Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Information-driven proactive maintenance systems

SCIENTIFIC STUDENTS' ASSOCIATIONS CONFERENCE PAPER

Author
Attila Ernő Frankó

Supervisor
István Moldován

October 27, 2017

Contents

[1] This section is mostly based on [16].

[2]This step of design was a joint work with Csaba Hegedűs. This part from another aspect and a full project proposal can be found in his work: [11]

Kivonat

A negyedik ipari forradalom küszöbén állunk, ami várhatóan gyökeres változásokat hoz a gazdaság és az ipar számos ágazatában. Az úgynevezett Ipar 4.0. keretében – többek között – megjelennek az „okos gyárak”, ahol a teljes termelési folyamatot kiberfizikai rendszerek (CPS) figyelik meg. Ennek érdekében, a különböző eszközöket és berendezéseket össze kell kötni, melyek így egy speciális hálózatot fognak alkotni, amit ipari célú Internet of Things-nek (IIoT) nevezünk. Azok a kiberfizikai rendszerek, melyek ezen IIoT szerves részét alkotják, képesek lesznek elosztott döntéshozást és önoptimalizáló folyamatokat megvalósítani, ezzel megalkotva egy önállóbb, alkalmazkodóbb gyártásmechanizmust.

Ebben az új korszakban, számos paradigma meg fog változni, többek között a karbantartás területén is. Napjainkban a karbantartás az ipar minden résztvevője számára terhet jelent, a magas költségek és az elvesztegetett idő miatt. Az Ipar 4.0-ban a karbantartási folyamatok információvezéreltté válnak, alapjait az előrejelzések adják, így képesek leszünk úgy ütemezni az egyesi feladatokat, hogy elkerüljük a váratlan, nem kívánatos leállásokat. Ezen megközelítés legfőbb előnyei, hogy megóvja eszközeink állapotát illetve számos karbantartásra szánt időt és költséget takarít meg.

Jelen dokumentumban összefoglalom az új karbantartási szemlélet alapjait továbbá azokat az algoritmusokat, melyek a fejlett diagnosztika és prognosztika gerincét képzik. Továbbá bemutatom, hogy milyen kihívásokkal kell szembenéznünk egy okos karbantartó és felügyelő rendszer megvalósítása során. Fő célom pedig egy olyan CPS prototípusának a megalkotása, mely a hozzáillesztett szenzorok adatait gyűjti és a lényeges információkat továbbítja a felhőbe – ahol feldolgozásra és elemzésre kerülnek, így a későbbiekben felhasználhatóvá válnak. Munkám során kitérek arra a kérdésre, hogy miként modellezhető egy eszköz állapotának romlása: milyen adatokat szükséges gyűjteni, hogy megalkothassunk egy modellt, illetve milyen kommunikációs és rendelkezésre állási kritériumokat kell teljesítenie egy elkészült rendszernek. Ezeket a kérdéseket egy valós problémán, vasúti váltók proaktív karbantartásán keresztül fogom feltárni.

Abstract

We are on the verge of the fourth industrial revolution, which is predicted to fundamentally change many sectors of economy and industry. In this so-called Industry 4.0 – among others – "smart factories" will be introduced, where whole production processes will be monitored by cyber-physical systems (CPS) continuously. To this end, various equipment and devices will be connected together to become a special Internet of Things for industrial purposes (IIoT). These cyber-physical systems within the IIoT paradigm strives for decentralized decision making and self-optimized processes to make a more autonomous and adaptive manufacturing.

In this new era, many paradigms will change, and within that maintenance as well. Nowadays maintenance is a burden for everyone in the industry, due to its high cost and time wasted. In Industry 4.0, maintenance processes will be information-driven and forecast-based. This way, we shall be able to schedule these types of tasks well in advance to avoid unexpected, unnecessary breakdowns. This approach should allow to keep the equipment in good condition and reduce time and effort spent on its repair.

In this paper, I present the basics of these new maintenance approaches, and the algorithms that can be used for advanced diagnostics and prognostics. Moreover, I show what sort of challenges we face with during an implementation of a smart maintenance and management system. My main goal is to implement a simple prototype of a CPS that can gather data with its own sensors and transmit the relevant information to the cloud – where it will be processed and analyzed for further use. This work then discusses how the degradation of an equipment can be modeled: what kind of data needs to be collected to establish a model and what kind of communication and availability requirements a such system should meet. These questions will be detailed through a proactive maintenance use case for railway switches.

Introduction

The convergence of information and communications technology is about to make a lot of changes in economy, industry and our everyday life. Some changes have already made due to the emerging of new paradigms and applications that are mostly based on interoperability of various systems and related to Internet of Things. These applications like intelligent transportation systems (ITS), smart grid, smart home are getting more and more impact on our life and similar technologies will transform the sectors of industry too.

The process of changes in production is the so-called fourth industrial revolution or Industry 4.0. It affects many areas from service and business models throughout industry value chain up to the operation of factories and manufacturing. Maintenance is an integral part of production chain but it's often a burden on companies because of unforeseen extra costs due to the failures of equipment. Furthermore, failing of equipment means not only that it has to be repaired, but in the meantime it doesn't participate in production which causes decreasing proceeds.

Nowadays *Reactive* and *Corrective* approaches are still the current way of performing maintenance tasks despite their disadvantages. Reactive maintenance is performed when a piece of equipment has already failed so it's a run-to-failure method therefore the time – and money – spent on its repair is large and it's not guaranteed that the failure can be fixed. Corrective maintenance uses a different manner: When the machine is still working but not properly then maintenance will be planned to restore equipment to an operational condition. As it's seen both of them require a lot of effort spent when equipment fails however one of Industry 4.0's goals – and our goal too – to realize a smart maintenance system that reduce time and effort spent on repairing equipment by keeping them in good condition. Such a system can transform maintenance tasks from difficult, expensive and unwanted liability to an effective and beneficial part of production.

To achieve it, all important parts of a device that have impact on operation of equipment will be monitored by sensors. The data gathered by sensors will be forwarded to processing units which analyze the collected information and build a degradation model up to estimate the equipment's remaining useful life. Based on this information maintenance tasks can be scheduled before the equipment fails thus the date of repair can be planned in order to minimize its costs and duration. Moreover, processing units not only forecast malfunctions but also investigate root causes when an unexpected failure occurs to improve their models of equipment and to ease performing tasks.

In this paper I will introduce the fundamentals of condition based maintenance included the commonly used algorithms and their implementations from a quite new point of view. The goal of

my work is to design and implement such a system the will be used for maintenance of railroad switches. I will detail what circumstances affected certain designing phases meanwhile I show how these methods would be used as standardized steps in further development. In the end of this paper I will present and verify the finished device by simulating an operation of railroad switch.

Chapter 1

Basics of a smart maintenance system

In the following sections I'll detail proactive maintenance and its related technologies, approaches and methods in general. Nevertheless, I will frequently note that how certain routines or models can fit into our use case that is to say how we can adapt the parts of proactive maintenance. Therefore, first of all railroad switches will be introduced shortly and the expectations about smart maintenance of these switches will be discussed.

1.1 About maintenance of railroad switches

In the most cases when we say *railroad switch* in vulgar tongue, we mean *railway junction*, because switch is only a part of a junction however; we are talking about switches now. The switch consists of the pair of linked tapering rails, known as points (or point blades) lying between the diverging outer rails (the stock rails) as it's shown in Figure 1.2 and 1.1. These points can be moved laterally into one of two positions to direct a train coming from the point blades toward the straight path or the diverging path. A train has two possible type of direction to move through the switch: facing-point movement (from narrow end toward the points) and trailing-point movement (from either of the converging ends toward narrow end) [5][14].

One of the most common cause of failure – besides natural wearing out that is mostly influenced by temperature and physical impacts that cause dilatation of rails – is splitting, when the train during a trailing-point movement and the points are set in the wrong position, the train's wheels will force the points into the correct position. In certain cases, splitting doesn't affect the condition of the equipment but in most cases – especially if the switch was locked – splitting can damage the device. After splitting the junction has to be overseen, because if the points stay at the wrong position, the following train that does a facing-point movement could potentially go off the rails [14].[1] It's worth to note that, each switch has a facing point lock that fixes a set of points in position. Officially, trains are only allowed to pass through locked switches, but between scheduled trains the lock is open, because an unexpected splitting can destroy the lock [18].

[1] This is what exactly happened at Szajol in 1994

Figure 1.1. *A junction's facing-point with blades pointing to left – the straight way – at Budapest-Soroksári út railway station*

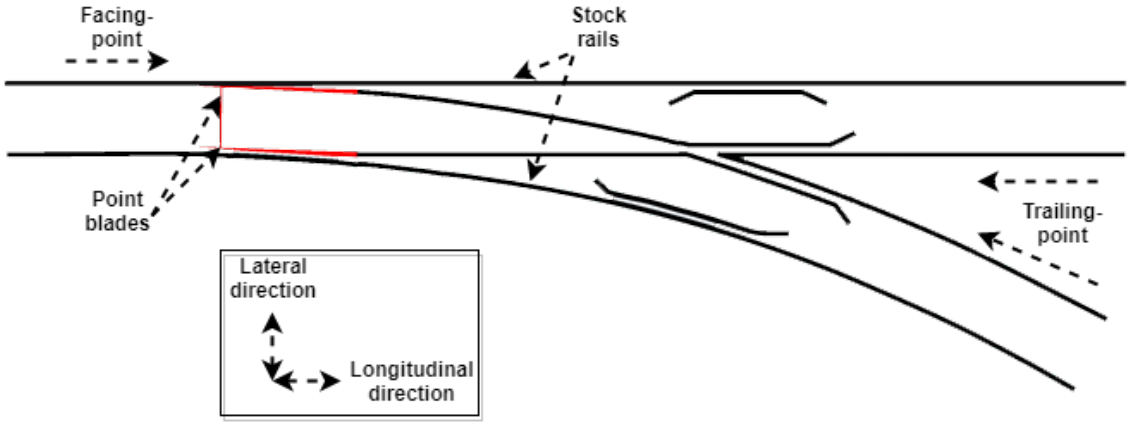

Figure 1.2. *A schematic figure of a railroad switch included directions of displacement*

After this short description the question is: How can proactive maintenance improve the management of switches? To illustrate the possibilities, it's enough to look at a simple problem. A real issue was raised by our industrial partner: they can't effectively schedule the periodic maintenance of switches, because of large distances – in their country. For example: when two switches get out of order – or it's necessary to oversee one of them – at a great distance from each other and the mechanics, repairmen or experts have to reach both places: it's definitely a waste of time and

in the meantime trains cannot use them until they're fixed. A condition based, scheduled maintenance would be a solution, when all of the switches that are in poor condition will be serviced in certain regions. To provide relevant information it's enough to implement a simple system at first that can *collect data by sensors*. If this system measures such quantities as lateral and longitudinal displacement of points, the behaviour (move) of points during switching can be observed statistically and their location in real time.[2] Temperature and humidity can be measured too, so slight additional data about natural degradation of the equipment will be available.

Later, such a system can be developed to provide additional information – e.g. static and dynamic forces on the rail, state of lock – that can improve maintenance tasks. More opportunities will be detailed in further chapters. In the following sections I present proactive paradigm and its parts that are the core of a smart maintenance system.

1.2 Evolution of maintenance

In those days, when maintenance wasn't treated as a science and didn't have its own literature, some people recognized that the way how they maintain their equipment is not effective. Not long after engineers and scientists came up with new ideas to develop maintenance processes but only a few ones – based on the current state of the art – were practicable. The other ones were visions rather than realistic or applicable methods. Later, new technologies were emerging and some visions came true, but they didn't bring significant changes. In the 90's, *reactive* and *corrective* methods – or as a collective noun *breakdown maintenance* – still ruled the field of maintenance[3], although new[4] paradigms like *preventive* and *predictive maintenance* had been tested. The results of tests proved that these new paradigms could be more efficient than the currently used ones.

The basic idea behind the preventive method is scheduling maintenance tasks more frequently to prevent equipment from wearing out. On the one hand this way of service works well, because the number of failures reduced, but on the other hand it's still not efficient enough due to numerous unnecessary condition surveys and other maintenance tasks [9]. The predictive paradigm offers a way more effective solution based on condition monitoring. In this case scheduling tasks rely on the real condition of the equipment, so repairing and maintenance tasks can be performed, when they're necessary. It sounds good, but there was a main problem with that: in many cases they didn't have tools to implement condition monitoring at an appropriate level to get useful information or if it was achievable it had a high cost. Surprisingly, despite their shortcomings both methods performed quite good during tests and in real environments too, but in many cases the old paradigms were more efficient.

A quite different approach emerged, when E. C. Fitch introduced the predictive maintenance based *proactive* paradigm. In this case the focus isn't on failure symptoms but on root causes [8]. As James C. Fitch wrote: "*The emphasis is on machine wellness, not machine sickness.*", so the objective of proactive maintenance is to keep equipment in good condition, thereby extending a

[2] It can sign that if the switching wasn't successful i. e. points didn't reach their end positions

[3] As nowadays

[4] Both of them was invented in the 60's & 70's but due to the lack of technology, none of them was usable

machine's operating life [10]. Following this paradigm offers low upkeep due to the longer operating life, but there is a huge disadvantage as it's shown in Figure 1.3.[5] The price of building a condition monitoring system is pretty high, but a system that can explore root causes is way more expensive. Fortunately, we have technology nowadays like cheap sensors, microcontrollers, wireless communication tools and cloud computing to reduce these costs and create effective, applicable maintenance systems.

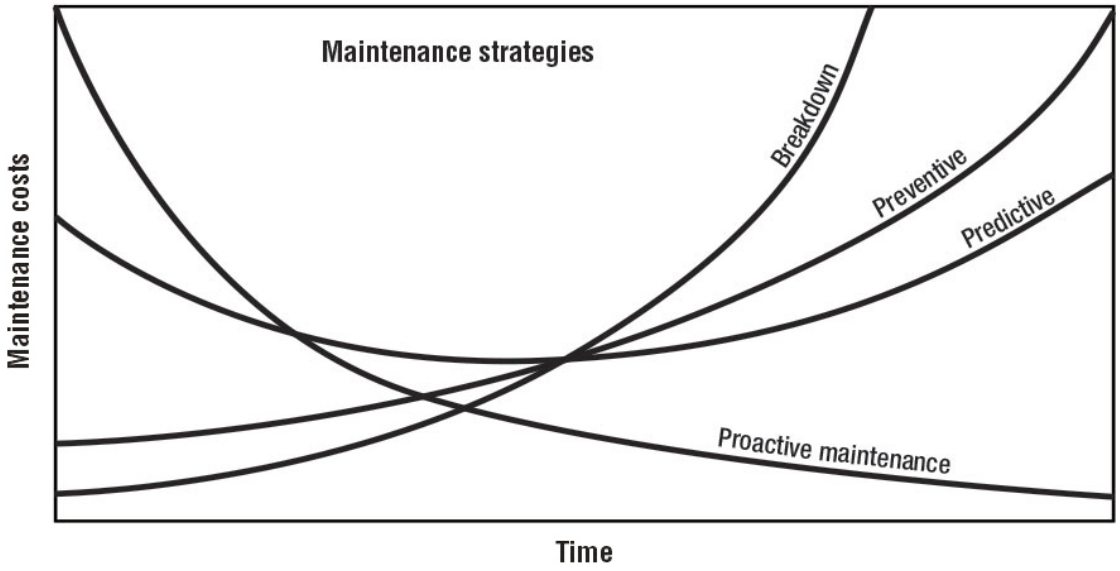

Figure 1.3. *Investment and maintenance costs at different methods [8]*

1.3 Fundamentals of diagnostics

1.3.1 Diagnostic standards

Before talking about implementation options, the two most important parts of the maintenance process must be discussed. The first one is diagnostics, which involves identifying and quantifying the damage that has occurred the other one is prognostics. Traditionally diagnostics can be described with a main process that consists of fault detection, localization and identification, but from our point of view, it's more important to discover root causes. There is no general "best solution" because implementation highly depends on use case however, the ISO-13374 (Condition Monitoring and Diagnostics of Machines) standard defines the six blocks of functionality in a condition monitoring system – as well as the general inputs and outputs of those six blocks – as it's shown in Figure 1.4 [1][13]. In this section these blocks – some better, others less – will be discussed, mainly from technical aspect however, a short general description can't be skipped.

[5]This figure is made by E. C. Fitch, so it's a bit optimistic, but it shows well the difference between maintenance paradigms.

The functional blocks can be organized around a data processing pipeline. The data from different sources (which can be very different: from raw sensor data to high-level user reports) is collected. This function is the Data Acquisition. The collected data is integrated - possibly to a common format. This function is the Data Manipulation, which consists of the pre-processing and transformation of the data. The amount of formatted data can be huge - a distributed system is needed. State Detection, Health Assessment and Prognostic Assessment functions can be placed along the data processing pipeline, but we need to take into consideration that these functions will be distributed between different levels (i.e. machine level and higher). Detection and assessment can happen in real-time, for instance when based on historical data it is immediately clear what is the cause, but in many cases off-line with intermediate storage of the data when the cause of the event is unknown and complex processing is required. The results and a possible advisory should be presented on the HMI[6].

The *Data Acquisition* block may represent specialized module, that collect data by analogue and digital sensors or a server of calibrated digitized sensor data records. In our case I refer to this block as a hardware module or *low-level*, because it's not available any historical data, that's why sensor based data acquisition has a big role to play in our implementation. These levels will be discussed in the following chapter from a use case related aspect.

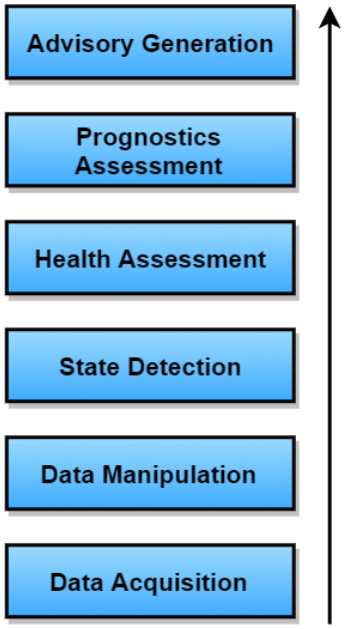

Figure 1.4. *ISO-13374 standard's functional blocks, the arrow sings the typical direction of dataflow*

1.3.2 Generic data analysis process in condition based maintenance

Diagnostics has – more or less – always been information-driven, but in the most cases information meant expert knowledge. In smart maintenance diagnostic processes are built upon collected

[6]Human Machine Interface

data by sensors (or other sources) and expert knowledge has a different role in understanding data. Knowledge engineering refers to any domain specific knowledge that is elicited, compiled and formalized. Once formalized it may be digitally recorded and processed further. For example maintenance expertise may be formalized as an expert system that aid technicians in diagnosing and solving machine breakdowns. As it's shown in Figure 1.6 knowledge from domain experts has a big impact on analyzing data from any source – that includes raw sensors data, databases, ERP systems. After preprocessing – that basically means the step required to make data ready for automated processing – *data analysis* is the key process of both prognostics and diagnostics, but of course it covers different procedures that provide data to one or the other. The biggest difficulty is choosing the right algorithms and models. Because every block and process are built upon each other it's not adequate to validate a model or algorithm at the end of the process. As it can be seen in Figure 1.5 data mining should be an iterative process in which the interactions between subtasks leads us to an appropriate solution (model or algorithm).

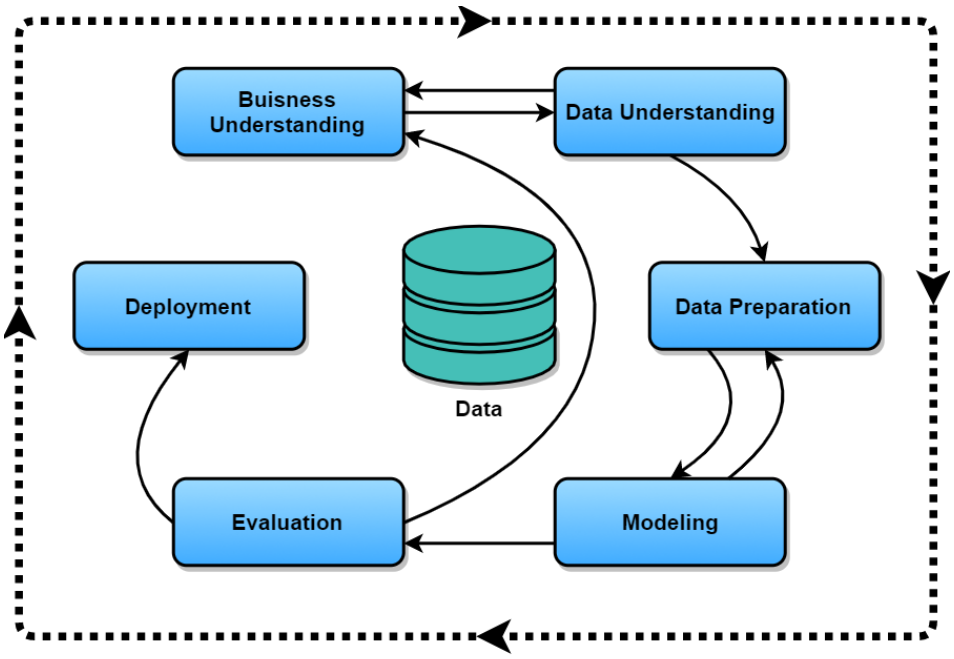

Figure 1.5. *The standard iterative model/process for data mining and model verifying*

That's why data analysis refers to all automated data processing that generates pattern recognition, regression and classification models. It includes a vast number of techniques and algorithms available in the machine learning literature. Here I refer to any of the algorithms – e. g. various filtering, correlation, pattern recognition and trend detection methods – that may be applied specifically to one of the following types of maintenance analysis: failure (state) detection, diagnostics (health assessment) and prognostics (prognostics assessment). In the following section I focus on the health assessment block, more precisely an important algorithm of diagnostics.

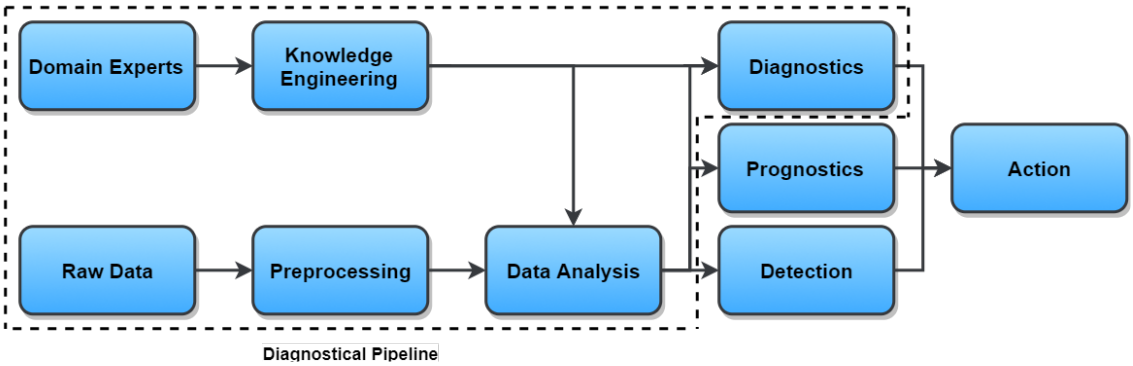

Figure 1.6. *Generic data analysis pipeline*

1.3.3 Root Cause Analysis in diagnostics

As I have already mentioned in smart maintenance discovering the roots of problems and failures is one of the most important task of diagnostics. The output of state detection provides us the current condition of a piece of equipment's parts and we use this information to investigate the failure. A possible solution is the *RCA (Root Cause Analysis)* algorithm, which is a standard, wieldy used method to identify root causes. It's important to emphasize that RCA is general process or even more an approach and it doesn't have restrictions on the used tools, processes and routines. The basic approach is the following [7]:

- Define the problem and formulate hypotheses
- Gather data and evidence to classify causes and parameters of condition in a sequence of effects
- Investigate key parameters through routines and event correlation checks to find root causes
- Once a result of a check is available, it starts new routines using the new pieces of information until the root cause is found and no further checks are envisioned

This procedure can be implemented in many different ways depending on the problem that we want to solve. There are two different approach: *model driven* – when a physical description of the system is given – and *data driven* – if the behaviour of system is unknown. In the last case correlation rules will be defined by data analytics tools e.g. pattern recognition and machine learning [20]. In next chapter I will show a method in details but it's worth to mention other possible solutions:

- Codebook approach – rule and case based reasoning
- Model based approaches – Kalman filtering
- Probabilistic approach – Bayesian networks, Dempster-Shafer evidence theory, neural networks, fuzzy logic

The result of the RCA is one or more possible root causes, marked with different probabilities.

1.4 Prognostics: the backbone of proactive maintenance

1.4.1 Role of Remaining Useful Life[7]

The output of RCA – so diagnostics – included root cause of failure – and many others – will be used in prognostics, which is another important process of maintenance beside diagnostics. There are many different definitons of prognostics, so I only briefly formulate the essence: The main task of prognostics is forecast the equipment remaining useful life, in other words predicting an expecting time to failure. It's relies heavily upon the results of diagnostics, mainly the information about time progression of the equipment's condtition. Of course there are other important tasks too – that I won't detail in this paper – like: findig correlation and interrelationship between failure modes, deterioration rate analysis and examining the effect of maintenance. However, the boundary between prognostics and diagnostics isn't well defined, so certain processes would be assigned to both of them.

The product of prognostics is the *RUL (Remaining Useful Life)*, which is one of the most important *calculated* information about the equipment. Associated confidence limit is another important output of prognostics. It tells us how reliable the estimated value of RUL is. In this paper I don't focus confidence limit, because from our point of view – in this phase of desgin and implmentation – it does not play a crucial role. Scheduling maintenance tasks is built upon RUL so the efficiency of a proactive maintenance system depends on the realibility of the estimated value. so on the prognostic model that we chose. To understand what the prognostic model and its role is, we have to take a look at the general method. When calculating RUL, we need to know the behavoiur of the equipment and its components, to anwser those questions, which lead us to the result, like:

- How quickly does the condition of the component deteriorate?
- How severe is the degradation now?
- What events will change this expected degradation behaviour?
- How may and how many other factors affect the estimated value?

1.4.2 RUL prediction models

It's obvious we can build a model that describes how the component's condition is wearing out and its ability to estimate RUL depends on how accurate is. There are different approaches how models can be implemented and these offers different confidence levels. Of course, models that predict RUL more effectively are more complex, therefore the level of the expert knowledge we have about the equipment and the available data affects which approach we choose. A short classification of different type of models by their complexity can be seen in Appendix F.2.

[7] This section is mostly based on [16].

Knowledge based models search similarity between an observed situation – state of wear-out – and a databank of predefined failures deduce the life expectancy from previous events. Two approaches are involved here:

- *Expert systems* contain a large number of fixed rules that are precisely formulated *if-else* statements by human experts, thus the system can emulate how a human expert does the estimate of RUL. It's a simple prognostic model and relatively easy to implement, but it requires a lot of rules and precise inputs and its efficiency by far lower than the other ones.
- *Fuzzy systems* are based on fuzzy logic, but still similar to the ones have fixed rules. This model still needs experts to define rules, but there are less restrictions on the rules and the quality of inputs, however it requires a huge quantity of continuous input data.

Life expectancy models – as well as the statistical ones – determine the remaining useful life of individual machine components with respect to the expected risk of deterioration under known operating conditions. It contains two main categories:

- *Stochastic models* provide reliability-related information, such as Mean Time to Failure (MTTF) as probabilities of failure with respect to time. They are based on the assumption that the times to failure of identical components can be considered statistically identical and independent random variables and thus be described by a probability density function. This type of model has numerous implementations like Markov, Hidden Markov and Bayesian techniques based models that offer many different benefits, like: it's easy to implement it with software or can be used to model multivariate, dynamic processes, although in general it can be said: each implementation requires a large volume of sample or training data or have restrictions on their availability.
- *Statistical models* estimate the time to failure of individual components by damage progression which relies on previous inspection results on similar or the same machine. Forecasting of future deterioration is often based on comparing these results with models representing "good" behaviour or calculated from complex mathematical expressions that describe the behaviour. It is worth noting that these expressions aren't fixed rules but a model of components' behaviour built from previous data sets that are come from condition monitoring. Therefore, these models often categorised as "data driven" models, that are used when a suitable dynamic model of the physical process is not available. Statistical models are easy to develop and historical input data is not necessary in every case, but simplicity is its huge disadvantage too. It's easy to develop a model that may be statistically adequate but physically meaningless and there aren't clear guidelines on the selection of parametric estimation technique.

Artificial Neural Networks (ANN) compute RUL from a mathematical representation of an individual component or the whole system. This representation comes from measured and historical

data sets (like condition monitoring) which used as input and training data. This type of models is efficient at modeling non-linear complex systems, without understanding the physical behaviour of the system. The main disadvantages here are the required big amount of training data, the fairly long time to determinate the most appropriate model and it's difficult to implement.

Physical models quantitatively characterize the behaviour of a failure mode using physical laws. These models estimate RUL by solving a deterministic equation or set of equations derived from extensive empirical data. Using this model includes not only common scientific and engineering knowledge, but specific expert knowledge and field experimentation. Therefore, a huge disadvantage in this case the need of specific knowledge and the challenging implementation, but it offers the most precise RUL estimates of all modelling options.

Each model has different advantages and disadvantages and it's our job to find the appropriate model that matches to our expectations and capabilities. In our case – proactive maintenance of railway junctions/switches – a huge amount of data will be collected by sensors and processed by Big Data technologies that will be introduced in section 2.2. We don't have the adequate depth of expert knowledge to implement a physical model, but we want to get reliable results. Moreover, it's beneficial to implement a maintenance system that can be easily adopted to other use cases, so choosing neural networks would be an excellent choice. However, *now* a statistical model is more suitable, so in the next chapter I detail PHM model and after I discuss why it's preferred in the early phase of design and implementation.

Chapter 2

Cyber-physical system based maintenance methods

In this chapter the most often used technologies and applications in smart maintenance will be introduced. I'll present how diagnostic and prognostic processes are adopted to the use case and the overall concept of such a maintenance system and the requirements and expectations about the designed one.

2.1 Requirements and expectations

2.1.1 Main objectives

Before talking about the functionality and details, it's worth to formulate the overall concept: This project strives for designing and implementing a cyber-physical system based proactive maintenance service for railway switches. Of course our ultimate goal is formulating the standard design principles for such a system which can be integral part of smart production after further developments. I will discuss some opportunities in the last chapter, but now let's stay focus on railroad switches and take a look at the scope of functionality. Such a – general – system will be able to:

- Reduce the adverse impact of maintenance on productivity and costs
- Increase the availability of assets
- Reduce time required for maintenance tasks
- Improve the quality of the maintenance service and products
- Improve labor working conditions and maintenance performance
- Increase sustainability by preventing material loss

As I mentioned in the section 1.1 the main issue that was raised by our industrial partner is the unscheduled maintenance due to the unexpected failures and great distances. In this case solving the issue covers each point above and it's necessary to remember that's a pressing problem of our partner.[1] To fulfil these requirements it's necessary to provide relevant data about the condition of the equipment in real time and statistical information e.g. estimated RUL. Of course providing relevant information assumes that relevant data is gathered by sensors, that's why I detail this topic in the following chapter. The relationship between the requirements and the system's functions is shown in Figure 2.1.

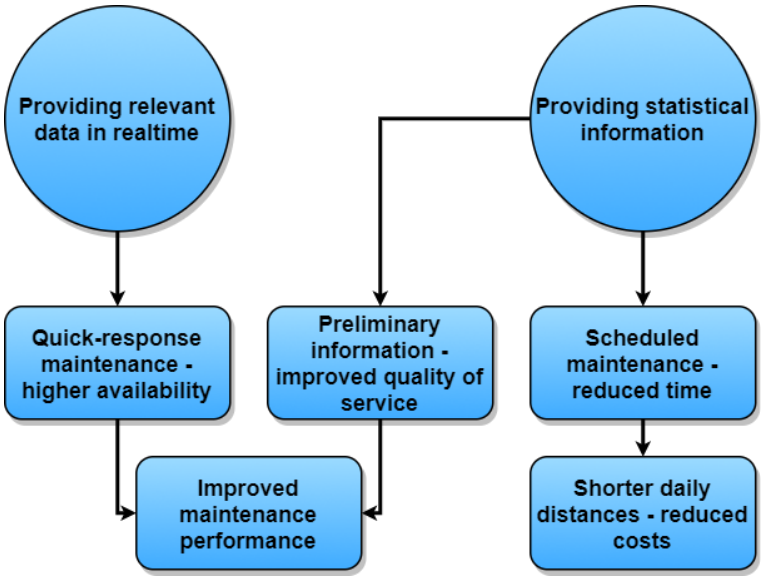

Figure 2.1. *The impact of the system on the maintenance tasks – the "preventing material loss" requirement is missing, because it highly depends on the collected data*

It's still unclear how to verify these requirements generally, when we are talking about an adaptive solution. There are numerous options, but in my opinion a simple method could be an iterative way of verification by matching it towards the different use cases. As the use cases evolve with the findings of the project, a modular system will be validated by checking that it covers the essential parts of the use cases, and will be gradually refined by incorporating use case specific requirements to fully cover all the use cases.

2.1.2 System architecture

The modern proactive approach, this whole project, and actually Industry 4.0 concepts are built on cyber-physical systems, so the architecture of our system too. It consists of two different levels: a *low-level* that was mentioned in section 1.3.1 and a high-level as it's shown in Figure 2.2. Low-level represents the data acquisition block of ISO-13374 standard which is often referred to as *sensors* however, it's much more than only sensors. The main device in low-level is an embedded system that operates sensor moreover it preprocesses the gathered data – in most cases it means

[1] However, I have to note that a highly developed system can offer much more

converting them to an appropriate format – and handles the communication with the high-level. The physical system is a part of the low-level too as the maintenance tasks which isn't only represent itself, but a human-machine interface (HMI) too. HMI is responsible for visualizing the processed data that's provided by the high-level, so repairmen and mechanics are able to get the relevant information. The extended, detailed low-level can be seen in Figure 2.3.

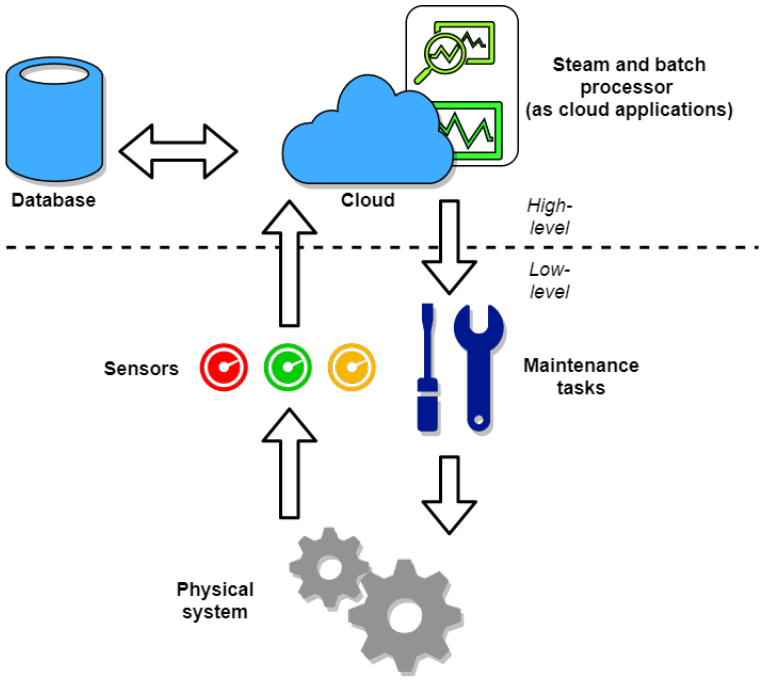

Figure 2.2. *An overall picture about the architecture of the system*

The high-level of such a CPS architecture like our system is often referred as *cloud* that can mean a lot of things which requires high performance and scalability. In our case the most important parts of high-level are: steam processor, batch processor and database. In Figure 2.2 I refer to stream and batch processor as cloud and handle database separately. Gateways that aggregate the dataflow is a part of the high-level in an extended, loaded system when a huge number of low-level devices provide the collected data.

The different processors implement those algorithms and methods that are detailed in chapter 1 as the most important parts of condition monitoring based maintenance. The stream processor handles the incoming dataflow and provides information towards HMI. It can predict events and failures in real-time by using RUL prediction models and trigger the batch processor to start RCA in case of unexpected failure. Batch processor is able to run more complex algorithms that can't be executed in real-time e. g. the learning phase of certain machine learning algorithms that are used to establish RUL models. This module does RCA too and it can demand additional data from the database to ensure an accurate result.

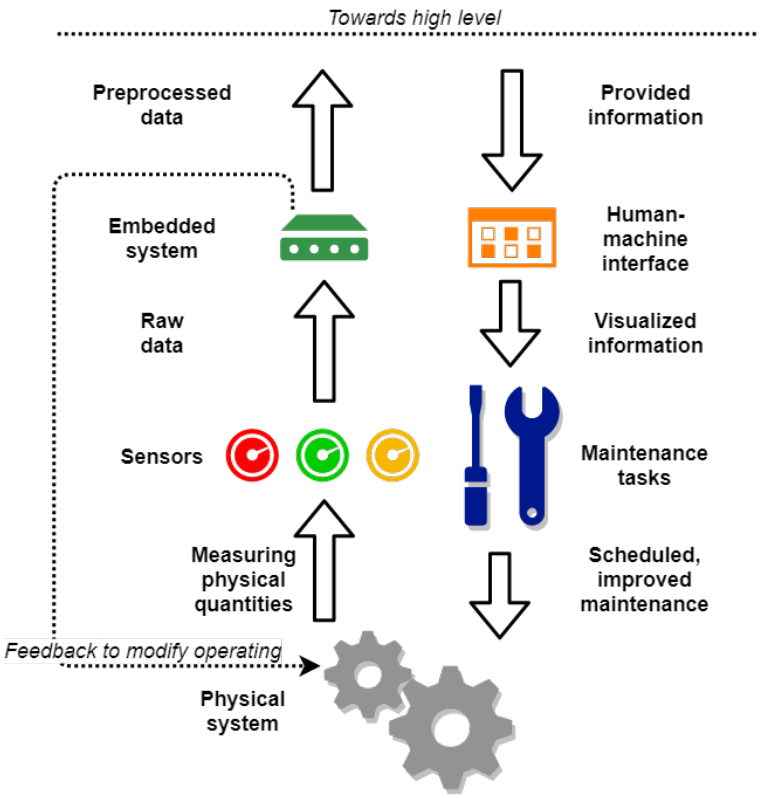

Figure 2.3. *This figure shows an extended low-level. In further design plans the operating of physical system can be influenced by the embedded system based on processed information.*

2.2 Challenges in high-level processing

2.2.1 Could based data processing

At low-level choosing the right tools depends on the use case, but the requirements of these devices and their running processes – for example level of power supply or performance – don't vary on a large scale. In this case I'm about to implement a system for smart maintenance of only one railway switch first, although it's planned to expand the system after successful tests and validation. To fulfill the requirements of high-level processes at small number of switches (or generally, at any equipment) is not critical, almost any available cloud can handle this.

Now we let put the use case aside for a moment and start to think in a common system, scalability and latency (in real-time applications) become more critical factors. However, cloud computing offers large scalability the services are distributed the architecture of the cloud is really centralized and doesn't fit into the IoT paradigm that can't provide the required QoS level (in latency). Using fog or edge computing that use near-user edge devices would be a efficient solution in mission-critical systems due to the reduced needed bandwidth. This approach can effect the efficient operating of Big Data technologies too, that are essential parts of such an system. A further idea is connecting the corresponding CPS based smart maintenance systems to create a collaborating System of systems (SoS) [12], so different systems can communicate with each other moreover they can share data and their results (for example: RUL prediction models for a certain type of

equipment). This approach is still a new research area, but "on-demand" usage of processing and storage resources can provide the basics of this paradigm, so extending such an information-driven maintenance system in the future probably won't be an issue.

To implement high-level functions and processes we have a large number of available solution and it's our job to choose the right ones. There are numerous cloud service provider and mostly they offer the same, so before choosing one we should check them one by one. Primary criteria would be: security, architecture, manageability, support and one of the most important is costs. In the field of Big Data open-source frameworks are very popular, so they have a high-level support. Hadoop (batch), Samza (stream), Spark (hybrid), Strom (stream), Flink (hybrid) and a lot of other projects by Apache are probably the best choices for the role of a stream and batch processor [3][4].

2.2.2 Scalable data uploading and aggregation

There is a part of our system that we haven't talked about yet, because of its special role in the architecture namely the communication. The real problem is that the data doesn't flow only toward one direction between devices: e.g. an embedded system send gathered data to the cloud – more precisely to the stream processor, but the cloud can send messages or commands to the embedded system (in the next chapter I give you an example). That's why the standard server-client connection doesn't fit into this model, we need a bidirectional, persistent communication method. MOM (Message Oriented Middleware) infrastructure are able to implement such a platform independent way of communication. A lightweight realization of MOM is MQTT (Message Queue Telemetry Transport) that is *publish-subscribe* based messaging protocol for use on top of the TCP/IP protocol. There are huge benefits of the MQTT:

- Asynchronous send/receive method – The broker forwards the messages to the communicating parties, thus it saves synchronizing processes.
- Lightweight and distributed – The minimum amount of overhead makes it very useful where the bandwidth is limited moreover MQTT brokers can distribute load, so increase the performance.
- Retain and QoS – QoS levels are extremely important to implement robust communication systems for harsh environments. Retaining messages can be very useful when a command – that is transmitted by cloud/processor – affects all of the low-level devices, thus if a new one joins to the system and get the message it can operate in the same way as the others.

Another problem cloud be the consistency of data formats. Different systems represent data in different ways so the structure of messages will be diverging too. It's worth to mention this question is strongly related to choosing the appropriate database. NoSQL solutions support this type of freedom as regards the format and the developers of Hadoop recommend to use them in Big Data applications. However, NoSQL databases are less mature and still have problems arising from time to time that have not been fully solved yet nonetheless a NoSQL database can be a good choice

for a prototype. Another way to ensure data consistency is using an industrial standard framework. The MIMOSA OSA-EAI [2] standard offers a general, detailed data model for industrial usage and can be implemented in any standard SQL database. Using this approach requires standardised messages correspondingly and offers interoperability with different systems e.g. ERP[3] systems. The structure of messages will be detailed in Section 3.3.2 [2].

2.3 Diagnostic and prognostic methods in maintenance of railroad switches

2.3.1 Implementing RCA – Petri net based failure investigation

Since in section 1.3.3 I presented general processes in the field of diagnostics and now I show how they can be implemented to the actual problem. In this case we don't have – almost – any knowledge about the behaviour of physical system therefore a rule based data-driven approach is used which relies on gathered data. Input data represents parameters of the equipment such as temperature of switch, vibration of rails, other physical quantities and events like malfunctions. Our main goal is finding of the first problem that started a chain of events which ended in a failure. In this method we start to investigate the problem that finally appeared and to check the input data to find out what caused the problem. If it seems a hypothesis is valid, we start a new investigation, but on the new problem and so on until we find a root. If there isn't new available data at the current hypothesis we start to investigate a new one, until we checked all of them. A short example to RCA can be seen in Figure 2.4. It can be seen this method is based on data availability and requires simultaneous task execution. *Dataflow-driven* paradigm - as it was mentioned – fulfils our requirements and according to: [17], it can be implemented effectively with Petri nets, which is a quite new approach.

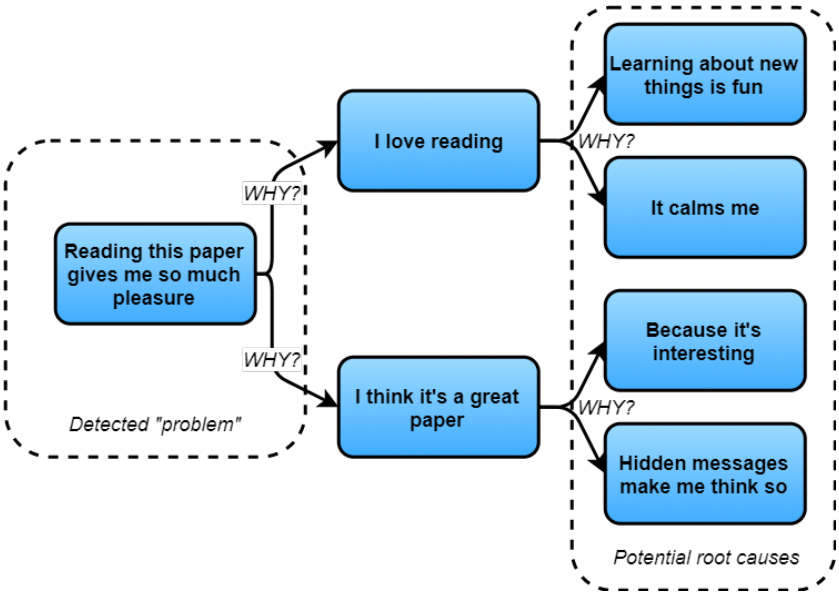

Figure 2.4. *A simple explanation of RCA including two steps (2 Whys)*

[2]Open System Architecture for Enterprise Application Integration
[3]Enterprise Resource Planning

Petri net is directed bipartite graph which is used to model distributed systems with discrete states. It consists two type of nodes: places (circles) and transactions (rectangles). Lines connects only places with transactions and vice versa, but not the nodes in the same group. A possible mathematical description of Petri net:

$$\text{place: } p \in P, \text{ transaction: } t \in T, \text{ line: } e \in E : (P \times T \cup T \times P)$$

In the Petri net tokens (marks) are used to describe states by which places contain them. A transaction can *fire* if sufficient number of tokens are available in all of its input places. Then a new token will be created its output places and all of the input tokens that were used to fire will be deleted [20].

It's clear that Petri net based RCA method requires a huge amount of input data that is painful disadvantage in the early phase of implementation however, the depending on data availability turns into a really important advantage later. It was stated before stream processor triggers batch processor to start an RCA if a failure occurs but batch processor also can trigger various data sources to provide data during running RCA. This function is really because RCA can offer a more confident output due to available historical data in database. Moreover, it can verify failure by triggering low-level device which the exact nature of the failure can be determined from as it's presented in section 4.2.2.

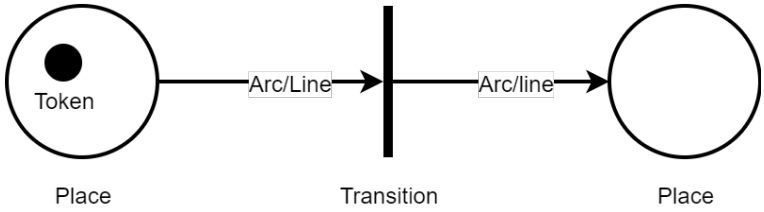

Figure 2.5. *Representation of a petri net – places are circles, transitions are rectangles and tokens are black dots*

2.3.2 Demonstrating RCA method – "Coffee machine problem"

Now, I demonstrate the operation of the mechanism on a simple example – because the case of railroad switches is much more complex – based on a real problem: I want to make a cup of coffee with my espresso machine, but for an unknown reason the machine doesn't work and some of the lights are flashing. To find the root causes, I use the RCA algorithm and construct a Petri net to model the problem. In this case the input data is the fact itself that I cannot make coffee and every lights are flashing. According to this a token is created in the place named *"No coffee"* and another one in the place *"Every lights are flashing"*, so a transaction fires, as it's shown in Figure 2.6.

As the machine's manual says, if every lights are flashing then there is too much steam in the system, so in the Petri net a token is created in the place named like that. In this state the new available input data has to be checked. Since, I'm an experienced user I know that there is two

possible reason: I used the milk steaming wand and I forgot let the steam out from the system or I put too much coffee into the machine, so the water and hot steam can't flow through fine ground coffee. It's easy to find out: if I used steaming wand probably the first answer is correct. If I didn't used, then the second is correct one, like in this case. It means that the place which represent the input data has a token, so a new transaction can fire. In the final state of the process there is one token in the place that represents too much coffee in the machine, so we discover root cause of the failure. The whole process is observable in the figures on appendix F.1.

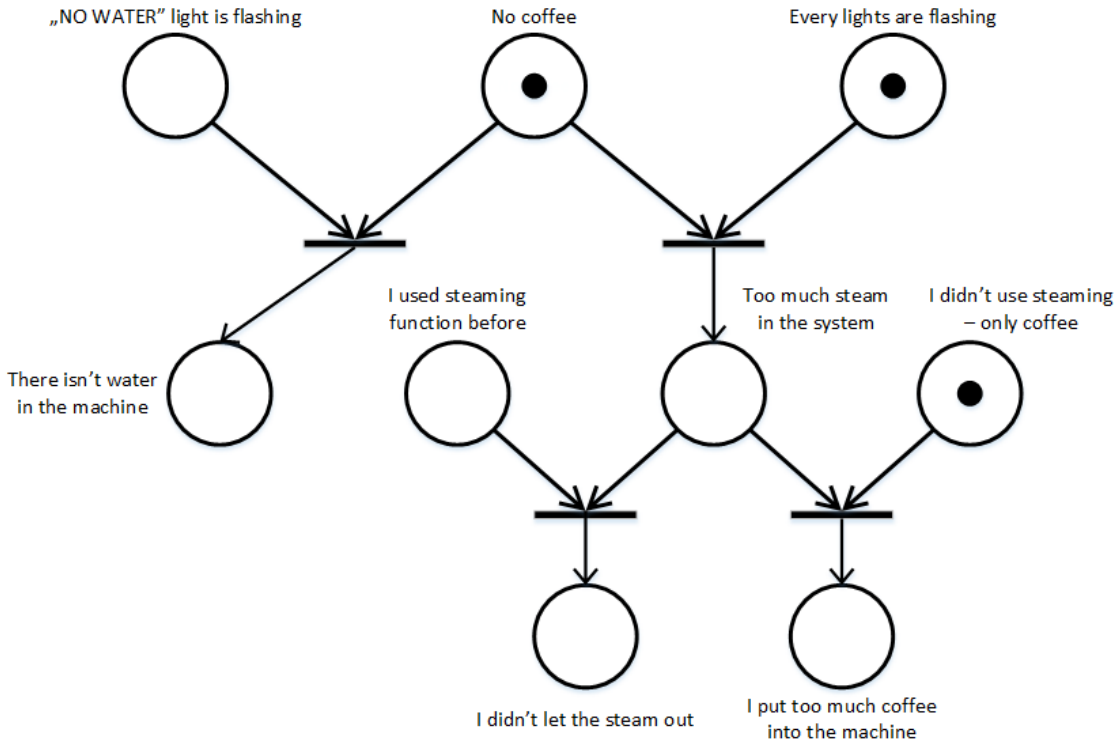

Figure 2.6. *A simple problem solving with RCA and Petri net – at starting state*

2.3.3 Proportional hazards models based RUL prediction

Proportional hazards models (PHM) are the most commonly and widely used models in prognostics. Since D. R. Cox has introduced this type of model for the first time, it has evolved and there are other more complex models now, but its basics haven't changed. In this approach we model, how the explanatory variables – will hereafter be cited as covariates affect the condition of the equipment. The most basic regression models are linear, so they assume the equipment or system will always wear out the same way in contrast to PHM assumes covariates have an effect on the hazard rate. RUL can be estimated from the survival function, which can be estimated from the product of a baseline hazard function and a positive function that is described by a vector of covariates and an associated vector of unknown regression parameters [16].

Our main goal is the identification of the *hazard rate* – $h(t)$ –, which shows the probability of a failure in a short time interval – $(t+\Delta t)$ – given survival to time **t**. I'll show a mathematical proof in appendix F.3 for those interested, how hazard rate is related to density function and reliability

function – $f(t)$ and $R(t)$ – but now we treat it as a fact. A basic hazard model offers a time influenced hazard rate:

$$\lambda(t) = \frac{f(t)}{R(t)} \tag{2.1}$$

However, in general not only the time, but covariates. that are associated with the system or equipment have an influence on hazard rate too. That's why PHM defines hazard rate as a product of a *baseline hazard rate* – $\lambda_0(t)$ and a *positive function* – $\psi(\mathbf{z};\beta)$, where $\mathbf{z}$ is a row vector consisting of the covariates and β is a column vector consisting of the *regression parameters*. These parameters are unkownn and associated with the model, defining the effects of covariates [15]. Positive function has different forms, now I present the common exponential form – included covariates and parameters – due to its simplicity.

$$\lambda(t;\mathbf{z}) = \lambda_0(t)\psi(\mathbf{z};\beta) = \lambda_0(t)e^{\mathbf{z}\beta} = \lambda_0(t)exp(\sum_{j=1}\beta_i z_i) \tag{2.2}$$

To establish a model, it's necessary to estimate the β regression without making any assumptions about the functional form of the baseline hazard rate. There are a few ways to calculate these values from collected data: maximum likelihood method is widely used, but other approaches are available too. Another challenge is choosing the significant covariates, because having irrelevant factors in the model affect calculating the parameters. When all covariates are defined baseline hazard function can be estimated, but in most cases it's assumed the function is exponential to facilitate the use of common regression methods. Now, survival function and RUL can be estimated from baseline hazard function:

$$R_0(t) = exp[-\int_0^t \lambda_0(x)dx] \tag{2.3}$$

In what follows, I outline why PHM is an appropriate first pick to model degradation of an equipment and how we should use it to fit into our expectations. A detailed description about PHM and other calculations – like handling measurement errors and misclassification – can be found here: [6].

2.3.4 Fitting PHM to the use case – An appropriate solution to estimate RUL

The most important question in this whole section that I have to answer is: Why PHM? We can choose any model to predict RUL and most of them are fulfill our requirements, for example a really adaptive implementation would be based on neural networks. Deep learning is an upcoming part of machine learning methods and it seems to be that trend won't change soon. So, why shouldn't we choose a fresh technology that probably offers a quite accurate and reliable solution to us? The short answer is: because we can't.

The biggest problem here is that we are in an early phase of implementation of such a "smart" system. ANNs require a huge amount of input/training data that we can't provide, but it isn't only about the quantity. Gathering relevant information is quite more important, that's why having human experts is necessary during designing and implementing a system. However, it's not an easy challenge to pick relevant parameters because we don't have that expert knowledge and for our partners it's a whole new situation. I demonstrate this with an example: When we talk about proactive maintenance and management of railway switches we don't specify the type of the switch, it's handled as a *"general type"*. The truth is that – according to an expert with whom I had a conversation about it – each type of switch has its own relevant parameters or physical quantities. For example, longitudinal displacement of *point blades* (the rails that are moving) has a great impact on the operation of certain types of switches, but on the other types it does not [14]. It's more efficient to recognize and to measure only the common relevant parameters in the first place when we design a prototype. Therefore, it's not affordable to use a complex model which is hard to implement, because when can't utilize the more precise results in this early phase.

Bearing in mind we are about to design a prototype, the main requirements for our model are: the less expert knowledge, easy implementation based on input data and handling the properties of degradation, which depends on multiple factors. Statistical models are the best choice for this case but, proportional hazards models have some advantages over other techniques like *trend extrapolation*. All of them are simple and rely on input dataset moreover they are highly supported in data mining thus it's available to establish a model manually with tools like *R language*, but only PHM can handle the multiplicative effects of covariates, which is one of our most important requirement.

Unfortunately, PHM has its limits that I have to discuss too. One of the biggest one is: a simple standard PHM model can't handle time-dependent covariates. It's obvious temperature, humidity, level of dust and a lot of other physical quantities that would have an impact of degradation of a piece of equipment is time-dependent one by one. Extended proportional hazards models could be a suitable solution, but in this case we lose the simplicity. Easy and affordable method to substitute variables that depend on time with its average values (or weighted average). Now we have a simplified (but still efficiently usable) tool to model degradation and estimate RUL. We can state without any doubt: this model will overfit and it's probably none usable for any other use-case, but it's powerful to validate and verify relevant parameters (covariates) that we want to use in the future as a part of a more complex prognostic model.

Chapter 3

Implementation of a smart maintenance system for railroad switches

In previous chapters I outlined certain points of the use case and presented the essential parts of railway switches and their operations in our point of view. Now I won't repeat them, but I will refine the expectations, investigate the opportunities and finally first design then implement the actual system – focusing on its low-level parts.

3.1 Design fundamentals and elements

3.1.1 Defining use case based requirements

In this project our industrial partner is a company that designs and manufactures mostly signaling systems. In the first round I'll design and implement a low-level device that they can use in maintenance of railroad switches. Collecting data by low-level device and validating the system are their tasks but if a huge amount of data is gathered – and also experience – we can refine the existing device and the whole system. Based on this I formulate the actual expectations which design can start from:

- The system must be capable of measuring the relevant physical quantities that represents the condition of a railway switch
- The gathered data has to be preprocessed and forwarded to further analysis in an eligible format (inside a message)
- The system has to provide statistical information about the degradation of physical system (failure trend) and real-time information about the current status of operating[1]

These principles will be used as directives that I will follow during design and implementation. Each of them are consistent with the previously established general conceptions, but fulfiling them won't be an easy challenge and I will soon show you why.

[1] It will be seen providing statistical information is out of our scope in this case

3.1.2 Parameters of the physical system

The first problem is choosing the physical quantities which the system has to measure and this issue arises from several factors. As the first implementation of the system is just a prototype it's suitable to gather only a few parameters that are influence the condition of the equipment the most. Another difficulty is the different mechanical construction in different type of switches consequently, the relevance of certain parameters highly depends upon what type of switch we talk about. For example – as I mentioned before in section 2.3.4 – the operation of certain switches is not sensitive to the longitudinal displacement of point blades, but it has a great impact on operation of other types [14]. According to this, I try to find common parameters first.

It's pretty sure that environment parameters are common, although their degree of influence on the condition of an equipment may be different. Measuring temperature seems to be a good choice, because it affects every mechanical system and can be measured easily with external sensor. A really confusing question is what can I call temperature? The ambient temperature or the temperature of the rails. A well-known possible malfunction is the dilation of rails which is caused by temperature in most cases – but, I have to notice that not only thermal expansion can deform rails and by the way the lock.[2] Therefore rail temperature has to be measured as one of the most important parameter. Another environmental quantity is humidity which plays a lead role in corrosion that doesn't only affect the railway but all part of the switch so the switching mechanism too. Because humidity is related to ambient temperature and the most humidity sensor can measure temperature and vices versa, it's obvious both of them have to be measured. We shouldn't forget the model has to be kept simple, so let's move on mechanical parameters.

Gathering data about lateral displacement of point blades is essential in this case, because that can provide real-time information about the status of operating. The state of point blades clearly defines if a failure has occurred. If the actual state of the points isn't at the edge of the measurement range the switching operation failed. We can detect splitting too based on the speed of operating (the moving of the blades). The longitudnal displacement has been mentioned many times and we can lay down it's not a common parameter unlike lateral one. Now it's a dilemma to choose a type specific parameter to measure. In my opinion longitudinal displacement would be an efficient choice, because the most common switches in duty are sensitive to this type of movement and other special parameters like even radius of curvature are hard to be measured. However, it wasn't me who made a decision: our partner denoted that longitudinal displacement is relevant in the case of test switch.

Now we have three environmental factors and two mechanical factors so it should be checked what are they good for. Lateral displacement represents the state of the points thus the status of the switching operation that is indispensable to provide real-time information. The other parameters are related to the condition of the equipment, itemized:

- Value of humidity and ambient temperature influence the degree of corrosion, that affect not only the condition but the operation too

[2] For example: dynamic forces generated by moving trains

- Value of rail temperature is related to dilation of the rails so the usability of the junction
- Degree of points' longitudinal displacement affects the operation and the condition of every part of the switch included the points and the lock

This list shows that system meets the requirements for limited functionality based on the measured data. The way of measurement and handle data will be presented in the following sections.

3.2 Implementation of hardware

The first things I have to deal with the components of low device. We should consider the sensors to be given because our partner's suggestions were accepted so we have already chosen them with the required resolution. What we have now is: four analogue displacement sensor (two for one direction and two for another one), an analogue temperature sensor for measuring rail temperature with a supplementary ADC unit – that communicates via SPI, and a combined temperature and humidity sensor for measuring ambient quantities – that communicates via I^2C.

The main part of the embedded system is the microcontroller unit (MCU) that will read the data from the sensors and communicate with auxiliary units. The MCU has to preprocess a huge amount of data, convert analog signal to digital data and handle different communication line so what we need is a high performance unit with rich peripheral set. In addition, I'd like to avoid further design and assemblage so the best solution would be an MCU that is integrated to a board which has the necessary supplementary components. Finally, a popular STM32F407 based development was chosen that fulfills all requirement. The STM32F407 offers the performance of the Cortex^{TM}-M4 core (with floating point unit) running at 168 MHz thus preprocessing and building messages up (some message can reach the size of 64 KByte) won't be an issue. It has numerous communication interfaces, Analog-to-digital converters, 192 KBytes of SRAM just to mention what we need.

The second most important component is the communication module that I haven't talked about yet. In this case the low-level device will be placed into a junction that can be anywhere therefore it's necessary to offer wireless connection. Wi-Fi network won't be available that's why we have to use mobile network to connect to internet. To solve this issue a SIM-800 chip based modem FONA by Adafruit was chosen to play the role of communication module. It can be controlled by AT commands (Hayes command set) that are sent by MCU via UART line. The modem is not only used for MQTT communication but to synchronize MCU's real time clock (RTC) with UTC[3] via HTTP.

The assembling of the modules was made by jumper cables during the first tests, but each module have been soldered to the board in certain cases via cables later. Since the device will be placed into a waterproof box later the modules don't require any mechanical stability. A picture of a very first prototype of the system – the final version has not been photographed – can be found in Appendix F.4 and a schematic draft in Figure 3.1. Figure 3.2 provides a picture of overall system

[3] Coordinated Universal Time

included high-level when where message sent by low-level device will be processed by Apache Big Data framework and stored in Mimosa database.[4]

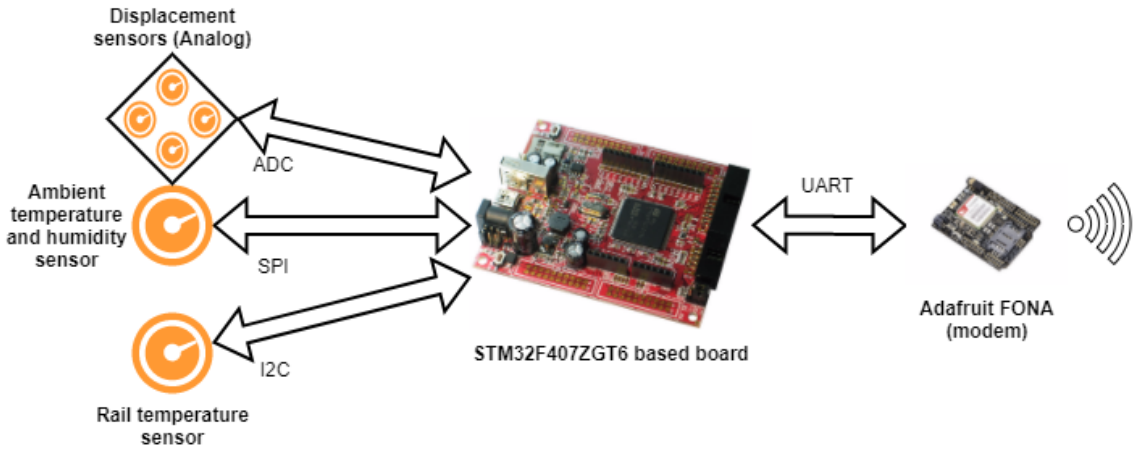

Figure 3.1. *A schematic figure of components of low-level device*

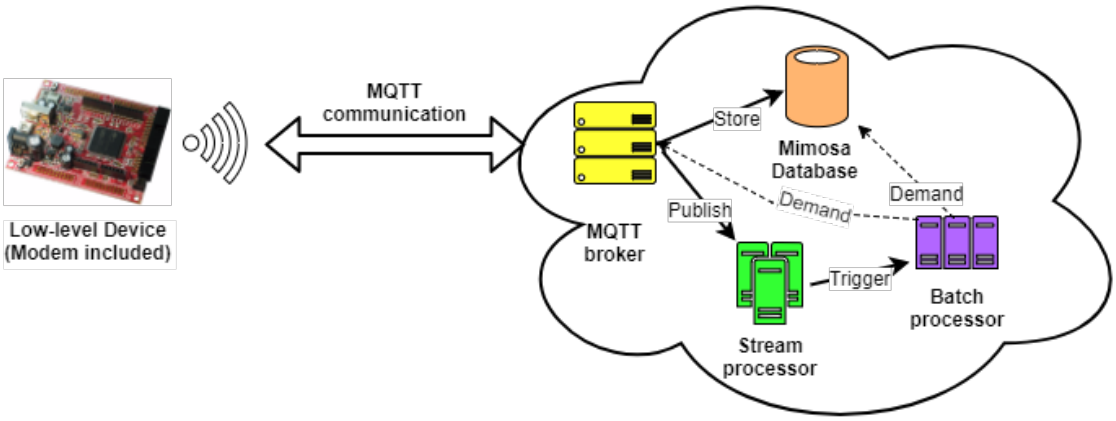

Figure 3.2. *A schematic of overall system*

3.3 Challenges in software designing

The next step is the design of embedded device, which collects data and transmits it. In this section I focus first on the data acquisition and preprocessing sequence and later the communication.

3.3.1 Design of measurement cycle[5]

The easy part of this step is measuring the environmental quantities, because neither the humidity nor the temperature varies fast. An effective solution can be sampling with a given frequency thus

[4] A detailed description about implementation of high-level architecture and data mining can be found in Zoltán Umlauf's work: [19]

[5] This step of design was a joint work with Csaba Hegedűs. This part from another aspect and a full project proposal can be found in his work: [11]

we don't have to implement event-driven operation, but the question is how detailed data we need. Because the answerr is unknown yet, it's practical to handle the sampling frequency – not to be confused with f_s from Nyquist-Shannon sampling theorem – as a parameter hereby making this function more adaptive.

Measuring displacements is bigger issue, because we don't require data at constant time intervals, but all the times when switching operation happens. To achieve it the device has to be triggered by the sensor to start a measurement, but that's not the interesting part. When the switching operation ends the measurement has to be stopped and that raises a question: How long time will a switching operation take? According to our partner the points pass from one end position to another within approximately four seconds but a maximum of ten seconds. Another option to set a second trigger when the points reach their end position but that may cause infinitely long measurement when switching operation fails and the points stay somewhere in the middle of the range. It would be the best if we will integrate the two approaches into one that means: When a switching operation begins it triggers the device to start a measurement that will run until the points reach their end positions or a timer expires. In this case it would be efficient to define the maximum interval as a variable thus it can be changed anytime if we want to increase the measurement interval so the amount of gathered data. This measurement sequence can be described with a final-state machine (FSM) that makes it easier to implement. The final version of the FSM that is shown in Figure 3.3.

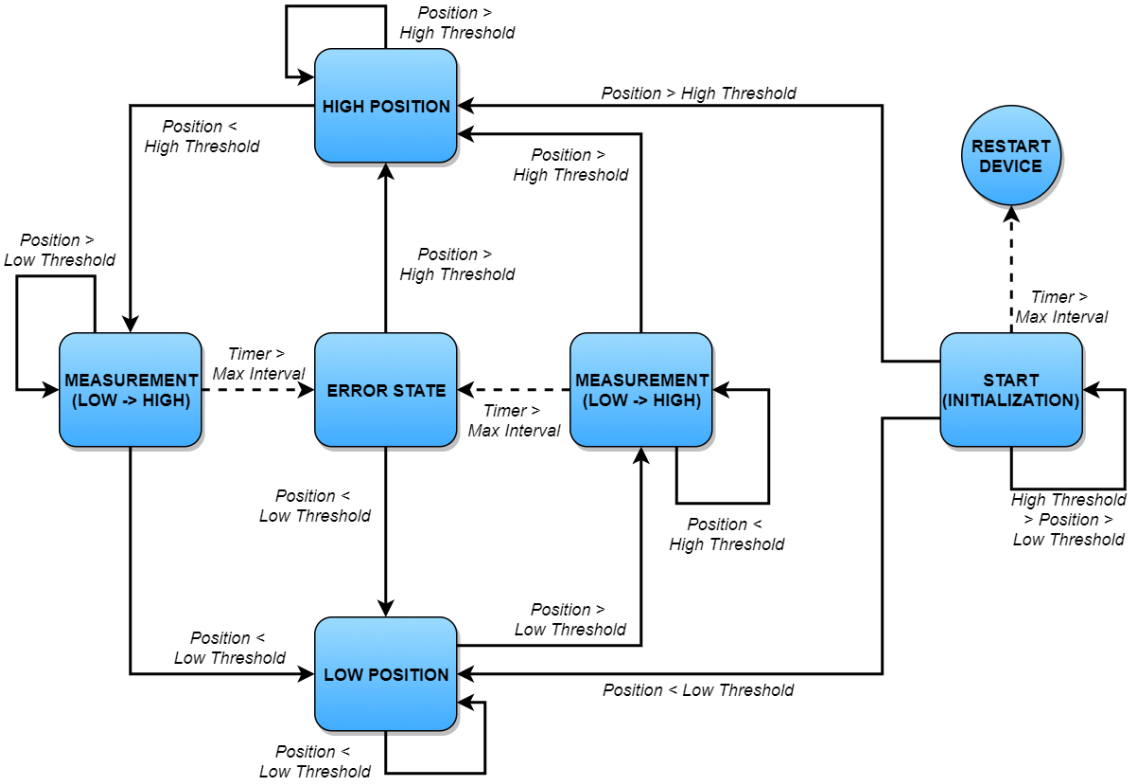

Figure 3.3. *An FSM representation of the measurement sequence*

As it's shown several different states have been defined which require explanation. *High Position* and *Low Position* represent the end states of points but technically they are interchangeable, because it doesn't matter which end position will be fixed to the value of zero the distance between

the positions is constant. The low and high appellations refer to the displacement sensor's value and thus the two states are more distinguishable then they would have called *state A* and *state B*. There are two *Measurement* state that are distinguished from each other in the figure but as we change the end positions the measurement state will be reversed too. Basically the direction of switching doesn't affect the measurement procedure only the events that can be triggered. There is a state that I haven't talked about yet the *Error* state. When a measurement starts the process steps into Measurement state and it can step forward when if points reach end positions within the expected time which is shown as Max Interval in the Figure. Otherwise the process steps into Error state where a warning message will be generated to indicate the failure. The process can leave Error state to any end position without any limitations upon time.

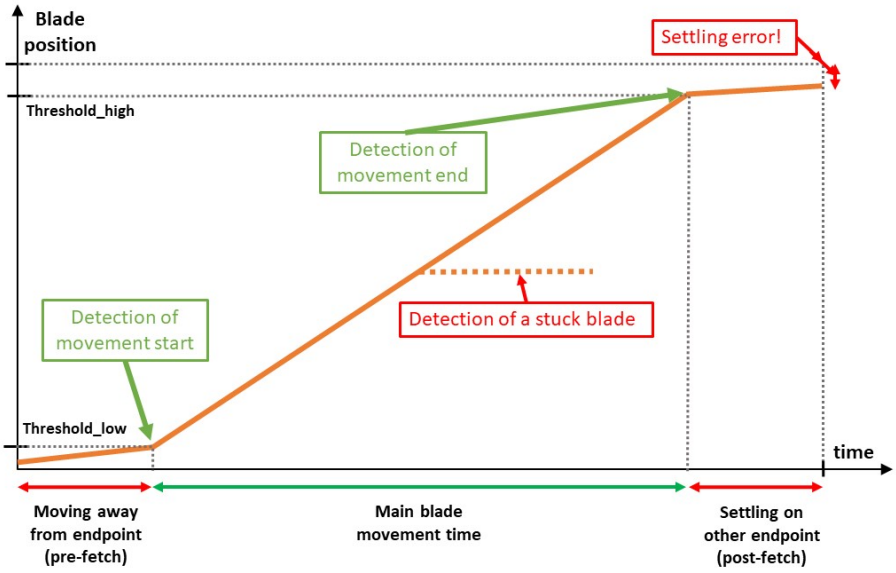

Figure 3.4. *A schematic figure of measuring actual position of points [11]*

Since the sensors gather data in high resolution the value of the end positions can vary on a – relatively small – interval. Therefore, two threshold levels were defined which means that every value above of high threshold represent high state and every value under low threshold represent low state. If the value of the actual position – in the figure it's simply *position* – pass one of thresholds over it triggers the events inside the measurement sequence. Moreover, threshold levels secure that the fluctuation of *position* doesn't trigger the measurement event. The level of thresholds can be set manually as other parameters. A schematic figure of measurement can be found in Figure 3.4.

The figure contains a *Start* state which represents the starting of low-level device (embedded system) and the initialization processes. Basically the system is waiting in Start status until the actual position will be detected so the process can step into both low and high position. If the position is still between the threshold levels after maximum time interval the device restarts, because it's probably due to the wrong settings of threshold levels.

In many cases it may be useful to check the position of points between switching operations. A *command* message from the cloud (batch processor) triggers a quick measurement to get informa-

tion about the actual end positions. However, a trigger event by threshold levels has higher priority than command message therefore if measurement is going on command message doesn't trigger another measurement.

3.3.2 Data format and message structure

In the previous chapter I've talked about communication as a part of the architecture and presented MQTT protocol that the system will use. In this section I'll detail how a message will look like and how communication will work between low and high-level. As the messaging depends on the information that will be sent first I summarize what type and amount of data we have. A manifest classification of data is based on its nature so the way of measurement. For example: in this case we have humidity, ambient temperature, rail temperature, lateral displacement and longitudinal displacement. The first three quantities are related to the environment and that's why they will be measured periodically. The last two ones are related to the operation of switch therefore, this type of measurement will be started by trigger events. This grouping can lead us to define two types of messages: *environment* message for the environmental quantities and *movement* message for lateral and longitudinal displacements. The different messages will be published at different topics to operate efficiently thus the last of each measurement will be stored in the broker. An additional topic will be reserved for commands that are sent by batch processor to embedded system.

Another challenge that I've faced with is to design the inner structure of messages. The first step is to choose an appropriate data format (*file format*) that fits into MQTT based communication and can be handled easily – mostly by the embedded system. The widely used JSON (*JavaScript Object Notation*) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types. The JSON-MQTT coupling is a commonly used solution for systems that require lightweight messaging and certain QoS requirements in the world of IoT.

The next thing I have to deal with is: what should a JSON formatted environment or movement message contain besides data. As I've mentioned in Section 2.2.2 using general data models would be efficient in several aspects, but the content of messages will be restricted. An essential part of messages should be the identifiers that describe the type of a value. Not only MIMOSA, but every database uses *universally unique identifiers* (UUID) as unique keys which represents quantities in this case. Another important information is the "location" i.e. the identifier of a device which allows the measurement of different devices to be separated. The last real data that will be included is a timestamp in UTC format which will be generated by the device. Any other attribute in the file that wasn't detailed before is related to the general data model and has a role in data description.

In Figure 3.5 the first part of an environment message is shown. The additional attributes make the file hard to read so in the following I give a short explanation. The first two "attribute-value" pairs are related to classification of data: "DA" stands for Data Acquisition and "env_meas" means environmental measurement. These attributes aren't important in any processes, but they allow us

to search in the database manually, because we can omit the using of UUIDs. It can be seen the values of ambient temperature, ambient humidity and rail temperature are attached to UUID 364, 365 and 366, respectively.


```
{
    "hasAttributes": [
        {
            "hasAttributeClassType": "DA",
            "hasName": "env_meas",
            "hasValueContainers": [
                {
                    "hasValue": {
                        "value": 21.200256
                    },
                    "uuid": {
                        "value": "364"
                    }
                },
                {
                    "hasValue": {
                        "value": 51.855469
                    },
                    "uuid": {
                        "value": "365"
                    }
                },
                {
                    "hasValue": {
                        "value": 19.819996
                    },
                    "uuid": {
                        "value": "366"
                    }
                }
            ]
        }
    ],
```


Figure 3.5. *The first part (1) of an environment message that contains the actual data*

The Figure 3.6 shows the first part of a movement message. The values of the measurement are stored in an array because of the huge amount of data. As the displacements are measured with two-two sensors – one for lateral direction and one for longitudinal direction at both switch – therefore, four different values constitute an element of the array. The last value that can be seen is the duration of measurement in milliseconds. Vigilant readers can notice that a UUID's value is "move" and that's because the actual "uuid" hasn't been created yet.

The common second part of messages is shown in Figure 3.7. Both environmental message and movement message contain the same attributes the only difference is the value of "Event Description UUID": Environmental – 3143, Movement – 5392. The end of a message consists of the identifier of the location and the timestamp.

There is a third message type the command message, but its structure is still not standardised. In the implementation the command message contains two strings. One for the type of the command and one for theidentifier of destination device.

```
{
    "hasAttributes": [
        {
            "hasAttributeClassType": "DA",
            "hasName": "mov_meas",
            "hasValueContainers": [
                {
                    "hasValue": {
                        "values": [
                                    "4095-2290-4095-2293",
                                    "4083-2284-4095-2298",
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    "0018-2293-0026-2287",
                                    "0016-2284-0013-2291",
                                    "0014-2289-0002-2285",
                                    "0000-2280-0001-2296"
                        ]
                    },
                    "uuid": {
                        "value": "move"
                    }
                },
                {
                    "hasValue": {
                        "value": 4420
                    },
                    "uuid": {
                        "value": "duration"
                    }
                }
            ]
        }
    ],
```


Figure 3.6. *The first part (1) of an environment message that contains the actual data and the duration of measurement*


```
            ],
            "hasEventDescriptionUuid": {
                "value": "3143"
            },
            "hasLocation": {
                "measLocId": {
                    "value": "DEVICE_001"
                }
            },
            "hasTimestamp": {
                "hasUTCDateTime": {
                    "format": "DATETIME",
                    "value": "2017.10.10.16:11:15"
                }
            }
        }
```


Figure 3.7. *The Figure shows the second part (2) of a movement message that is quite the same as at an environmental message*

3.3.3 Software development

One of the most important part of development is designing the software. The whole program that runs on the MCU was written mainly by me in C language, however it uses the STM's Hardware Abstraction Layer's (HAL) API and the *parson* library (JSON) that increased the speed of development. The program implements the measurement sequence, building up JSON structures and MQTT communication besides many other functions some of which will be presented detailed.

In the previous sections I've talked about a few parameters that affect the operation of device for example: sampling frequency, maximum time interval of measurements, etc. These values are often related to one given device furthermore there are other parameters that are required to build up communication e.g. MQTT broker's name, APN[6] of network – modem needs that information to connect to GPRS network and few others. Because these parameters are required when the system is just offline they have to be stored on a physical medium. Since the board has a MiniSD reader thus it's an obvious solution to use it for that purpose. The settings are stored in a JSON file that can be modified maunally as it's shown in Figure 3.8 and after powering the board on the device will read it.

```
{
    "threshold_min": 500,
    "threshold_max": 3800,
    "meas_timeout": 10000,
    "meas_offset": 500,
    "env_meas_freq": 180,
    "mqtt_host": "mantis1.tmit.bme.hu",
    "port": "1883",
    "gsm_apn": "internet.telekom",
    "ping_retry": 3,
    "client_name": "LL_MEAS_DEV_002"
}
```

Figure 3.8. *The config file of the system consists of the parameters that required for the proper operation*

The first two attributes store the value of threshold levels that can be set between value of 0 and 4096 – which represent the possible outputs of 12-bit ADCs. These values are not converted to physical quantities because of the firm request of our partner. The *"meas_timeout"* is the maximum time of a measurement in seconds, while *"meas_offset"* is the value of prefetch and postfetch time window in second. A measurement runs while the actual position is between thresholds, but in most cases we want to know what happened before and after the switching operation. This parameter tells that how many seconds of data should be provided before the start and after the end of measurement. *"env_meas_freq"* is the so-called sampling frequency but because environmental quantities aren't required in high resolution this parameter can be given in seconds. The next three parameters are the address of MQTT broker, the port number which it is available through and APN's name of the provider of SIM card, respectively. *"ping_retry"* is a legacy parameter that was defined during the first tests because of the poor quality of the modem which I used. If

[6] Access Point Name

the broker doesn't reply to a ping message sent by the device in *ping_retry · keepAlive*[7] time the device restarts the modem. The last parameter of the file is the *"client_name"* which identifies the device during MQTT communication and in database.

The software implements two types of measurement as it was planned a movement measurement and an environmental one. The last one is not a hard task, because each sensor use I^2C or SPI[8] to communicate with the MCU and the HAL's API allows data transfer by only a few function call. The raw data are read from sensors at intervals specified in the config file. A timer runs until it reaches the specified value then it sets a flag which triggers the measurement. After the measurement phase the MCU preprocesses raw data or more precisely converts values to actual information then builds a – serialized – JSON file that will be transmitted as a message via MQTT.[9]

The topic of movement measurement is more interesting because in this case data are read from sensor all the time. The four sensors are connected to different ADC channels which are used in circular mode. A timer runs with a frequency of 400Hz and triggers the reading function that store raw values of ADCs in sufficiently large circular buffers (arrays). Since this timer has the highest priority nothing can interrupt the measurement cycle. If the value of current position crosses a threshold a flag will be set – due to the implemented FSR – which signs that a switching operation is in progress. Then the values from circular buffers will be copied into so-called output buffers in chronological order moreover offset data will be copied too. All of these processes are controlled by further inner flags and trigger events. When actual position reaches the other threshold level copying will be stopped – except copying of offset data – and the content of output buffers will be converted to a format that can be seen in Figure 3.6: Associated values separated by dashes constitute a string and these type of strings will be placed into the JSON formatted message.

A special movement measurement will be executed if the device receives a command message. In this case only – prefetch and postfetch – offset data will be stored in output buffers because there is no actual measured value. Therefore, if the value of *"meas_offset"* is set to $\mathbf{1000}^{[ms]}$ then the "length" of measurement will be **2** seconds long.

In this chapter I've showed how I designed and implemented the low-level device of the system to fulfil the requirements which were defined earlier. I haven't dealt with high-level implementation because our partner take care of it within doors and my works mainly related to low-level as it's seen according to this chapter.

[7] The keep alive functionality assures that the connection is still open and both broker and client are connected to one another in MQTT communication

[8] I^2C and SPI are serial buses for communication between MCUs and peripherals

[9] The detailed process of conversion can be found in sensors' datasheets

Chapter 4

Verification of the developed device

In the following sections I will show how I verified the system that I implemented and how it fulfills the requirements. Since the methods is based on messaging I have to draw attention to that I can't show the whole content of messages due to the limited space however I'll explain every detail that is required to support my statements.

4.1 Verification principles

The system – in this case we're only talking about the low-level device – is considered to be verified if it's capable of preforming each necessary function. These requirements were laid in the Section 3.1.1 thus I only summarize them here: The system has to measure physical quantities, send the values in messages, and provide information about the condition of equipment.

The functionality of system can be proved by monitoring communication, because the measured values are integral parts of messages. Since I have no access to our partner's high-level implementation I will use other tools to show the communication between device and MQTT broker. The application that I use as a HMI is the MQTT-Spy by Eclipse Paho – it's seen in Figure 4.1 – which is an open source utility with GUI[1] to monitor MQTT activities. MQTT-Spy is basically an MQTT client which I subscribe with to topics so I will see the messages between device and broker. Moreover, I'll simulate request/command messages of high level by publishing messages to the command topic.

What I'll present is a test or more precisely I will simulate how the condition of a switch is deteriorating. The process of wearing out is obviously much more longer than in the test but the messaging process is the same in every occasion. The role of switch will be played by a liner slide potentiometer that is connected to the ADC channel which triggers the measurement sequence. In this case further channels (one lateral and the two longitudinally directions) will be tied to ground. The switching operations will be simulated by moving potentiometer from one end position to another – which is very similar to the operation of displacement sensors. First I will move it fast

[1] Graphical User Interface

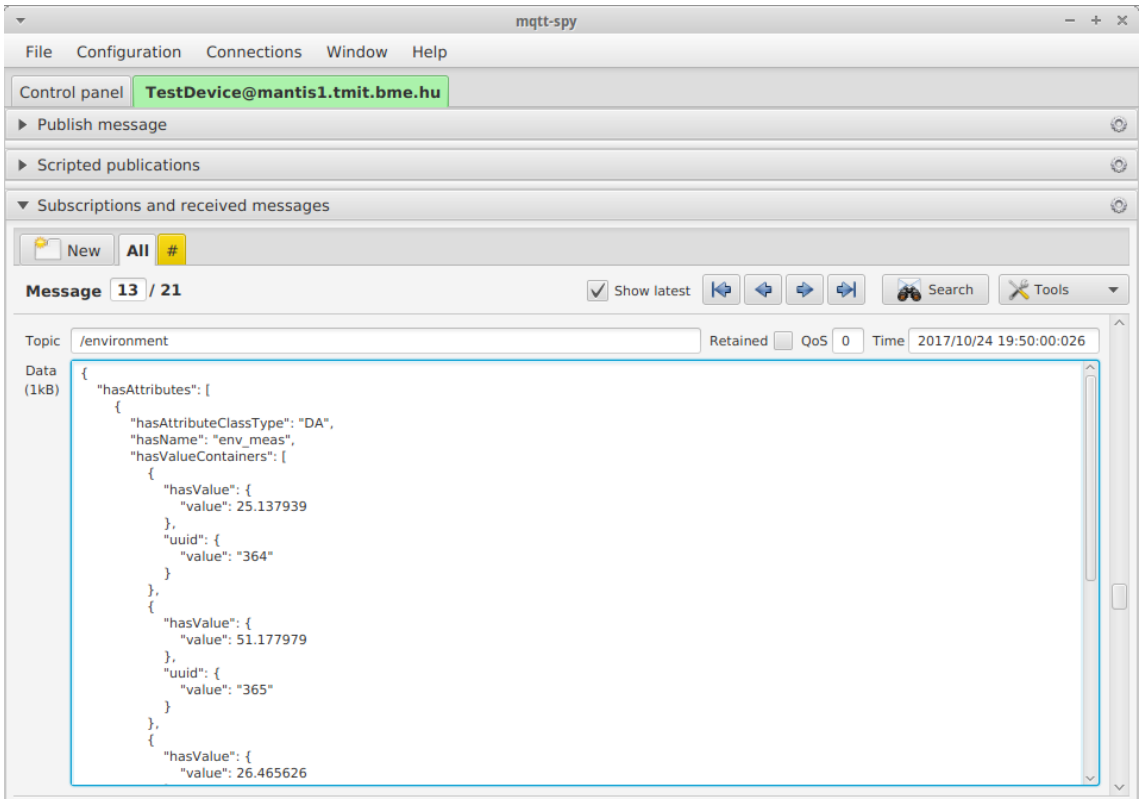

Figure 4.1. *The user interface of MQTT-Spy that shows an environment message*

then the moves will be getting slower and finally I'll leave it at a middle position between thresholds to indicate an error. When the message that contains the measurement of *"failed switching operation"* I'll send a message into the command topic that triggers the device to do a short measurement which shows the actual position of the points or in this case the potentiometer. Then I'll analyze, evaluate and summarize the results of test in as it's seen in the next section.

4.2 Measurements and proof of concept tests

4.2.1 Functional testing

During the test I triggered seven different measurements by sliding the potentiometer while I used the settings showed in Figure 3.8. The Table 4.1. contains the most important properties of measurement.

Table 4.1. *The most important properties of different simulated switching operations*

Number of measurement	Length/Duration[2]	Result of operation
1st Switching	2210 ms	Succeeded
2nd Switching	2590 ms	Succeeded
3rd Switching	3770 ms	Succeeded
4th Switching	5730 ms	Succeeded
5th Switching	8130 ms	Succeeded
6th Switching	11000 ms	Failed
7th Switching	994 ms	Commanded

As it's seen the first five measurement succeeded and only the last one failed which has length of 11000ms. That is the sum of maximum interval of a measurement and *"offset time"*, so the system limits the length of measurement and indicates an error message – that was not specified before thus I used a simple string that was published into the *warning* topic. As it's seen in Figure 4.2 the trigger events are working precisely as well as prefetch and postfetch time windows. The bolded line at value of 3800 and 500 signs when was the device triggered by threshold events and it can be seen each measurement cross that line 500 milliseconds after start, that is exactly is the value of *"offset time"* therefore I can state that the *trigger events of measurements*, the *timeout* function and providing *offset data* works well. It can be read from the chart that the offset time was too short in this case because at certain long measurement – when the moving was slow in the beginning – prefetch values don't contain the start position therefore should be set at a longer value. Since the lengths of measurements were different postfetch time windows started at different moments, but it can be clearly seen that the lengths of these are 500ms too – from crossing the threshold lines.

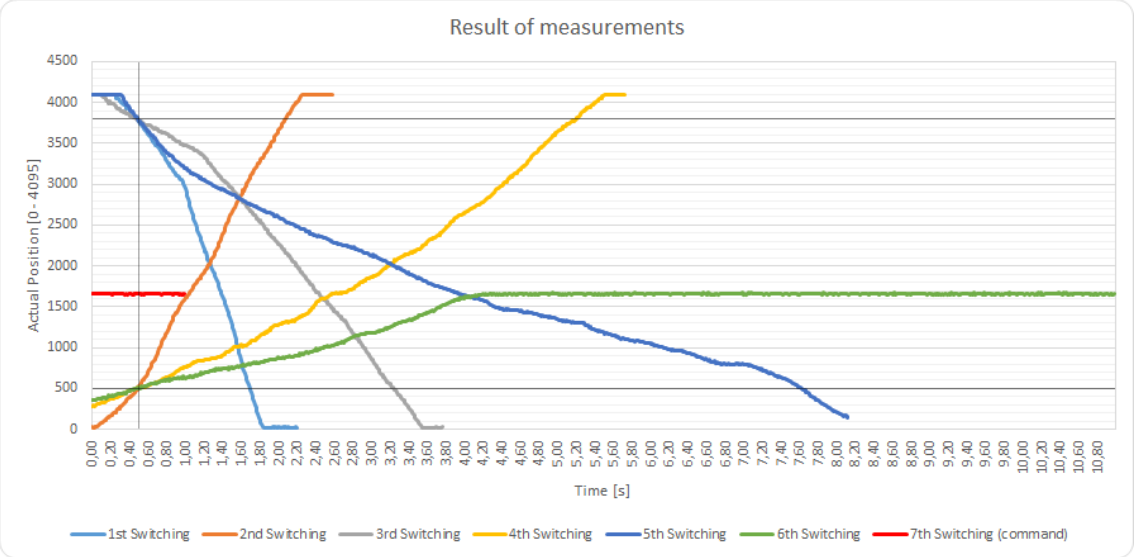

Figure 4.2. *The time course of different measurements as the result of the test – it can be found in Appendix F.5 in bigger size*

Now I focus on the failed measurement and the last one. As it's shown in the figure, the sixth switching operation goes from the low position to the approximately value of 1660 – the value of end position is fluctuating between 1650 – 1675. The accurate start position is unknown because of prefetch time window doesn't contain it due to really slow moving. If we take look at the last measurement that was triggered by a command message – that simulates operation of batch processor – we'll see it's still fluctuating around value of 1660 which means that the actual position hasn't changed since the last measurement. Another important thing that needs to be checked is length of this measurement which is 994ms. That is almost double *offset time*[3] i.e. it wasn't triggered by threshold levels consequently this function works well too as I described in the previous chapter.

I haven't mentioned environment measurement yet and it's because I did the measurements in a room and thus environment quantities don't change significantly. Nonetheless while I was testing

[2] According to the duration value of messages
[3] The reason of this related to the implementation but now I don't detail it

movement measurement the device sent environment message three times (at every 180 seconds) – one is shown in Figure 4.1 therefore I can say the device is capable of gathering all relevant data – ambient quantities, rail temperature and of course displacement – by reading sensors.

It's time to summarize what I did in this chapter so the result of measurements in short:

- I attempted to verify the system which I designed and implement. The verifying method is based on messaging but each critic point of operation was covered by this manner.
- Sending movement messages and the related tasks like starting measurement by trigger events, reading data from sensors and providing prefetch and postfetch data were tested. The moving of displacement sensor was simulated by sliding a potentiometer that is fairly good approximation.
- The device did all of the tasks which means that it sent messages in JSON format that contained the measured values of actual position. Based on the messages – that were represented in a common figure (Figure 4.2) – I can say that the device fulsfiled the criteria which were defined previously (e.g. triggering by threshold levels, max possible measurement time). Moreover, the device was triggered by a command message – that simulated request for data by batch processor – and did a special measurement of the actual position of "points" which was transmitted as message later.
- According to this list I can state that: **The implemented low-level device fulfills all criteria** and **it's capable of providing data for high-level processing units as a part of an information-driven maintenance system.**

4.2.2 Demonstration of high-level processing

In this short subsection I'll give you an example how high-level processing units would analyze gathered data. As it's known batch processor implements RCA methods by a dataflow-driven solution namely Petri-net. I made a simple Petri-net – similar to "Coffee-machine problem" – based on the actual measurements to explain how batch processor will execute fault detection and investigating root causes. The whole process can be found in Figure 4.3 where I used regular notations except *dotted circles* which denote the further places of the process and *rings* that represent input data provided by low-level device *after* batch processor triggered it.

In the start state there is a token in "Failure Detected" place which means batch processor so RCA was triggered by a failed measurement. Now RCA can require additional data by triggering low-level device to confirm the failure is still existing. After firing transaction one token is created in the place named "Actual Failure". Now RCA can trigger low-level device again – or in this case the previous triggered (commanded) message contains the relevant dataset to specify the type of failure. The result of the RCA in this case that the engine of switch failed because the actual position of points is far from threshold level there it's probably not a strange object between points and stock blades caused the failure but the malfunction of switch engine.

Since a required amount of data isn't available I can't demonstrate estimating RUL with the actual processing units however, in the future these type of data will be usable to establish a RUL prediction model in which humidity, rail temperature and ambient temperature can play the roles of covariates. For example, a possible – simple – hazard function would be: $\lambda(t) = \lambda_0(t) \cdot exp(hum \cdot z_1 + ambTemp \cdot z_2 + railTemp \cdot z_3)$ where z_i are covariates and $\lambda_0(t)$ can be estimated from the gathered data. Since the stream processor does calculation on the data and refine regression model constantly the estimated RUL is up-to-date in every case.

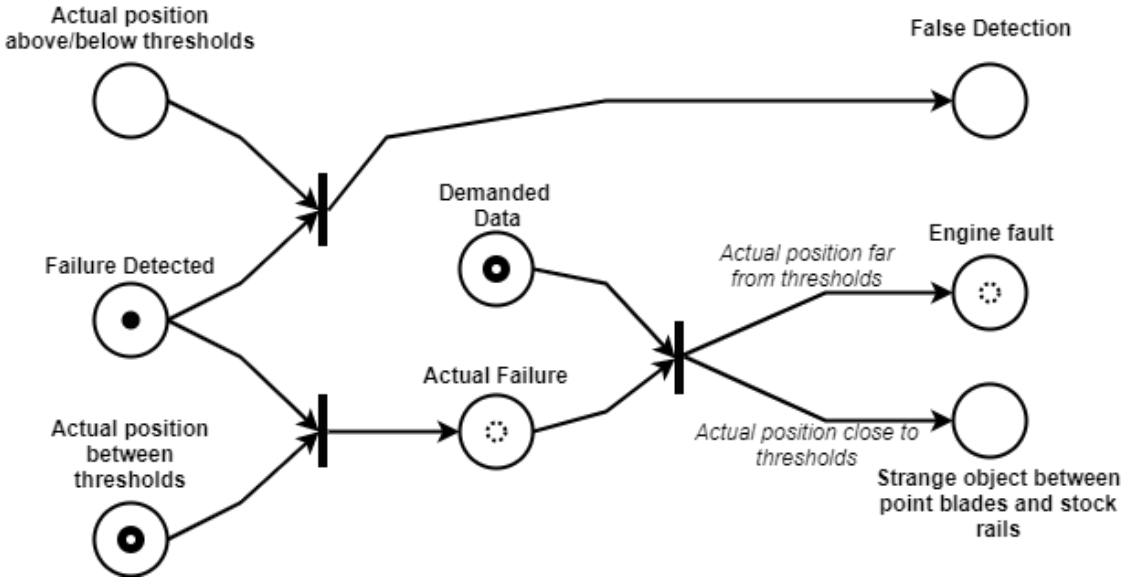

Figure 4.3. *The time course of different measurement*

Chapter 5

Conclusion

To conclude, in this paper I discussed a concept of reforming industrial maintenance by using cyber-physical systems given emphasis on the case of railroad switches and presented my work included from designing a low-level device throughout the implementation up to verifying it.

In the first place I showed the disadvantages of current maintenance approaches, the result problems in the case of railroad switches and how another approaches offer a solution to turn performing maintenance tasks into an effective and cost-efficient method. Important subareas of maintenance like diagnostics and prognostics were introduced generally in order to lay down the theoretical basics of a condition based maintenance and management system.

An overall concept and architecture of a cyber-physical system were presented and I discussed how it can fulfil the expectations and requirements of the use case which were defined previously. Then, certain parts of architecture were detailed as well as the possible implementations. The appropriates methods

In the major part of this work the main design principles were founded which is highly related to the actual physical system and defined requirements. These include choosing relevant quantities to measure, event detection methods, data gathering cycles and information providing – which consists of message format and communication protocol. An implementation of the low-level device and its most important parts and tasks were presented.

Finally, I attempted to verify the finished subsystem by simulating the operation of a real switch. Based on the results it has shown the implemented low-level device is capable of detecting trigger events and failures, providing gathered data in a predetermined format to high-level processing units and receive and execute commands sent by them.

Acknowledgement

I'd like to thank my supervisor István Moldován for all of his guidance, advices and mainly for the great conversations that led me to write this paper.

Bibliography

[1] *ISO-13374: Condition monitoring and diagnostics of machines - Data processing, communication and presentation*.

[2] *MIMOSA, OSA-CBM: Open System Architecture for Condition-Based Maintenance*.

[3] *Hadoop distributed file system*, 2017.

[4] *Storm stream processor*, 2017.

[5] Basil Cooper. Points, locks & bolts. *Rail Enthusiast*, February 1984.

[6] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society*, 4(2):187–220, 1972.

[7] Larry D. Dell, Paul F. Wilson, and Gaylord F. Anderson. *Root Cause Analysis: A Tool for Total Quality Management*. ASQC Quality Press, 1993.

[8] E. C. Fitch. *Proactive Maintenance for Mechanical Systems*. Elsevier Science Publishers Ltd, 1992.

[9] James C. Fitch. Proactive maintenance can yield more than a 10-fold savings over conventional predictive/preventive maintenance programs. Indaba, South Africa, 1992.

[10] James C. Fitch and Holly J. Borden. *Profitable Condition Monitoring*, chapter Interpreting Contaminant Analysis Trends Into a Proactive and Predictive Maintenance Strategy. Kluwer Academic Publishers, 1993.

[11] Csaba Hegedűs. Interoperability and data processing solutions for current industrial iot challenges. https://tdk.bme.hu/VIK/Halozatok2/Egyuttmukodesi-es-adatfeldolgozasi-megoldasok, 2017.

[12] E. Jantunen, U. Gorostegi, U. Zurutuza, F. Larrinaga, M. Albano, G. Di Orio4, P. Malo, and C. Hegedűs. The way cyber physical systems will revolutionise maintenance. 30th Conference on Condition Monitoring and Diagnostic Engineering Management, 2017.

[13] Erkki Jantunen, Urko Zurutuza, Luis Lino Ferreira, and Pal Varga. Optimising maintenance: What are the expectations for cyber physical systems. Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC) 3rd International Workshop, April 2016.

[14] Kálmán Kirilly. Personal interview, 2017.

[15] Dhananjay Kumar and Bengt Klefsjö. Proportional hazards model: a review. *Reliability and Engineering System Safety*, 44:177–188, 1994.

[16] Lin Ma, Melinda Hodkiewicz, and Joanna Sikorska. Prognostic modeling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, December 2011.

[17] István Moldován and Pál Varga. Integration of service-level monitoring with fault management for end-to-end multi-provider ethernet services. *IEEE Transactions on Network and Serivce Management*, 4(1), June 2007.

[18] British Board of Trade. Requirements in regard to the opening of railways, 1892.

[19] Zoltán Umlauf. Adatfeldolgozási módszerek és megoldások kiberfizikai rendszerek proaktív karbantartásához. https://tdk.bme.hu/VIK/Jelfeld/Adatfeldolgozasi-modszerek-es-megoldasok, 2017.

[20] Pál Varga. *Service Assurance Methods and Metrics for Packet Switched Networks*. PhD thesis, Budapest University of Technology and Economics, 2010.

Appendices

F.1 Petri net representation of the “coffee machine” problem

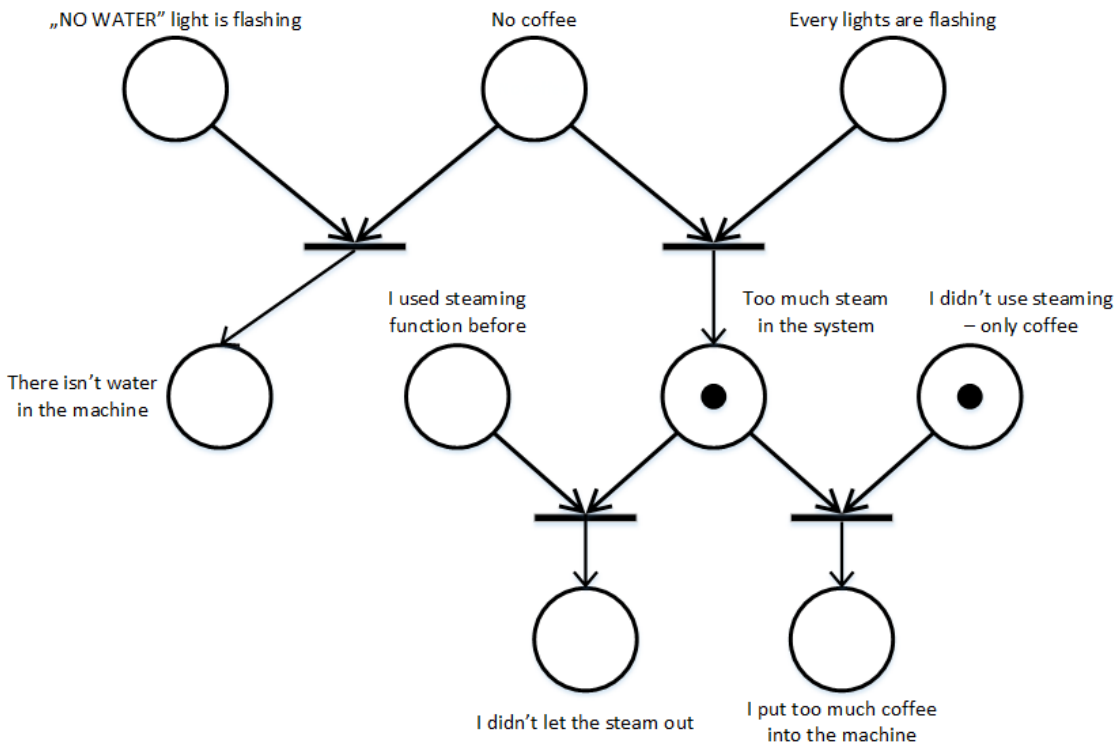

Figure F.1.1. *A simple problem solving with RCA and Petri net – middle state*

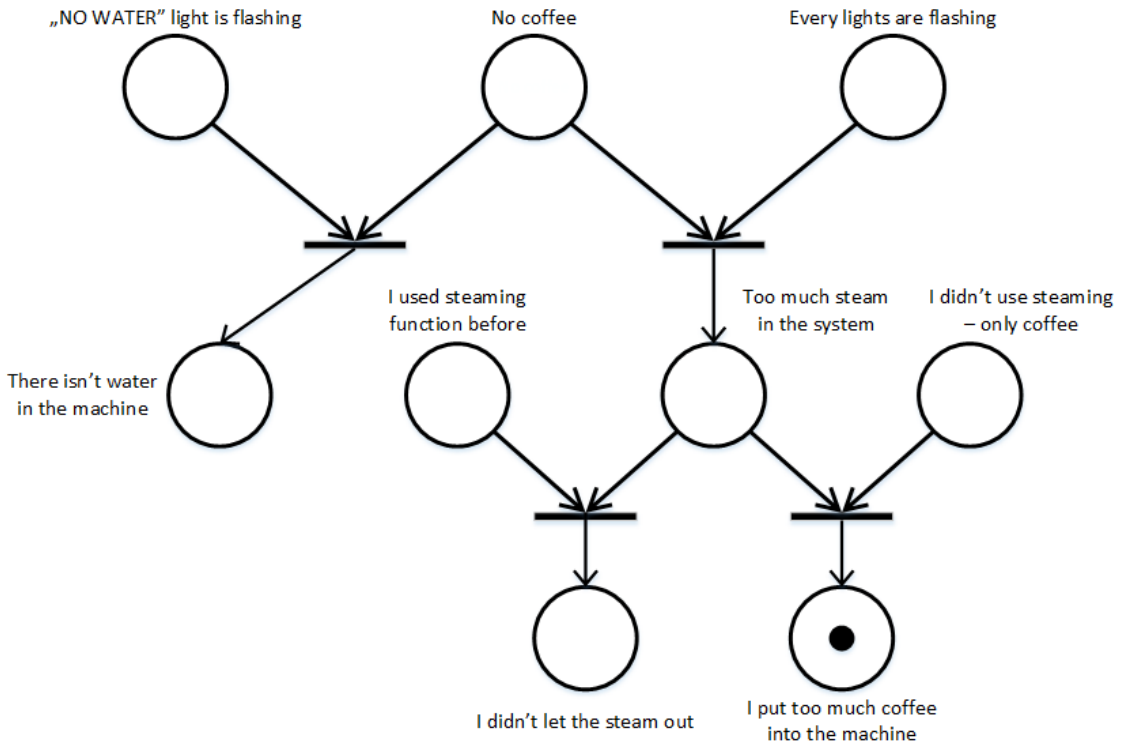

Figure F.1.2. *A simple problem solving with RCA and Petri net – final state*

F.2 RUL prediction models

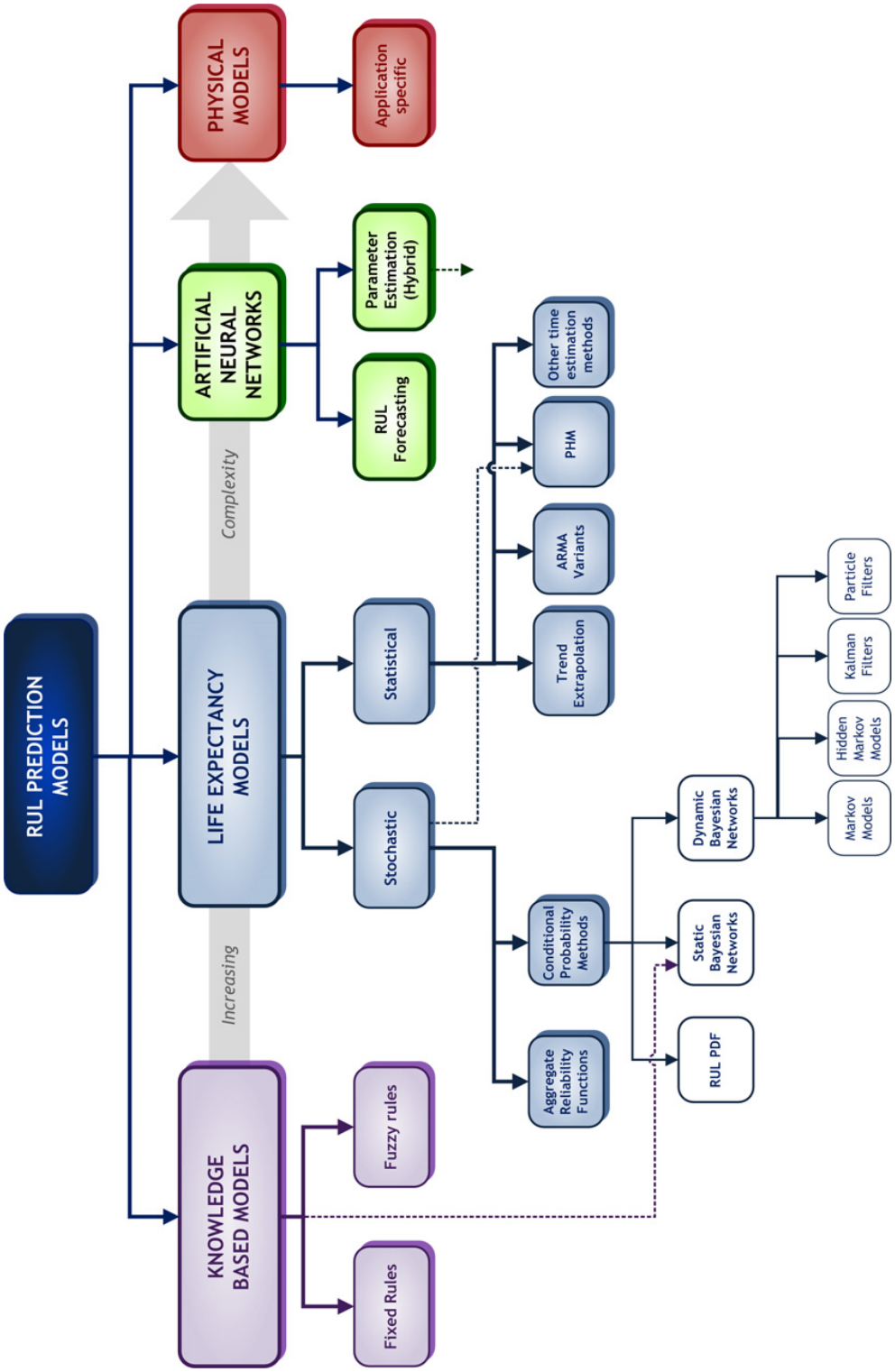

Figure F.2.1. *Different RUL prediction models classified by complexity [16]*

F.3 Mathematical proof of the relation between hazard rate and statistical functions

To calculate the hazard rate, I'll show it's related to the probability density function and survival or reliability function [6]. We assume **t** is a continuous stochastic variable and $F(t)$ is the cumulative distribution function. We have to use some definitions too, where $F(t+\Delta t) - F(t) = P(t < T < t+\Delta t)$, the reliability function is $R(t) = 1 - F(t) = P(T > t)$ and density function is $f(t) = F'(t)$. Now we can formulate the hazard rate in a mathematical way:

$$h(t) = \lim_{\Delta t \to 0+} \frac{P(T < t+\Delta t | T \geq t)}{\Delta t} \tag{F.3.1}$$

Now we use conditional probability defined by Kolmogorov: $P(A|B) = \frac{P(A \cap B)}{P(B)}$.

$$\lim_{\Delta t \to 0+} \frac{P(T < t+\Delta t | T \geq t)}{\Delta t} = \lim_{\Delta t \to 0+} \frac{P(t \leq T < t+\Delta t)}{P(T \geq t)\Delta t} \tag{F.3.2}$$

We use the definition of the reliability function – $R(t)$ – and the cumulative distribution function – $F(t)$ – to simplify the expression.

$$\lim_{\Delta t \to 0+} \frac{P(t \leq T < t+\Delta t)}{P(T \geq t)\Delta t} = \frac{1}{R(t)} \lim_{\Delta t \to 0+} \frac{F(t+\Delta t) - F(t)}{\Delta t} \tag{F.3.3}$$

It can be seen, the expression above contains the definition of derivation, that is:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

We use the defintion of derivation and density function – f(t) – to get a simple form of hazard rate.

$$\frac{1}{R(t)} \lim_{\Delta t \to 0+} \frac{F(t+\Delta t) - F(t)}{\Delta t} = \frac{F'(t)}{R(t)} = \frac{f(t)}{R(t)} = h(t) \tag{F.3.4}$$

$$h(t) = \frac{f(t)}{R(t)}, \text{ where } R(t) \neq 0 \tag{F.3.5}$$

This is the final form of hazard rate, that shows its relation to density function and the reliability function. The result of this proof was used in section 2.3.3, but instead of $h(t)$ I use $\lambda(t)$, because of the frequent usage of exponential function as a positive functional term.

F.4 A picture of the very first prototype of the low level device

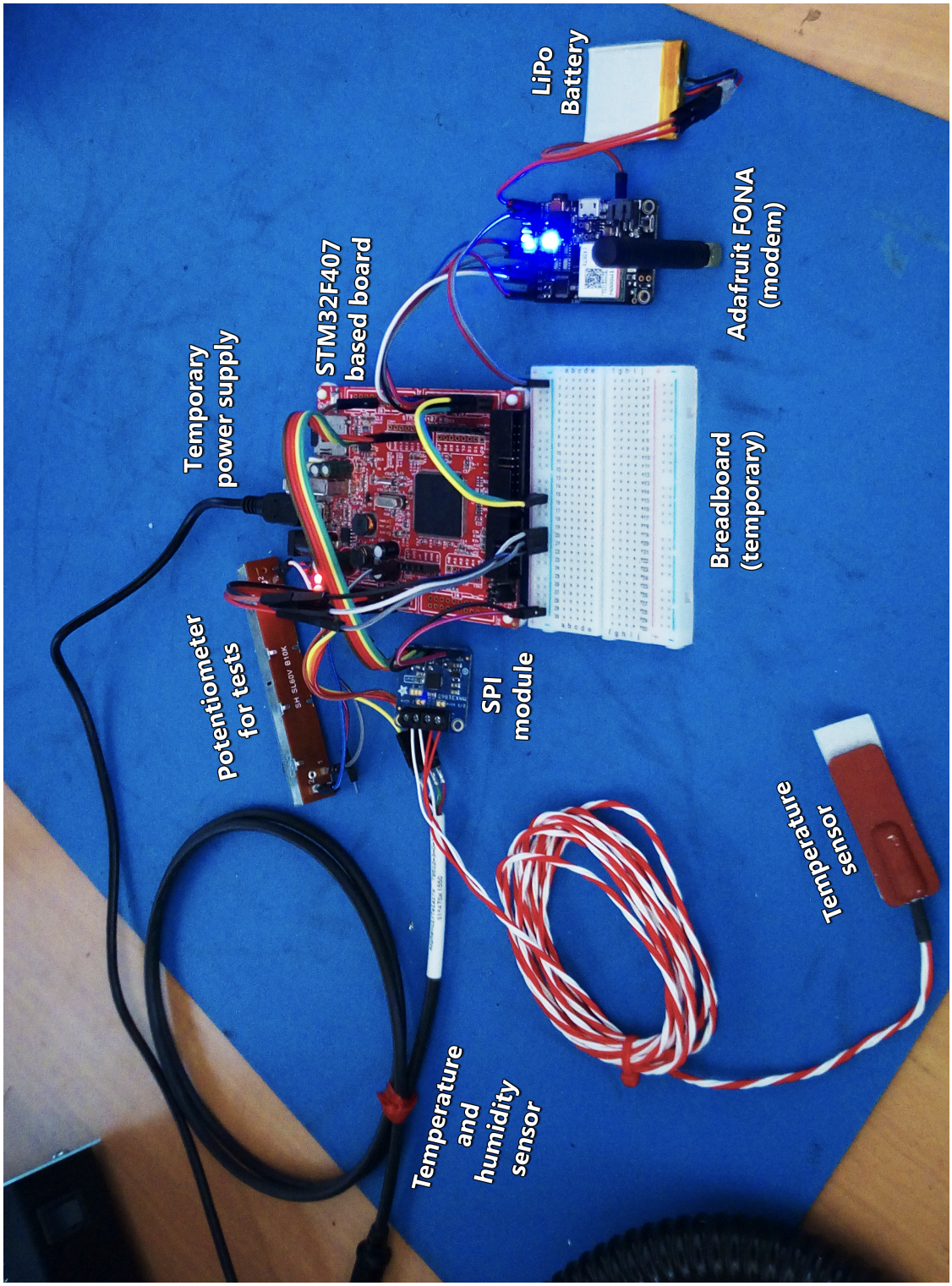

Figure F.4.1. *A very first prototype of the system with additional temporary parts in action*

F.5 Results of measurements

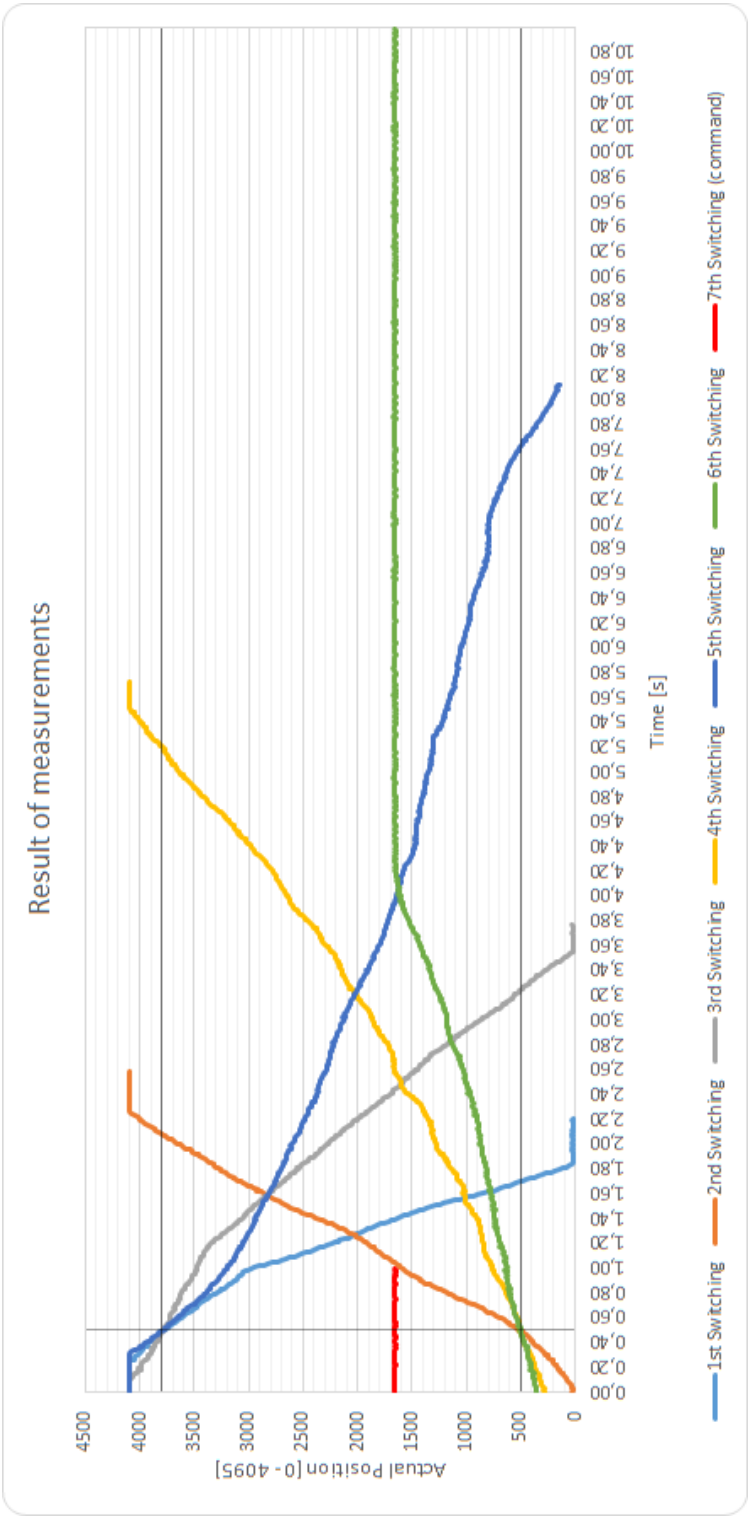

Figure F.5.1. *The time course of different measurements as the result of the test*