WIRELESS FAULT MONITORING OF ROBOTIC CELLS USING ANDROID TABLET PCS WITHIN AUTOMOTIVE PRODUCTION

A. CUMBERLEGE & T. VAN NIEKERK
NELSON MANDELA METROPOLITAN UNIVERSITY, SOUTH AFRICA

Abstract

This paper presents the work on the development of software for an Android based mobile device. The goal being not only for a working application, but also to optimise the development process to ensure a user friendly application. Android is an ideal platform for connecting man and machine in a fast moving production environment. With its current popularity and market share it allows engineers to easily create flexible software applications to monitor and control industrial processes and machines. This software if not properly designed can lead to stability and maintenance problems. The experience gained from developing, maintaining and improving a mobile application for monitoring production faults, results in the improved response between man and machine. Further on it will be shown how the application can be used for data capturing purposes, thereby simplifying the data capturing process.

Keywords: Android, Industrial software, Embedded, Java, Scripting

1. INTRODUCTION

In the automotive manufacturing environment there is a constant strive for improving productivity and efficiency. Processes constantly need to be monitored to determine their effectiveness, thereby improving such processes. Due to the high demand for increased productivity and minimum product manufacturing cycle times there is an increased focus on reducing downtime of automated equipment. Those working in production and manufacturing environments know that time is money and the faster they and the machines around them can work, the better their companies can perform. As production lines become less labour intensive they are becoming more dependent on computers and automated systems. It is becoming clear that the production environment is a competitive race to produce products to the market at a rapid pace due to high public demand. Cycle times for production processes are normally accounted for during the design stage of a production line. A few seconds difference in cycle time could substantially affect the company's profit. This invention relates to production process monitoring which mainly involve robots used by manufacturing industries. The constant strive for improving the efficiency of production processes has become a worldwide challenge.

In high speed production facilities the response time between man and machine is of utmost importance when financial implications are taken into account. In order to accommodate this, the timeous notification of production process-faults and proposed solutions to problems should take place effectively and efficiently.

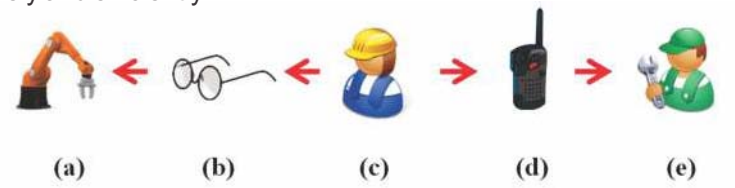

Figure1: Manual fault notification: (a) Industrial robot; (b) Visual; (c) Operator; (d) Two-way radio (e) Technician.

Figure 1 illustrates how many industries rely on production operators to visually detect production faults and then inform a technician by means of a two-way radio. The elapsed time from the occurrence of a fault to the time the technician is informed can lead to unnecessary production losses and would therefore require an improved process of informing a technician when and where these faults occur.

2. OBJECTIVE

The object of this invention is to develop a generic software application to establish wireless communication between a production technician and a production process by means of an Android mobile device which communicates with a Programmable Logic Controller (PLC) [Advanced Micro Controls 2009-2014] on a wireless network for the purpose of improving the process of informing technicians of production-faults. The mobile application should be capable of informing the user when and where production-faults occur by means of a sounding alarm and vibration and have the ability to automatically log all downtime to an Excel spread sheet and allow the user to add data such as a fault location, fault description and corrective action. The interpretation of the fault on the mobile handset could be manipulated by the PLC, depending on what the user would like to monitor. For example a fault indication on the application could represent either a generic (overall) fault or a specific fault depending on the range of possible faults or the user's requirement. For example the system could be used to notify a user when pre-determined equipment approaches the end of its lifespan thereby allowing for spares to be ordered prior to it being worn or damaged.

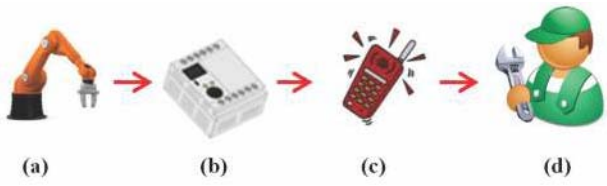

Figure 2: Wireless fault notification: (a) Industrial robot; (b) PLC; (c) Android mobile device; (d) Technician.

2.1 Advantages of the Application

Since the mobile device is based on the Android platform it makes the application very flexible as it can easily be used on any device supported by this platform by simply downloading the application onto the device. The Graphical User Interface (GUI) [Meier 2009] is developed in such a way to enhance ease of use by the user. In addition to this the data capturing within the application allows for paper-process to be eliminated. The application has also been developed in a generic way which allows the user to adapt the application to the required environment. The Android platform is important for two main reasons amongst many others. Firstly, Android is suited for a production and maintenance environment due to its powerful capabilities and flexibility. Secondly, Android is important to the mobile platform because of its application model and hardware capabilities. Its applications are not menu driven which requires a person to click or tap numerous buttons. It makes use of an element within its architecture known as intents, making Android a graphical experience by nature.

2.2 User Access

The mobile application will have two User-access levels with the ability of adding an additional user:

- Operator (Default) - This level will only allow for the monitoring of a process.
- Administrator - This level would be required when recording information related to a breakdown and has the option of adding additional users.

2.3 Software Requirements

During the initial planning phase of any software application, the scope of the project needs to be determined. It is a critical step into the development process to establish all the requirements for a particular project. Before Android development can take place the following software needs to be installed on a computer that will be used for development:

- Eclipse IDE [Friesen 2012] - This is a free and open source integrated development environment (IDE) which is meant for Java developers.
- Android Software Development Kit (SDK) [Meier 2009] - This is a set of software development tools that allow for the creation of a specific type of platform.
- Android development tools (ADT) [Prassler 2004] - ADT extends the capabilities of Eclipse allowing you to set up new Android projects, create an application user interface (UI), add packages based on the Android Framework application programming interface (API), debug your applications using the Android SDK tools and export signed or unsigned .apk files in order to distribute your application.

Many IDE's are available for developing software depending on the developer's choice. To find the right IDE the development language should be taken into consideration and should be able to compile and debug the application. Most IDE's supports more than one language such as Eclipse, Netbeans and IntelliJ of which the two most widely used IDE's are Eclipse and Netbeans.

2.3.1 Eclipse versus Netbeans

With Eclipse being associated with IBM and Netbeans with Oracle, both IDE's have been in the market since the late 1990's. While each IDE provides very similar features it becomes a decision of which one is most favourable. The image below in Figure 3 summarizes the comparison between Eclipse and Netbeans [Tee 2012]. The sections marked in green in Figure 3 indicate the higher rated IDE for that specific category.

	Eclipse	Netbeans
Ease of Use	More complex to use	Easy to use
Plug-ins	More available	Less available
Market Rating	61%	8%
Auto-Generate Imports	Yes	No
Performance	Faster	Slower
Support	Better user support	Not as good
Software bugs	More	Less

Figure 3: Eclipse versus Netbeans

Once all the required software packages have been installed and the framework of the application has been created, software development can begin. Figure 4 shows the GUI within Eclipse where each complex activity-screen is developed.

Figure 4: GUI design within Eclipse

With a comparison made between Eclipse and Netbeans and having taken all the above mentioned aspects into consideration, it was evident that Eclipse was the better option.

2.4 Software Architecture

The building blocks of the Android framework combined with leading edge software development make it the leading smartphone operating system. In simple terms Android is an open source operating system. The Android Open Source Project is a project designed by Open Handset Alliance and the Open Source Community that make it possible for developers all over the world to exchange source code to build the platform. Android aims at taking full advantage of what mobile handsets' hardware capability has to offer. Open Source means that the operating system is freely available to anyone who wants to develop an application that is intended to run on an Android device.

The fault conditions that are monitored by the device are sent from the robot via Interbus to the PLC and then via Wi-Fi to the Android device. The robot is connected to the PLC by means of a slave card which uses its master card to communicate with technology packages such as spot-welding, stud-welding, sealer applications etc. Therefore the faults that need to be monitored between the robot and the Android device are limited by the number of faults that are sent to the robot by the technology package over the robot's master Interbus circle. When a fault occurs it is indicated on the main screen of the application next to each zone by means of a warning sign. The log file will indicate the zone and robot number where the fault occurred. When tapping the faulty zone, a screen will appear where all the robots used in that zone are illustrated with a green dot. A fault will be represented by a green and red flashing dot.

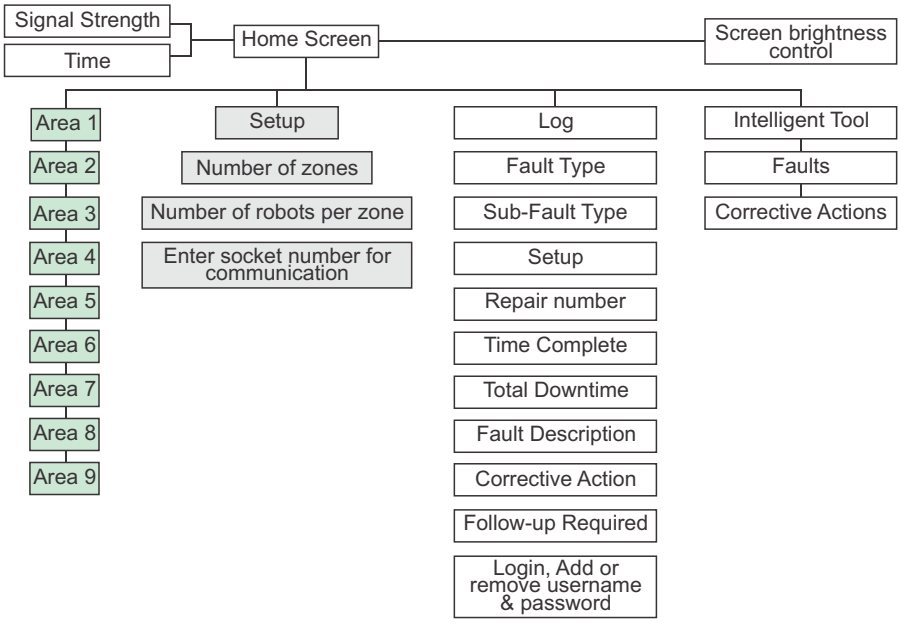

Figure 5: Software architecture of the application

The overview of the application is used for interaction between the user and the robot via the wireless network. Software architecture of the entire application can be seen in Figure 5. The Android environment is one that is structured in layers and built on a Linux Kernel Foundation. This open source technology is programmed in Java language and executed on a virtual machine known as the Dalvik Virtual Machine rather than a traditional Java Virtual Machine. The Android platform architecture is divided into five layers as illustrated in Figure 6 below.

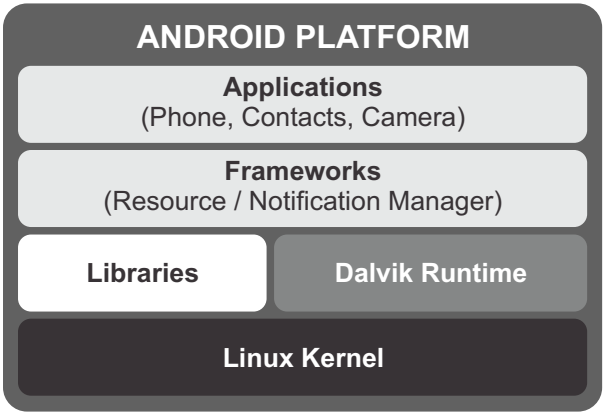

Figure 6: Android architecture [APCMag 2012]

2.5 Welding Robotic Cell Development Configuration

In conjunction with the development of the application it was necessary to setup an experimental facility where the performance and functionality of the application could be tested without affecting the performance of production.

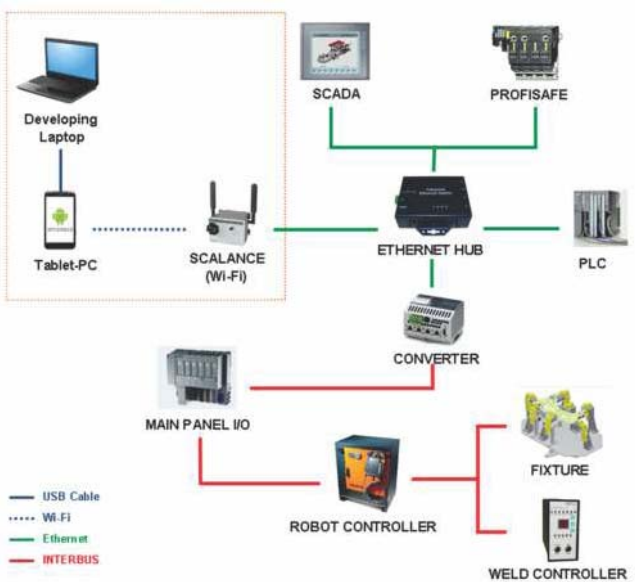

Figure 7: Hardware architecture of the experimental setup

Figure 7 illustrates the Hardware Architecture of the experimental setup that was used prior to implementation within Production. [Siemens Simatic Net 2008] [Phoenix Contact 2001]

3. APPLICATION MODEL

The applications home screen is used for interaction between the user and the robot via the wireless network. A screenshot of the application's home screen is displayed in Figure 8, with the green light indicating a broader production zone in which an error occurred.

Figure 8: Application Home Screen

3.1 Production Zone Details

When a fault occurs in a specific area it will be indicated with a warning sign next to that area on the home screen. When the button is pressed where the fault occurred, the screen illustrated in Figure 9 will appear indicating which robot cell currently has a fault.

Figure 9: Area 1 Fault Screen

3.2 Camera View

The camera button allows the technician to access the camera feature directly from within the application when images are needed for breakdown analysis.

3.3 Breakdown Logging

Once a breakdown has been corrected the technician can log the relevant information within the application. When storing this information the application scans through the existing file and appends the new information to the next available line. Before a breakdown can be logged the technician must log in with his username and password. This enables the person reading the breakdown report to see who logged each breakdown. Figure 10 illustrates the page used for logging a breakdown. The technician selects the fault type, sub-fault type and station number from a scrollable list including the additional input fields as illustrated below. All the information is captured and written to one line with dedicated columns for each data type. The reason why this format was chosen for storing the captured breakdown information is because of it being widely used in the industry and can easily be viewed on any computer.

Figure 10: Area 1 Fault Screen

The purpose of logging each breakdown is to track the failure rate of individual components over time. The more frequent problems of the same category occur the more predictable they become and can therefore be prevented.

3.3.1 File Transfer for Centralized Fault Logging

The File Transfer Protocol enables anyone to access the Excel file remotely while the user is busy using the device in the field as illustrated in Figure 11.

Figure 11: Area 1 Fault Screen

3.4 Configuring a Zone and Robotic Cell

When the setup button on the home screen of the application is pressed, the user has the ability of setting up and configuring the application within the application itself making it very flexible and generic. Rather than developing an application that is hard coded for a specific area the approach was to develop it in such a way that the user can set it up in a way to suit his/her requirements. This enables the user to use the application in different areas without having to download a modified version to the device when the device containing the application needs to be used in a different area. The user can change the following parameters within the application during initial setup or when reconfiguring the application:

- Renaming of the home screen heading and each zone heading.
- Renaming of each robot name for each zone.
- Change the number of robots for each zone - Figure 12.
- Add usernames and passwords
- Change the port number over which wireless communication takes place with the PLC - Figure 12.

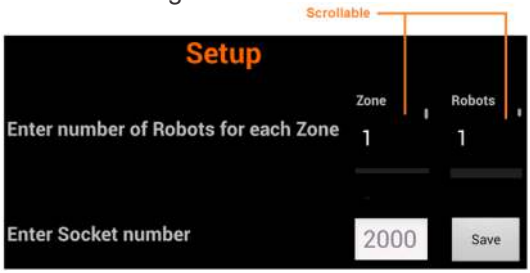

Figure 12: Setup Screen

3.5 Towards an Intelligent Maintenance Tool

This feature allows the user to build up a database of corrective actions for specific type of faults over a period of time – Refer to Figure 13. When a spot-weld gun is not welding there could be many reasons for this. The user can then enter the type of corrective action taken, thereby saving it within the application. A further development is possible which would count the number of times each corrective action is taken for a specific fault thereby providing the user with a most probable cause based on its history.

Figure 13: Intelligent Maintenance Tool

4. SCRIPTING JAVA

Java technology consists mainly of a programming language and a platform. The programming language is made up of a high-level language that is object orientated and has a unique syntax and style. On the other hand the platform is made up of a special environment that enables the programming language to run on [Friesen 2012].

4.1 Asynchronous Task

The application listens if any conditions have changed by reading-in a range of characters from within the PLC's memory on a continuous basis. The scan is executed by means of an asynchronous task. This is all done in the background to prevent the graphical user interface from responding slower since it runs on the main thread of the application. Android handles and processes all events from one user interface thread which is normally referred to as the main thread. If a developer does not use additional threads to process data, all data will be processed in the main thread causing the main thread to respond slower. When long-lasting operations are performed within the main thread such as network processing, the user interface of the main activity will be blocked until all the code has been processed. Android will automatically open a dialogue requesting the user to wait or close the activity when it's not responding for more than five seconds.

4.2 Bit Extraction

A specific class is used to take care of all the decision-making based on the data read from the PLC. A range of 270 characters are read in and by means of an AND function and comparing the result to not equal zero, it determines if the value of a specific character is equal to zero or not. This technique was used within the application to monitor the condition of individual characters that represent the status of a fault condition for a specific robot. The code below was used to establish a connection between the tablet and the wireless network and listening on port 2000 and reading the value of 270 characters.

```
ServerSocket serverSocket = new ServerSocket(2000);

while (true){
try
{
socket = serverSocket.accept();
in = new BufferedReader (new InputStreamReader (socket.getInputStream
()));
buffer = new char[270];
in.read(buffer, buffer[0], 270);
} catch (SocketException e)
{
e.printStackTrace();
} catch (IOException e)
{
e.printStackTrace();
break;
}
}
```

4.3 Display

Eclipse makes use of an Extensible Markup Language (XML) [Vogal 2011] class where the layout and display are created. Widgets [Jain 2010] are visualized and positioned to represent the screen when the application is executed. Within the XML class a lot of work is done for the developer such as translating encoded characters. The XML class serves as a framework that makes the developer's work much easier by creating a common format which can be used to share data over the internet. Typical information which will be stored in the XML class would be a widget's id, width, height, orientation, text size etc. The source code used for creating a button is as follows:

```
<Button
   android:id="@+id/next_button"
   android:layout_width="70dp"
   android:layout_height="44dp"
```

```
   android:layout_x="560dp"
   android:layout_y="135dp"
   android:text="Log"
   android:textSize="15sp"
   android:textStyle="bold" >
</Button>
```

The graphical layout of the XML class is where all the widgets are visualized. Lists of all types of widgets are available which can simply be dragged and dropped into the graphical display and the java code will automatically be added to the XML.

4.4 Communication Time

The communication time between the device and the PLC ranges up to no more than a few hundred milliseconds depending on the distance between the device and the access point. This means that whenever a fault is detected by the PLC the Android device is informed within approximately 500 milliseconds. This is calculated by means of starting a timer upon transmission and stopping it once the handshake is completed.

5. TESTING BREAKDOWN RESPONSE TIME

A comparison was done on the basis of a study on Breakdown Times with the Tablet-PC versus without Tablet-PC. Breakdown times for specific faults were identified and these faults each have a set repair procedure and thus comparable repair times:

- Tip Dress Error – Robot stuck in Tip dresser
- Robot Waiting for Water flow - Electrode Came Off

Fault	Corrective Action	Ave. time before	Ave. time after	Time (Saving)
Tip Dress Error	Drive robot out manually, Reset Error. Switch Robot to Auto	11 min	7 min	4 min (36%)
Robot waiting for water flow	Drive robot Clear, Replace Electrode, Re-weld	9 min	3 min	6 min (67%)

Figure 14: Breakdown Comparison Results

The time saving shown in Figure 14 is a result of the direct notification and subsequently improved response times as well as more accurate breakdown time capturing. Potential effect on productivity:

- Increased OPR and Line Availability
- Reduced Repair Times (Mean Time To Repair)
- Reduced Vehicle Build Time

6. CHALLENGES

Some of the obstacles that were overcome during the setup and configuration included the successful implementation and operation of the Profi-safe safety circuit. This was achieved by researching the correct wiring method to be used and steps to be followed for the configuration of the hardware and software. Another obstacle was the failure of reconnection between the tablet and the wireless network as the application froze when the tablet exceeded the operating range of the wireless network. This happened when the technician moved out of the wireless network and lost connection with the PLC. Once the tablet connected within the range it functioned properly as every scan within the tablet's application successfully transmitted and received data to and from the PLC. However when the application did not receive any response from the PLC due to it being out of range it caused the application not to respond. This was overcome by using a wake-lock function in the Java application. The wake-lock prevents the device from going into sleep mode when the lock button is pressed. Multiprocessing was one of the major challenges that needed to be overcome. The application's network processing took between one and two seconds on every scan upon execution. This caused the main thread to pause until the network processing was completed, which in turn prevented the user to interact with the tablet during this time. This was overcome by performing the network processing in the background by means of a service and multi-threading. As a result, all communication was executed in the background allowing the main thread to run without any delays. In addition to the above possible challenges can be experienced due to obstacle intervention of the wireless signal. Fortunately in this application, it was overcome by mounting the Scalance access point in the center of the required area and also mounting it approx. 10 meters in the air. The range in which the Scalance can operate is strongly influenced by its surroundings. It operates up to 50 meters indoors and 100 meters outdoors which can easily be extended using additional Scalance devices. The approximate wireless range that was achieved was nearly 60 meter radius between the Android device and the access point.

7. CONCLUSION

In an automotive production environment time is valuable as vehicles are being produced within minutes. To achieve and maintain this, largely robotic based automated processes are used with complex software programs that require a lot of time for research and development. Factories that produce a product at a fast rate are constantly looking at ways to improve their processes in order to remain competitive.

This application mainly focused on a proof of concept basis which will be rolled out in the production facilities over a period of time. The concept showed promising results with the intention of being taken to the next level in the future. The current method used to inform the maintenance department when a breakdown has occurred is by means of a production personnel contacting maintenance using a two-way radio. This can take up to several minutes for the responsible production personnel to identify that a breakdown has occurred thereby justifying the concept of automatic error notification via mobile android technology. The design of the wireless production line monitoring system has evolved substantially since its initial design concept and has opened a window of nearly endless possibilities and for this reason it is necessary to take action and plan for future improvements.

8. REFERENCES

Advanced Micro Controls, Inc. (1999-2014) What Is A Programmable Logic Controller (PLC)? [Online] Available from: http://www.amci.com/tutorials/tutorials-what-is-programmable-logic-controller.asp. [Accessed: 10th November 2012].

APCMag. (2011) APC. [Online] Available from: http://apcmag.com/whats-android.htm. [Accessed: 23th April 2013].

Friesen, J. (2012) Learn Java for Android Development. [Online] Apress. Available from: http://www.free-ebooks-library.com. [Accessed: 12th September 2012].

Jain, A. (2010) Introduction to Android development Using Eclipse and Android widgets. [Online] Available from: http://www.ibm.com/developerworks/opensource/tutorials/os-eclipse-androidwidget/. [Accessed: 12th September 2012].

Meier, R. (2009) Professional Android Application Development. Indianapolis: Wiley Publishing, Inc.

Meier, R. (2009) Professional Android 4 Application Development. Indianapolis: John Wiley & Sons, Inc.

Phoenix Contact. (2001) User Manual, Configuring and Installing Interbus. Release 11. [Online] Available from: http://www.siemens.com/entry/cc/en/ [Accessed: 10th November 2012].

Prassler, E. et al. (2004) Advances in Human-Robot Interaction. STAR 14. Berlin: Springer.

Siemens Simatic Net. (2008) Scalance W-700, Configuration Manual. Release 11. [Online] Available from: http://www.siemens.com/entry/cc/en/ [Accessed: 10th November 2012].

Tee, J. (2012) What's the big IDE? Comparing Eclipse and Netbeans. [Online] Available from: http://www.theserverside.com/feature/Whats-the-Big-IDE-Comparing-Eclipse-vs-NetBeans. [Accessed: 19th March 2013].

Vogal, L. (2011) Android 4.0 Development Tutorial. Version 8.5. 2011. [Online] Available from: http:// www.cs.virginia.edu/~cs201/labs/Vogella-Android-Development-Tutorial.pdf [Accessed: 4th August 2012].