Reality* system is augmenting the real world scene necessitating that the user maintains a sense of presence in that world. The virtual images are merged with the real view to create the augmented display. There must be a mechanism to combine the real and virtual that is not present in other virtual reality work.

Developing the technology for merging the real and virtual image streams is an active research topic.

Milgram [Mil 94] describes a taxonomy that identifies how *Augmented Reality* and *Virtual Reality* work are related. He defines the *Reality-Virtuality* continuum shown as Figure 6.

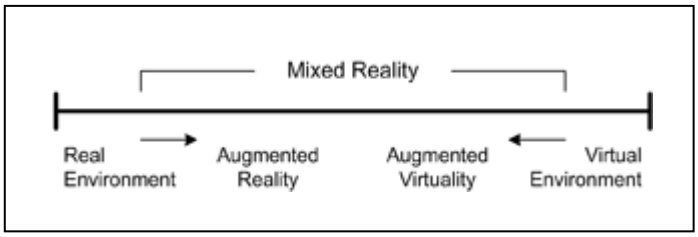

Figure 6: Milgram's Reality-Virtuality Continuum

The real world and a totally virtual environment are at the two ends of this continuum with the middle area called *Mixed Reality*. *Augmented Reality* lies near the real world end of the line with the predominate perception being the real world augmented by computer generated data. *Augmented Virtuality* is a term created by *Milgram* to identify systems which are mostly synthetic with some real world imagery added such as texture mapping video onto virtual objects.

This is a distinction that will show as the technology improves and the virtual elements in the scene become less distinguishable from the real ones [Val 06].

3.1.4 Augmented Reality Application Domains

Recently, the capabilities of real-time video image processing, computer graphic systems and new display technologies have converged to enable the display of a virtual graphical image correctly registered with a view of the 3D environment surrounding the user.

Researchers working with *Augmented Reality* systems have proposed them as solutions in many domains as:

- Medical
- Entertainment
- Military Training
- Engineering Design
- Consumer Design
- Manufacturing, Maintenance and Repair

The next section further explains this last domain, the *Augmented Reality in the Manufacturing* [Val 06].

3.1.5 Augmented Reality in Manufacturing

The breakthrough of AR technologies enables new innovations for the industry. An overview about AR applications in industrial manufacturing is given by *Azuma* in 1997.

Since then new applications were established, but a broad exploitation in manufacturing has not yet been reached.

In the following, typical problems within these application scenarios are shortly described, and it is shown how these can benefit by using AR [Sch11]:

Maintenance

Plant maintenance is generally characterized by high complexity and ambiguity: one maintenance operation might vary widely from another. The service technician has to deal with a large number of different devices and components and the processes and locations are often less well known or not known at all.
The correct sequence and execution of the maintenance task is crucial for the worker's safety and process reliability. Therefore AR delivers information about the devices and the sequence of the maintenance task. In this way instructions are easier to understand and the operator can stay focused on the on-going task.
By using *head-mounted-displays,* as shown in Figure 7, a hand-free montage is possible.

Figure 7: AR in Maintenance

Logistics

In manual order-picking tasks a worker has to collect several parts from different locations to complete the assignment. Depending on the dimensions of the storage and its segmentation this leads to varying picking times. In addition traditional picking methods contain several disadvantages. For instance it is impossible to use both hands during picking via picking lists without delay. Therefore AR offers two mayor advantages: hand-free operating and optimized picking routes which leads the worker directly to the needed parts.

Operator training

Today job rotation is a more and more common type of labour organization, i.e. one worker has to know different manufacturing tasks for products, parts and all of their

variants and works at different workstations. For flexible usage of workforce, workers need to be able to quickly adopt tasks from different departments or workstations. AR-systems can be used to support workers via on-the-job training: instructions of the given task are shown in the field of view of the worker. The system is able to monitor his work and give certain information, i.e. if the worker did a specific task wrong.

3.1.6 Augmented Reality System Architecture

This section will describe the components of which a typical *Augmented Reality* system is made up and will highlight the possibility of *Augmented Reality* as an area in which multiple technologies flow together into a single system. The fields of computer vision, computer graphics and user interfaces are actively contributing to advances in *Augmented Reality* systems [Val 06].

Typical Augmented Reality System

The next Figure 8 shows the multiple reference frames that must be related in an *Augmented Reality* system.

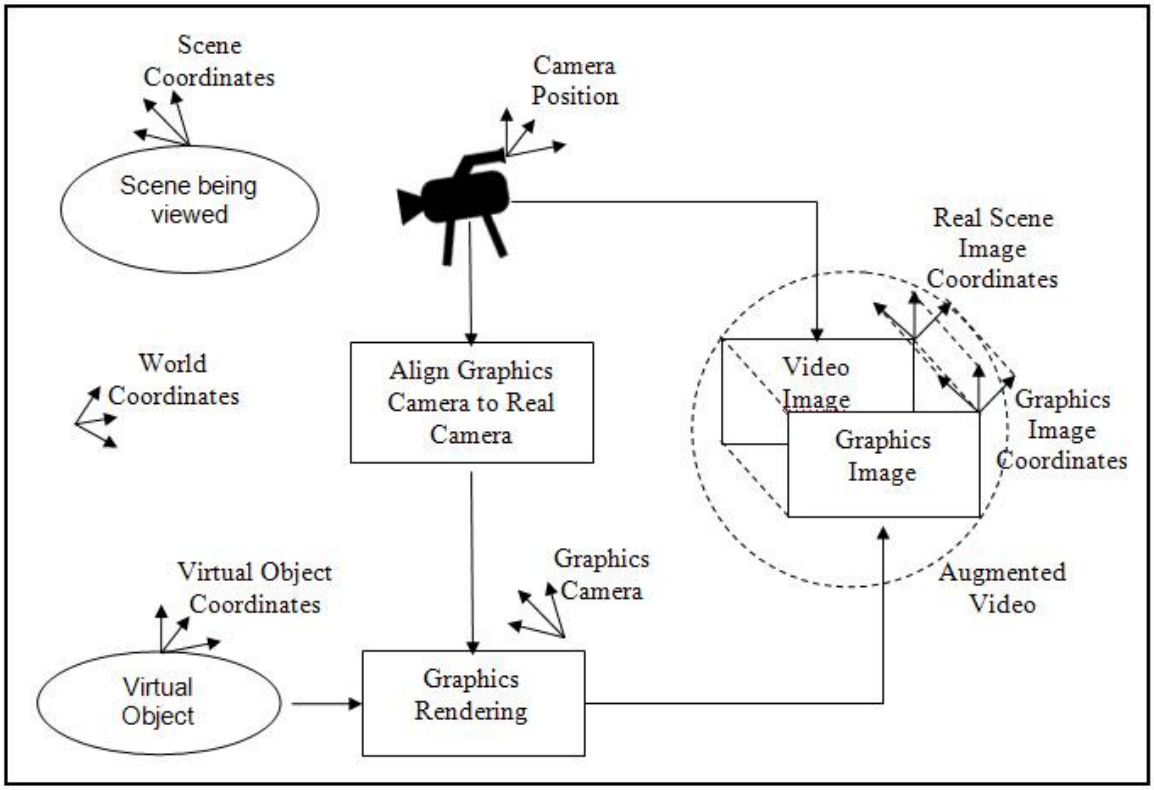

Figure 8: Components of an Augmented Reality System

The scene is viewed by an imaging device, which in this case is depicted as a video camera. The camera performs a perspective projection of the 3D world onto a 2D image

plane. The intrinsic (focal length and lens distortion) and extrinsic (position and pose) parameters of the device determine exactly what is projected onto its image plane. The generation of the virtual image is done with a standard computer graphics system. The virtual objects are modelled in an object reference frame.

The graphics system requires information about the imaging of the real scene so that it can correctly render these objects. This data will control the synthetic camera that is used to generate the image of the virtual objects. This image is then merged with the image of the real scene to form the *Augmented Reality* image.

Performance Issue

Augmented Reality systems are expected to run in real-time so that a user will be able to move freely within the scene and see a properly rendered augmented image. This places two performance criteria on the system which are:

- Update rate for generating the augmenting image,
- Accuracy of the registration of real and virtual image.

Display Technologies in Augmented Reality

The combination of real and virtual images into a single image results in new technical challenges for designers of *Augmented Reality* systems. The designer's decision of how to do this merging of the two images is a basic one.

To increase the sense of presence other display technologies are needed. Head-mounted displays *(HMD)*, as shown in Figure 9 have been widely used in virtual reality systems. Augmented reality researchers have been working with two types of HMD.

Figure 9: Video Head-Mounted Display

These are called video see-through and optical see-through. The "see-through" designation comes from the user's need to be able to see the real world view that is immediately in front of him even when wearing the HMD. The standard HMD used in virtual reality work gives the user complete visual isolation from the surrounding environment. Since the display is visually isolating the system must use video cameras that are aligned with the display to obtain the view of the real world.

The optical see-through HMD *(Manhart, Malcolm et al. 1993)* eliminates the video channel that is looking at the real scene. Instead, as shown in Figure 10, the merging of real world and virtual augmentation is done optically in front of the user. This technology is similar to heads up displays (HUD) that commonly appear in military airplane cockpits and recently some experimental automobiles. In this case, the optical merging of the two images is done on the head mounted display, rather than the cockpit window or auto windshield, prompting the nickname of HUD on a head.

Figure 10: Optical Head-Mounted Display

There are advantages and disadvantages to each of these types of displays. They are discussed in greater detail by *Azuma* [Azu 95]. There are some performance issues, however, that will be highlighted here. With an optical see-through display the view of the real world is instantaneous so it is not possible to compensate for system delays in other areas. On the other hand, with monitor based and video see-through displays a video camera is viewing the real scene. An advantage of this is that the image generated by the camera is available to the system to provide tracking information.

3.2 Android Operating System

3.2.1 Overview

On November 5th, 2007 leading technology and wireless companies came together to announce the future development of a truly open platform for all kinds of mobile devices – **Android**. Leading this development are *Google Inc*, *T-Mobile*, *Intel*, *HTC*, *Qualcomm*, *Motorola* along with many other companies under the umbrella of the *Open Handset Alliance* – a global alliance between technology and mobile industry leaders.

The *Open Handset Alliance's* common goal is to foster and develop a new breed of innovation for mobile devices allowing a far better user experience than today's current mobile platforms. The OHA will provide a far greater degree of openness that will enable developers to work and collaborate in ways never before seen, *Android* will greatly improve and speed up the process in which new and innovative mobile services are development and made available to the end user.

Through the development of *Android*, developers, manufacturers and operators will be far better positioned to ship out new and innovative products far quicker and far cheaper than today's standards. The *Android* platform will consist of an operating system, middleware, a user-friendly interface and powerful applications. This fully integrated bundle of software will significantly lower the current costs of developing mobile devices and services.

The *Android* platform is licensed under one of the most progressive open-source licenses available giving operators and manufacturers unprecedented freedom to design, build and distribute their own products [Tal 10].

3.2.2 Features

Current features and specifications of Android:

- **Handset layouts:** The platform is adaptable to larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications, and traditional SmartPhone layouts.
- **Storage:** SQLite, a lightweight relational database, is used for data storage purposes.
- **Connectivity:** Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
- **Web browser:** The web browser available in Android is based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine. The browser scores a 93/100 on the Acid3 Test.
- **Media support**: Android supports the most of the actual audio/video/still media formats.

- **Streaming media support**: RTP/RTSP streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic Streaming are supported by the Flash plug-in. Apple HTTP Live Streaming is supported by RealPlayer for Mobile, and by the operating system in Android 3.0 (Honeycomb). Microsoft Smooth Streaming is planned to be supported through the awaited port of Silverlight plug-in to Android.
- **Additional hardware support**: Android can use video/still cameras, touch-screens, GPS, accelerometers, gyroscopes, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers and accelerated 3D graphics.
- **Multi-touch**: Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.
- **Bluetooth**: Supports A2DP, AVRCP, sending files (OPP), accessing the phone book (PBAP), voice dialing and sending contacts between phones. Keyboard, mouse and joystick (HID) support is available through manufacturer customizations and third-party applications. Full HID support is planned for Android 3.0 (Honeycomb).
- **Video calling**: Android does not provide native video calling support, but some handsets have a customized version of the operating system that support it, either via the UMTS network (like the Samsung Galaxy S) or over IP. Video calling through Google Talk is available in Android 2.3.4 and later.
- **Multitasking**: Multitasking of applications is available.
- **Voice based features**: Google search through voice has been available since initial release. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards.
- **Tethering**: Android supports tethering, which allows a phone to be used as a wireless/wired hotspot. Prior to Android 2.2 this was supported by third-party applications or manufacturer customizations.

3.2.3 Architecture

The Figure 11 shows the major components of the Android operating system. Each section is described in more detail below [And 11].

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

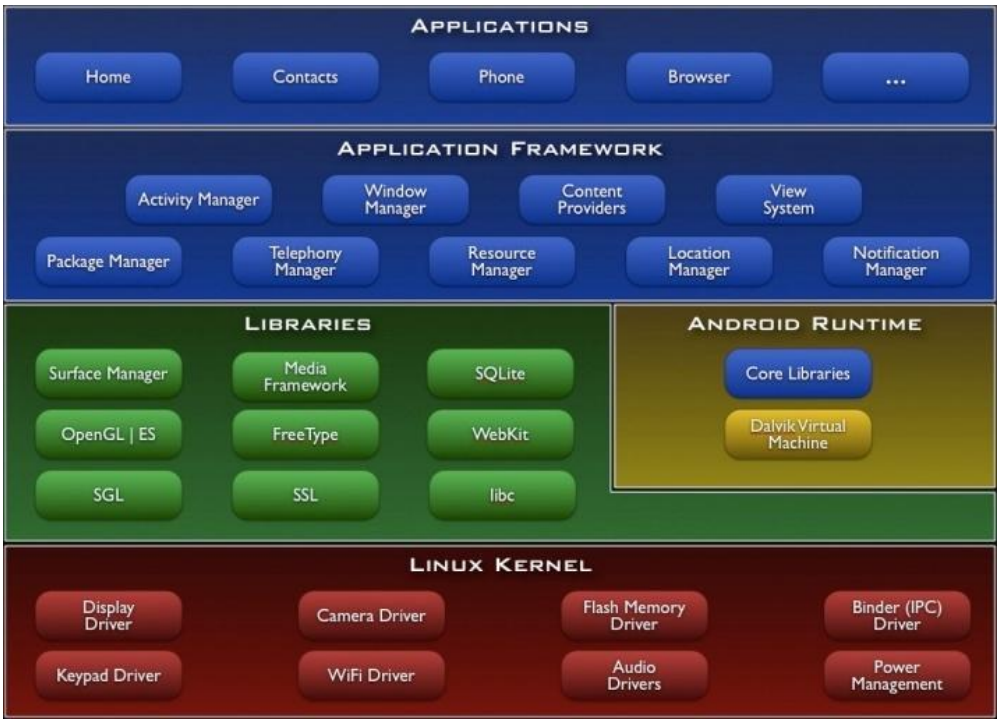

Figure 11: Android Architecture

Android runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **Free Type** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications

Application Framework

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Applications

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

3.2.4 Application Fundamentals

Android applications are written in the Java programming language. The Android SDK tools compile the code—along with any data and resource files—into an *Android package*, an archive file with an `.apk` suffix. All the code in a single `.apk` file is considered to be one application and is the file that Android-powered devices use to install the application.

While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on the Dalvik virtual machine. Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.

Once installed on a device, each Android application lives in its own security sandbox:

The Android operating system is a multi-user Linux system in which each application is a different user.

By default, the system assigns each application a unique Linux user ID (the ID is used only by the system and is unknown to the application). The system sets permissions for all the files in an application so that only the user ID assigned to that application can access them.

Each process has its own virtual machine (VM), so an application's code runs in isolation from other applications.

By default, every application runs in its own Linux process. Android starts the process when any of the application's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other applications.

In this way, the Android system implements the *principle of least privilege*. That is, each application, by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an application cannot access parts of the system for which it is not given permission.

However, there are ways for an application to share data with other applications and for an application to access system services:

It's possible to arrange for two applications to share the same Linux user ID, in which case they are able to access each other's files. To conserve system resources, applications with the same user ID can also arrange to run in the same Linux process and share the same VM (the applications must also be signed with the same certificate).

An application can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more. All application permissions must be granted by the user at install time [And 11].

3.2.5 Application Components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role—each one is a unique building block that helps define your application's overall behaviour.

There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

The four types of application components are [And 11]:

Activities
An *activity* represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email application, each one is independent of the others.

Services
A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface

Content providers
A *content provider* manages a shared set of application data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your application can access. Through the content provider, other applications can query or even modify the data (if the content provider allows it).
Content providers are also useful for reading and writing data that is private to your application and not shared.

Broadcast receivers
A *broadcast receiver* is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system like a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Applications can also initiate broadcasts—for example, to let other applications know that some data has been downloaded to the device and is available for them to use. Although broadcast

receivers do not display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work. For instance, it might initiate a service to perform some work based on the event.

Intents

A unique aspect of the Android system design is that any application can start another application's component. For example, if you want the user to capture a photo with the device camera, there is probably another application that does that and your application can also use it, instead of developing an activity to capture a photo yourself. You don't need to incorporate or even link to the code from the camera application. Instead, you can simply start the activity in the camera application that captures a photo. When complete, the photo is even returned to your application so you can use it. To the user, it seems as if the camera is actually a part of your application.

When the system starts a component, it starts the process for that application (if it's not already running) and instantiates the classes needed for the component. For example, if your application starts the activity in the camera application that captures a photo, that activity runs in the process that belongs to the camera application, not in your application's process. Therefore, unlike applications on most other systems, Android applications don't have a single entry point (there is no `main()` function, for example).

Because the system runs each application in a separate process with file permissions that restrict access to other applications, your application cannot directly activate a component from another application. The Android system, however, can. So, to activate a component in another application, you must deliver a message to the system that specifies your *intent* to start a particular component. The system then activates the component for you.

3.3 AndAR Library

3.3.1 Overview

AndAR is a project that enables *Augmented Reality* on the *Android* platform. The whole project is released under the GNU *General Public License*. It offers a pure Java API to the ARToolkit.

3.3.2 Structure

AndAR is an *Augmented Reality Framework* for *Android*. It not only offers a pure *Java* API but is also object oriented. The Figure 12 shows a simplified class diagram of an application that makes use of *AndAR*.

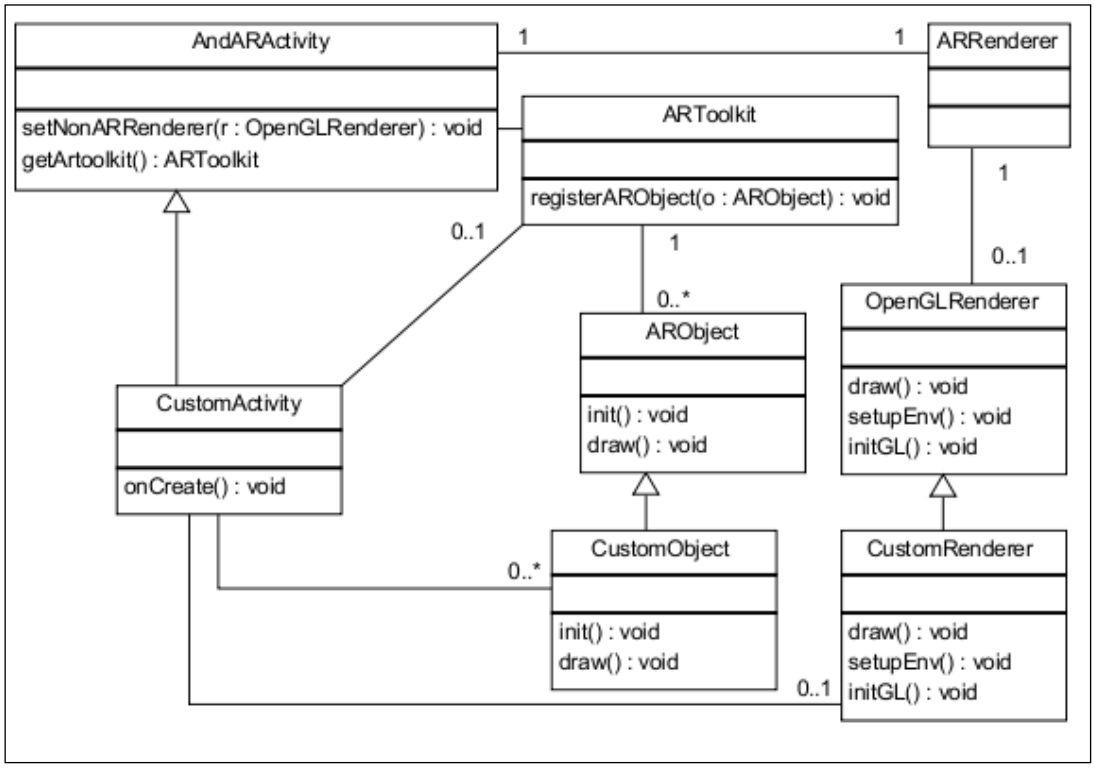

Figure 12: AndAR Class Diagram

Every *Android* application consists of one or more Activities. An *Activity* is a visual user interface, targeted to a single purpose. Only one may be active at a time. In order to write an *Augmented Reality* application, one has to extend the abstract class *AndARActivity*. This class already handles everything related to *Augmented Reality*, like opening the camera, detecting the markers and displaying the video stream. The application would run already, by just doing that. However it would not detect any markers.

In order to do so, you have to register *ARObjects* to an instance of *ARToolkit*. This instance can be retrieved from the *AndARActivity*. The *ARObject* class itself is abstract. This means, it has to be extended, too. It expects the file name of a pattern file in its constructor. This file must be located in the assets folder of the *Eclipse* project.

Pattern files can be created by a tool called *mk_patt*, found in the *ARToolkit* website. They are used to distinguish different markers. In order to draw a custom object, the method draw has to be overridden. Before this method is invoked a transformation matrix will

already have been applied. This means the object will be aligned to the marker, without any further steps. This method will not be invoked, if the marker belonging to this object is not visible.

The class *ARRenderer* is responsible for everything OpenGL related. If you want to mix augmented with non augmented 3D objects you may provide a class implementing the *OpenGLRenderer* interface. There are three methods defined by this interface. *initGL* being called only once, when the OpenGL surface is initialized. Whereas *setupEnv* is called once before the augmented objects are drawn. It can be used to issue OpenGL commands that shall affect all *ARObjects*, like initializing the lighting. In the draw method you may draw any non augmented 3D objects. It will be called once for every frame. Specifying in this way the described renderer is optional.

The *AndARActivity* furthermore offers a method that allows the application to take screenshots.

During the development of this project, some of the methods of this library have been modified to add new functionalities to the Augmented Reality. They will be explained in the next chapters.

3.4 *SmartFactory*^{KL}

3.4.1 Introduction

The manufacturer-independent research and demonstration plant of *SmartFactory*^{KL} (Figure 13) illustrates the vision of the factory of the future. Diverse components of different manufacturers (networking) are connected with each other within it. Intelligent components are able to take over independently contextual tasks and to work self-sufficiently (self-organization).

Figure 13: View on the demonstration plant SmartFactoryKL

In addition to that, *SmartFactory*^{KL} is modifiable and extendable in a user-defined way (flexibility). And last but not least, major focus is put on the user-friendly configuration of the operating systems (user orientation) despite the fact of increasing complexity [Sma 2011].

3.4.2 Description

Products and concepts of information and communication technology (ICT) are applied in many areas of life. They rapidly develop in respect to their fields of application and their capability. The devices used in every-day life and application of consumer's electronic systems like *SmartPhone*, navigation systems, *Bluetooth* or *Wireless-LAN* are to replace traditional principles of operation and conception in industry since these so-called *"smart"* technologies have not yet entered into the world of a factory.

SmartFactory^{KL} was established in order to change this and to transfer new technologies and concepts into industrial environments in a useful way. Innovative systems of automation are tested, refined and supplemented continuously by using the example of a complete typical industrial production plant. Thus, *SmartFactory*^{KL} is the interface between research and industrial practice.

The technology initiative *SmartFactory*^{KL} has been established as an association in order to develop new ideas with partners and to put these ideas into practice in common projects which range from fundamental work on basic technologies to development of marketable products. Within that the transfer between science and application is always on the focus. Members, sponsors and promoters create a living partnership in order to realize the vision of a future industrial landscape with modern and innovative means [Sma 2011].

3.4.3 Architectures

Due to the availability of new intelligent field devices and thus the increasing decentralized automation, there is a need for methods which reduce integration effort(s). The concept of the service-oriented architecture (SOA) deriving from enterprise software is a powerful decentralized approach to integrate software modules with defined functionality into large IT systems. This offers a great opportunity for the area of automation technology to complement with adequate software architecture the developments in the field of mechatronic functionality integration and intelligent field devices.

Research topics are the transfer of architectures, methods, protocols and tools from the area of business software to the field of automation as well as the realization of the *Plug&Play* principle in industrial field devices [Sma 2011].

3.4.4 Projects

Within a project, new innovative technologies are developed until they reach industrial feasibility by including interested partners. The technologies which are provided by the partners are tested and extended at SmartFactoryKL. In this connection, SmartFactoryKL can be considered as a platform which enables the transfer between science and industry. Fundamental research and applied science are linked with each other.

BMBF Top Cluster

Within the frame of the high-tech top cluster competition of BMBF, *SmartFactory*^{KL} stands as one of three *Living Labs* in the focus of the development and evaluation of emergent software for the domain of process and production industry.

EU Cognito

Cognito is a project of the 7th EU research framework program and will last three years. Together with six project partners from England, France, Germany and Portugal, a user-centered assistance system will be developed which supports complex assembly operations based on *Augmented Reality Technologies*.

SemProM

Smart labels provide products a memory and support intelligent logistics. Within the ICT-2020 research program of the *German Ministry of Education and Research* (BMBF), the innovation alliance *"Digitales Produktgedächtnis"* ("Digital Product Memory") develops with the joint project *SemProM* key technologies for the *Internet of Things*.

Demonstrator development for parameterization of field devices via Bluetooth

First, the project aimed to design a Proof of Concept to demonstrate the possibility of wireless parameterization and control of field devices. This shall show the potential of a universal, this means a manufacturer- and component-independent, several standards dominating and on standard operator units executable, operator system. The result is a java-program which can be used on several mobile phones and can communicate on different channels with 20 single components in *SmartFactory*^{KL} at the moment.

3.4.5 Plant

Central research and demonstration platform of the *SmartFactory*^{KL} is its hybrid demonstration plant (Figure 14). It contains a procedural and technical production of piece goods, and is built strictly according to industrial standards.

Figure 14: SmartFactoryKL demonstration plant

The plant can produce a customized product (soap bottles, shown in Figure 15) in the batch size 1 to customer specification. Terms of requirements, structure and complexity of the laboratory system with industrial production in practice is absolutely comparable. Functional electrical components (controllers, sensors, actuators) from different vendors are flexible networked. Both within the system as well as to the overall control levels many communication systems are running wireless.

Together with the modular design of the system it has been achieved the greatest possible degree of flexibility: reconfiguration of individual process steps and integration of new modules into the overall system is very simple.

Figure 15: SmartFactoryKL customized product

The plant forms the core of the demonstration center, where the interaction of individual "smart" technologies into a functioning whole can be tested and experienced.

4 Approach

This project was born from the concept of providing a tool to support the technicians in the field of industrial manufacturing. After studying the needs identified, the decision of the technology to be used for the development of the application was taken, concluding that the new concept of *Augmented Reality* could greatly facilitate the work of maintenance and communication between technicians within an industry.

For the development of this project various phases were defined at the beginning. They are discussed in detail in following chapters.

Requirements Analysis

This section is intended to describe what the application is supposed to do. Here, the two main use cases are explained: the *Information Scenario* and the *Communication Scenario.* Also the *Functional Requirements* are described. This means the expected behaviour of the application. And, at last, the *Environmental Requirements,* the physical conditions for the correct working of the system are introduced.

Concept

Once defined the requirements, it is necessary to explain the concept of the system, which means the abstract idea of it. This concept is explained with the help a paper written by some engineers of the *German Research Center for Artificial Intelligence* for the *International Conference on Manufacturing Science and Education* in year 2011 in *Sibiu* (Romania).

Development

In this step the abstract idea becomes concrete. Therefore, some tools and applications are necessary. In this section the *Components Used for the Development*, like *Eclipse* or *MySQL,* the *Software Architecture* that will be used by the application, the *Database Tables* needed for the storage of the data, some of the most important *java classes* used and a description about every functions concerning each of the screen developed for the application are explained.

Implementation and Testing

Once the application has been developed it is time for its implementation and tests to proof its correct behaviour. Every test sessions have been realized in the *SmartFactory*^{KL}, a demonstration plant in Kaiserslautern (Germany). The device used for launching the application has been the *Acer Iconia Tab A500*, one of the first devices with the *Android 3.0 O.S.* For the data storage, the technicians working in the *SmartFactory*^{KL} created a *MySQL* database, the device connection works via a *Wireless-LAN,* For the identification of the modules some *Fiduciary Markers* have been created.

5 System and Concept Description

Every work done in this project is fully described in this chapter, from the concept to the final version of the application. There are also references to future developments and some improvements.

5.1 Requirements Analysis

5.1.1 Overview

The main objective of the *Augmented.SmartFactory*[KL] application is to provide efficient and effective real-time information to technicians in a manufacturing industry.

The application should detect the markers in each module, recognise them, and display the status information and other parameters.

It should also implement a virtual communication board where technicians can write messages concerning any incidence or problem detected in the modules.

5.1.2 Use Cases

The application is designed to satisfy two specific use cases. The first one should provide the real-time and status information to the technicians. The second use case deals with the possibility of providing a communication system to the users. They both are fully explained hereafter.

First Use Case: Information Scenario

The user or technician launches the application in the device. Once the camera is ready, it is time to start with the *Augmented Information*. The user can walk around the whole plant pointing to the markers placed at the modules. When a marker is detected by the camera, information related to the module and a virtual box indicating the state with its colour are displayed on the screen, as shown in the Figure 16 . The user can also easily identify if there is any problem looking to the colour of the *virtual box* placed on the marker. The user can also browse to the *Module Screen* where he finds more information about the

module. Here the user can see information related to the *specification, maintenances, live status, etc.*

Figure 16: First Use Case, Information Scenario

Second Use Case: Communication Scenario

The user or technician launches the application on the device. Once the camera is ready it is time to start with the *Augmented Communication*. When the camera detects a marker and the module information is displayed on the screen, the user can check if this module has a message associated. There are new messages when a little star is drawn beside the module picture in the *Information Panel* on the right of the screen. Clicking on this star, an *information board* is displayed, showing the messages posted in the module, like in the Figure 17.

Figure 17: Second Use Case, Communication Scenario

The user can answer to these messages or write new ones going to the *Module Screen*. There, the user can check the date, the priority and the owner of the message. Only the high priority messages will be displayed in the *Information Panel.*

5.1.3 Functional Requirements

- The application should connect to the database hosted on the server at the initialising time
- If the application is not connected to the database it will not work
- The user can change the server *URL* in the options menu. The application will try to reconnect to the server with the new *URL*
- Users can go to the *Module Screen* through the *Augmented Camera* or through the *Modules List*
- In the *Modules List* the user can select one of the module listed and go to the *Module Screen* to interact with it
- When the user select the *Augmented Camera* option, the camera will be initialised and the application will wait to one marker to be pointed to
- When a marker is detected, the application will recognise it and will display the data stored in the database about that module
- Together with the module information two buttons will be shown. One is the *More Information* button, its function is to show the *Module Screen*. The other is the *Take a Screenshot* button, for taking a picture of the virtual objects of the screen
- In the *Module Screen* the user can browse through the different tabs as well as read and write message on the *Information Board.*
- At any time, the user can go back to the *Main Menu* pressing the back button on the left-bottom of the screen

5.1.4 Environmental Requirements

- The application should be executed on a *SmartPhone* or *Tablet-PC*.
- The operating system should be *Android O.S.*
- The application needs a server with a database to get the data. The database created for the project is based in *MySQL*
- A *PHP Servlet* is required to enable the communication between the device and the database
- A *Wireless-LAN* is needed for the connection between the device and the server
- The factory plant should be equipped with an adequate illumination, necessary for the right performance of the camera
- Each module should have its unique *Fiduciary Marker* placed on it. The markers should be clearly-defined and composed of a black square on a white background and, inside this square, a big character identifying the module

5.2 Concept

The concept behind the *Augmented.SmartFactory*^{KL} is to design an application which runs on a *Tablet-PC* or *SmartPhone*. With these devices, the service technicians in a manufacturing plant can identify and receive real-time information regarding the current state of the production line, which includes e.g. the process configuration and machine parameters. Also the complete real-time overview of the whole process can be outlined and displayed, highlighting the real important information. E.g. if it is considered that the current field of view is the terminal phase of the production-line and the remainder of the production-line is not visible, and an error occurs in the initial or middle stages, the error message is shown to the user, although this information is not related to the current field of view, but it is important [Sch 11].

Another feature of the application is the possibility of insert comments relating to each of the modules or components in the factory. In this way, the different technicians responsible for the maintenance of the *SmartFactory*^{KL} can communicate with each other typing messages which are stored in the database. Thanks to these messages is possible to have an historical record of each module, with information about every actions and maintenances performed on them.

The application is implemented as video see-through *Augmented Reality*, where virtual images and texts are superimposed on a live video of the real world. The idea is that each of the modules of the *SmartFactory*^{KL} has *Fiduciary Markers* put on them. These markers can then be tracked via a camera integrated in the *Tablet-PC* or *SmartPhone*. The application is used to display important virtual, text-based information, augmented into the

video input of the camera, and associated with each module. The application gives an overview of the current, real-time process state, e.g. parameters such as temperature, pressure, status and utilization [Sch 11].

5.3 Development

In this chapter every development details, the tools used and some diagrams are specified. Every class has been programmed exclusively for this project, with the exception of the *AndAR library*, although some of the classes of this library were modified to satisfy some needs of the application.

5.3.1 Components Used for the Development

In the following every component or applications which were used for the development of this project are explained:

Eclipse Classic 3.7

The *Augmented.SmartFactory*[KL] application has been developed in *Eclipse*.

Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle. It is known as a *Java IDE* but *Eclipse* is much more than a Java IDE.

The *Eclipse* open source community has over 200 open source projects. These projects can be conceptually organized into seven different "pillars" or categories:

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)

The Eclipse community is also supported by a large and vibrant ecosystem of major IT solution providers, innovative start-ups, universities and research institutions and individuals that extend, support and complement the *Eclipse Platform*.

The *Android SDK* and the *ADT plug-in* are necessaries for developing *Android* applications in *Eclipse.* They can be found at the *Android* website [Ecl 11].

Android SDK

The Android SDK provides the tools and libraries necessary to begin developing applications that run on Android-powered devices. Some of these tools are:

- Development environment
- Debugging environment
- Libraries
- Phone Emulator
- Documentation
- Tutorials
- Example code

Apache HTTP Server

The *Apache HTTP Server* Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including *UNIX* and *Windows NT*. This project aims to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

Apache httpd has been the most popular web server on the Internet since April 1996, and celebrated its 15th birthday as a project this February.

The *Apache HTTP Server* ("httpd") is a project of *The Apache Software Foundation* [Apa 11].

This is the *HTTP Server* used in this project for the communication between the application and the database hosted on a server in the *SmartFactory*^{KL}. A *PHP Servlet* is used for the data request and responses.

PHP 5.3.5

PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

What distinguishes PHP from something like client-side *JavaScript* is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was [Php 11].

The *Servlet* used by the application to retrieve the data from the database is completely done in this language. *PHP* has native tools for database management, allowing queries, updates and deletes in an easy way.

MySQL 5.5.8

The *MySQL* database has become the world's most popular open source database because of its high performance, high reliability and ease of use. It is also the database of

choice for a new generation of applications built on the LAMP stack (*Linux*, *Apache*, *MySQL*, *PHP / Perl / Python*.) Many of the world's largest and fastest-growing organizations including *Facebook*, *Google*, *Adobe*, *Alcatel Lucent* and *Zappos* rely on *MySQL* to save time and money powering their high-volume Web sites, business-critical systems and packaged software.

MySQL runs on more than 20 platforms including *Linux, Windows, Mac OS, Solaris, HP-UX, IBM AIX*, giving a great flexibility for the control. For begginers to database technology, experienced developer or DBA, *MySQL* offers a comprehensive range of database tools, support, training and consulting services to make them successful [Ora 11].

This server is responsible for the storage of the application data. For the project only two tables have been created, but the server can maintain several of them.

WampServer 2.0

WampServer is a *Windows* web development environment. It enables a creation of web applications with *Apache*, *PHP* and the *MySQL* database. It also comes with *PHPMyAdmin* to easily manage the databases.

WampServer installs automatically (installer), and its usage is very intuitive. It is possible to tune your server without even touching the setting files.

WampServer is the only packaged solution that will allow one to reproduce a production server. Once WampServer is installed, it is possible to add as many Apache, MySQL and PHP releases as one wants [Wam 11].

5.3.2 Software Architecture

This system implements a typical *Client-Server (C/S)* based architecture. The *Client–Server* model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called *Servers*, and service requesters, called *Clients*. Often clients and servers communicate via a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Features

In this architecture, the sender of a request is known as *Client*, and its features are:

- Sends requests, which means that it has an active role in communication (master device)

- Waits and receives responses from the *Server*
- Usually, it can connect to multiple servers at once
- Typically, it interacts directly with end-users through a graphical user interface

The recipient of the request sent by the client is known as *Server.* Its features are:

- At the beginning, it waits for requests from clients, and then it plays a passive role in communication (slave device)
- Upon it receives a request, the server processes it and sends the response to the client
- Usually, it accepts connections from a large number of clients
- It is not common that it interacts with end-users directly

Advantages

- ***Control centralization***: Access, resources and data integrity are controlled by the server, so that an unauthorized or defective client application cannot damage the system. This centralization also facilitates the task of updating data or other resources
- ***Scalability***: It is possible to increase the capacity of clients and servers separately
- ***Easy Maintenance***: Roles and responsibilities are distributed among several independent computers, so it is easy to repair, replace, update or even move a server while the clients will not be affected by this change. This is also know by *encapsulation*

Disadvantages

- ***Traffic congestion:*** This was always a problem in this architecture. When a large number of clients sends simultaneous requests to the same server, it can cause many problems
- ***Less robustness:*** One of the typical problems of the C/S architecture is that when a server is down, it cannot satisfy the requests of the clients
- ***High cost:*** Server software and hardware are here very decisive. A regular PC hardware may not be able to serve several numbers of clients. Specific hardware and software is needed, which increase the costs
- ***Less resources:*** The client does not have the same resources than the server

In this case, the *Augmented.SmartFactory*^{KL} application is the *Client*, and the machine hosting the database with all the data about the modules in the *SmartFactory*^{KL} is the *Server.* The application sends requests to the server with the information required, and the server answer with that information. This type of server is called *Database Server*, and provides database services to the client application.

As shown in Figure 18, the application installed on a *Tablet-PC* is conected through *Wireless-LAN* to the server. The server is hosting a *MySQL database* which is updated frequently with the data of the *SmartFactory*^{KL}. Every module of this factory is identified with the *Fiduciary Marker* and when the camera of the *Tablet-PC* detects one of these markers, the application makes a request to the server asking for the appropriate data. The technician can see this data on the screen of the device.

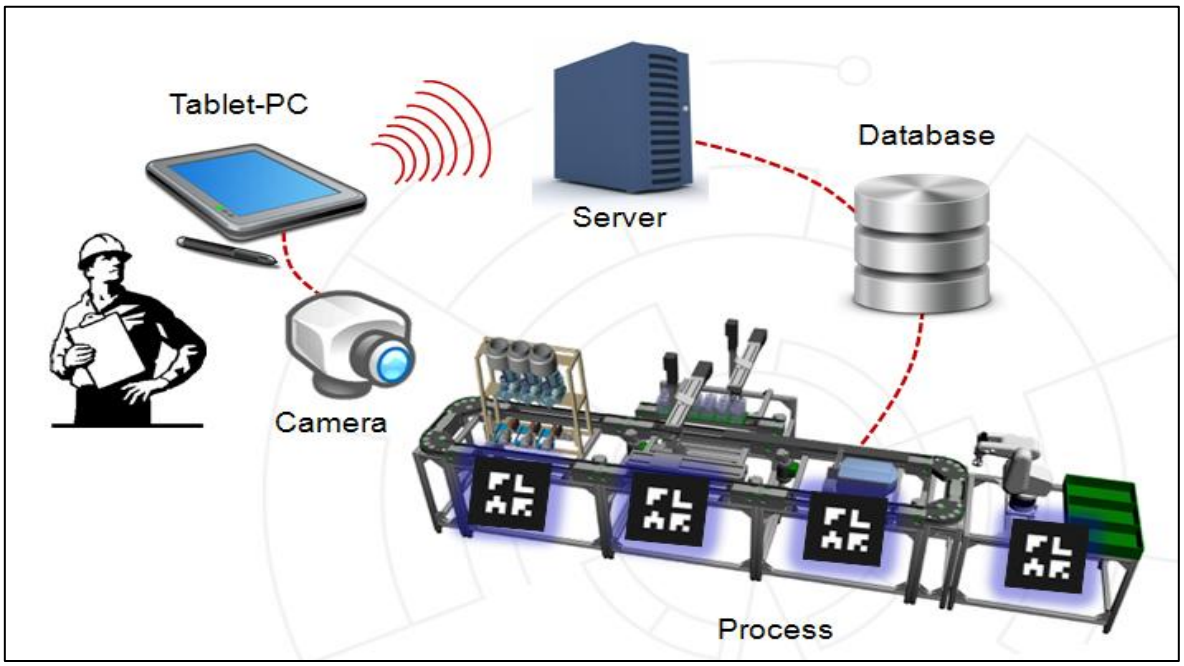

Figure 18: System Architecture

5.3.3 Database Tables

The application needs two tables for store the data. One of them is *Modules* whit all the information concerning the modules in the *SmartFactory*^{KL}. The other table is *Maintenance,* where the comments and actions related with the maintenance of every module are stored. These tables are explained hereafter.

Table 'modules'

The structure of this table is shown in the next creation script:

```
CREATE TABLE IF NOT EXISTS `modules` (
  `id_module` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `description` varchar(2000) NOT NULL,
  `ins_date` date NOT NULL,
  `icon` varchar(40) NOT NULL,
  `image` varchar(40) NOT NULL,
  `pattern` varchar(6) NOT NULL,
```

```
  `status` int(11) NOT NULL,
  `utilization` int(11) NOT NULL,
  `consumption` float NOT NULL,
  PRIMARY KEY (`id_module`)
)
```

The fields of this table are:

- *id_module:* the identifier of the module and the primary key of the table
- *name:* the name of the module which will be shown in the application
- *description:* the full description of the module
- *ins_date:* the date when the module was inserted into the database
- *icon:* the name of the icon file used for the lists
- *image:* the name of the image file of the module
- *pattern:* the name of the pattern file used to identify the module with *AR*
- *status:* an integer describing the status of the module: *(0: OK, 1:ERROR)*
- *utilization:* the utilization of the module in percentage
- *consumption:* the consumption in watts of the module

Table 'maintenance'

The structure of this table is shown in the next creation script:

```
CREATE TABLE IF NOT EXISTS `maintenance` (
  `id_mant` int(11) NOT NULL AUTO_INCREMENT,
  `id_module` int(11) NOT NULL,
  `date` date NOT NULL,
  `user` varchar(50) NOT NULL,
  `priority` int(1) NOT NULL,
  `description` varchar(3000) NOT NULL,
  PRIMARY KEY (`id_mant`),
  KEY `id_module` (`id_module`)
)
```

The fields of this table are:

- *id_mant:* the identifier of the maintenance and the primary key of the table
- *id_module:* the id of the module owner of this maintenance
- *date:* the date when the maintenance was inserted
- *user:* the name of the user who wrote the maintenance
- *priority:* an integer describing the priority of the maintenance: *(0:Normal, 1:High)*
- *description:* the full description of the module

5.3.4 Highlighted Classes

In this section the most important and significant classes in the project will be explained:

ARObject.java

Inside this class, the augmented object is defined. This kind of objects are the registered ones by the *ARToolkit*, and when the application detects the pattern of this object, it executes by default the *draw(GL10 gl)* of this class. This method is defined here but it can be overridden by the class extending this parent class, like in the *CustomObject* class, explained later, and it is used for the drawing of the virtual box on the screen.

Some modifications have been done in this class to allow the application showing the information about the module detected. The abstract method *text()* has been added here to show this information. This method will be overridden in the class extending this *ARObject.* That class will be explained forward. To let the application knows what method should be executed one more attribute has been added. It is the *int show*, and its value will represent if the object should be drawn o texted in the *ARToolkit* class, which will be explained in the next section.

```
//final values for the show parameter
public static final int SHOW_MODEL=0;
public static final int SHOW_TEXT=1;

//parameter for setting up the method to execute
private int show = ARObject.SHOW_MODEL;

//method to show the information on the screen
public abstract void text();
```

ARToolkit.java

This is the most important class of the *AndAR* library. Here is where most of the functions concerning the *Augmented Reality* are implemented, like the method to register the *ARObjects* with *registerARObject(ARObject arobject)*. This method stores all the registered objects in a *Vector* and it will be explored to know the detected object.

When an object is detected the method *draw(GL10 gl)* written here is executed, and here is where another modification has been done. This method now check the *show* attribute of the object, and depending of it, it will execute the method *draw()* or the method *text()*.

```
public final void draw(GL10 gl) {
      if(initialized) {
            if(Config.DEBUG)
                  Log.i("MarkerInfo", "going to draw opengl stuff now");
            //explore the arobjects vector
            for (ARObject obj : arobjects) {
                  if(obj.isVisible()){
                        if (obj.getShow() == ARObject.SHOW_MODEL)
                              obj.draw(gl);
                        else
                              obj.text();
                  }
```

```
            }
        }
}
```

AugmentedInfoActivity.java

This class implements *Augmented Reality* in the application. For that, it extends the *AndARActivity* class of the *AndAR* library.

In the constructor every object is registered into the *ARToolkit,* after consulting them on the database. There are two types of objects: *CustomObjects,* for drawing the *virtual box* on the screen when a marker is detected, and the *InformationObjects,* for displaying the information concerning the module detected on the right panel of the display. These two classes will be explained later.

The *AndAR* library, as a extension of the *ARToolkit,* has a specific method for register the objects. So, after the instantiation of these objects, they are registered like:

```
//instance of the CustomObject
object = new CustomObject("test1",pattern,80.0,new double[]{0,0}, color);

//register the object created
artoolkit.registerARObject(object);
```

With this method, the object is registered into the *ARToolkit* memory, so it is ready to be detected by the camera.

This class also implements the method for taking the screenshot of the virtual objects drown on the screen. For that an asynchronous task is used:

```
//asynchronous task for the screenshot
class TakeAsyncScreenshot extends AsyncTask<Void, Void, Void>
```

CustomObject.java

This class is used to draw the *virtual boxs* on the screen, and for that, it extends the class from the *AndAR* library, *ARObject.* Every class with this extension should have a method called *draw(GL10 gl),* that will be executed every time the camera detects a marker. In this class, the method draws a box by calling the method *draw()* of the *SimpleBox* class after some rendering settings:

```
@Override
public final void draw(GL10 gl) {
      super.draw(gl);
      //render settings
      gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_SPECULAR, flash);
      gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_SHININESS, flash);
      gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, diffuse);
      gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, ambient);
```

```
      //color and position
      gl.glColor4f(0, 1.0f, 0, 1.0f);
      gl.glTranslatef(0.0f, 0.0f, 12.5f);

      //draw the box
      box.draw(gl);
}
```

InformationObject.java

This is the class used for showing the real-time information about the module detected in the right panel.

When the application creates an instance of this class, it connects to the database to retrieve all the information that will be shown about the module. The parameter to identify the module is the name of the *pattern*.

When the camera detects a marker, unlike the *CustomObject.java* this class executes the method *text()*. This method writes all the information returned by the database in the User Interface. For this, the method should call the *runOnUiThread(new Runnable),* because only the User Interface Thread can interact with the elements on the screen:

```
@Override
public void text() {
      aug.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                  showInformation();
            }
      });
}
```

5.3.5 Functional Specifications

In this section the requested behaviour of the application is explained. The description is divided in screens for the best understanding of the reader.

Main Menu

This is the first screen of the application (Figure 19). It should display a menu with four options:

- *Go to Modules Maintenance:* By pressing this option the application will show another screen with a list of every module in the database
- *Start the Augmented Camera:* This option will launch the camera with the *Augmented Reality*
- *Settings...:* If the user presses this button, the *Settings Screen* will appear, where some options can be configured

- *About:* This option will show a panel with the information about the creators and developers of the application

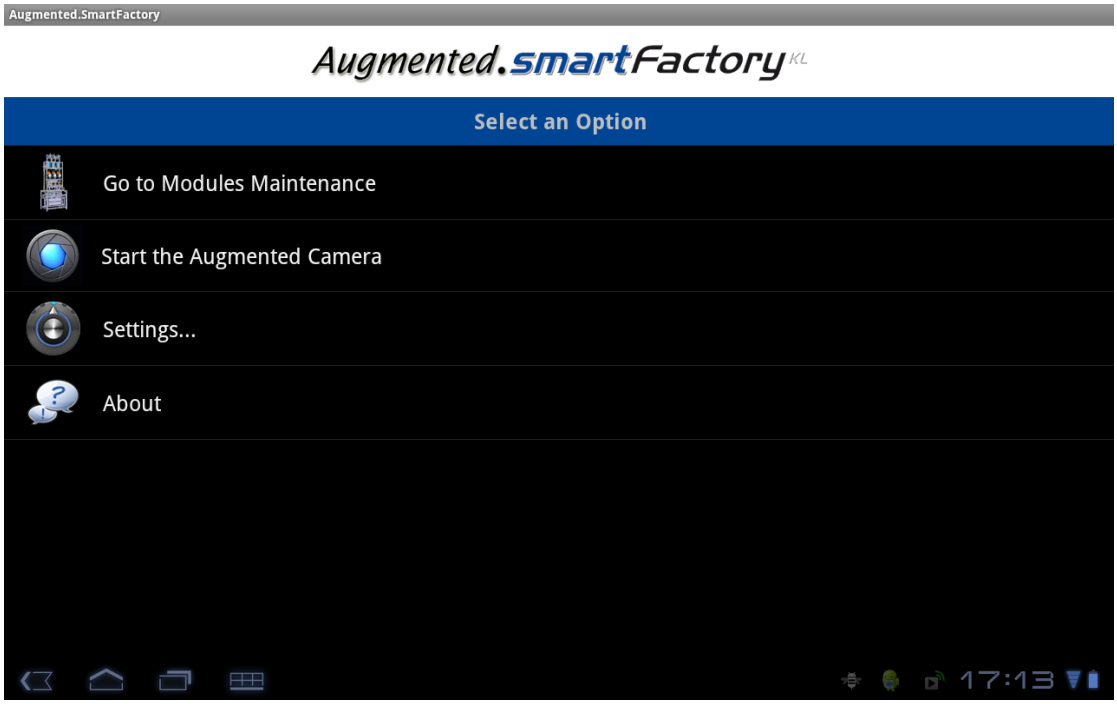

Figure 19: Main Menu

If the application is not able to connect to the database in the server and any of the first two options are pressed, the application will produce an error message with the cause of the problem.

Modules List

In this screen (Figure 20) a list with every module stored in the database is shown. The list is composed of a small icon of the module and the name of it. By pressing on any of them the application will show the *Module Screen* with all the information about it.

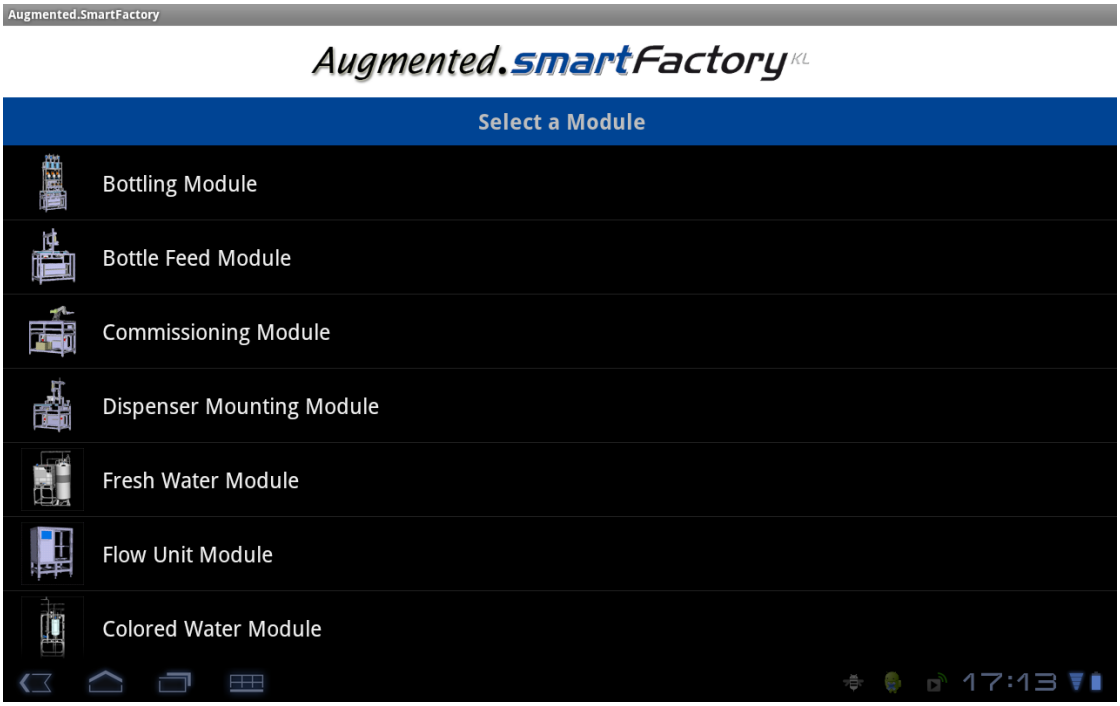

Figure 20: Modules List

Module Screen

This is the main screen concerning the module information (Figure 21). Here, the live status, utilisation, consumption, maintenance and other features about the module is displayed. The user can see four different tabs in the top side of the screen, and pressing on them, the screen will change showing the proper information. These tabs are:

- *Live Status:* If this tab is active the user can see the main live information of the module: the status, the utilisation and the consumption. By pressing on any of these options a pop-up panel will be shown with charts about the historical data concerning the actual module *(not implemented)*
- *Description:* This tab shows a short description about the module and about its functionality
- *Specifications:* Here the user can see some technical information about the actual module *(not implemented)*
- *Maintenance:* In this tab various instructions about the maintenance of the module will be shown. This will help to the technicians in the maintenance works *(not implemented)*

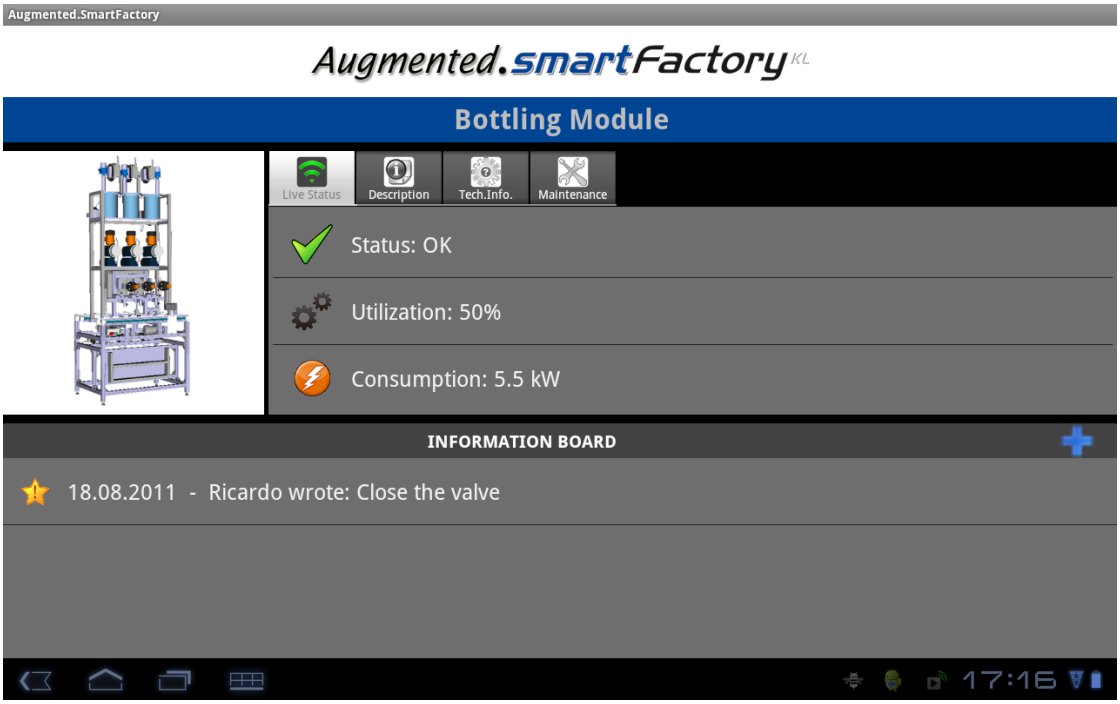

Figure 21: Module Screen

In this screen is also the location where the user will be able to insert any comment or message to be attached to the module. By pressing the "+" symbol, a *pop-up* screen will be shown (Figure 22) and there the user can write the message data:

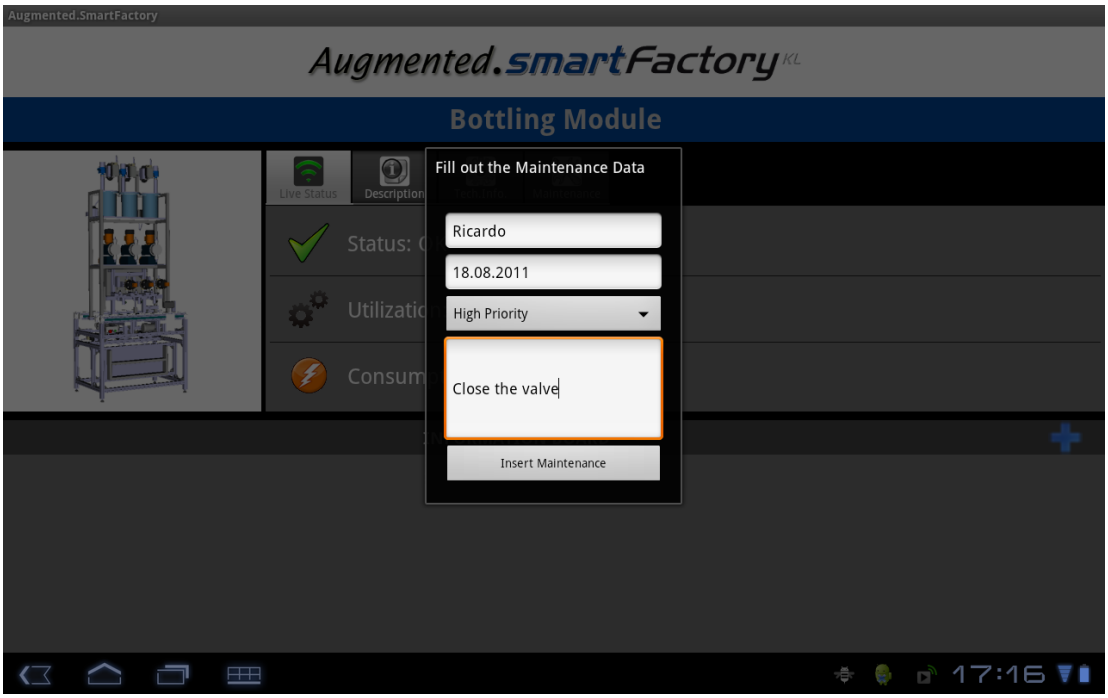

Figure 22: Insert Message Screen

Augmented Camera

This is the most important screen of the application (Figure 23). Here is where the *Augmented Reality* is working. After the initialisation, the application waits for the user to point any of the markers placed on the modules. Once a marker is detected by the camera, the application should connect to the database to retrieve all the data and shows it on the screen.

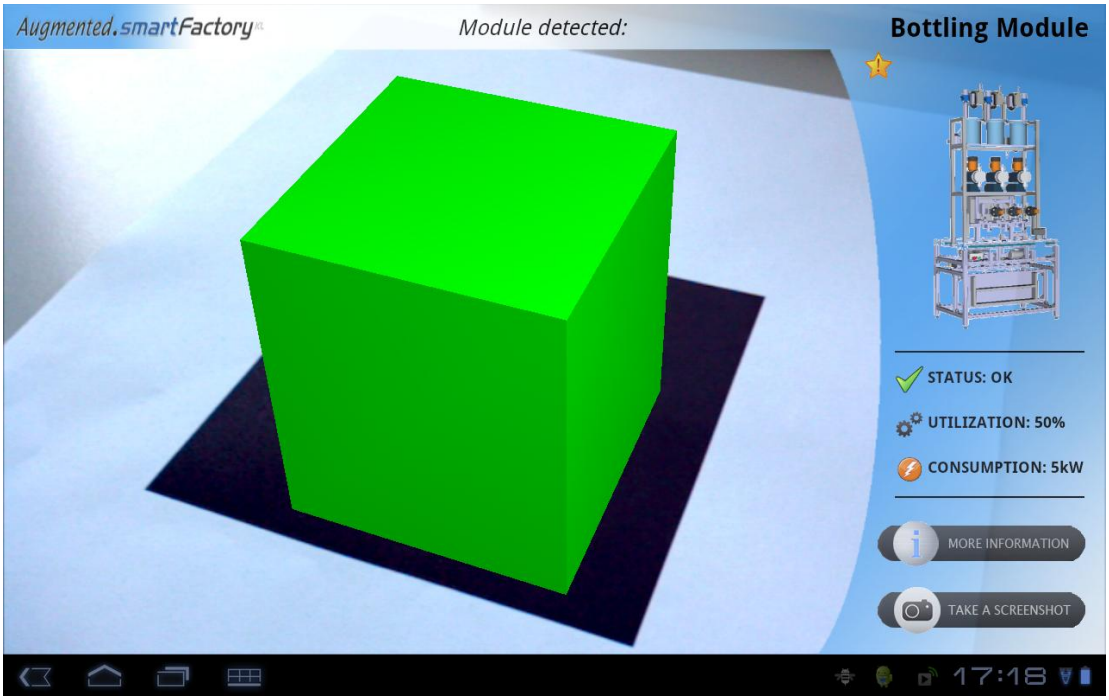

Figure 23: Augmented Camera Screen

The data is displayed on the right panel of the screen. There the user can see the name of the module pointed, a *3D* picture, some information about the live status and two buttons. One of them is the *More Information* button, used for going to the *Module Screen*; the other is the *Take a Screenshot* button, for taking a picture of the virtual objects of the screen.

If there is a little star close to the *3D* picture, as in the Figure 24 that means that there are messages with high priority attached to the module. To display that messages the user must touch that star and a pop-up panel will be shown with the messages.

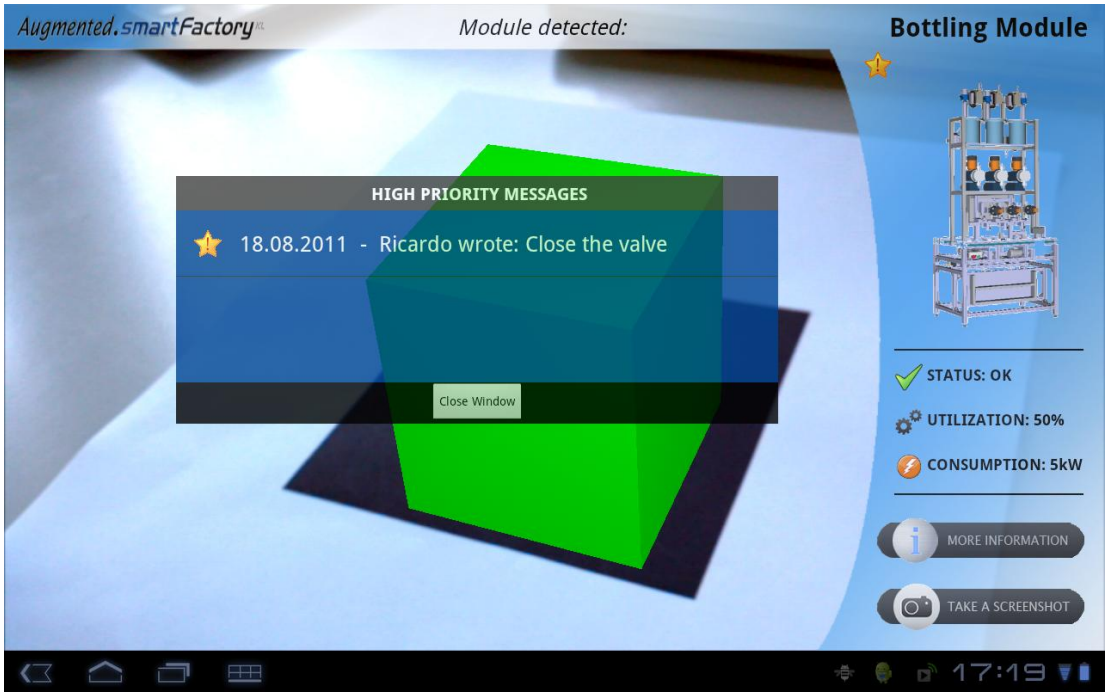

Figure 24: Virtual Information Panel

In this screen the user will be able to detect easily if a module has any problem in its status just looking to the colour of the virtual box. If the colour is green, it means that everything is working correctly and there are no problems. On the other hand, if the virtual box is red, like in the Figure 25, it means that there is a problem in the module and it should be fixed.

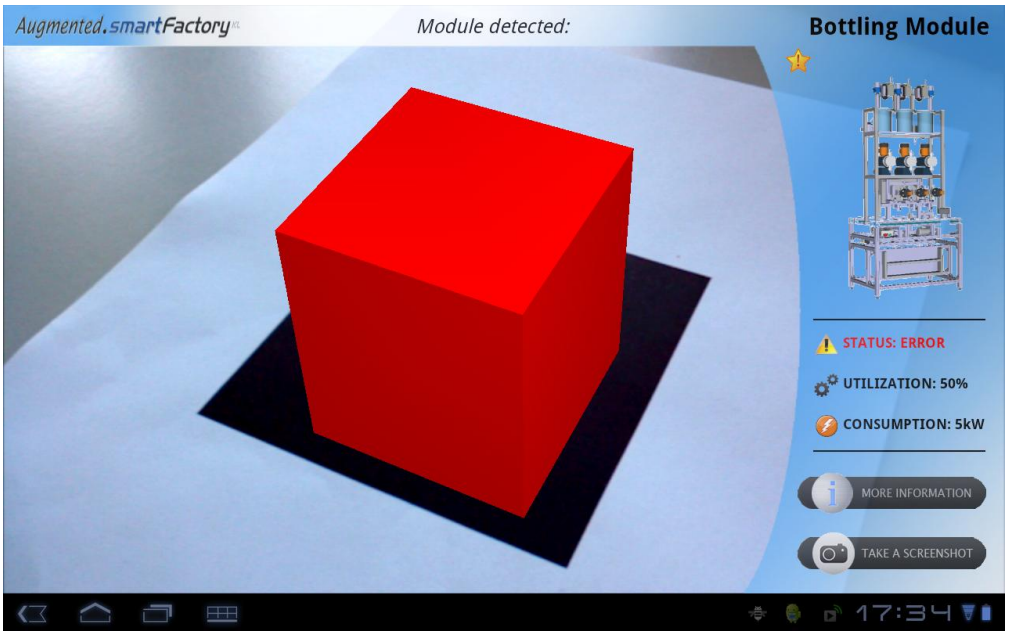

Figure 25: Red Box Showing a Problem

Every time the camera detects another marker, the appropriate information is displayed on the right panel.

5.4 Implementation and Testing

In this chapter is described how and where the project has been implemented and tested. Once developed, the software has to be adapted and implemented to the environment where it will work. There, typically some problems and difficulties emerge, which are explained in the following section:

5.4.1 Acer Iconia Tab A500

The first step to test the application was to run it on a physical device. At the beginning the application had been launched in the virtual device, provided by the *Android SDK.* This tool was very useful for the designing of the application but it is not compatible with the camera, so it is not useful for testing the *Augmented Reality* application.

So, once the *Acer Iconia Tab A500* (Figure 26) was acquired, it was possible to start with the deployment of the application. Although the application can be executed on any *Android* device, it was decided to use a *Tablet-PC* due to its big screen and its handling characteristics. It is more robust than a *SmartPhone,* and this is also critical in industrial environments like the one for this project.

Figure 26: Acer Iconia Tab A500

Like every devices with the *Android O.S.,* this device has a tool for helping the developers to debug the application through the *Universal Serial Port* (USB). It consists in allowing to the development software, like *Eclipse,* to connect straight to the device and execute the application on it.

5.4.2 Database Server

During the development of the application, a local database was created in the same computer used for programming. The *WampServer* (explained before) was installed there and with this, it was easy to setup a complete *MySQL* database. This local database was very useful for the tests inside the office, but in the case of the *SmartFactory*^{KL}, it was necessary to place a new database. So, the technicians created a new database in one of the servers inside the *SmartFactory*^{KL}, allowing the application to connect to it through a local *Wireless-LAN.*

5.4.3 Fiduciary Markers

One of the essential components of the project are the *Fiduciary Markers* and their appropriate design. These markers are the physical identifier of each module, so they should be very clear and different one from the others.

The first tests of the *Augmented Reality* were realized with some markers already designed for the previous version of the application done in *Flash.* These markers were geometrical shapes inside a white square with black border, as shown in Figure 27.

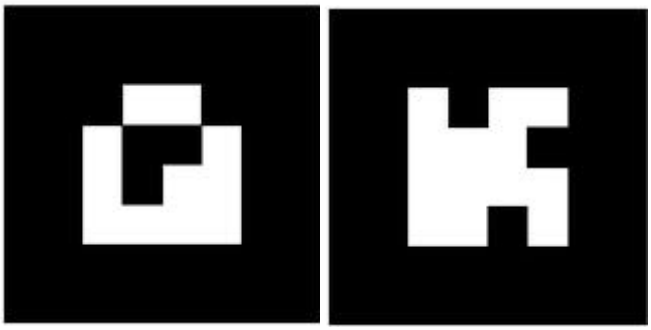

Figure 27: First Fiduciary Markers

After the test with these markers, the conclusion was that with two or three markers, the application worked quite well, but the problem occurred when more markers were stored. The application sometimes confused the markers, showing information about the wrong module. This could happen for different reasons: one of them is because the markers

were very similar and the camera could not track them correctly. The other reason could be the camera resolution, and maybe can be solved with some parameters adjustments.

After this first test, it was decided to use another kind of markers. So, now the markers are containing a big character identifying the module, like in Figure 28. These characters should be as different as possible to avoid this *"tracking error"*.

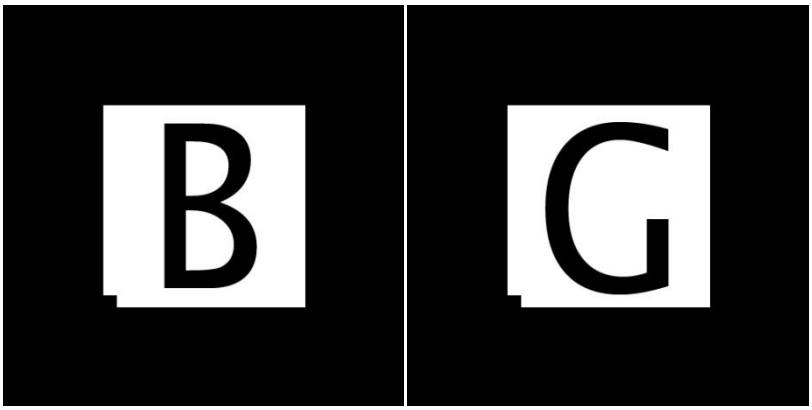

Figure 28: Fiduciary Markers with Characters

These markers were designed in *Adobe Photoshop CS5*, and after being printed, pasted on a cardboard to make them stronger, important in an environment like factories.

With this kind of markers, the *tracking* resulted very better, but sometimes occurs a little error. The solution for this is to design a specific markers with complex shapes and so different between them.

For introducing the markers into the application and make them identifiable for it, it is necessary to create a file with the extension *".pat"*, corresponding to a pattern file. There are some applications for doing that, like the one inside the *ARToolkit,* but for this project an online tool called *ARToolkit Marker Generator Online* found in this website was used. For using this a webcam tool is required. When the application starts, it is possible to create *".pat"* files just pointing the camera to the desired marker. The application recognizes this with a red square and shows it in a little window, as can be seen in Figure 29. Once there, it is possible to save it into the computer.

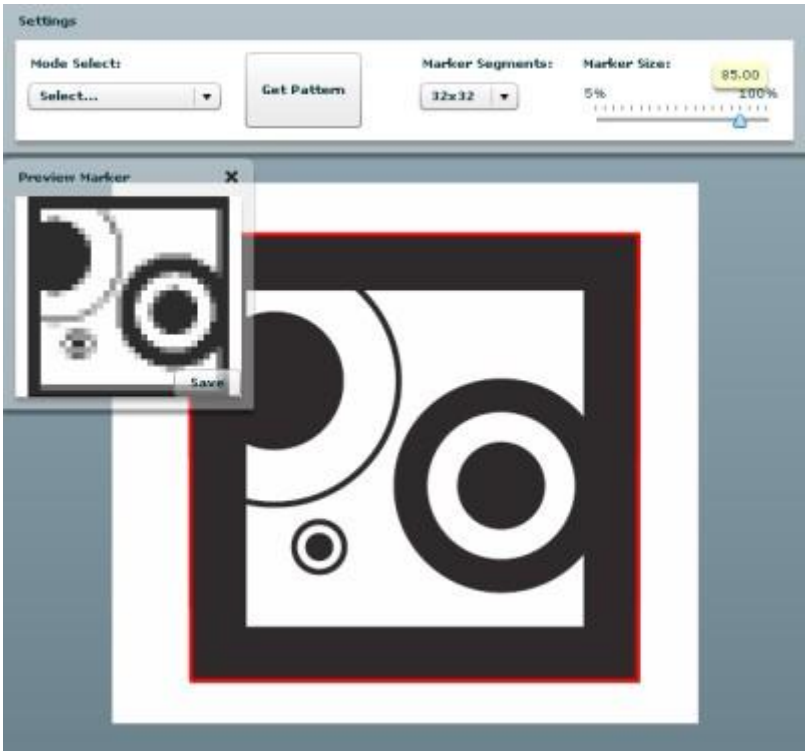

Figure 29: ARToolkit Marker Generator Online

These *".pat"* files have to be saved inside the *assets* folder of the project, where the application will search for to find the patterns.

5.4.4 *SmartFactory*[KL]

All the tests and demonstrations for this application have been made in the research and demonstration plant *SmartFactory*[KL]. It was necessary to understand the real-world requirements and problems which can occur in the shop-floor, so the application could be adapted to it.

As explained above, in the chapter Requirements Analysis, the demonstration plant should be properly illuminated for the right *tracking* of the markers. If the plant is dark some undesired *tracking errors* could occur, which means a malfunctioning of the application.

About the markers, they should be placed in a visible place on every module, and if it is possible, pointing in the same direction, which will make it easier to technicians to maintain the plant. In the following each module and its marker are described:

Fresh Water Module

This module is actually the storage module, it holds the water, white tank of 1000 litres and the vacuum chamber which holds 750 litres of rinse water for cleaning the pipes and other chambers. Its marker is shown in Figure 30.

Figure 30: Fresh Water Marker

Flow Unit Module

This is the module that circulates water throughout the process side of the plant. Components found in this module are pumps and throttle valves, controlled by a separate PLC with a user friendly interface and a *Bluetooth* connection in order to facilitate the change of parameters and also create an easy and efficient maintenance of the module. The marker is shown in Figure 31.

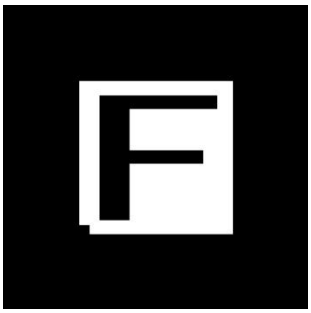

Figure 31: Flow Unit Marker

Color Dosage Module

This module consists of three reservoirs of 1000 millilitres of color: red yellow and blue and the delivery systems for them. The delivery system is made up of six pumps, two for each color. One that has a capacity of 32liters/hour and another with 4,4liters/hour, this responds to the need of different delivery capacities, could be it small or large. The marker is like in Figure 32.

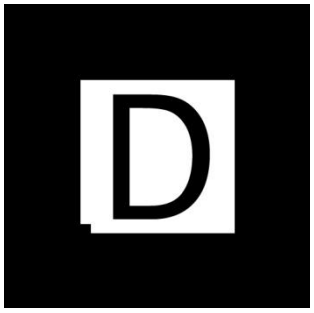

Figure 32: Color Dosage Marker

Colored Water Module

This module consists of an acrylic 120 litres transparent tank. In the production phase it is possible to encounter the following scenario: A big amount of water of only one color is needed. This tank represents a buffer for such a request. The color is dosed in the tank and the Flow Unit fills the required amount of water. The marker is shown in Figure 33.

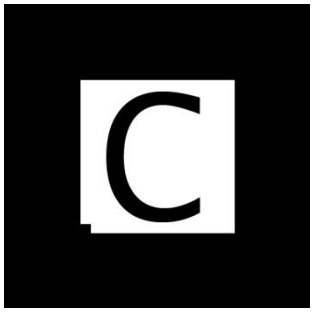

Figure 33: Colored Water Marker

Raw Soap and Mixing Module

This is a reactor tank of 20 litres which is double laired, circulating warm water on the outside, and it is used to create a constant, higher temperature of the soap in order to reduce the viscosity. A heat exchanger is needed to provide the warm water and a gear pump and other valves to regulate the flow of soap. Its marker is shown in the Figure 34.

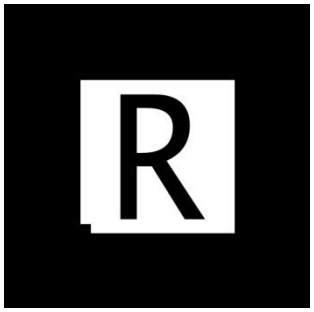

Figure 34: Raw Soap and Mixing Marker

Bottling Module

This is the module where for the first time the soap meets with its final container, the bottle. The bottle holds the information about its contents and the module reads it and fills the bottle with the required fluid be it different by color or nature (water or soap). The marker is shown in Figure 35.

Figure 35: Bottling Marker

Dispenser Mounting Module

After bottling it is needed to close the bottles, this happens at this module. After reading the *RFID-tec* the robotic arm takes a dispenser from the tray and screws it on to the bottle. The bottle is maintained in place by guide rails, and stopped in position by pneumatic actuators controlled by the PLC of the module. The marker placed on this module is shown in Figure 36**.**

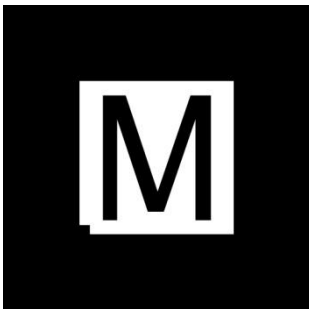

Figure 36: Dispenser Mounting Marker

Bottle Feed Module

This module supports a robotic arm, programmed to place empty bottles on the transporters that further take them to bottling. Human aid is needed in order to feed the robot with a fresh supply of bottles, the maximum capacity of a tray being of nine bottles. The marker is like in Figure 37.

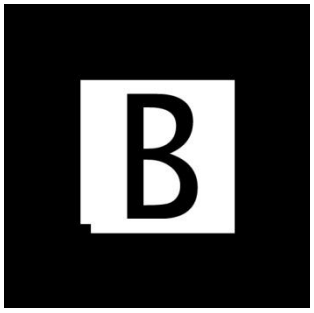

Figure 37: Bottle Feed Marker

Labeling Module

The bottles also need labels, and they are provided at this station, the positioning is critical, any mistake being visible on the product, be it bad centering or skewed (tilted) labels. The labels get to this station ready printed and are first just stuck onto the bottle, not entirely glued, the next step is to press the hole surface of the label against the bottle. The marker is shown in Figure 38.

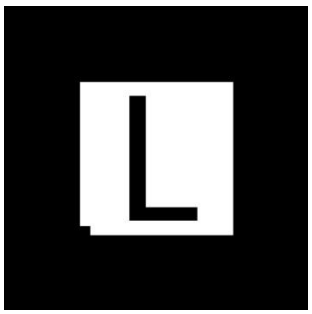

Figure 38: Labeling Marker

Commissioning Module

This is the final station of the plant, represented by a robotic arm that takes the bottles of the transporter.. Even if the station is the last one, transporters can circle the discrete part of the factory with empty bottles, or without dispensers, or unlabeled ones, until the problem is solved, if there is one. The marker is shown in Figure 39.

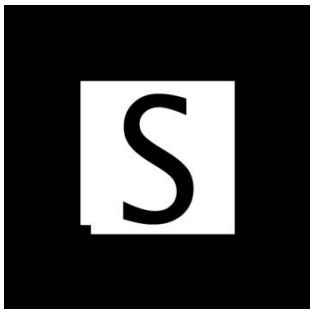

Figure 39: Commisioning Marker

6 Evaluation and Outlook

From the beginning, the project comprises different state-of-the art technologies and applies them to the manufacturing world. These advances can help the workers in their job, making it more efficient and faster.

After the implementation of the project in the *SmartFactory*^{KL} the first impressions were really convinced, because the targets previously defined were reached. The application works fine, recognizing the patterns and showing the state and other important real-time information about the modules inside the plant. With every change, the performance works better, like with the modification of the markers. This results in the opinion that working more in this project means to develop a very complete application.

Like in every application, there are some problems, bugs or future improvements to be done. Some of them are explained here:

Touching the Augmented Objects

One of the first improvements to be done is the possibility of touching the virtual objects drawn on the screen. This will allow many new features to the application but the main one is that the technicians will be able to select what module they want to check when more than one is detected by the application.

There are some developments with this feature but they are not still implemented in the *Augmented.SmartFactory*^{KL} application. This should be done in the future as one of the first improvements.

Users Management

In this release of the application there is no *users management*. It could be interesting to control the users access to the application and to handle the different roles. With this systems the user will have to *login* into the application and he will be able to enter some parts of the application and will be denied to others.

In the communication board, now the users must write their name in the messages, but with this system it will not be necessary anymore, because the application will know at every moment which user is using the application.

Module Screen

This screen is where all the information related to one module is displayed. In this version this screen is just a prototype to show the possibilities of it. In the future new features can be added, like a system to support the technicians to repair any module or component

inside the factory. This system will guide the users for fixing any problem related to the machines, by a step-by-step manual and also using the *Augmented Reality* concept to help them.

Communication Board

The communication board developed for this project is just a first impression for a complete system. In the future this system should be more powerful, with full monitoring of every event occurring inside the factory. This communication board should implement a complete *answer-response* system, and also a *topic search* or *previous actions,* so the technicians will be able to see how the problems were fixed in the past.

7 Bibliography

[Del 07] Delclós, Tomás: The “Internet of Things” challenge. http://www.elpais.com/articulo/portada/reto/Internet/cosas/elpeputeccib/20070517elpcibpor_1/Tes 17.05.2007

[Luc 11] Lucid Agency: Mobile Battles. http://www.lucidagency.com/infographics/mobile-os-battle-iphone-vs-android/ 24.06.2011

[Nic 11] Nickinson, Phil: Google anounces Q1 2011 earnings. http://www.androidcentral.com/google-announces-q1-2011-earnings 14.04.2011

[Mil 94a] Milgram, Paul, and Fumio Kishino. A Taxonomy of Mixed Reality Virtual Displays. IEICE Transactions on Information and Systems E77-D, 9 (September 1994), 1321-1329.

[Mil 94b] Milgram, Paul, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. *SPIE Proceedings volume 2351: Telemanipulator and Telepresence Technologies* (Boston, MA, 31 October - 4 November 1994), 282-292.

[Azu 95] Azuma, R. SIGGRAPH '95 Course Notes: A Survey of Augmented Reality. Los Angeles, Association for Computing Machinery. 1995.

[Jon 08] Jongedijk, Lara: History of AR and Key Researches. http://augreality.pbworks.com/w/page/9469037/History-of-AR-and-Key-Researchers 15.07.08

[Auk 92] Aukstakalnis, S. and D. Blatner. Silicon Mirage - The Art and Science of Virtual Reality. Berkeley, CA, Peachpit Press. 1992.

[Mil 94] Milgram, P. and F. Kishino. "A Taxonomy of Mixed Reality Visual Displays." IEICE Transactions on Information Systems**.** 1994.

[Val 06] Vallino, Jim: Augmented Reality, Research and Professional Development. http://www.se.rit.edu/~jrv/index.html 03.09.2006

[Sch 11] Schaumlöffel, P., Talha, M., Gorecky, D. and Meixner G.: Augmented Reality Applications for Future Manufacturing. International Conference on Manufacturing Science and Education, MSE 2011, Sibiu, Romania.

[Tal 10] Anonymous, What is Android? Talk Android. http://www.talkandroid.com/google-android-faq/ 2010

[And 11] Android Developers. Development Guide. http://developer.android.com/ 2011

[Sma 2011] SmartFactory^{KL}. About us. http://www.smartfactory-kl.de/ 2011

[Ecl 11] The Eclipse Foundation. What is Eclipse? http://www.eclipse.org 2011

[Apa 11] The Apache Software Foundation. HTTP Server Project. http://httpd.apache.org/ 2011

[Php 11] The PHP Group. What is PHP? http://www.php.net/ 2011

[Ora 11] Oracle Corporation, Why MySQL? http://www.mysql.com 2011

[Wam 11] WampServer. Presentation. http://www.wampserver.com/en/ 2011