Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Computer Science with specialization in Embedded Systems 30.0 credits

MOBILE INTERACTION WITH SAFETY CRITICAL SYSTEMS: A feasibility study

Erik Jonsson
ejn12027@student.mdh.se

Examiner: Johan Åkerberg
Mälardalen University, Västerås, Sweden

Supervisor: Adnan Causevic
Mälardalen University, Västerås, Sweden

Company supervisor: Markus Wallmyr,
Maximatecc, Uppsala, Sweden

November 16, 2015

Abstract

Embedded systems exists everywhere around us and the number of applications seems to be ever growing. They are found in electrical devices from coffee machines to aircrafts. The common denominator is that they are designed for the specific purpose of the application. Some of them are used in safety critical systems where it is crucial that they operate correct and as intended in order to avoid accidents that can harm humans or properties. Meanwhile, general purpose Commercial Off The Shelf (COTS) devices that can be found in the retail store, such as smartphones and tablets, has become a natural part of everyday life in the society. New applications are developed every day that improves everyday living, but numerous are also coupled to specific devices in order to control its functionality. Interaction between embedded systems and the flexible devices do however not come without issues. Security, safety and ethical aspects are some of the issues that should be considered.

In this thesis, a case study was performed to investigate the feasibility of using mobile COTS products in interaction with safety critical systems with respect to functional safety. Six user scenarios were identified for investigation, which could be of interest for industrial applications; The operator presented live machine data, The operator controlling the machine remotely, The service technician using mobile device in maintenance, service technician reading machine logs from the office, the production manager analyzing machine productivity logs from the office and the software manager uploading software. Restrictions in the functional safety standard, IEC 61508, and the characteristics of COTS devices, leads to the conclusion that real time interaction with safety systems is not allowed if the certification is to be preserved. Extracting information used to analyze the system where data is only sent from the machine would be allowed. All scenarios where the machine sends data to the user, and the data is only used as information, are hence allowed if isolation properties are guaranteed. A prototype system was designed and parts of it were implemented to show how sending and logging information can be performed using the company developed communication platform Data Engine.

Keywords: Safety Critical Systems, COTS interaction

Table of Contents

1 Introduction

Almost everywhere around us, embedded systems can be found. In coffee machines, amplifiers, in cars and even implanted inside humans, pacemakers can be found to keep up the heart rate. These type of systems utilize one or several computers that are designed and optimized for the specific purpose. In contradiction, general purpose computers such as laptops and stationary computers, are designed to provide high flexibility while being used for many different tasks. Some embedded systems are used in interaction with humans, and if they fail or run out of control it can have catastrophic effects. In the worst situations, human lives can be lost due to the failure. Such systems are considered safety critical systems. The company maximatecc AB have throughout the years designed embedded systems to control machinery such as seaport container cranes, harvesters and excavators. When controlling machines that posses great powers and currents, it is obvious that failures can cause big damage. To minimize the risks of hazards, it is important to design and test systems in such a way that failures are unlikely to occur. The International Electrotechnical Commission (IEC) have, together with the industry, develop standards that classifies safety related systems. Many machineries, that maximatecc supply control systems to, are certified under IEC 61508 for functional safety at different SIL (Safety Integrity Level) [1]. The standard gives guidelines on how each step in the development process should be performed, which documentation is necessary, which types of tests should be performed and how the product shall be maintained throughout the whole life cycle of the product. New revisions of the EU machine directives is also making it more important to certify products to be able to compete on the market.

A centrally controlled system has one control unit that controls all parts of the system. However to control machines operating on currents of several amperes the cable costs grows with the size of the machine. Hence, it is many times preferable to have a distributed system where the different parts in the machine communicates via data buses. Input/Output (I/O) controller nodes controls the input from sensors and output to actuators in the system. Accordingly, they usually also contain a communication interface to send and receive parameters from other parts of the system.

The last decade there have been an explosion of applications for wireless devices, and Custom Of The Shelf (COTS) mobile devices, such as smartphones and tablets. They offer high functionality with many types of sensors and large computational power. Many applications for these devices have been developed to make everyday life easier. With the wide functionality and usability of such devices the interest of using wireless technology have also attracted the industry and safety critical system suppliers. A wireless interface from a COTS device to the safety critical system would open up many opportunities to new user interfaces. For example it could be possible to read sensor values from the machine and present it on the external device, or even control the system remotely. However, even if existing techniques enables this communication, it is not obvious that it should be done from a safety perspective.

1.1 Problem Formulation

This thesis investigates how and under which circumstances an external, not safety certified, Commercial Off the Shelf (COTS) device, like a smartphone or tablet, can interact with a safety critical system such that the safety system still can be certified under the desired SIL in IEC 61508. It also investigates for which type of applications such a communication would be appropriate regarding safety. The problem is defined by a set of user scenarios. In these scenarios two main questions are considered.

1. Is it feasible to use the COTS devices in interaction with the Safety Critical Systems?
2. How and under which circumstances, can an application running on the COTS device communicate with the Safety Critical System without violating safety restrictions given by the standard?

1.2 Report organization

In the following section the research method of the thesis is presented followed by introduction of user cases for the study in section 3. In section 4 the functional safety standard IEC 61508, State Of The Art (SOTA) literature study, characteristics of COTS devices and existing company developed software is presented. In section 5 the user scenarios are analyzed from a safety perspective resulting in recommendations and a system proposal. Feasible scenarios in the recommendation are implemented in a prototype design. The work is concluded in section 6 and future works are suggested in section 7

2 Research Method

The nature of a rather wide problem, as formulated in section 1.1 requires a well measured method to define boundaries for the research. For example, there are many variants of COTS devices with wireless connectivity enabled and many types of Safety Critical Systems.

Case studies are widely used in social science to study specific situations in human interactions, but also suitable for Software Engineering problems when interacting with technology [2].

An example is in the startup of a large project. Here it is common to investigate if the invention is feasible to avoid unnecessary investments at an early stage. In such a case study, the goal is to find feasible features and demonstrate their concepts.

Runesson and Höst [2] gives guidelines on how to conduct a case study in software engineering. They define five major steps in a case study process:

1. Case study design: objectives are defined and the case study is planned.
2. Preparation for data collection: procedures and protocols for data collection are defined.
3. Collecting evidence: execution with data collection on the studied case.
4. Analysis of collected data
5. Reporting

The research method used in this thesis is designed as a technical feasibility study. A set of user scenarios is defined for which the technical feasibility is investigated. The aim is to explore techniques to enable the scenarios and to explain why or why not the scenarios are feasible from the safety perspective. Furthermore, COTS devices are defined with some general features that can be found in most of the smartphones and tablets on the market. Their nature is described in terms of safety features and their wireless connectivity. A few devices from the State-Of-Art will serve as raw models to describe the characteristics.

Subsequently, evidence is collected from SOTA research and standardization restrictions. The approach is hence qualitative, and exploratory in the sense of trying to find techniques to enable the scenarios, and explanatory in the sense of explaining why or why not they are feasible. The collecting of evidence has been made in iterations. When new information was discovered it opened up new traces to follow up, and the research was hence performed in an incremental way.

2.1 Contribution

The contribution resulting from this thesis work is a set of safety recommendations and suggestions on appropriate applications for mobile COTS interaction with a Safety Critical System. A prototype is designed and implemented to show how the feasible scenarios can be implemented.

3 Use Cases

Since nodes in a machinery system usually have some communication interface, it could also be possible to connect them to the Internet of Things (IoT). If a node contains a radio transmitter it is also possible to establish a wireless connection to the Internet. If only wired communication is enabled on the node, a connection to IoT could also be realized by wire through another node, serving as a wireless access point or gateway. This section starts with a short presentation of the field of IoT. From that, a set of user scenarios are chosen and described for investigation.

3.1 The Internet Of Things

IoT is the paradigm shift the computer science community have been talking about for more than a decade now. The idea is to equip embedded systems with Internet connectivity devices. Only imagination limits what can be done if systems can communicate over the Internet or other networks. This also comes with potential issues such as usability, security, privacy and safety of the innovation. In surveys like [3] and [4] the big variety of applications thought of in the IoT is presented. Every electrical device from small tags to space crafts fits into the category of things. The only feature that is mandatory on the devices is that they contain some kind of network interface that allows the device to connect to the Internet. There are many ideas on what can be useful as future applications. It spans from Cyber-Physical Systems where the system interacts with the physical environment, to smart sensor networks monitoring and collecting information about the environment. Small RFID chips sending its position on demand that can be attached to single products and machine to machine (m2m) networks where machines communicate to perform shared or related tasks are another possible applications. The ideas also span over all possible domains that humans might act such as smart cities, transport, buildings, energy, living, health and industry. The visions also includes the possibility to utilize cloud services such as analyzing of big data.

Vermessan et al.[5] presents current ideas and technologies for the IoT. They also list application areas and research challenges. For the scope of this thesis the areas of smart vehicles and manufacturing are the one's that are closest when considering industrial machines. Business areas like mining, forestry and agriculture are deeply dependent of vehicles in order to be productive.

Application examples described for the industry are:

- system monitoring
- repair and maintenance
- system control
- augmented reality

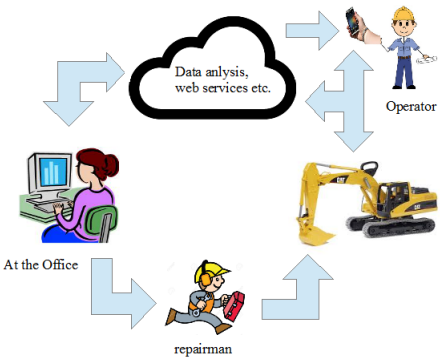

Figure 1: Eco system of several useful applications that can be enabled using wireless communication

In figure 1 several scenarios are depicted that can be enabled for an industry application. Useful applications could be to upload and read diagnostics or logs from the machine. This could for example support the repairman or service technician with information of which spare parts to bring without seeing the machine and alert if a module is getting worn out between regular maintenance. This would be a good tool to reduce costly system downtimes. It could also be possible to alert the operator in runtime that the machine health is getting bad or to upload and install new software or firmware to the machine from the office. If Internet connectivity is unavailable at the working site where the machine operates, it could still be possible to have communication and transmit data in real-time to and from a hand held device when walking around the machine. This could also enable augmented reality applications that could help the user in troubleshooting or performing guided maintenance.

3.2 User scenarios

From discussions with the company supervisor, six possible scenarios where identified where an wireless Human Machine Interface (HMI) could be interesting

1. **The Operator presented with live machine data:**
   The user in this scenario is presented live machine parameters. It investigate how live data, can be presented in real time, to the operator in a COTS device. The user may be anywhere around the machine and still have machine parameters available. It can be useful to know the status of the machine from a remote location in order to make decisions of which actions to take.

2. **The Operator remotely controlling the machine:**
   There can be situations where the operator would like to control the machine remotely from a position beside the machine. For example, a crane controlled using a wireless connection increases the flexibility for the operator and if the line of sight is limited it could be useful to move around beside the machine to get a better view of the situation.

   In this scenario parameters shall be set from the COTS device, to an I/O controller in the machine, in order to control its behavior. This could also be done in combination with the first scenario, to display the current parameter values.

3. **The Service Technician using the mobile device as help in maintenance:**
   With smart image recognition and positioning techniques, augmented reality can be realized. This will allow the service technician to look at the machine trough an augmented reality filter. It will enable a connection to the machine that can present guidance in investigating and solving problems without having to read the manual to find the specific error messages. This filter could for example be implemented in Google glasses or a tablet. It would be very helpful in diagnostic applications.

4. **The Service Technician reading machine logs from the office:**
   To help in planning repairs and/or maintenance it would be possible to send machine health parameters over the Internet to the corporation cloud. For example, by utilizing the computation power in the cloud, smart pattern recognition computations can be performed to identify erroneous conditions. This would also enable ordering of spare parts in advance without examining the machine, and hence reduce costly downtimes. With statistics from machine data it could also be possible to analyze and find trends from an entire fleet of machines.

5. **The Production Manager:**
   This scenario is built on the same idea as the previous one and can be used to automatically create daily reports and make them available from the office in the corporation cloud. This would enable opportunities to business process improvements.

6. **The Software Manager:**
   This scenario will investigate the possibility to update firmware and/or software remotely from the office. The machine is set to a specific state used for updates and the software manager uploads and installs the software.

4 Background

In this section Safety Critical Systems and the standard for functional safety (IEC 61508) is presented, which is the object for the considered certification. Furthermore, related works and SOTA research will be presented. It includes Mixed Critical Systems (MCS), virtualization techniques, distributed systems, COTS device characteristics. Lastly a company developed software communication platform is presented. These parts lies as the foundation for the analysis and the system proposal in the following section.

4.1 Safety Critical Systems

Many types of embedded systems exists that are designed for different specific purposes. Some of them are designed to control apparatus that, if they get out of control, could cause big damage that may harm humans or properties. These kind of systems are regarded as safety critical systems and are found for example in the military, machine and medical care industry. Knight[6] describes safety critical systems as "*a system that if it fails can have unacceptable consequences*". The more people that could get harmed from a hazard, the more cautious and careful the system has to be designed, implemented and maintained. For example if the control system of a commercial aircraft with many passengers fails, and the captain cannot steer it anymore, the catastrophe will likely have deadly outcome. Safety put humans in the first place, but it also consider the risk to lose valuable resources or big investments. There are some well known example of Safety Critical System failures.

> The Ariane 5 flight 501 space craft self-destructed 1996 at an altitude of 3700 m about 40 seconds after take off due to a software bug when converting a 64 bit floating point to a 16 bit integer. No humans where harmed but huge amounts of resources where lost [7].
>
> Therac 25 used in the mid 1980:s is an example where a software problem have killed people. The radiation therapy machine emitted radiation doses several levels of magnitudes higher than intended because of programming errors. Six persons got severe wounds and four of them died [8].
>
> In US military the Patriot missile defense system failed to demolish incoming SCUD missiles from the Iraqi army during the Gulf war. The failure was due to inaccurate precision in the arithmetic calculations in the system. This led to a loss of 28 soldiers and wounding around 100 civilians [9].

Aftermath from these and other accidents have increased the awareness of safety and the demands for risk reduction when new systems are developed. From these discussions, safety standards have evolved.

It can be argued that risks shall always be minimized as much as possible. The loss of a life is invaluable. Doing that would however postpone release dates and hence also technological advantages such as environmental and energy savings, and at the end, there would still be a small risk that something fails. Hence, it is more common to balance the risks and the costs and reduce the risk to a tolerable level.

It is important to realize that safety and reliability are not the same thing. Safety system always puts the safety of humans in the first position. A highly reliable system could be a gun that always fires when the trigger is pulled, and such a system could obviously harm somebody if it's aimed in the wrong direction. What is aimed for is a reliable safety system.

When developing a product that is meant for safety purposes it is usually a good idea to aim to certify it, not least considering laws regulating it in some countries. In some countries it would also be possible to sue a company delivering products not meeting safety requirements when an accident have occurred.

4.1.1 Security

Every embedded system that have the equipment to connect to external sources is under potential security risks. Security of data aims to protect data from unauthorized access and manipulation.

Hackers have over the years been able to intrude in systems such as banking and National Aeronautics and Space Administration (NASA) that should be considered having high awareness of security. Recently it has also been shown that modern cars can be hacked and controlled remotely [10][11] and last year (2014) we could read the news that somebody had hacked into a nuclear plant in South Korea [12] and stolen information about the system and the facilities.

In the computer science security community it is common to discuss confidentiality, integrity and availability of systems. Confidentiality targets that unauthorized users should not be able to spy and read data from the system, and integrity that changes to data should not be possible. A typical threat against availability is a Denial Of Service (DoS) attacks when the system is disrupted or overflooded so that authorized request to the system becomes slow or even totally unavailable. Many books and articles such as [13] [14] covers techniques to handle these issues. Security should of course be considered when designing Safety Critical Systems, but it will not be the main focus in this report.

4.1.2 Functional safety

Functional safety considers the design and implementation of safety related functionality, mainly meaning functions of the system that are interacting with humans and environments. This section presents a brief summary of IEC 61508 and a selection of commonly used techniques. It also presents the significance of a rigorous and careful development process.

Overview of IEC 61508
IEC have developed standards which guides engineers in development of new safety related products. IEC 61508 [1] is the standard for functional safety. In IEC 61508 -1 the aim of the standard is described:

> It *"sets out a generic approach for all safety lifecycle activities for systems comprised of electrical and/or electronic and/or programmable electronic (E/E/PE) elements that are used to perform safety functions."*

In IEC 61508 -4, safety is defined as:

> *"the freedom from unacceptable risks"*

and functional safety as:

> *"part of the safety relating to the EUC and the EUC control system that depends on the correct functioning of the E/E/PE safety-related systems and other risk reduction measures"*

where EUC is short for Equipment Under Control.

IEC 61508 consists of 7 parts.

- Part 1 describes the general requirements and the development life cycle activities.
- Part 2 focus on the realization phase of the system, mainly referring to hardware.
- Part 3 part focus on the realization of the software.
- Part 4 gives abbreviations and definitions of terms.
- Part 5 gives examples of methods to perform risk and hazard analysis and to determine the SIL.
- Part 6 gives guidelines for the application of parts 2 and 3.
- Part 7 presents an overview of safety techniques and measures.

An independent authorized person, department or organization will examine if the product conforms to the standard and if it does, it will be certified. For high SILs it has to be an independent organization that examines the system and for low SILs it can be a single person. An important

part of the process when developing a safety related product is accordingly to collect evidence for conformance to the standard.

Development process and techniques
The first part of the standard, describing the general requirements, presents four important aspects that should be covered. Documentation, management of functional safety, overall life cycle requirements and functional safety assessment.

The objectives of the documentation is to provide and specify all necessary information in order to enable that all phases in the life cycle can be carried out effectively. New additional documentation will be produced and added in each phase. The documentation requires that sufficient information shall be provided for all phases, that it is well structured, and that it shall be reviewed to ensure that it is accurate, concise, easy to understand for those who are to make use of it and suit the purpose for its intention.

In the management of functional safety, the objectives are to specify activities and responsibilities of who in the project team that is responsible for which phase, or safety related E/E/EP systems, and who will carry out which specific activity. One person can, for example, be responsible for the hardware safety and another one for the overall design. Except from assigning responsibilities, the requirements includes making sure that persons assigned to the activities have the right competence, and that all persons involved will get the right training to perform assigned tasks. This also means that experienced persons have to refresh and update their current knowledge of safety. It also provide means to assure that the communication channels and procedures in the project are clear and the interfaces between different phases, where different persons are responsible, is planned for and handed over in a proper way. Procedures should also be developed to ensure prompt follow-up and resolution to problems raised, with an extra attention to hazardous events. This also includes all related analysis and documentation matters.

The objective of the functional safety assessment is to specify necessary activities in order to be able to investigate and judge if the E/E/EP safety related systems will comply to the standard.

The overall life cycle requirement presents a framework that defines 16 phases for the product life cycle from concept to decommissioning. Each step has its objectives and requirement to fulfill. The precess flow shown in figure 2 presents each phase and in which order they shall be carried out. Each phase has its inputs and expected outputs. The realization phase (10) is described explicitly for hardware and software respectively. Generic objectives for each phase can be found in table 1 in IEC 61508 -1.

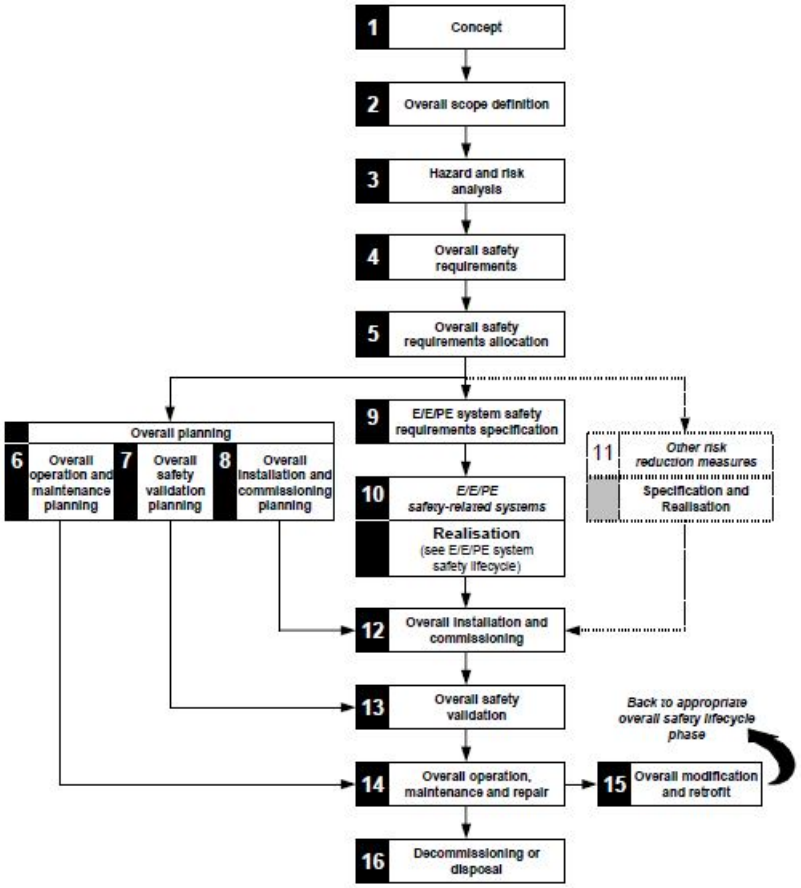

Figure 2: The Overall Safety process described in IEC 61508 -1 [1]

Many of the phases can be found in any project process model such as the V-model. Scope definition, allocating requirements and specification, realization and validation are all such examples. Just like in any project there may also be reasons that forces the project to loop back to an earlier phase to complement that phase and all its subsequent phases. Examples are if the product does not pass the validation tests or if a new hazardous event is discovered. The similarity of the processes makes it relatively easy to include safety in a project, but also increases the requirements of how the phases shall be performed. The phase that sets the character of the safety process is the hazard and risk analysis which all other subsequent phases will depend on. In this phase the objective is to determine hazards and hazardous events in the scope relating to the EUC. The analysis will be used in phase 4 to specify the overall safety requirements which in turn is used to determine SIL when the overall safety requirements are allocated in phase 5. There are 4 SILs in IEC 61508, 1-4, where 4 is the highest level with the most stringent requirements.

To determine the SIL, measurements of the risks and the probability that they occur are performed. Low demand probabilities of failure for safety functions that are rarely used, for example airbags, is measured in Probability of Failure on Demand (PFD). Functions that are often or continuously used are measured in Probability of Failure per Hour (PFH). The SIL and corresponding probabilities are presented in table 1.

SIL	PFH	PFD
4	$10^{-9} \leq p < 10^{-8}$	$10^{-5} \leq p < 10^{-4}$
3	$10^{-8} \leq p < 10^{-7}$	$10^{-4} \leq p < 10^{-3}$
2	$10^{-7} \leq p < 10^{-6}$	$10^{-3} \leq p < 10^{-2}$
1	$10^{-6} \leq p < 10^{-5}$	$10^{-2} \leq p < 10^{-1}$

Table 1: Probabilities p for determination of SIL targets

IEC 61508 -5 presents a set of techniques to perform the risk and hazard analysis and for determining SIL targets. The objective is to identify the hazards and hazardous events to enable collection of evidence that such events will fit to the aimed SIL. This can be carried out by quantitative or qualitative methods or in combination.

One technique to perform risk analysis is to construct a Fault Tree and perform a Fault Tree Analysis (FTA). The tree is drawn from the hazardous event and backwards to track what can cause it. An example of a hazardous event can be that the brakes fails. In the first step, all causes that makes this happen are considered. In the next step the reasons ending up in these situations/states are considered, and so on, in a sequence of steps. In the last step, the tree will show one or more things that when they fail that potentially can cause the hazard directly or in chain reactions leading to it. The final tree will look like a system of logical gates with a number of inputs followed by AND and OR gates. By using this analysis it will be possible to make quantitative measures in the design phase to produce a reliable system within the tolerance of the intended SIL. The human factor should be considered in the analysis, but to quantify and determine frequencies of human behavior is hard and is not a requirement in the standard.

SIL targets can also be established using risk analysis graphs where qualitative judgments are involved. Functions and failures are categorized from the severity of the failure like in the risk classification approach shown in Table 2. It involves assessments of consequences and frequencies where a negligible risk could be that the background lights on the instrument panel is malfunctioning in the driver cabin while a catastrophic failure could be that it will become impossible to use the brakes.

FREQUENCY	CONSEQUENCE			
	Catastrophic	Critical	Marginal	Negligible
Frequent	1	1	1	2
Probable	1	1	2	3
Occasional	1	2	3	3
Remote	2	3	3	4
Improbable	3	3	4	4
Incredible	4	4	4	4

Table 2: Example of Risk Classification approach

Intolerable region is covered by 1, tolerable region of 2 and 3, and the acceptable region of 4. For region 2 and 3 it is good practice to reduce the risk to a level As Low As Reasonable Practicable (ALARP). The method considers, as the name implies, that the tolerable risks should be reduced as much as possible until the time and cost expenses would be unproportional to the project.

It should be noticed that the specification will be a week point in the process. If it is missing a requirement or contain ambiguity it will be a single point of failure which will be propagated into the design and implementation phases. This is accordingly a very important part of the process to make correct and complete. Therefore formal methods are used when high SILs are considered. Formal methods specifies notations that shall be used in the specification of the system requirement. With specific notations these methods makes it possible to prove certain properties by logic reasoning. For example, it can be proven that the property: it should always be possible to start the failsafe procedure, holds. The property will in this case be true for every state of the machine.

Quantitative evidence rely on things that can be measured and calculated by statistics. These techniques heavily refers to physical issues in the hardware where components or modules have documented failure modes and probability for them to occur. Random hardware failures can be

caused for example by worn out components or extreme values in the physical environment. As evidence from the development of the hardware it is common to provide a Failure Modes Effect Diagnostic Analysis (FMEDA) report. FMEDA is a quantitative method used to prove that the hardware failure probability is at least as low as the SIL requirement. For example it is possible to calculate Bit Error Rate (BER) depending on the distance between two wires, or to measure the failure rates of single components such a capacitor and resistors. These rates and the modes of failures are usually provided by the manufacturer. The calculations are performed in Failure In Time (FIT) units, where 1 FIT is equal to the Probability of Failure per Hour (PFH) of 10^{-9}. For example a system with SIL 2 should have a FIT value between 100 and 1000 in analogy with Table 1. When interconnecting two modules of the system the FITs are added and the total FIT will now be the sum of the two modules. By adding redundant modules the FIT value will decrease. In redundancy M out of N (MooN) components is used where for example 3oo5 means that 3 out of 5 available units must work correctly for the device not to go dangerous.

In hardware it is also common to use read-backs of the output. This technique is employed to make sure that what is sent is the same thing as intended. This is achieved by looping back the output signal and compare it to the sent signals. If the signals differ too much a fail safe procedure shall be triggered.

Systematic failures such as flaws in the specification or bugs in the software however are not easy to calculate by statistics. Even if one would try, it would be hard to find the distribution of the errors. Evidence for reduction of this kind of failures are hence provided by using qualitative methods.

Hazard and Operability Analysis (HAZOP) is a structural method to examine documentation and to analyze the system design together with other engineers in the project team through a series of meetings to ensure its correctness. The aim is to determine safety hazards in the proposed or existing system, their possible causes and their consequences, and to recommend techniques to minimize the risk of their occurrence. The group leader shall encourage the participants to creatively expose all potential hazards. HAZOP can be performed at any phases of the development cycle and is recommended to be applied at an early stage to avoid mishaps later on.

One technique to implement fault tolerance is to use redundant modules, meaning that there are several instances of the same module in the system. It can for example be several sensors of the same kind at almost the same spot. A voting system compares the sensor values and validate which values to use in the system. If something is wrong the failsafe procedure shall be started. If a systematic fault in a module exists, all the redundant modules can be expected to make the same mistakes. Therefore, it is also common to implement diversity in systems. Given some input and expected output there are many ways to produce the output. With diversity, heterogeneous systems are considered. The developers find different ways of implementing the module. All solutions should of course fulfill the specification. For software development this approach is called N-version programming [15]. A voting system is used here as well. One voter alone would present a single point of failure, but it is also possible to have several voters for the same inputs to enhance the safety. With several actuators the safety can be enhanced even further. This increases the fault tolerance in the system. Another strength with N-versions, regarding software development, is that it is possible to use different programming languages or compilers in the development to increase the diversity even more.

Another often used technique for software development is defensive programming where it is checked in the code that incoming values are within acceptable ranges. Depending on the application, this can for example be min and max boundaries from the current value or simply extremes.

Standards also give restrictions and recommendations on methods to use when building the software. For example it is for higher SILs required that dynamic memory should not be used since it is easy to make programming mistakes with for example reading or writing outside an array. It may also be complex to keep track of where data are stored in memory and how much that is allocated at a specific point. Running out of memory during execution can cause unexpected behavior. Many times, the number of commands used in a programming language is reduced to a subset of commands to reduce the options and simplify the analysis.

The standard also take in aspect related good practices in code structure, such as well commented and readable code. The compiler shall also be chosen with respect to proven functionality

and should have been tested in use.

Watchdog timers can be used to monitor functionality. It is a clock ticking downwards. If the timer reach zero it will start the fail safe procedure. A properly working system will reset the timer periodically to a given value. Several other types of monitoring techniques also exist. For example one module can monitor that another module is working properly and making correct computations.

It is also the responsibility of a safety system to fail in a safe way if a hazardous event is detected, often with the meaning that all actuators shall be disabled without any delays.

In the project process, planning for maintenance, validation, installation and commissioning are performed in parallel with the implementation phase. The final phase in the process is decommissioning and with that the whole life cycle of the product have been covered.

An important aspect to keep in mind when developing a safety related product is that evidence of conformance to the standard is necessary for all parts of the system, both hardware and software if it should pass the certification. There should also be evidence showing that the process was carried out compliant to the standard and that the documentation is sufficient.

Another thing to notice is that since the safety functions have to be validated and verified extra rigorous. Hence, it is better to keep them as simple as possible. The more complex they grow the harder will it be to verify their correctness.

4.2 Mixed Criticality Systems

A Safety Critical System can hold functionality of different critically levels with both highly critical tasks, less critical tasks and regular tasks. Such systems are called a Mixed Critical Systems (MCS). One example can be a pacemaker, which have the critical task of giving electrical pulses to keep the contraction pace of the heart and other tasks such as sending heart rate data and/or battery status to an external device. Another example is in a commercial aircraft, where we have both the steering and video entertainment systems in the aircraft. To make a system safe, safety have to be considered for the whole system including the entertainment application. It would of course be costly and a waste to certify the entertainment system for safety. Standards do however allow mixed criticality levels within the same system if they can be isolated in such a way that the safety of the highly critical steering system cannot be affected by the entertainment system. To guarantee this, the functions have to be isolated and separated both in temporal and spatial domains. The temporal domain refers to the separation in time and spatial to memory and resource handling. It includes both the user memory accessed during runtime of the application, and the kernel memory handling. This can, for example, be done by statically allocating and dedicating memory for different functions.

Burnes and Davis [16] presents a review of the research from 2007 to 2014 and present challenges for further research regarding MCS. Much attention has been drawn to find good scheduling algorithms to separate tasks in temporal domain on single processors, such as Fixed Priority Scheduling (FPS) and Earliest Deadline First (EDF). There are plenty of work done on how to effectively schedule jobs to ensure that mixed criticality functions do not interfere with each other [17][18][19]. Embedded systems with several processors and modules on the same chip consumes less energy than solutions using single processor and multiple chips. To analyze these systems, sometimes including Networks On a Chip (NoC), presents challenges to researchers in order to isolate functions of different criticality level.

Pellizzoni et al. [20] presents a design methodology for Systems on Chip (SoC) to guarantee temporal and spatial isolation in a MCS. A formal method is used to together with a set of certificates that describes the intended applications behavior. The tools presented in the paper that applies the methods, automatically generate hardware wrappers that enforce the behavior defined by the certificates. In particular, run-time monitoring is employed to formally check all data communication in the system is temporally isolated. The tools are demonstrated and tested in a case study of a pacemaker system, where mixed critical tasks of diagnostics, logging and heart contraction runs in the same system. A RF transmitter enables wireless communication and setting of certain parameters without intruding the highly critical tasks.

If jobs or functions of different criticality are to communicate, we face a tougher challenge. The functions are now interacting and the high integrity level for the safety critical functions

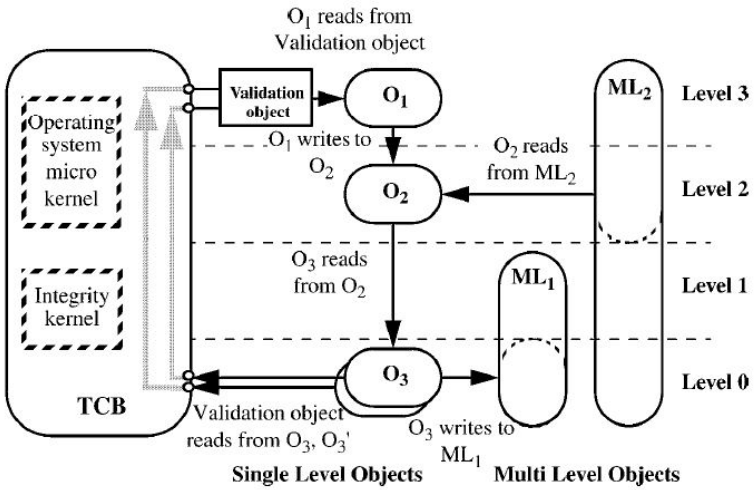

Figure 3: Totel's integrity model for communication between different SIL in a MCS [21]

should be preserved. One way to achieve this, as mentioned, is the costly way to design all modules in the system at the highest intended integrity level. Totel et al. [21] however, presents an integrity model that enables communication between different levels of integrity. The only concern for downstream communication from a higher SIL to a lower is to make sure that there is no backward leakage. This is defined as the diod property. Totel's model shown in figure 3 also suggests a special channel through which data from a lower SIL can be upgraded and used at a higher level. To ensure safety, all information going upstreams have to be validated to the intended level. This channel is secured by a Trusted Computing Base (TCB), that offers a specific entry point for the upstream communication and the diod properties. The TCB consists of a micro kernel used to schedule messages and an integrity kernel that ensures that the information moves in the correct direction of the TCB. Before the data is upgraded to the new level it also have to be validated by a level specific validation object (VO). These validations are performed using common safety techniques. A specific TCB have to be implemented for each SIL to SIL communication. The model also presents multi level objects that could be implemented for the highest intended SIL but operate at any level. When it is used on a lower level though there is no way to upgrade such an object back to a higher level again. Such an object would require that the highly critical task is executed before tasks on lower levels and would hence not enable upwards communication. Laarouchi et al. [22][21] evolves Totel's model to be useful in distributed systems and purposes use of virtualization in avionic systems. Furthermore, they use Totel's model to enable the communication to the avionic system in the Airbus from a COTS laptop, and present how the techniques can be used for maintenance scenarios [23]. In addition the take-off procedures case for avionic systems is considered in [24] that also proposes the use of proxy TCBs where one TCB is implemented between each consecutive SIL instead of one for each SIL to SIL. In this way a communication from level 1 to 4 have to pass through 3 TCBs and validated by a validation object at each level before it can be used.

Wasicek et al [25], inspired by the work of Laarouchi et al. also evolves Totel's model to work on a MultiProcessor System on Chip (MPSoC) architecture in the ACROSS system for vehicles. The proposed design uses virtualization and is implemented on a time triggered architecture. Furthermore they demonstrate the design in a car system where an omnimeter sensor is used both for counting traveled miles and the braking system utilizes. This seems to be a promising approach but Wasicek et al. concludes that more research need to be done in order certify this kind of systems.

From papers citing the model proposed by Totel et al only a few papers focuses on connecting COTS to critical systems and it seems to be limited to one research group. A weakness besides this is that the hardware of COTS is never considered. From standards such as IEC 61508, that is an important part of certifying a product for safety.

4.3 Virtualizaton

As several papers in section 4.2 have stated virtualization would offer isolation properties that is a requirement for MCS. This section presents these techniques with attention to how they could be utilized in a MCS with respect to the goal of communicating with a COTS device.

The term virtualization refers to a software abstraction layer where an OS or application running above it is unaware of it. The virtualization mimics the real environment that the program or OS is adapted to. Hence, for example the OS, can operate in the same way as it should have done if it ran directly on the hardware. This virtualized environment is usually called a Virtual Machine (VM).

The first virtualization technique was developed by IBM back in 1964 [26] and has been used in servers ever since. The increasing numbers of transistors on chip and cheaper hardware however made the interest of virtualization fade during the following decades. With the increased usage of the Internet and the need for storage and analysis of big data in servers, new virtualization implementations came in the late 1990:s and during the 2000:s. The Xen and Azure hypervisors gave virtualization a new push forward and the techniques are today widely used in data centers.

During the last decade the power of virtualization has been exploited in the domain of cloud services. Big companies such as Amazon and Microsoft are nowadays offering Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as as Service (SaaS). The IaaS provides an infrastructure and hardware resources for virtualized OSes and servers. This infrastructure makes it possible for a company to rent hardware on a data center instead of buying and keeping own in the house. PaaS provides a platform and all needed to care about is to develop the application meant to run on that platform. For example, a web server platform can be rent, with all the support (linux, apache, mysql, php, etc.) needed to run a website. All the tenant have to care about is the coding. SaaS separates the application software from the platform and can give access to different editors for example through the web browser. This for example, enables the user to edit documents or to send e-mails without installing any editor or mail client on the own system.

The Virtual Machine Monitor (VMM) also called the hypervisor is a specially designed OS that handles and manages resources among the Virtual Machines (VM) running on it. An OS that runs in a VM is called the guest OS and the underlying hypervisor OS i called the host. There are mainly two types of hypervisors today.

Type 1, also called ”bare metal”, runs directly on the hardware and manages the real hardware resources among the guests running on it. This architecture makes it possible to move a whole guest OS from one piece of hardware to another. Hence a server crash in a data center do not have to make the services unavailable for a very long time since the OS easily can be migrated to another machine. Running several different guests on the same piece of hardware also enables better resource utilization. In data centers, several lightly loaded server systems can for example run on the same hardware and still deliver its services with adequate quality. If the resources becomes a limit for the performance at some point one of the guest OS can be migrated to another less loaded piece of hardware and in that way balance the total load among the hardware in the data center. It is also possible to run different types of OS such as (General Purpose OS) GPOS and (Real Time OS) RTOS on the same machine as shown in Figure 4. The GPOS could for example be used for regular applications, a RTOS for controlling hard real time tasks.

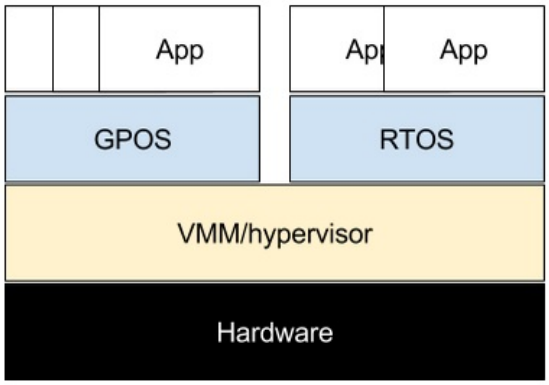

Figure 4: Type 1, bare metal, hypervisor design

Type 2 hypervisor is called "hosted". As illustrated in Figure 5 it runs like an application in another OS. This makes it possible for a programmer or designer to run and test applications in different environments in a convenient way on the same machine saving a lot of time and space.

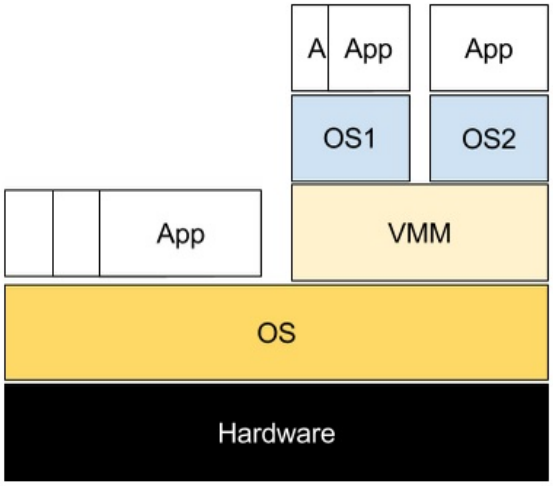

Figure 5: Hosted, type 2 hypervisor design

Hypervisors are furthermore divided into two main types of virtualization.

In native or full virtualization, a binary copy of a guest OS can be installed directly in the VM on the hypervisor without any modifications. The hypervisor is responsible for trapping and handling all requests from the guests. It is rather easy to port an OS to this architecture but it has been shown that extra overhead especially in the memory handling has bad impact on the performance. The issues with the virtual memory handling in page tables and system calls where addressed by Barham et al. [27] which introduced paravirtualization in the Xen hypervisor. With as small as possible modifications to a number of OSes they managed to reduce these overhead costs of the hypervisor.

In paravirtualization the guest OS is modified to replace all privileged instructions with explicit calls to the hypervisor API. These calls are called hypercalls. Paravirtualization also allows to replace multiple privileged instructions with a single hypercall, reducing the number of context switches between privileged and unprivileged modes. These rather simple changes makes the paravirtualized versions perform closer to an OS deployed directly on the hardware and is the most widely used technique today.

Reinhard and Morgan [28] demonstrates that a hypervisor can be built to run on a rather simple processor and run multiple paravirtualized VMs on it and states that the bare metal hypervisor could be useful for running applications separated on the same ECU especially where isolation is

important. Landau et al. [29], have also shown that it is possible to implement hypervisors for multi core processors and even dedicating CPUs to specific VMs.

Gudeth et al. [30] argues that bare metal hypervisors are to be preferred for security applications. All hypervisors found during this background research, that are designed for security purposes, are also of this type. The argument that the extra layer in a hosted solution would be a security risk can directly be transfered to a safety related application. The extra software and complexity would also make it harder to analyze. For security and safety applications the Trusted Computing Base (TCB) of the kernel should be kept small. The type 1 hypervisor seems to be preferable over type 2 for our purpose.

Hypervisors are however not the only way to achieve virtualization. Microkernels such as L4 [31] has been demonstrated to support paravirtualized OS. While hypervisors are developed to run several VMs simultaneously on the same platform the microkernels aim to keep the kernel as small and simple as possible, by minimizing the amount of privileged code and moving things that do not necessary need to be in the kernel out of it such as device drivers, file systems and I/O. The TCB is kept small but microkernels seems to suffer from a high load of context switches and calls to the kernel. Microvisor is what OKL4 [31] call their hypervisor of which they have tried to reduce the hypervisor kernel to a minimum. More recent microkernels have improved the performance to acceptable levels.

In distributed systems like in cars with many ECUs, keeping the Bill Of Material (BOM) down is a selling argument for virtualization techniques. By virtualizing, it is possible to physically remove ECUs close to each other and use one instead with virtual machines running on it and for example, allowing MCS like ABS braking system and the radio receiver at the same unit.

The speed, size and energy consumption have usually been optimized for the purpose of the application in embedded systems. Hence, efficiency and utilization problems that motivates virtualization in data center are usually not an issue for mobile devices. Some properties do however make virtualization interesting for mobile applications as well.

Many applications have been developed and used in the highly flexible OSes provided with the mobile devices. When viruses, trojans and other malware have been intruding more frequently in these softwares, the motivation for securing these devices has increased. The isolation properties of virtualization drew the attention to mobile vendors.

Both ARM and OK labs published white-papers [32][33] a few years ago with software techniques that potentially may make smart-phones and tablets useful for security or safety applications. Both present virtualization techniques to isolate software in temporal and spatial domains.

In 2009 Motorola released the first smartphone, Evoke QA4, with support for virtualization. Many others have followed and in recent years several hypervisors have reached the market supporting ARM and Intel processors widely used in mobile devices.

One of the applications that have been a driving force in the development of virtualization for mobile platforms comes from the idea: Bring Your Own Device (BOYD) to work. The idea origins from the fact that many employees have to carry two smartphones in their pockets. One private and one for the corporation. Companies want to keep their data secure and hence private phones, with a big diversity of softwares, are not trustworthy. However, by running an isolated VM on the smartphone it is possible to design and implement a secure interface to the corporation network. Some high tech companies are already using this technique and reducing the Bill Of Material (BOM) since they don't need to supply all employees with company phones anymore. BYOD could also enable employees to work remotely on devices that they are familiar with instead of having to adapt and learn a new system. These things and the flexibility to work anywhere could also increase the productivity. VMWare among others, have also demonstrated that an user friendly implementation makes it possible to switch from one VM to another almost like switching desktop screen [34]. Cells [35] is another solutions for running several virtualized phones on the same platform that enables BYOD.

Just like these kind of security applications it could be possible to use virtualization and VMs to isolate tasks for safety purposes from others. Prioritized RTOS have been shown to meet deadlines and ensuring real time behavior, and vendors like Wind River, Green Hill and PikeOS offer safety certified hypervisors for ARM platforms that often are found in mobile products. These hypervisors have been tested for applications in avionics and vehicle industries. Some other virtualization giants such as KVM, Xen, OKL4, VMWare and Red Bend are also developing mobile hypervisors [36].

Virtualization techniques may be used both on the machine side and/or on the mobile COTS device side to offer the possibility to implement safe software platforms beside regular applications on the same hardware.

4.4 Distributed Systems

Many systems are built by several subsystems or devices called nodes in a distributed system. To make the whole system work properly the nodes need to communicate and exchange data in order to perform its job. Machinery systems are often distributed system where nodes are placed closed to its operational functionality in order make installation simpler and to save power cable lengths. A combine harvester harvesting grain crops, for example, have nodes for controlling the cutting mechanisms, the threshing drum, straw walker and the straw chopper, distributed over the machine.

This section presents some typical machine nodes, how communication between nodes works and the meaning of safe communication. In this thesis, wireless communication between mobile COTS devices and machinery systems is considered. Therefore some extra attention is spent on such protocols, and less on internal machine communication.

4.4.1 Machine nodes

An I/O controller is a device that controls inputs from sensors and outputs to actuators. In machine control systems it typically contain a DC/DC unit that transforms power input into all necessary voltage and current levels. It typically also have communication peripherals such as CAN and/or Ethernet connections. Hence the I/O controller can communicate with other parts in the system to give or receive instructions. Analog inputs from sensors are usually converted to digital using an Analog to Digital Converter (ADC). Motors and hydraulics are typically controlled by coils and electromagnetism using Pulse Width Modulation (PWM) signals. For example in a hydraulic cylinder, a PWM signal is modeled to control the valve that regulates the pressure and hence the force that the machine needs to perform its task. Other typical input and output are digital on/off signals. The micro-controller is the brain in the I/O controller. It is responsible for performing calculations and making decisions according to its programmed instructions.

Machine functional safety is not in the scope when developing a new I/O controller. It cannot be predicted how throttles, motors or valves will be connected or which ports that will be used. It can only be ensured that the product developed performs it intended actions properly and safe and holds features to perform or trigger safety functionality. It can for example be verified that the PWM-signal emitted is correct, and if its not, handle the error in a safe way.

A safety I/O node is usually delivered with certified hardware, firmware and an Application Programmers Interface (API). Software such as a certified RTOS can also be provided.

Another important node often found in distributed machine systems are the job computer, which works as a central node to coordinate the I/O nodes.

Other typical nodes found in machines are I/O nodes connected to user interfaces such as joysticks and displays.

4.4.2 Communication

To enable communication there have to be a sender and a receiver. A simple example is Alice and Bob talking to each other. When Alice says something, sound waves are sent/transmitted trough the air and received by the receptors in Bobs ears. Bob then processes the information in his brain and responds. Typical problems would be if Bob do not listen, occupied with something else, or if Alice speaks a language Bob doesn't understand. To enable electronics to communicate, protocols have been defined to make the sender and receiver understand each other. The Open Systems Interconnection (OSI) reference model shown in figure 6, defines 7 layers of protocols that can is used by nodes to communicate. Each layer is responsible for different parts in the communication, provides services for the layer above and presents data to the layer below in the predefined form. Tannenbaum [37] presents a comprehensive description of computer networks including the OSI model depicted in figure 6.

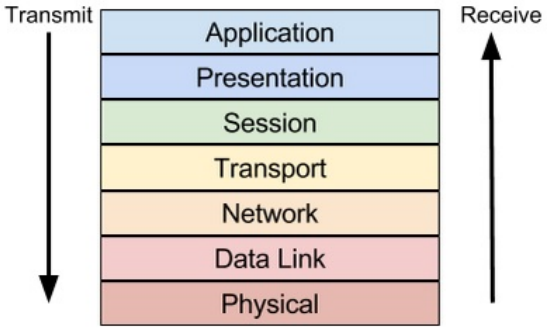

Figure 6: The seven layers of the OSI model

The first layer handles the physical aspects of the communication such as radio transmitters, connectors and cables. The protocol in this layer defines voltages, frequencies and electrical pulses. For wireless communication it is considered how the signal is modulated on the carrier wave. This can be done by defining the meaning of different amplitudes, frequencies or phase shifts. Another physical aspect to consider for communication is the range. Higher frequency gives shorter ranges than lower due to attenuation. Higher frequencies however deliver higher data rates.

The second layer is the data link layer. The main task of the layer is to produce a data frames in the size defined by its protocol and present them to layer one. The layer concerns the link between two nodes and that messages are transmitted. An important part of this is the Medium Access Control (MAC) layer, which defines how the nodes may utilize the medium. The MAC layer is divided in two main variants of protocols: contention-free and contention-based. Contention-free protocols are designed to avoid that nodes within the network interfere each other. This can for example be done by dedicating time slots or frequencies for different nodes. If some node do not use its slots this will be a waste. An example of such a protocol is (Time Division Multiple Access) TDMA. Contention-based protocols calculate on having contention in the network, but nodes do on the other hand not have to wait for its slot before transmitting. The Carrier Sense Multiple Access (CSMA) widely used in computer network is such an example. Problems can occur if two or more nodes transmits at the same time. There exists several techniques to handle this. The two most widely employed ones are the Collision Detection(CD) and the Collision Avoidance(CA). These protocols uses small Request To Send (RTS), Clear To Send (CTS) and Acknowledgment (ACK) packages to inform other nodes of its transmission state. If a node sends a RTS package the surrounding nodes can hold there own packages until the transmission is finished. Many more MAC-protocols exists.

The third layer is the network layer. It controls how the package shall be routed from source to destination in a network. Routing protocols controls where a received messaged shall be forwarded to reach its destination. There are several known protocols here as well. A simple example is the flooding algorithms where packets are sent on to all other nodes expect from where it comes from. Nodes keep records of which packets that they have already flooded and do hence only flood once. Other protocols utilize routing tables to direct the communication to specific nodes. This usually decreases the number of transmission and hence saves energy. The Internet Protocol (IP) provides uses IP-addresses to in order to route packets. Subnetting is a course on its own regarding this.

The fourth layer is the transport layer. The main task of this layer is to provide a reliable and cost-effective data transport from source to destination. The layer is a glue between the underlying layers and the upper ones in the TCP/IP model, making it possible for the programmer to implement applications with less care on how packets are routed and transmitted from source to destination, usually referred to as socket programming. The transport layer provides that service. The two main Internet protocols are the connectionless User Datagram Protocol (UDP) and the connection-oriented Transport Control Protocol (TCP). UDP is connectionless meaning that it do not care of establishing a connection to the receiver before sending, which is the case for TCP. UDP is more used for sending single datagrams of data while TCP is used to stream much data.

The fifth layer is the session layer. It controls communication sessions between computers, and manages the connection between local and remote applications.

The sixth layer is the presentation layer. Unlike lower layers moving bits around, this layer is concerned with syntax and semantics. In order to make it possible for different applications on different platforms to communicate this layer handles the presentation of the data. It maps it in an for the application understandable way.

The seventh layer in the model is the application layer. It is the layer closest to the user. This is where data is presented to the user and where he can interact with the system. A typical protocol in this layer is the Hyper Text Transfer Protocol (HTTP), which is widely used in web browsers. Other well known protocols are File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), and Secure Socket Layer (SSL).

A network application employs the OSI stack to enable its features. When a message are to be transmitted it works downwards in the stack attaching headers for the protocols on the different layers. The message is then sent over the physical channel. The receiver reads the headers in each layer in reversed order to direct the message to the receiving application.

Not all nodes involved in the communication reads all headers. For example a router working in the network layer only reads and removes the first three layers. With that information it decides where to route the packet and works it's way down through the underlaying layers attaching new headers and forward it to the next node. A data switch, working on the data link layer, only consider the two lowest layers.

Which protocols to use for a specific application is a design issue and depends on what properties that are important. Aspects that could be of interest are for example reliability, scalability, latency and energy consumption.

Moreover communication networks can be formed in different topologies as depicted in figure 7, which describes how the nodes are connected to each other.

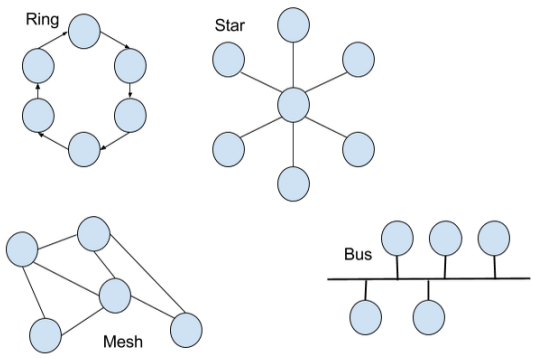

Figure 7: Network topologies

They can for example be connected in a ring where all nodes have two neighbors. One to the left and one to the right. The line topology is almost the same, but it have end nodes which only have one neighbor. The star topology have one node in the middle to which all other need to send to in order to communicate to the other nodes. Another topology is the tree where nodes have one parent and child nodes. Especially in wireless network the topology is not fixed since nodes are mobile. In such applications it is common to work with Ad-Hoc networks to which new nodes can connect. When nodes can be connected to each other in different ways it is called a mesh. In a fully connected topology all nodes will be connected to every other node in the network.

In vehicles and machinery systems the communication is usually performed on data buses. A bus is a network topology where all nodes communicate over the same channel. The Controller Area Network (CAN), widely used in the automotive industry, is an example of a protocol defined for the two lowest layers in the OSI stack operating on the bus topology. CAN is designed for time critical systems, where a deterministic behavior is crucial. In CAN all nodes have an unique id which also presents the priority of the messages from the node. A message from a highly prioritized node will always be sent before a lower one if the nodes try to send at the same time. This makes it possible to calculate worst case times for messages of different priority on the bus. Other protocols, such as Ethernet, are also popular in machines for less time dependent tasks.

Wireless Communication

To enable wireless communication to a mobile COTS device it is necessary to have some wireless communication device deployed within the machinery system. It will work as an access point or gateway to the machine.

It is not always an easy task to choose the right technique for a new application. Ferro and Potorti [38] gives a survey and comparison between Wireless Fidelity (WiFi) and BlueTooth (BT). Lee et al. [39] also includes Ultra Wide Band (UWB) and Zigbee in their survey. The survey aims to be helpful for engineers to choose technique for their specific application. The essences of the comparisons is described in the following paragraphs. In addition the from the surveys Universal Mobile Telecommunications System (WCDMA) protocol used in 3G will be presented.

Both WiFi and BT presents wireless techniques for the two first layers in the OSI model, the physical layer and the data link layer. The protocols where however designed for different originating purposes and are hence different in some aspects. WiFi is designed for business and home networks where several computers are connected. The protocols are optimized for speed and security. BT on the other hand is designed to replace cables in small Personal Area Networks (PAN) such as mouses or headsets. Hence the range of a BT network is much shorter but accordingly also less power consuming. While new versions of WiFi tries to push the limits of high data rates by for example using multiple antennas to increase the throughput, BT have not considered to send a lot of data even though later versions have higher bit rates.

Both protocols operates at the Industry, Scientific and Medical (ISM) band and can hence together with many other products like microwave owns and other radio links interfere with each other. The ISM band is considered as free natural resources around the globe.

On top of the two lowest layers from the OSI model more layers can be pushed to fit what we need for the actual application.

The MAC in WiFi uses the contention based protocol of CSMA/CA. In intention to avoid collisions, small packets of Request To Send (RTS) and Clear To Send, (CTS) is used. To confirm that a data transmission was successful acknowledgments (ACK) are used. Networks using this protocol can be formed in any topology.

BT is design for master-slave relationships between nodes in the network. There can be only one master per network. The master is in control of the slaves and tells them when and what to do. A slave may not communicate directly with another slave. The message has to be passed via the master. There can be 7 active slave nodes at the same time but up to 255 in total. Passive nodes are mainly in a sleeping mode. The communication is divided into time slots where the master talks on even slots and slaves in odd slots. The slots are Time Division Multiplexed (TDM). As implicated the master also controls which slave that may use the channel and when. The fact that BT uses TDM makes it more time predictable in comparison to CSMA/CA. One should however notice that data can be lost even if time slots are dedicated due to for example interference or attenuation.

Another younger candidate protocol is the 802.15.4 Zigbee. It is designed to be used in Wireless Sensor Networks (WSN) where low energy consumption is regarded as especially important. Since sensors often are deployed far away from a power source they have to rely on battery or energy harvesting.

The main advantage of the Ultra Wide Band (UWB) is the high data rates which make it interesting for replacement of USB and such in home and video appliances. It is intended for short range.

The International Engineering Consortium and others [40] gives an overview of the history of protocols aimed to meet new demands for digital mobile communication. New applications beyond calling and sending SMS such as surfing the Internet and streaming videos have pushed the development towards higher data rates. Universal Mobile Telecommunications System (UMTS) is the successor of the Global System for Mobile communication (GSM). UMTS is also referred to as the Wideband Code Division Multiple Access (WCDMA). It employs a 5 MHz channel bandwidth and data rates up to 2 Mbps can be achieved. Just like the WiFi protocols it is the lower layers of the OSI model that makes it special. The physical layer, the data link layer and the network layer controls the communication to a base station which in next step routes the communication. Base stations are deployed in the landscape to form a grid. They are strategically deployed to support

as many users as possible in a cost effective way. If a device cannot connect to a base station the services cannot be utilized. A Subscriber Identity Module (SIM) card is also required to identify the mobile device in order to utilize the services.

WCDMA protocol is backwards compatible with the predecessor GSM and hence 2G mobile communication protocols can be used instead, which with lower carrier frequencies has longer transmission ranges.

In upgrades of WCDMA such as High Speed Downlink Packet Access (HSDPA) and High-Speed Uplink Packet Access (HSUPA) transmission speed have been increased to 14.4 Mbps. A downside is that sharing of bandwith with voice calls and other Internet traffic can cause unpredictable delays in the communication. Another downside is that the 3G grid do not cover all possible location for a machines such as deep in the forest or in a mine.

Inter-VM Communcication

If the two instances of different criticality level are to be run on the same ECU, virtualization, as presented in section 4.3, is a good alternative in order to ensure isolation. Virtual machines can communicate using regular communication protocols such as those described above, using their virtualized network interfaces. If the instances share the same hardware platform this could however cause unnecessary overheads for latency and throughput. Wang [41] presents a number of techniques based on the Xen hypervisor that bypasses the isolation barrier and enables that VM instances on the same hardware can communicate directly. The main technique is to use a shared memory area, like a pipe, into which one VM can write data that immediately will be available for the receiving VM. XenSockets apply this technique using a circular buffer. The hypercalls to the hypervisor are similar to socket programming used for communicate between application. XenSocket only support one way communication. XWay is another inter VM communication technique that uses shared memory. In addition to the XenSocket this interface supports bidirectional communication. Inter Virtual Machine (IVC) and MNET are two other techniques that also support bidirectional communication. Wang compares the techniques and concludes that they all have pros and cons regarding the performance issues. Diakhate et al.[42] also investigates inter VM communication and presents a virtualized Message Passing Interface (MPI) that do not require any modification of the hypervisor kernel. An important feature that have to be considered when using this types of channels in a MCS as described in section 4.2 is that the isolation properties have to be preserved in order not to degrade higher SIL.

4.4.3 Safe Communication

Safe communication aims to ensure that the data sent from the source is correctly received by the destination node. Communication protocols such as CANOpen Safety, openSAFETY and SafeEthernet have been developed to provide high reliability and safe communication for safety related systems. These kind of protocols typically implements one or more of the following types of techniques [43]

1. Sequential numbering, the method assigns packet numbers incrementally in an defined way.
2. Time stamp, adds time stamps to each packets to see when it was created and hence also how old the data is.
3. Acknowledgments, essentially like ACK used in contention based protocols to acknowledge that the transmission was successful, but here whole message is sent back to make it possible for the sender to verify that the right message was received. The message can be an echo or manipulated in a predefined way. The sender will the acknowledge the transmission.
4. Identification, identifies sender and receiver by recognizing them by a specified identifier that is added to the packet.
5. Data safety, tests data content for correct transmission at the receiver with for example Cyclic Redundancy Check (CRC) and sometimes with Hamming Code to enable correction of bit errors.

6. Redundancy with cross comparison by sending the same package several times, or the same data but with inverted data bits, that are compared to verify that the data is correct.

The medium of the channel or the number of nodes the message passes before arriving at its destination is not important in the protocols. As long as it can be verified that what was sent also is what arrived at the destination. This is what's called a gray channel. Anything may happen in the transmission as long as the receiver can verify that the arrived packet is correct. Safety certified protocol such as safeEthernet or OPENSafety is designed to meet SIL 3 requirements, by showing that the rate of undetected bit errors are below the acceptable probability.

To make sure that the received data is correct the CRC is the most important and widely used technique. It outperforms the simpler parity check bits that only can detect a few bit errors. CRC can detect several bit errors and sequences. The amount it can detect depends on the size of the CRC.

To calculate a CRC a polynomial division is used on the package intended to send. The checksum is then added to the packet. The receiver performs the same calculation on the packet and compare with the CRC bits. If they do not match the packet is regarded as corrupt. The polynomials used are usually 16 or 32 bits. More details on how the CRC is calculated is well described by its inventors Peterson and Brown in [44]. A corrupt package can be repaired by using techniques like Hamming Code but in safety application it is more common to simply discard the faulty data and ask for a new package.

OpenSafety provides what they call a black channel assessment where the safety protocol is implemented in the application layer in the OSI model. The safety layer is added in the payload as an extra layer. This decreases the size of the payload but it is on the other hand relatively easy to implement. Sending information from one node to the other can accordingly be done like in any other network. The medium used, nodes visited in between, as mentioned, will not matter as long as the packet arrives and the receiver can verify it according to the safety protocol.

If time constraints are important for the communication on the other hand the media may matter. For hard real time applications a wired communication channel is preferred over wireless due to its higher package delivery ratios. Wireless communications are more frequent interfered by other sources that can cause packet losses and retransmissions. For time critical applications it can be important that bit errors and retransmissions of packets are rare since it would be impossible to predict the timing and latencies otherwise. For a safe application it would also be appropriate to implement some safe procedure for deadline misses.

Pendil et al. [45] presents derived mathematical models that can be used to perform safety related calculations on wireless channels with respect to the IEC 61508 standard. Calculations of PFH, is used to obtain the SIL level. The calculations regards both Bit Error Rate (BER) and bit erasure. Pendli et al further adds safety functions to the bluetooth stack in [46] and [47] and shows by simulation in a General Erasures Channels (GEC) that combines the two types of bit errors, that their implementation can reach SIL 3. The 2n-bits method invented by Jitsukawa [48] is used to guard the communication from attenuation and bit error problems by adding the inverted bit to every data bit. The word is sent twice and protected by a 32 bits CRC.

The Vehicle-to-Vehicle (V2V)[49] system is derived from the 802.11 (WiFi) standard and is a protocol for communication between cars. Cars can communicate with other cars up to a range of 250 m to detect and inform each of their speeds or warn for slippery roads ahead to prevent accidents. The cars forms Ad Hoc networks with the other cars in range.

4.5 COTS

This section presents a brief overview of a number of State Of The Art (SOTA) Commercial Off The Shelf (COTS) smart phones and tablets to get a picture of which features that characterize them. The set consists of Apple iPhone 6, Samsung Galaxy S6, Microsoft Lumia 640 LTE, Apple iPad Air, Microsoft Surface Pro3 and Google Nexus 9. The products are chosen to give a span over different popular vendors and recent versions.

4.5.1 Software

iOS, Android and Windows are OSes that supports applications such as facebook, google maps etc. running in the system. They also provides a number of background services for maintaining the OSes, and the kernels are based on the high level programming languages Objective C, C and C++. As can be seen in table 3 all products use either Apple iOS, Windows 8/8.1 or Android 5.1 operating systems. All product support Bluetooth (BT) and WiFi but the tablets do not support 3G.

type	name	OS	BT	Wi-Fi	3G
phone	Apple iPhone 6	Apple iOS 8	yes	yes	yes
phone	Samsung Galaxy S6 SM-G920F	Android 5.0 (Lollipop)	yes	yes	yes
phone	Microsoft Lumia 640 LTE	Windows Phone 8.1	yes	yes	yes
tablet	Apple iPad Air	Apple iOS 8	yes	yes	no
tablet	Microsoft Surface Pro 3	Windows 8/8.1 Professional	yes	yes	no
tablet	Google Nexus 9	Android 5.1 (Lollipop)	yes	yes	no

Table 3: SOTA COTS device SW characteristics

4.5.2 Hardware

Smartphones and tablets are complex systems with a number of input sources such as accelerometer, GPS and touchscreen, but more important is the processor which the devices wouldn't work without. These devices would have to be examined as a part of a whole safety system if used in interaction with such.

In table 4 the processors of six relatively new COTS that can be acquired on the retail market are presented. All processors are at least dual core processors. Most of them are also SoC architectures which, as described in section 4.2, increases the complexity of the hardware analysis.

In addition, not shown in the table, all devices do of course contain wireless connectivity interfaces.

type	name	processor cores	processor type	SoC
phone	Apple iPhone 6	Dual Core	Apple Cyclone	yes
phone	Samsung Galaxy S6 SM-G920F	Octa Core	ARM Cortex A-57	yes
phone	Microsoft Lumia 640 LTE	Quad Core	3 ARM Cortex A-7	yes
tablet	Apple iPad Air	Dual core	Apple Cyclone	yes
tablet	Microsoft Surface Pro 3	Dual core	Intel Core i5 4300U	no
tablet	Google Nexus 9	Dual core	ARM Cortex-A15	yes

Table 4: SOTA COTS device HW characteristics

System On Chip

Circuits in COTS devices today contain many modules. Since less area decreases the energy consumption SoC is interesting for these devices. Nano transistor sizes makes the chips in modern devices small and possible to integrate several processing cores, memory and Graphical Processing Unit (GPU) on the chip. Systems integrating several processors are often referred to as MultiProcessor System on Chip MPSoC. Internal buses or NoC enables communication between modules on the same chip, and makes them very powerful.

Bates et al. [50] investigates the using of super scalar processors in safety critical systems with respect to certification arguments for processors. They compare two types of processors, COTS and bespoke processors. COTS have the advantage of being widely spread and it is easy to obtain software tools. Bespoke processors are designed for specific purposes and safety arguments can be built when developing it. The core arguments considered are operational functionality, WCET predictability, mitigation of design errors and reliability. Even though the paper was published more than a decade ago, the main conclusions are still important and valid. COTS processors are not designed to be used for safety critical purposes and hence there is a lack of protection from

systematic errors made in the design, such as random hardware faults caused by physical effects and fault tolerance. According to IEC 61508 described in section 4.1.2 a safety system have to consider both hardware and software in order to be able to be certifiable.

More recently, Mohan et al. [51] investigated the use of multicore processor architectures in cyber-physical systems where the machine interacts with the environment. They too, conclude that COTS multicores have serious problems with timing and predictability.

4.6 Data Engine

Data Engine (DE) is a Software Application Platform (SAP) designed by maximatecc AB.

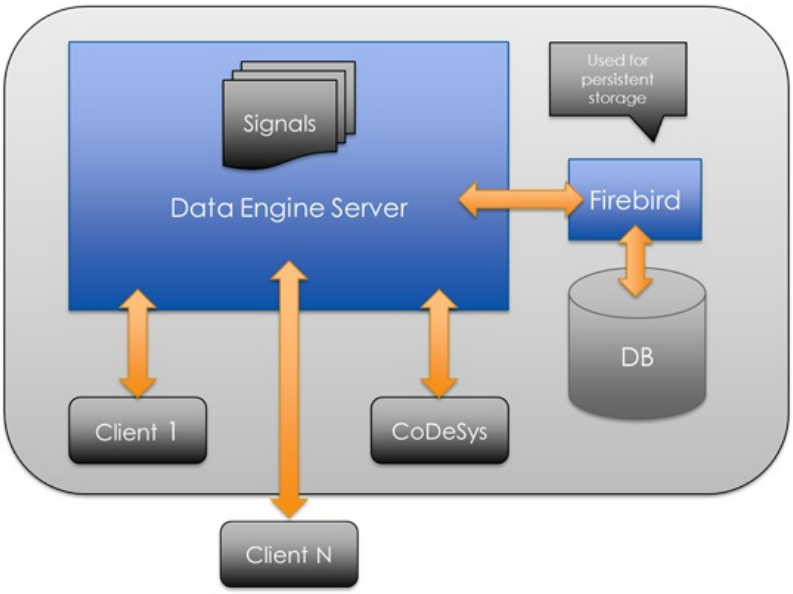

Figure 8: Overview of the Data Engine and the information flow

Figure 8 shows an overview of the DE design. Basically, DE can be thought of as a memory space for storing data with services to distribute the data to the connected clients. Every data entity in DE is identified by an unique integer value coupled to the parameter value. When connected to the server the clients can subscribe to data values as producer, consumer or both meaning that they can send data to or retrieve data from the DE. Every data entity is coupled to an unique identifier. Moreover a special design communication protocol is designed for this purpose.

DE is also connected to a FireBird database in the backend which enables persistent storage of parameters when the power is switched off. Signals that need to be stored until the power is turned on again can be stored here. Features to log data here is not enabled. DE communicates with the database using SQL queries. An interface to the, in machinery systems commonly used, Controller Development System (CoDeSys) is also available.

The most interesting functionality for the user scenarios presented in section 3.2 is the handling of reading and writing parameters in DE. By writing machine parameters in the DE it is possible to read them and present them in a GUI application, immediately when changed or by reading them periodically.

DE is however not designed for safety and accordingly the software is not regarded as safe, but its properties makes it interesting for external access to machinery data which is the case for all scenarios described in section 3.2.

5 Results

This section presents suitable communication protocols depending on the application of the scenarios described in the section 3.2, safety analysis regarding the safety critical system communicating with the COTS device. Furthermore, this is summarized in safety recommendations for the user scenarios. A conceptual system design is proposed that enables feasible user scenarios, and an implementation realizing these scenarios is presented.

5.1 Communication

In section 4.4.2 three main candidates protocols where presented: BT, Wifi and WCDMA.

All protocols regards the lower layers in the OSI model and more layers can be added on top of them to enable all sorts of communication. For security there is more to find in WiFi and WCDMA. If data is stored within the machine BT or WiFi would be more suitable than 3G but if the cloud should be reached with outgoing information as described in the production manager scenario, WCDMA would be better. BT would be better if timing determinism is considered due to its master-slave structure.

Safety protocols using black channel assessment would be possible for all these protocols over a gray channel. Protocol stacks for using TCP and IP are usually also prepared and hence, it would not be a too big problem to change the underlying layers at a later point.

For sending log data extracted from the machine however, a safety protocol would not be necessary since the data is not used directly for safety functionality. It would however be appropriate to check that the data sent is correct when reaching its destination. CRC and checksums are implemented in many protocols to avoid corrupted data. CRC is for example implemented in TCP on headers and the payload. An extra safety layer would in this case only complicate the implementation for this type of application.

Since 3G is not available everywhere it would be appropriate to enable both 3G and WiFi stacks. A WiFi network can be set up and used in the surroundings of the machine.

If a machine system such as large cranes is considered, WiFi would more likely be able to cover the necessary ranges than BT. WCDMA would not be interesting for this internal machine communication. If WCDMA is going to be used it would be as a gateway to enable live logging in the cloud. It is also possible to store data in a mobile COTS device and submit it at the end of the day when Internet connectivity is available.

If two instances of different criticality level are to be run on the same physical device or ECU, virtualization is a good alternative with respect to its isolation properties. Guest VM can communicate using protocols such as those described above in this section. Performance improvements can be achieved by using VM-intercommunication techniques as described in section 4.4.2

The choice of protocol will hence depend on the application considered.

5.2 Safety analysis

This section analyzes the feasibility of the user scenarios presented in section 3.2 and the different nodes in such a system. The nodes considered in this analysis consists of a safety certified system designed to work in a machine and a mobile COTS device with the characteristics described in section 4.5. Figure 9 shows an simplified model of the considered system.

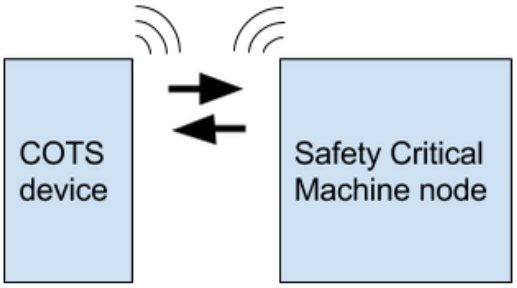

Figure 9: COTS device communicating wireless with a Machine node containing Safety Critical functions

As described in section 4.1.2 a certification shall include all devices considered in the system. This analysis will hence try to include the mobile COTS device into the safety system as much as possible without violating the safety of the already certified machine system. The communication interface between the safe system and the COTS device will give the main input to the safety strategy presented in next section.

5.2.1 Machines

Machine nodes specially designed for safety functions and aimed for certification shall always be developed following the considered standard in mind. In this thesis the functional safety standard IEC 61508, presented in section 4.1.2, is considered. IEC 61508 is only one example of a standard in industry, but serves as a good reference example. It shows the importance of following the rigorous process involved and employing adequate techniques to provide evidence of conformance. When engineers are in control of the process and collaborate with an authorized certification institution there are good chances to achieve the certification.

5.2.2 COTS

COTS devices do not have any restrictions on safety as described in IEC 61508 and are not developed with this in mind. As mentioned in section 4.1.2, things should be kept simple when designing safety critical software to make it as easy as possible to analyze and to provide evidence for safety. The OS:es used in COTS devices have thousands lines of code. Neither is the programming language or the compiler chosen with respect to the standard. Basically there is no chance to analyze this in reasonable time and, as implied, safety issues would likely be found soon, that would disqualify the safety arguments. Their software are hence not suitable platforms for safety critical tasks.

Safe software could be implemented using techniques to isolate safety critical tasks from regular tasks in MCS as presented in section 4.2.

Neither is the hardware designed to meet safety requirements. Probabilities and modes of hardware failures are hence unknown and it can by no means be regarded as safe. The lack of evidence both for software and hardware would make them unsuitable for safety critical functions and hard to argue for in a certification process. To utilize this hardware, extra safety monitoring functions would be needed that checks that the device is working correct and trigger a failure procedure if problems are detected.

Using another mobile and safe hardware platform, containing a wireless communication device, instead of COTS devices, would enable the whole set of user scenarios described in section 3.2. HMI 411 developed by maximatecc AB is an example of such a device. However replacing the COTS device from the system with this puts it out of the scope of this thesis.

5.2.3 Safety interface

All user scenarios described in section 3.2 consider sending or receiving data from the machine. In this part of the analysis the scenarios are grouped since they face the same base problems.

The first group include logging scenarios, live data presentation and repair and maintenance using augmented reality. All these scenarios are sending data from the machine.

The second group will consider machine control where data is received and used by the machine.

The third group covers the updating software scenario. It also receives data but this cannot be done when the machine is running. Here, the machine should be sent to a fail safe mode before starting the update. This special machine mode will be treated in the third group.

Machine to COTS device

When sending data from the machine, the whole process can be under control by the machine. The hardware and software is considered certified for safety and as seen in section 4.2 it should not cause any safety degradation if it is ensured that there is no backwards leakage. When data has been sent from the I/O node it is no longer the responsibility of that particular node to preserve the safety in the a distributed system it is a part of. It now depends on the receiver, in this case the COTS device. Considering a gray channel and using a safety protocol containing the properties presented in section 4.4.3 it can be ensured that the data is delivered and verified correctly. The receiver side however, needs further considerations. If the data presented to the user is going to be used for safety functions while interacting with the machine it is necessary to be able to trust the data presented on the COTS device. Wrong data can obviously cause that the human makes dangerous maneuvers. To ensure this the hardware failure rates needs to be known in order to provide the evidence.

To make the software implementation safe it is necessary to be certain that it is isolated from regular jobs in the system. As we have seen in section 4.3 a promising way to achieve this is through virtualization. The hypervisor used would have to be certified together with the software. Installation of this on a smart phone requires that the smart phone supports virtualization.

The hardware however, still does not present any evidence for safety. To enable the use of the COTS hardware for safety, the usage would have to be limited as much as possible in order to possibly enable safety analysis of a small part of that complex system.

The most promising option would be to create a gray channel directly to the video memory, holding whats displayed on the screen, in the COTS device. Then, that would be the only part needed to analyze in order to get a safe presentation of data. In this solution a bit representation for the video memory could be created at the machine and tunneled directly to the video memory. This is not an easy task and it will lock the solution to the specific piece of hardware and hence decrease the flexibility of installing an application at any mobile COTS device. Safety could also be increased by sending multiple pictures created by diverse functions to different part of the memory and hence presenting them side by side. The user could then decide from what he sees if it is trustworthy or not.

The age of the data, that could indicate the connectivity, would also be necessary to present either directly to the user, or to be monitored by the COTS device. Old data can obviously lead to wrong decisions. Showing a time stamp to the user would probably not be very convenient from a user perspective if he would have to compare it to some system clock, and putting this trust on a human would not be a good safety argument. Monitoring the time stamps by the CPU on the COTS device would hence likely be necessary, but as already mentioned, trusting COTS hardware is problematic.

Another safety technique in IEC 61508 -7 is to monitor processes and check that they are operating correct. Smart phones and tablets usually have both a GPU and multicore CPU in a MPSoC. Since they are heterogeneous the faster one of them could be utilized to monitor that the other one is writing correct data in the video memory which holds whats presented on the screen. This monitoring would decrease the probability of systematic failures, but since the the failure rates are unknown from the start it will still not provide any better safety arguments. This idea would also be complex to implement and would be bound to a specific platform.

Another technique that could be used to increase safety is to perform memory checks regularly to ensure that memories are working properly and that for example no bits are stuck.

These ideas will be complex to implement and as mentioned, bind the application to a specific COTS device, which would spoil the flexibility of using any COTS device.

However, another warning system that tells the user when the machine health is getting bad would not decrease safety and could be implemented as a runtime complement to a safety system.

From a functional safety perspective, the data presented will be a threat only if the user are to act upon it during operation.

The first user scenario, where the operator is presented live data, shall hence only be used as information, not in order to operate on safety functionality. For scenario three, where the technician uses machine data as help for maintenance, and the log scenarios (four and five) where data only is used as information, there would be no problems and neither would any specific safety requirements be necessary to treat the data after it has left the machine. Hence these user scenarios would be possible.

COTS device to Machine

In this part of the analysis the communication start with the COTS device as sender. As described in previous section there are severe problems from a safety perspective when it comes to COTS devices. It will not be possible to trust that the error rates of hardware are low enough for any SIL.

Safety communication protocols exists as described in section 4.4.3 that can be used, but is it possible to protect the machine from problems at the COTS device? Monitoring connectivity, analyzing time stamps of data and using common safety techniques such as defensive programming as recommended by IEC 61508 described in section 4.1.2 is a tempting idea for its simplicity and may protect the system from some bad input.

Lets consider a case with the safety certified I/O node. Defensive programming, is used to define which values that are allowed and refuses parameter values that are out of bounds, acting like a sort of firewall. Using state machine procedures that only allows certain transactions and input values could be another type of defensive programming. At the same time the connectivity to the wireless device is monitored by the machine and appropriate actions can be taken if connection is lost. Would it with these techniques be safe to get input from a non-safe source such as our mobile COTS device in this case?

Consider the random hardware fault *"uninteded setting of a value"*, meaning that any parameter value can be set and sent to the machine without any human interaction.

This may trigger an unintended state transaction, but all the defined possible state transactions should be safe and hence the procedure will still follow a specified sequence. If this would be the case the input source would not matter.

Unfortunately, there can be situations where the machine is running and a transaction or value is, and should be, allowed but under the specific situation making that move could put humans or properties in danger. Even if all safety measurements have been taken, there will still be values that have to be allowed. The allowed parameter interval can be tightly set but there can be sequences or locks on the same errors since the failure modes and rates are unknown. Such system errors cannot be argued for in a certification process and can hence not be allow as input. If an input is desired it would have to be obtained from another certified source to be able to preserve the safety.

This type of error could also be caused by a fatigued human with sloppy hands. Humans are unreliable in that sense. IEC 61508 however, only prescribe that human errors should be considered in the design, while systematic failures should be reduced to meet the probability of failure required by the intended SIL. When two modules communicate like this to perform safety functional operations they have to be considered as a cohesive system. The system itself should be designed not to accept unintended actions generated within the technical equipment.

Neither will the value be detected as erroneous by a safety protocol at the destination node. A, for the situation, dangerous value can hence be set, sent and propagated into the machine that makes an unexpected and unintended maneuver. Since the failure modes and rates are unknown it cannot be argue for in safety certification process.

A technique that could be used to enhance safety would be to use a sort of dead mans grip on the multi touch screen. The user have to hold one finger on a certain area to allow touch inputs from the safety parameter area on the screen. This will decrease the risk of for human errors, or to set the value in the pocket, on the COTS device side but it is only a software implementation. It will not make the hardware any safer.

Hence, scenario two where the operator are to control the machine from the COTS device will be disqualified for safety purposes.

As seen in section 4.2 Laarouchi et al. describes some promising experiments using a COTS

laptop communicating with the avionic system on an Airbus [24]. The research team used Totel's model to upgrade criticality level. If they succeed with a safety certification for that type of system it would also be possible to do something similar in this type of system.

Fail Safe Mode
To enable updates of soft or firmware, an incoming parameter that sends the machine to a specific fail safe state has to be allowed. The channel into the system has to be safely controlled. If information comes from a not trustworthy source it should be discarded in a gateway firewall. If the packet comes from an authorized source that is specifically marked with the *"software update"* marker, it should instead start the procedure sending the machine to a fail safe state where new software can be uploaded and installed. As described earlier in this section this can be done unintended from a COTS device, but starting a safety procedure should not endanger the safety of the system.

Since the machine in this state, should not fail or make any actions it should be possible to update firmware or software. It has to be controlled though that the new software packet is the intended one and correctly received and installed. The new software should, of course, also have gone through a safety process as described in IEC 61508 -3 to be certified before uploaded. The new software also have to be tested and verified in the system according to the standard. When uploading the new software appropriate checksums shall be used in order to ensure that correct packets was uploaded. This also comes with security issues. It must be ensured the the new software comes from a trustworthy source and is not manipulated by any evil source on the way.

5.2.4 Summary

As seen in the summary in table 5, there are four scenarios that are feasible. None of them advocates real time interaction with the safety critical system. As described in section 5.2, enabling logging of machine parameters requires extracting data from the machine during runtime and would hence also enable scenarios one and three. Live data should however not be used for real time interaction with the system since the data can not be trusted, but the information could be used as a complementary runtime warning system, alerting the user when machine health is getting bad. This would increase the safety of the system, but it would not be valid as an argument in a certification process. Scenario three can be helpful for troubleshooting, and as long as repair is not done in real time while the machine is running it would be safe to use it.

Scenario	Feasible	Comment
1. Live Data Presentation	no	Data can not be trusted
2. Machine Control	no	COTS not safe to use as input
3. Augmented Repair	yes	Data may only be used as information
4. Machine Health Logs	yes	Not functional safety, only information
5. Production Logs	yes	Not functional safety, only information
6. Soft/Firm ware Update	yes	Should be done in fail safe mode

Table 5: Safety Strategy for COTS device interaction with safety critical system

5.3 Recommendations

This section presents recommendations extracted form the analysis in section 5.2 of the user scenarios presented in section 3.2. It describes which user scenarios that are feasible and which techniques that would be necessary to preserve the safety properties. The recommendations aims to be used as a guide for future system development. Moreover, a conceptual system is proposed that enables the feasible scenarios in the recommendations.

5.3.1 User scenarios

1. **The Operator presented with live machine data:**
   The scenario shall not be implemented using existing COTS devices mainly due to the problems regarding the hardware in such devices. To realize this user scenarios a certified mobile platform have to be used. Wifi or BT are suitable bases to build the safety communication protocol on, depending on desired ranges. BT is to prefer if time predictability is of concern.

2. **The Operator remotely controlling the machine:**
   This scenario face the same problem with the hardware as the first and hence the recommendations are the same.

3. **The Service Technician using the mobile device as help in maintenance:**
   This case only consider extracting information of the system and will hence be feasible from a safety perspective. To not ruin safety, it have to be ensured that only outgoing traffic from the safety node is possible. As for the two first scenarios the communication will be chosen depending on desired ranges. Time predictability of the protocol is no main concern. Hence both BT and Wifi would be suitable.

4. **The Service Technician reading machine logs from the office:**
   As in the maintenance scenario data will only be used as information and will hence have the same safety precautions. If data are to be sent to the office, Internet connectivity is required. For a mobile machines working in the field, 3G or GPRS communication is to recommend, depending on the data rates required.

5. **The Production Manager:**
   This case also enables logging and can be treated as the service technician.

6. **The Software Manager:**
   Software could be uploaded from a failsafe mode. The main considerations are related to ensuring that the uploaded data is correct. A properly chosen checksum that can detect errors for the SIL required probability is necessary. Confirmation of correctly performed uploads should be provided. Moreover, security issues are strongly recommended to consider if this scenario is to be enabled.

5.4 System proposal

The system proposed is based on the recommendations above. It can be implemented with currently available COTS products (i.e non safety classified COTS HW)

The safety critical functions in the machine have to be isolated from regular jobs. As seen in the recommendations above only outgoing data can be allowed from that module. The interface between safe and regular functions could be placed right between the machine and the COTS device when only outgoing data is allowed. The DE presented in section 4.6 however presents a nice interface that also could be used for other user interfaces in the machine and is hence proposed as a middle ware communication platform. An other benefit of placing the safety interface within the machine is that the interface can be controlled by the engineers developing the specific machine system.

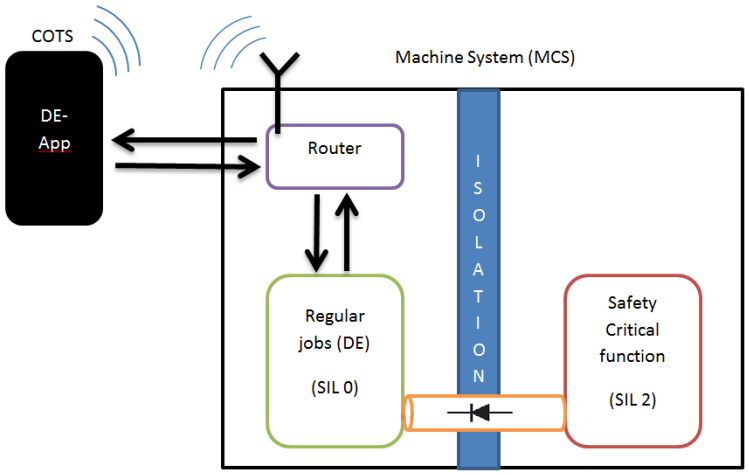

Figure 10: MCS system design that isolates the safety functions , preserving safety and enabling wireless communication to COTS device

The proposed system as depicted in figure 10 hence present an isolation between the safety critical functions and the DE regular job. The isolation can be performed by techniques chosen from IEC 61508. If virtualization is used a safe hypervisor have to be chosen. The communication channel can be done by using regular communication protocols, but if low latencies and high throughput is considered a shared memory area as presented in section 4.4.2 can be used. Concerning safety, the most important part of the channel is that diod properties are ensured, meaning that the information only can flow out from the safety critical functions.

A router as a standalone module or as an integrated part of another machine node, or a similar device is needed to enable the wireless connection to the COTS device. It usually also has security features that would block unauthorized access to the system.

5.5 Prototype System

This section presents a design and an implementation enabling the logging scenarios. In addition, an interface to develop the augmented reality application will also be presented. The safety part is not included due to safety reasons. As seen in section 4.1.2 developing safety software is a rigorous process that should not be done by one person alone. The regular jobs on the other hand do not consider safety and can be handled by any engineer. Hence, the system will only be prepared for safety interaction.

For logging scenarios a web server is considered as receiving COTS device. Such a server would hold the same type of unsafe properties as described for COTS devices in section 5.2, since they are not designed for safety. For the augmented troubleshooting scenario a smartphone or tablet is considered. The machine node writes data to DE and the COTS device reads it from the DE. In that way a channel from the safe machine node to the COTS device is created.

A database to store the logs will be set up at the receiving device.

To make analysis of logged data easier time stamps could be useful. Time stamps can also be useful for monitoring heartbeats and in a way the connectivity for the troubleshooting. A watchdog timer used to check the age of the latest heartbeat is used to indicate if the age of the date is satisfiable and alert if not.

An application could be implemented to react on the signals, emitted by DE when new values arrives, and log all new data. Another option would be to read data periodically. In this case the sample period will have to be chosen depending on the parameters and the application. This issue will have to be decided depending on the specific application, but both options are available in the SAP. Both options are possible using the DE.

5.5.1 Design

Figure 11 shows an overview of the components used in the design. The Machine node serves as the machine part connected by wire to the wireless router. These parts are considered located at the machine. The mobile COTS device connects wireless to the router and can hence be anywhere within the connectivity range of the router. The machine node could be any machine node on which DE can run that has an Ethernet port. The router handles security issues just like in a home appliance network and will require authentication for establishing a connection. Hence, only users that know the password will be able to log in and communicate with the DE server running on the machine node. In this prototype the router will be connected to Internet by wire. By replacing it with a 3G router the services could be installed in a machine and communicate to the Internet over 3G, to reach the company server in the cloud.

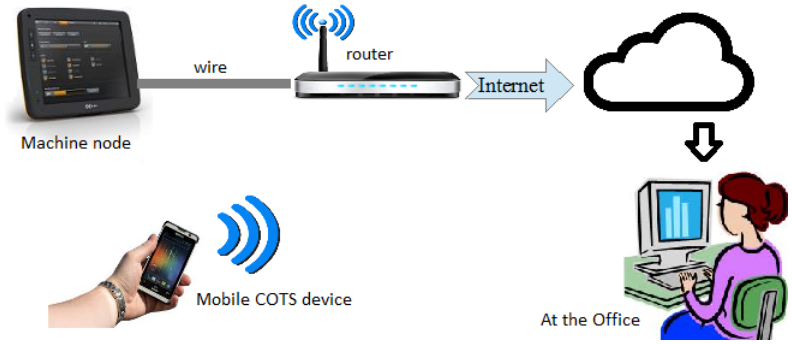

Figure 11: CC Pilot XS wired connection to router, and backend connection to machine. Mobile COTS device connected wireless to router, Router have connection to Internet.

The safety functions in the machine node, as shown in figure 10, should in a real scenario provide data to the regular side within the machine node through the safe channel. The function reading the data from the channel would then write the data to DE. The COTS device application reads data from DE periodically and sends it to a network database. In this prototype however all inputs to the DE will be retrieved from a touch screen connected to the machine node. The database can be deployed in the machine node localhost or remote in the company cloud. The prototype will enable both. Data can be sent to the cloud directly from the machine node but by implementing a user interface for the COTS device application it is also demonstrated that the implementation could be used for the augmented maintenance scenario. Reads are performed periodically and will present a new instance/row in the database together with the time stamp when data was read. The database can be accessed from the office using SQL queries. In this prototype the values are recorded and presented in the debug window and stored on the web using a free tier LAMP-server (Linux, Apache, MySQL, PhP) at Amazon EC2 cloud. In this way the logs are presented on the web and accessible from the COTS device using a web browser (a specific application could also be implemented for this purpose). Live data presentation in a simple GUI for the COTS device will be used to present information about the machine. For such a scenario it is interesting to monitor the heartbeats of the sender to know that the connectivity to DE is fine. For this purpose a heartbeat monitor is added.

5.5.2 Prototype Implementation

This section provides a description of the components and modules in hardware and software that were used for implementation of the design presented in previous section.

Hardware setup
Since the, by maximatecc AB, intended safety I/O node was still under development a similar equipment was used instead, namely the CC Pilot XS. CC Pilot XS is a main controller with a several different communication ports (Ethernet, CAN, J1939) and can serve as a communication gateway in a distributed system. It also has numerous I/O ports. CC Pilot XS also comes with a touchscreen attached, which can display graphics and sense touch inputs. (With this setup the safe I/O node could also be placed as a part of a distributed machine control system connected to the main controller.)

Since neither the CC Pilot XS, nor the intended I/O node have any WiFi interface some extra devices is needed to enable wireless communication. The CC Pilot XS has Ethernet interface and can hence be connected to a WiFi router to which also the COTS device can be connected wireless. When connected to the same router and network, it will be possible to subscribe to signals at the DE running on the CC Pilot XS with the security features that the router offers. The CC Pilot XS will serve as producer and the COTS device as consumer.

Google Nexus 7 version 2 running Android 5.1.1 (alias: Lollipop) will serve as COTS device.

The CC Pilot XS and the COTS device will connect to a Netgear 100 Mbps Wireless Firewall Router WGT624v2 to enable the wired to wireless communication between the nodes. The router is connected to Internet via an Ethernet cable.

Software development tools
Several different platforms are used to enable the logging functionality. The company specific DE binaries and development suite, the tool chains for the Android application running on the COTS device and the network database interface. All these parts needs to be ported in order to enable the communication with the DE server running on the CC Pilot XS. Qt cross-platform development suite is used to build both the sending and the receiving application and appropriate tool chains to compile the code for the different platforms. Using a cross-platform also makes it possible reuse code. This makes the coding more efficient and from a safety perspective it would reduce the lines of code that needs to be analyzed.

Qt
Qt is a cross-platform development suite. The environment provide tools for developing anything from simple console applications to widgets for mobile applications. It is based on C++ but also uses javascript and the Qt Meta Language (QML). A specific feature in Qt is signals and slots. Theses are very useful for intercommunication between different software layers and objects such as the GUI built in QML and C++ classes in the back end. Signals and slots are used in this prototype to engine the presentation of data in the GUI and to enable sending input data to the DE.

To understand signals and slots you can think of it as if you want to build a physical wired network. There is no way that a signal can be sent to another client in the network if the cable is not plugged in. When plugged in it is possible to emit the signal to the slot on the other client. By emitting a signal, which also can hold parameters, the slot function will be triggered. Signals can be used to trigger sporadic functions or timing behaviors controlled by timers. When declaring a C++ class in Qt signals and slots are declared as public or private. The programmer should choose the appropriate for the specific application. Signals and slots are connected in the `QtObject` class, using the following template line.

```
connect(*sender_object, SIGNAL(signal), *receiver_object, SLOT(function)
```

To present C++ class variables in a GUI the parameters have to be bound to the QML code. To enable this the macro `Q_PROPRERTY` is used. The variables and the functions are declared in the macro and the functions are implemented as usual methods to a class.

```
Q_PROPERTY(int intValue READ intValue WRITE setintValue NOTIFY intValueChanged)
```

In this example `intValue` will be the `READ` method which reads the `intValue`, usually simply implemented by a method returning the value of the variable. The `WRITE` creates a reference to the method that sets a new value to the variable and the `NOTIFY` refers to the signal that shall be emitted when the value is changed. This signal tells the GUI to update the graphics with the new value. Properties from the back end C++ code is in this way bound to enable the power of object oriented programming presented in the GUI.

Virtual Box and LinX development Suite
To reduce the time to set up a development environment with all necessary libraries, compilers, binaries and documentation, maximatecc AB offers a preconfigured image of Linux that contains Qt and all necessary libraries to start developing applications for the CC Pilot XS and its relatives. It is prepared to run in a virtual machine and is recommended to be installed in Virtual Box, which is used to develop this prototype.

Data Engine SAP
DE, described in section 4.6, is provided with an API that presents two main files: the sapcore and the admincore.

Sapcore provides a way for clients to subscribe to signals as producers, consumers or both. Producers writes data to the DE and Consumers reads it. This means that variables are stored in or read from the DE server. When new data is written to the DE there is an observer service that can be utilized. It signals that the value has been updated to its subscribers. It is also possible to read data from DE at any time chosen by the programmer. The most important functions in the API used to handle this are:

`Connect(clientName, hostName, port)`
Connects the client with the clientName to the host on the port. A heartbeat is also created with the clientName as reference.

`Disconnect()`
Disconnect client from DE server.

`Subscribe (name, datatype, sigType, persistent)`
Subscribes to a signal with the name and datatype. The sigType input tells if the subscribed signal is PRODUCER or CONSUMER or both. Persistent storage is set to true if the value should be kept in power off situations.

`GetValue(id, &value)`
Retrieves the value of the signal with the id number.

`SetValue(id, value)`
Sets the value of the signal with the id number.

`DataChanged (int id, "type" value)`
Signal emitted by DE when data has changed. The id identifies the signal and the value parameter is the new value.

`SyncChanged (int id, int value)` Signal emitted when a heartbeat synchsignal has changed. The id identifies the signal and the value parameter is the new value.

When a client is connected to DE it is obliged to update a heartbeat periodically. Subscription of this signal as PRODUCER is done by default while establishing the connection. It is possible for other clients to subscribe to these signals as CONSUMER in the same way as to any data value signal. These heartbeat signals are under the category syncsignals. If one client receives data from another it can sometimes be good to monitor that the sender is alive and able to update this heartbeat periodically.

The Admincore, as the name implies, offers administrative features to the Data Engine. For example, signal values can be overridden and detailed information of the signals can be retrieved.

The clients communicate with DE using TCP. It makes it relatively easy to connect clients to DE and communicate signals over a network. A specially design protocol is implemented, the

PUIProtocol, to handle the communication interface to the DE.

Android application, Win/LinX
Nexus 7 used for this demonstrator runs on Andriod 5.1.1 Lollipop. To develop and build android applications for Lollipop, some tools are required. First, since Android applications are built on Java the Java Development Kit (JDK) is required. Moreover, the Android Source Development Kit (SDK) is required. The SDK Manager helps the developer to download the API for the wanted Android version. For applications running on Lollipop, API 22 was used. Android Native Development Kit (NDK) is also required. It is a toolset that allows usage of native code languages such as C or C++ (Qt applications are, as mentioned, mainly coded in C++). Apache Ant is required to build the necessary tool chains for the compiler. In this demonstrator Java JDK 8u45, Android API 22, NDK r10e and Apache Ant 1.9.4 were used (information of the latest suitable tool chains can be found at Androids website [52]).

With the tools presented above there are two basic ways to start develop Android applications that enables a DE connection. The first way is to install all necessary packets in the virtual machine. The space in the preconfigured image is however limited. Hence, it is required to extend the size of the virtual machine to create space for all necessary packages. The other option is to extract binaries of the DE libraries and link to them using an other image or OS. This was the choice for this demonstrator and done in Win 7.

MySQL
MySQL network database is used to store the external logged data. A Qt plugin is installed for the version used. For some versions of Qt, the programmer will have to build this plugin to enable the cross-core functions that Qt presents. Information on how to build the plugin can be found at Qt developers pages.

Cloud
Amazon elastic cloud free tier LAMP-server is used to upload data to the MySQL server and by a simple php web application show that data can be presented in a web browser. It is also possible to store the data on the localhost. When the data has been stored in the database it can be used for other more specific applications. This only enables the possibilities.

Software Implementation
This subsection presents the software objects needed in the design described in section 5.5.1. The most important parts for the solution is covered in more detail. The sender and the receiver applications will share some functionality and will hence reuse code. The same objects will hence exist on both the sending and the receiving side. The implementation extends an existing example, developed by DE development group, with functionality for logging data external and monitoring heartbeats.

The implementation of the prototype design using components described in section 5.5.1 is built upon the SAP DE libraries. The DE API provides functions to read and write in the DE. The DE server running on the CC Pilot XS can be accessed by using TCP/IP with correct address and port number chosen when setting up the server.

The implementation consists of the four C++ classes described below and one small helper class. The GUI, shown in figure 13 and figure 14 is implemented in QML, and driven from the main program in which all necessary objects and GUI are declared. The GUI at the sender side has a number of slidebars to show that different types of data can be sent. They can be changed by the user. The text fields are updated to show the actual values. The receiver side will present the actual values retrieved from the DE server together with the name of the parameter. The receiver also holds an animation of a moving rectangle that indicates that the client can send data to DE within an acceptable time window. If not, the animation stops. Furthermore, pushbuttons are added to control the data base connectivity.

- `DataEngineControl` is the central part of the implementation. It will control the connection and communication with DE, and utilize the `DataLog` to log data in the database.
- `DataEngineSignal` defines the signals and variables that are subscribed to the DE server.

Signals will be emitted when values are changed in order to notify the GUI to update.

- `DataEngineObserver` listens to changes in the DE and emit signals with the values when they have changed.
- `DataLog` controls the interface to the external database.

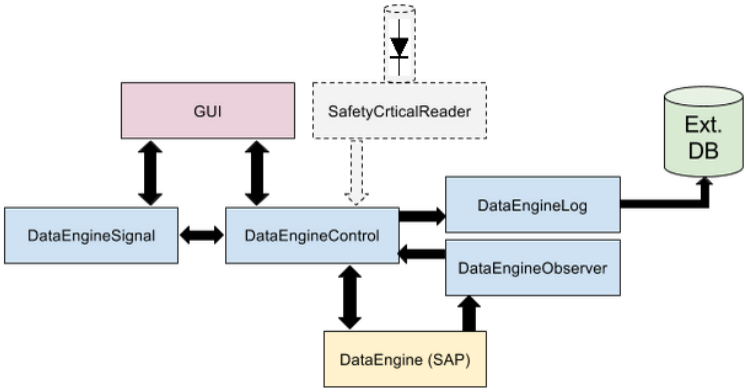

Figure 12: Objects in the implementation. The arrows shows how the information flows in the system. A placement for the SafetyCriticalReader, that is not included in the implementation, is suggested

The information flow and the internal communication between objects is depicted in figure 12. It is implemented in three layers. The object reading safety critical data from the specific channel as presented in figure 10 is excluded and the input to the DE will instead come from the sender GUI. The SAP will hence be the foundation onto which both the sender and the receiver GUIs writes and reads respectively. The SAP will in this way work as communication platform from one GUI to another and could hence be used bidirectional. The sending GUI is deployed on the CC Pilot XS and the receiver on the Nexus 7.

When the user gives new input at the sending application the `DataEgineControl` will write information to the DE. When new data is written the `DataEgineObserver` will notice it and if the receiver application is a subscriber of that data it will be updated at its GUI.

In this prototype, the touchscreen and Qt slidebars are used as the only input. When a slidebar is adjusted it will serve as new external input to the application The information flow hence starts from the sender GUI. The input parameter is bound to a `DataEngineSignal` and when the value is changed it signals and triggers the method `setValue` which emits two signals. The first one is to notify the GUI that a value has been changed and should be updated. The second signal, `sendToServer`, will trigger the method in the `DataEngineControl` to write the value in the DE. When the new value has been written in DE the `DataEngineObserver` will emit the signal that the value with the specific id has changed. The receiver application will notice it and emit the signal `dataChanged` to the `DataEngineControl` that triggers the method `valueChanged` that sets the new value in the receivers `DataEngineSignal` class. When the new value is set it will emit a signal that the value is changed and the new value will be presented at the receiver GUI.

The double arrow between GUI and `DataEngineControl` is used to be able to change the ip and port numbers to connect to the DE server.

Both the sender and the receiver will have the `DataEngineControl`, the `DataEngineObserver` and the `DataEngineLog` in common, meaning that the classes and their methods will be the same and can be used on both sides. The `DataEngineSignal` will look almost the same on both sides but the sender signals will be marked with the extension `_IN` to subscribe the signals as incoming values to the DE server. The receiver signals will be marked with `_OUT` since they should be values that are sent out from the DE server. The resulting GUIs are shown in figures 13 and 14.

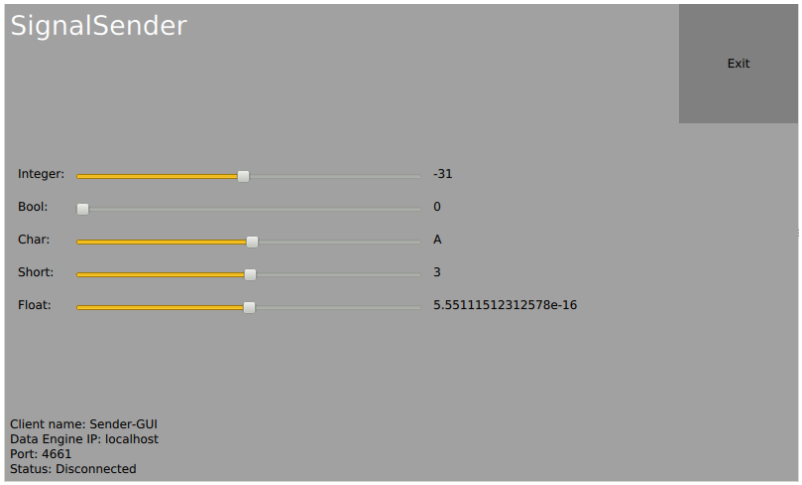

Figure 13: Sender application GUI

Figure 14: Receiver application GUI

In this application the port is chosen to 4661, which simply is a port outside the ones occupied by the well known standards such as ftp, http and MySQL ports 21, 80 and 3306 respectively. The IP address is given dynamically by the router.

main
In the main function a pointer is declared to the new `DataEngineControl` that is constructed on the heap. `DataEngineControl` constructs the `DataEngineSignals`, the `DataEngineObserver` and the `DataEngineLog`. The pointer to the `DataEngineSignal` handler is then fetched to enable the bindings to the variables for presentation in the GUI. Qts `QQmlApplicationEngine` class is used to drive the QML properties of the variables to and from the GUI.

```
engine.rootContext()->setContextProperty("dataEngineControl", dataEngineContext);

engine.rootContext()->setContextProperty("dataEngineSignal", signal);
```

DataEngineControl
This class is the core of the demonstrator. As mentioned, it constructs all the other classes and holds pointers to them. It will also hold a timer that gets values from the DE and sends it to the external DB periodically. The timeout signal of the timer will trigger the slot function `logData` that performs the operation using the `DataLog` class.

The most important members to understand the application are:

`SAP m_sap`

`DataEngineObserver* m_observer`

`DataEngineSignal* m_signalHandle`

`DataLog* m_dataLog`

`QHash<int, QString> m_signalIdLookupTable`

`QHash<QString, int> m_signalIdLookupTableNameToId`

`QHash<QString, QString> m_signalNameToPropertyName`

The most important methods are listed below with a short description.

`void InitConnection(QString host, int port, QString clientName)`
initiates the parameters used to connect to the DE server.

`void connectToServer()`
establishes the connection to the DE server.

`void SubscribeToSignals()`
subscribes to the parameters/signals defined in the `DataEngineSignal`. Parses the signal names to define if the signal is a `PRODUCER` or a `CONSUMER`

`void disconnectFromServer()`
disconnects from the DE server

`void sendToServer("type" value, QString signalName)`
sends a value to the server given the name of the variable. Id number for the signal is checked in the Hash tables

`void initDataLog(QString host, QString dbName, QString user, QString pwd)`
initiates the variables in the `DataLog` used to connect to the external MySQL database. Constructs and connects the timer that is used for periodically log data to the dataLog slot method.

`void startDataLog()`
starts the `[DataLog]` timer.

`void stopDataLog()`
stops the `[DataLog]` timer.

`void readFromTable(QString table)`
reads all values from the table in the external database. (Any method can be implemented to execute any SQL queries that the user is permitted)

`void logData()`
reads current data values from dataEngine server and writes it to the database

`void handleHeartBeatStatus(bool status)`
slot method that updates the connectivity status checked by `DataEngineObserver` (is the heartbeats updated and read in time). This will control an animation in the GUI, which visualize it to the user.

`void valueChanged (int id, "type" value)`
slot method tiggered by the `DataEngineObserver`

`void syncChanged (int id, int value)`
slot method tiggered by the `DataEngineObserver`

DataEngineObserver
This class inherits ISAPObserver which is a part of the SAP that holds methods to get callbacks to the application when new data is written to the DE server. It contain a small helper class used to monitor heartbeats.

HBStruct
This is a small helper class in which signal id together with last time stamp is stored. The HBStruct does only hold two members and the constructor. The members are `id` and `lastSyncTime`. `id` will hold the id of a heartbeat value stored in DE and `lastSyncTime` the time (QTime) of when it was last updated. HeartBeats are in data engine called synchsignals, therefore the name.

The only interesting member in the `DataEngineObserver` is:

```
QList<HBStruct*> m_heartBeats
```

The interesting methods are:

`void initHeartBeatMonitor()`
initiates the heartbeat monitor.

`void heartBeatMonitor()`
checks that heartbeats have been updated in time and emits a heartbeat status signal with a boolean parameter true if so and false if not.

`void DataChanged (int id, float value)`
emit signal with the value to trigger corresponding method in the `DataEngineControl`

`void SyncChanged (int id, int value)`
updates the heartbeat time stamp for this client and emits a signal that triggers corresponding method in the `DataEngineControl`

To notify `DataEgineControl` of changes in DE the following signals are used:

`void dataChanged (int index, "type" value)`
signals that the variable with the id index has been updated with the `value`.

`void syncChanged (int index, int value)`
signals that a heartbeat have been updated.

`void newSignal (int id, QString signalName, DataType type)`
handles the subscription of new signals to the DE.

DataLog
The DataLog class holds the interface to the external MySQL database. DB connectivity is enabled using Qt `QSqlDatabase` with a plugin driver for the MySQL connection. The data in the example is stored on localhost but can relatively be setup on a remote server. For this purpose the `QSqlDataBase` is declared as a member of the class.

```
QSqlDatabase m_sqlDB
```

The methods used to control the interface are:

`void initDB(QString host, QString dbName, QString user, QString pwd)`
initiates the variables needed to connect to the external database

`bool openDB()`
Opens the connection to the database.

`bool closeDB()`
Closes the connection to the database.

`void writeToDB(QString table, int id, int value)`
writes values in the `table` in the MySQL database

`void readFromDB(QString table)`
reads all values in `table` in the MySQL database

In order to enable the connection to the database the server it must be configured to allow remote access since this is not default in most MySQL servers. It can be enabled in the config file and optionally bound to a specific ip address. It is also proper to create a user with necessary privileges to prevent security issues like allowing root remote access. It is also important to check that there are no blocks on the MySQL TCP port 3306 in the server or any firewall in the communication channel, since this will block the attempts to access.

The connection is opened with a single command `m_sqlDB.open()` using the host name, database name, user name and password defined. The open method returns an error code and can hence be used to show the user if it was successful or not and why.

With the SQL plugin API it is possible to construct SQL queries to perform the wanted operations in the DB. For this application two types of queries are used:

```
INSERT INTO table VALUES()
SELECT * FROM table
```

The queries are created and executed by the following code snippet:

```
QSqlQuery q;
q.prepare(QString("INSERT INTO %1 (id, rpm, sampleTime) "
"VALUES (:id, :rpm, :sampleTime)").arg(tableName));
q.bindValue(":id", id);
q.bindValue(":rpm", value);
q.bindValue(":sampleTime", time);

if(!q.exec())
qDebug() << tableName << q.lastError().text();
```

DataEngineSignal
This class declares the signals that are subscribed to the DE. One of the memebers in the `DataEnginecontrol` is a pointer to this object. The purpose of having this class is to make the `DataEngineControl` more generic. Signals that goes into the DE are postfixed with `_IN` and signals coming out from DE with the postfix `_OUT`. This will be parsed by the `SubscribeToSignals` method in `DataEngineControl` to match the `CONSUMER` and `PRODUCER` properties used in SAP method `Subscribe`.

An example of how a member is defined is:

`bool m_boolValue_IN`

The methods in this class is also coupled to the `Q_PROPERTY` macro to enable updates in the GUI. Typical methods are:

`bool boolValue_IN()`

`void setboolValue_IN(bool arg)`

Signals used to notify the updates look like this:

`void sendSignal(bool value, QString signalName)`

`void boolValue_INChanged(bool arg)`

In figure 15 the database log example is shown in the web application connected to the MySQL server rented in the Amazon cloud.

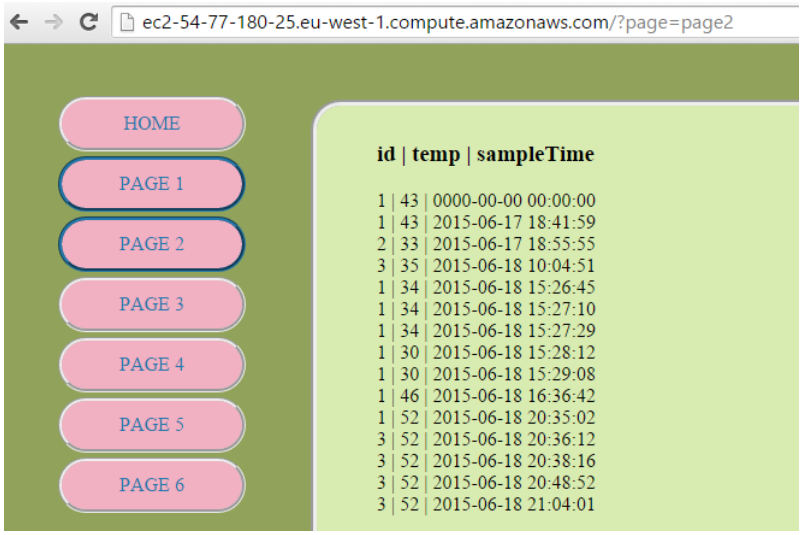

Figure 15: Web application showing contents from the MySQL database

6 Conclusions

It may be stated that safety is a conservative domain, always considering that hazards would happen, but it is restrictive for very good reasons. Lessons learned from catastrophes for the types of applications that interact with humans, environments or expensive equipment needs extra considerations. Safety systems should hence be designed to minimize the risk of harming humans and properties. When designing new systems for functional safety there is hence a rigorous process outlined to reduce the risks of hazards. This includes quantitative measures such as FMEDA reports on hardware and qualitative techniques to control that systematic failures are minimized in the process, specifications and implementation of software. As long as engineers are in control of the whole system development process it is possible to follow the industry standard to meet SIL requirements.

Safety communication protocols exists that are proven to ensure that the data sent is received and verified to be correct. Safety layers are usually a concern for higher layers in the OSI model and safety stacks can hence be created using standard lower layers.

COTS devices are not developed for safety applications. The time and cost to go through a certification process and implementing safety systems on such devices would likely make them too expensive for a normal person and the time to market release would be too long. At the pace these devices evolves and the market they are intended for, it would also consume too much time to analyze them to ensure safety. Before that job is done there would likely also be new products on the shelfs that customers want to use. Since the hardware of COTS devices most likely never will be safe there cannot be any safety guarantees for such devices. It is hence not recommended to use COTS devices for real time interaction with safety systems. In order to use COTS devices in safety applications they would hence have to be monitored by another device that holds safety properties.

If a safety node is to communicate with a non safe device, there will always be an interface between the systems. This interface has to be considered carefully in order to preserve safety. The non-safe node may not under any circumstances affect the safety node.

Since only information sharing, at this point, is recommended from the safety perspective, it might be better to let a main controller, such as CC Pilot XS, upload information through GPRS or 3G to the cloud and make an application connect to the service via Internet instead.

There exist some research on enabling communication between different integrity levels in MCS. The different levels of integrity have to be separated in both space and time to guarantee isolation in order to ensure that lower levels will not affect the higher ones. The industry is acting slow and cautious before trying new technology for safety applications. It is, of course, a good thing not to rush into new technology without validating and verifying it carefully. It is clear that sending information from a higher integrity level to a lower one does not violate the integrity of the higher level. However, upstream communication is more problematic and a topic for research.

Connecting systems of different criticality level is not an easy task and demands several considerations of the whole system, from the input source to the end device, either it is an actuator controlling machinery or if it is a display presenting an user interface. There is no simple way to allow communication from a low criticality level to a high one in a MCS if the integrity is to be preserved. Especially when the machine is supposed to act on the incoming values. Virtualization and Totel's model may presents some interesting opportunities in the future.

Virtualization by using an already certified hypervisor (Red Bend and Green Hill supplies this) can provide isolation of VMs and hence also guarantee separation of regular jobs from safety critical ones both in time and space, which is required for MCS according to the standard. This could provide a safe software platform to build the application on. In such a case incoming data to the machine from the mobile could have been made trustworthy and used as a part of the safety critical system.

When adding extra control layers such as a specific safety communication channel between VM instances in the hypervisor kernel it would also have to go through a new certification process.

To realize the prototype to run on the Nexus 7 with Android 5.1.1 Lollipop, toolchain to compile and install the application the device is necessary. This includes Java libraries, Android SDK and NDK plus Apache Ant toolchain builder. Neither of these have been developed for safety, and neither is the OS. Understood from this it would be a very long road ahead to use such COTS

device for controlling a safety critical functions.

DE enable communication between nodes but it is not designed for safety. All information coming from DE should hence be regarded as dirty from a safety perspective. If one would like to enable functional safety applications for DE, one would have to modify the code and add the safety perspective in a new development process.

Wherever there is an interface between a safe and non-safe system it will be important to be careful if the certified SIL at the safety node shall be preserved. As seen in the safety analysis in section 5.2 the case study in this thesis could be generalized to give guidelines of the challenges with such an interface.

- Safety of a system must include all parts
- Sending data from a safe system is feasible from a safety perspective.
- If the receiver of safety critical data is going to use the values for any functional safety, the values need to be trusted and hence that system also have to be a safe system.
- Outgoing data for informative purposes is always allowed.

7 Future work

The area of IoT presents many opportunities for new application that can make machines, industry, societies and home appliances more efficient. The opportunities do however come with challenges and issues regarding ethics, security and, as described in this thesis, safety.

Concerning the machinery applications in this thesis some future work that can be investigate remains.

Instead of utilizing an external database the Firebird in the DE could be used, but only for logging data locally in the machine. In this case the data can be extracted at a later point, for example, by uploading them to an external device such as a mobile phone and deploy them on the corporation server when Internet is available. This requires some extra storage space in a node at the machine. It have to be taken into account when designing the system. It should, for example, be possible to connect an external hard drive to the main controller or the router used in the demonstrator.

The implementation presented in section 5.5.2 do not include safety functions nor the specific channel described in section 5.4. This would of course be an object for inclusion in a safety process when designing and implementing a real system.

Many things that can be considered in order to get closer to a total inclusion of a COTS device in a safety system also exists. More research and analysis of COTS devices would be necessary.

Creating a gray channel from the machine node directly to the touchscreen on the COTS device would be an interesting solution that could minimize the influence of the COTS hardware. Then, only evidence for the touch screens and the video memory would be necessary. This rather complex solution will however bind it to some specific platforms. The same type of channel could be investigated for sending data from the COTS device to the machine.

Redundant inputs and output on the COTS device could be investigated. For example using the multi touch system, so that one have to hold one finger on a specific area in order to be able to change values. This would present a safety function reducing risks that the user sets values unintended. With redundant output from the machine, by for example producing two pictures of the same output using diverse programming, data correctness could be validated by the user. This two figures could be represented side by side on the COTS device, and the user could compare the figures and hence decide whether the output is trustworthy.

This type of redundant and safe inputs and outputs would have to be investigated aligned to the standard. Human errors are in this way considered in the system, but the faith is also putted on the human making decisions about the reliability.

It could also be of interest to analyze response times to see that the quality of the user experience will be preserved. Interference on the channel that could cause delays or retransmissions could be sources to increased response time.

There are many cloud services. The connection have been enabled by the logging scenarios. It could be investigated which ones that are useful. With big data it could for example be interesting to employ data analysis services on a whole fleet of machines or specific nodes.

If we want to log data for the production manager or for maintenance purposes it could be interesting to send a request for logs to the machine. In such a case it could be possible to stream data out to some storage where it later could be delivered to the corporation cloud.

Bonjour application where the machine presents which data that is available to read and the user can choose from a list which he wants to listen to would present another, more flexible, user interface.

Finally, the firmware and/or software update in fail-safe mode design and implementation was not covered in detail. This scenario need more attention and analysis to be enabled.

8 Acknowledgments

First I would like to acknowledge the support from my supervisors. Many thanks to Markus Wallmyr at maximatecc AB and Adnan Causevic from MDH.

Moreover there are two employees at the company that has meant more than others during the process. Christian Strytcz for the safety expertise input, and Jesper Melin for the help during the software implementation.

References

[1] I. E. Commission *et al.*, "Functional safety of electrical/electronic/programmable electronic safety related systems," *IEC 61508*, 2000.

[2] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, pp. 1–17, 2014.

[5] O. Vermesan and P. Friess, *Internet of Things-From Research and Innovation to Market Deployment*.

[6] J. C. Knight, "Safety critical systems: challenges and directions," in *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on.* IEEE, 2002, pp. 547–550.

[7] J.-L. Lions *et al.*, "Ariane 5 flight 501 failure," *Technical report, European Space Agency, July 1996. Available at https://www.ima.umn.edu/ arnold/disasters/ariane5rep.html (November 2015)*, 1996.

[8] N. G. Leveson and C. S. Turner, "An investigation of the therac-25 accidents," *Computer*, vol. 26, no. 7, pp. 18–41, 1993.

[9] M. Blair, S. Obenski, and P. Bridickas, "Patriot missile defense: Software problem led to system failure at dhahran," saudi arabia. Technical report, US Government Accountability Office (GAO), Tech. Rep., 1992.

[10] A. Saini and A. Marwah, "Amateurs hack system professionals hack cars," *International Journal of Research*, vol. 1, no. 10, pp. 509–518, 2014.

[11] A. Wright, "Hacking cars," *Communications of the ACM*, vol. 54, no. 11, pp. 18–19, 2011.

[12] S. Khandelwal. (2014, accessed mars 5, 2015) South korean nuclear power plant hacked @ONLINE. [Online]. Available: http://thehackernews.com/2014/12/Korea-nuclear-power-plant-hacked.html

[13] C. Kaufman, R. Perlman, and M. Speciner, *Network security: private communication in a public world*. Prentice Hall Press, 2002.

[14] B. A. Forouzan, *Cryptography & Network Security*. McGraw-Hill, Inc., 2007.

[15] A. Avizienis *et al.*, "The n-version approach to fault-tolerant software," *IEEE Trans. Software Eng.*, vol. 11, no. 12, pp. 1491–1501, 1985.

[16] A. Burns and R. Davis, "Mixed criticality systems-a review," *Department of Computer Science, University of York, Tech. Rep*, 2013.

[17] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*. IEEE, 2010, pp. 13–22.

[18] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *Computers, IEEE Transactions on*, vol. 61, no. 8, pp. 1140–1152, 2012.

[19] P. Ekberg and W. Yi, "Outstanding paper award: Bounding and shaping the demand of mixed-criticality sporadic tasks," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*. IEEE, 2012, pp. 135–144.

[20] R. Pellizzoni, P. Meredith, M.-Y. Nam, M. Sun, M. Caccamo, and L. Sha, "Handling mixed-criticality in soc-based real-time embedded systems," in *Proceedings of the seventh ACM international conference on Embedded software*. ACM, 2009, pp. 235–244.

[21] E. Totel, J.-P. Blanquart, Y. Deswarte, and D. Powell, "Supporting multiple levels of criticality," in *Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on*. IEEE, 1998, pp. 70–79.

[22] Y. Laarouchi, Y. Deswarte, D. Powell, and J. Arlat, "Safety and security architectures for avionics," in *DCSOFT*, 2008, pp. 46–52.

[23] Y. Laarouchi, Y. Deswarte, D. Powell, J. Arlat, and E. De Nadai, "Ensuring safety and security for avionics: a case study," *DAta Systems In Aerospace (DASIA 2009)*, pp. 26–29, 2009.

[24] Y. Laarouchi, Y. Deswarte, D. Powell, J. Arlat, and E. de Nadai, "Connecting commercial computers to avionics systems," in *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*. IEEE, 2009, pp. 6–D.

[25] A. Wasicek, "The across integrity model," in *IAENG Transactions on Engineering Technologies*. Springer, 2014, pp. 333–348.

[26] R. P. Goldberg, "Survey of virtual machine research," *Computer*, vol. 7, no. 6, pp. 34–45, 1974.

[27] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[28] D. Reinhardt and G. Morgan, "An embedded hypervisor for safety-relevant automotive e/e-systems," in *Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on*. IEEE, 2014, pp. 189–198.

[29] A. Landau, M. Ben-Yehuda, and A. Gordon, "Splitx: Split guest/hypervisor execution on multi-core." in *WIOV*, 2011.

[30] K. Gudeth, M. Pirretti, K. Hoeper, and R. Buskey, "Delivering secure applications on commercial mobile devices: the case for bare metal hypervisors," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 33–38.

[31] G. Heiser and B. Leslie, "The okl4 microvisor: Convergence point of microkernels and hypervisors," in *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*. ACM, 2010, pp. 19–24.

[32] R. Mijat and A. Nightingale, "Virtualization is coming to a platform near you," *ARM White Paper*, 2011.

[33] G. Heiser, "Virtualization for embedded systems," *Open Kernel Labs Technology White Paper*, 2007.

[34] K. Barr, P. Bungale, S. Deasy, V. Gyuris, P. Hung, C. Newell, H. Tuch, and B. Zoppis, "The vmware mobile virtualization platform: is that a hypervisor in your pocket?" *ACM SIGOPS Operating Systems Review*, vol. 44, no. 4, pp. 124–135, 2010.

[35] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh, "Cells: a virtual mobile smartphone architecture," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 173–187.

[36] D. Jaramillo, B. Furht, and A. Agarwal, *Virtualization techniques for mobile systems*. Springer, 2014.

[37] A. S. Tanenbaum, "Computer networks, 4-th edition," *ed: Prentice Hall*, 2003.

[38] E. Ferro and F. Potorti, "Bluetooth and wi-fi wireless protocols: a survey and a comparison," *Wireless Communications, IEEE*, vol. 12, no. 1, pp. 12–26, 2005.

[39] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*. IEEE, 2007, pp. 46–51.

[40] I. E. Consortium *et al.*, "Universal mobile telecommunications system (umts) protocols and protocol testing," 2004.

[41] J. Wang, "Survey of state-of-the-art in inter-vm communication mechanisms," *Research Proficiency Report*, 2009.

[42] F. Diakhaté, M. Perache, R. Namyst, and H. Jourdren, "Efficient shared memory message passing for inter-vm communications," in *Euro-Par 2008 Workshops-Parallel Processing*. Springer, 2009, pp. 53–62.

[43] J. Boercsoek, *Introduction in safety bus systems*. Hima, Bruehl, 2005.

[44] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[45] P. K. PENDLI, M. SCHWARZ, H.-D. WACKER, and J. BOERCSOEK, "Mathematical derivations for safety related systems with wireless communication," in *Proceedings of the 5th International Conference on Design and Product Development (ICDPD '14)*. NAUU, 2014, pp. 23–30.

[46] P. K. Pendli, M. Schwarz, H. D. Wacker, and J. Boercsoek, "Bluetooth for safety systems," in *Irish Signals and Systems Conference (ISSC)*, 2011.

[47] P. Pendli, M. Schwarz, H. Wacker, and J. Boercsoek, "Wireless communication modeling for safety related systems," *Naun International Journal of Circuits Systems and Signal Processing*, vol. 8, pp. 330–336, 2014.

[48] H. Jitsukawa and T. Maruyama, "Method for error detection and correction by majority voting," Apr. 16 1986, eP Patent App. EP19,850,109,179. [Online]. Available: http://www.google.com/patents/EP0177690A2?cl=en

[49] M. Khairnar, D. Vaishali, and D. S. Pradhan, "V2v communication survey wireless technology," *arXiv preprint arXiv:1403.3993*, 2014.

[50] I. Bate, P. Conmy, T. Kelly, and J. McDermid, "Use of modern processors in safety-critical applications," *The Computer Journal*, vol. 44, no. 6, pp. 531–543, 2001.

[51] S. Mohan, M. Caccamo, L. Sha, and R. K. L. Martin, "Using multicore architectures in cyber-physical systems," 2011.

[52] (2015, Jun.) Android developers site. [Online]. Available: http://developer.android.com/sdk/index.html