Richard Bjetak, BSc. BSc.

Data Analysis of Embedded Systems

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Mechanical Engineering

submitted to

Graz University of Technology

Supervisor

Ass.-Prof. Dipl-Ing. Dr.techn. Nikolaus Furian

Institute of Engineering and Business Informatics
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Siegfried Vössner

Graz, December 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Acknowledgement

First, I would like to thank my thesis advisors Dr. Konrad Diwold and Dipl. Ing. Amer Kajmaković for their great support and their collegiality. Furthermore, I would also like to thank Prof. Furian and Dr. Neubacher from the Institute of Engineering and Business Informatics for supervising this thesis.

Moreover, I want to thank Prof. Hörmann from the Institute of Statistics for his time and his expertise.

Also, my thanks go to my partner Stefanie for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Finally, I want to express my very profound gratitude to my dad and his unconditional support throughout all the years. Without his help this accomplishment would not have been possible.

Thank you.

Abstract

This thesis investigates the potential usage of already available but still unused time series data of an industrial embedded system. The given experimental setup consists of a programmable logic controller and its I/O-modules that are capable of measuring the temperature of their processor. The goal is to enhance the given setup with self-diagnostic capabilities. Therefore, the automation device is augmented with characteristics of self-awareness and contextual-awareness. On the one hand this is realized by developing an online change point detection method that is able to identify changes in the temperature values. On the other hand a linear regression model is conducted to make conclusions about the temperature in the environment. Moreover, ideas for a peer-to-peer comparison for individual I/O-modules are discussed. Following several approaches like the "roadmap to smart manufacturing" and the "IoT layer scheme" the idea is to provide the necessary analytical tools to create new services that enable process and product innovations.

Contents

List of Figures

List of Tables

1 Introduction

The ongoing data-driven transformation in manufacturing industry leads to an ever increasing amount of real time data. The progress to a smart manufacturing environment that makes use of this data can be decomposed into three phases:

- phase 1: data integration and contextualisation
- phase 2: simulation, modelling and analytics
- phase 3: process and product innovation

While the first phase is about making the data available and easily accessible, the next phase deals with processing the data to create manufacturing intelligence and to support decision-making processes within an organization. Only when having completed the first two phases, it is possible to create new services that lead to process and product innovations [1].

To achieve this, manufacturing facilities not only have to provide the infrastructure necessary to manage such an amount of data in terms of computation power, storage and latency, but also have to possess the analytical techniques to retrieve knowledge out of these datasets [2].

Therefore, cloud solutions offer a quickly deployable and scalable solution as they provide almost unlimited resources and a wide range of services to process the data. However, they are unable to meet the requirements of low latency, location awareness and mobility support. Especially when dealing with delay sensitive applications like safety related services, low latency is a problem. In this case, edge computing offers a suitable solution by extending cloud resources and services within an edge network. Edge computing describes a set of micro data centers that offer real-time processing and a drastic reduction of the data that needs to be sent to the cloud [3].

In order to develop the necessary analytical techniques, this thesis investigates the potential usage of already available but still unused time series data of an embedded system. With the help of low-cost Internet of Things (IoT) devices making up a small edge network the temperature data of a programmable logic controller and its Input/Output modules (I/O-modules) is analysed and processed. By developing an online change point detection algorithm as well as a temperature prediction model two features are presented that help to enhance an legacy automation device with characteristics of self-awareness and contextual-awareness.

The following section will give a deeper insight in the current situation of manufacturing systems linking the work presented in this thesis to the topic of cyber physical production systems (CPPS) and the creation of self-x services. Finally an overview of the structure of the work is given.

1.1 Current Situation

The advances in information technology such as Internet of Things, Cloud Computing, Big Data Analytics (BDA) and Artificial Intelligence (AI) often referred to as New IT have driven the manufacturing industry from an information-based age to a big data based age [4], [5]. This ongoing transformation has paved the way for a systematical deployment of Cyber-Physical Systems (CPS) in manufacturing environments [6]. In contrast to traditional embedded systems which are designed as stand-alone devices the focus of CPS lies on networking several devices [7], [8].

While embedded systems monitor and control the physical processes, the coupling of autonomous and cooperative elements and sub-systems results in a Cyber Physical Production System. This term describes an interconnected manufacturing system which can handle the actual operations in the physical world while simultaneously monitoring them in the cyber world. The foundation for linking these two worlds is provided by the steady increase of sensor and communication technologies leading to a system in which each process or piece of equipment makes event and status information available [9].

The real time transmission of information across the factory generates an enormous amount of data known as Big Data. According to Tao, Qi, Liu, *et al.* Big Data generated in manufacturing processes can be classified in: [4]

- management data (MES, ERP, CRM and PDM),
- equipment data collected with the help of IoT,
- user data from e-commerce platforms and social media,
- product data collected by smart products and
- public data collected by governments.

However, data is not useful unless it is put into context that can be understood by the right personnel or by other devices. Just connecting sensors to a machine will not create insights about any process. To become more competitive, manufacturers need to integrate advanced computing and Cyber Physical Systems [10].

The idea of smart manufacturing is to apply advanced analytics to Big Data, creating manufacturing intelligence in various areas such as production, supply chain, maintenance and diagnosis and quality management [1]. The potential benefits of manufacturing intelligence are manifold. Besides improvements in operational efficiency, asset utilization and sustainability it supports decision making by providing insights into pattern, trends, areas of inefficiency and potential risks of processes [11].

Many agendas such as “Industry 4.0” and “Made in China 2025” have put their focus on achieving a higher degree of intelligence in manufacturing [5]. A major focus of these agendas lies on the self-diagnosis capabilities for complex and distributed Cyber Physical Production Systems including detection of anomalies, suboptimal energy consumptions, error causes or wear [12].

That is exactly where this thesis tries to set in. Already available data of distributed automation devices shall be used to extend the possible self-diagnosis capabilities of existing production systems. Therefore, time series of temperature data of a PLC and its I/O-modules were made accessible. Doing so, it is possible to integrate a former embedded system into a cyber physical production system that is now able two make use of the measured values and provides the base for a variety of new services.

In general, expectations towards CPPS are high, for example: robustness, self-organization, self-maintenance, self-repair, safety, remote diagnosis real-time control, predictability, efficiency and many more. To fulfil those expectations, several research challenges related to CPPS have risen. One of them, is the topic of "Context adaptive and autonomous systems" with the goal to create methods for comprehensive, continuous context awareness for self-awareness in terms of knowledge about own situation, status and options for action [13].

Many embedded systems, especially when safety, security or both is crucial, benefit from some sort of sense of itself and its environment, and can improve robustness and sensibility of its own behaviour [14].

This includes the safety related automation devices that were provided in the experimental setup for this thesis. Therefore, the mentioned system is enhanced with characteristics of self-awareness and contextual-awareness.

According to the "hierarchy of self-x properties" introduced by Salehie and Tahvildari [15] (shown in Figure 1.1) both characteristics can be seen as the base level for any other self-x property in the major level under the umbrella of self-adaptiveness [15].

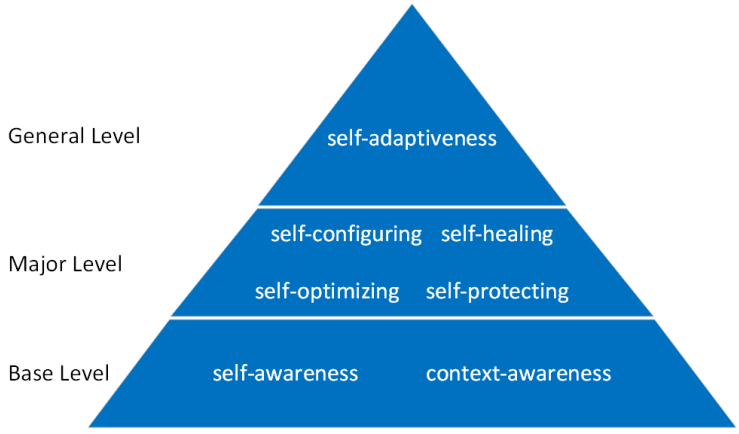


Figure 1.1: Hierarchy of the self-x properties, according to [15].

In case of this thesis, this is realized by implementing an online change point detection algorithm capable to monitor the own temperature state of the modules as well as a temperature prediction model to make conclusions

about the external environment. Both features create new possibilities within the subsystem itself but also within the cyber physical production system that can now make use of these extended diagnosis features.

This approach can also be interpreted using the "IoT layer scheme" of Weinberger, Bilgeri, and Fleisch presented in their paper "IoT business models in an industrial context IoT" as illustrated in figure 1.2 [16]. Both, the change point detection feature as well as the temperature prediction model represent a part of the data analytics layer. Together with the three other base layers of the model they form the prerequisite to create digital services in layer number five.

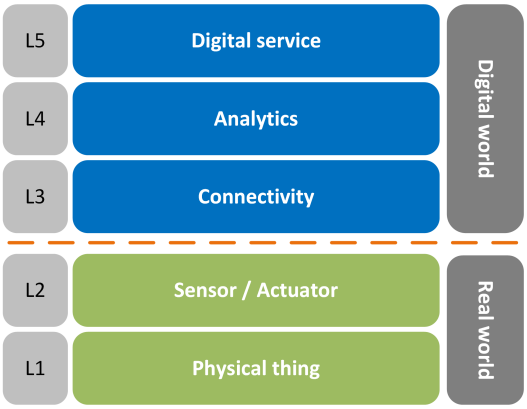


Figure 1.2: IoT layer scheme, according to [16].

To sum it up, the integration of stand-alone embedded systems into a cyber physical production system as well as the development of new services based on analysis techniques create a variety of possibilities to improve manufacturing intelligence and self-diagnosis capabilities. In the following chapter an example for such a service is given underlining the motivation of this thesis.

1.2 Structure

The thesis is structured as follows: Chapter 2 recaps the motivation of the thesis and provides an example for a safety related service of a PLC and its

I/O-modules. Furthermore, an overview of the related work is presented including the topic of data analysis in industrial environments, temperature prediction and change point detection.

Due to the nature of the data, chapter 3 is dedicated to the topic of time series. After introducing the characteristics of the term, different interpretations of time series analysis are presented.

In chapter 4 a detailed explanation of the characteristics of change point detection algorithms is given, differentiating between online and offline as well as supervised and unsupervised methods. This is followed by a description of two log-likelihood ratio based methods, namely control chart and CUSUM. Furthermore, different performance measurements are explained, including not only classification and regression metrics but also efficiency and computational complexity comparisons.

In chapter 5, the software and the hardware of the experimental setup are presented. The given setup is explained showing a schematic model of the used devices. Furthermore, the physical modelling of such a system is briefly presented to revise the basics of heat transfer and thermal conductance. Having these basics in mind parts of the collected data are shown and further discussed.

Chapter 6 presents the developed online change point detection algorithm and the temperature prediction model. This includes a detailed explanation of both methods and a discussion of its shortcomings. The final results are then presented at the end of each subchapter. To support a better understanding of both methods the chapter is facilitated with many figures that illustrate their functionality and performance. Furthermore, ideas for similarity comparison among the I/O-modules are discussed.

Finally, the conclusion of the work and an outlook on a possible end-to-end safety application are presented in chapter 7 and chapter 8.

2 Motivation & Related Work

The idea for this thesis was established at the COMET competence center for Excellent Technologies Pro^{2}Future at the Technical University of Graz together with the Research & Development department of SIEMENS Graz focused on system & safety engineering for automation devices. Therefore, a Siemens PLC system with Fail-safe I/O-modules that are capable of measuring their own processor temperature was provided. The goal was to investigate the possible usage of that yet unused time series data.

In a first step this required to gain a deeper understanding of the characteristics of time series data itself. As more and more devices offer the possibility to retrieve data, it is crucial to understand the statistical behaviour of time series for further analyses.

According to Gartner, the amount of connected devices in use making up the Internet of things will increase from 14.2 billion in 2019 to about 25 billion in 2021 [17]. Their wide range of sensors such as motion sensors, environmental sensors or position sensors create an enormous amount of data in form of time series that need to be handled [18].

Therefore, the following chapter is concerned with the topic of time series analysis.

As a next step, the capability to detect changes within the measurements as well as the possibility to draw conclusions about the environment were identified as core analysis features for further services.

One example for such a service is the ability of I/O-modules to contextualize themselves and their surrounding and detect fault situations. This includes failures due to external influences such as temperature extrema like fire or frost but also faults due to internal influences like hardware or software errors. Especially when comparing the behaviour of one module to its

own history or to its peers it is possible to detect failures due to internal influences, that would have been missed otherwise.

Given the broad scope of the topics related to the implementation of data analytics and services in embedded systems, the related work is manifold. In this thesis the following three main areas shall be outlined:

The first area focuses on the topic of future production and product technologies from a general point of view. As investigated by researchers of Pro^{2}Future in the vision paper "Data Analytics for Industrial Process Improvement A Vision Paper" by Thalmann, Mangler, Schreck, *et al.* [19], manufacturers are nowadays capable to collect data from different sources of manufacturing equipment but not yet ready to use analytics that go beyond interpreting historical performance. Hence, the data is not yet used as a base for advanced analytical techniques such as machine learning and predictive algorithms.

As already mentioned, this thesis is trying to fill this gap and therefore investigates possibilities to use the time series data of a programmable logic controller and its I/O-modules.

Another work that served as a base for this thesis is "Improving the Safety of Industrial Environments through Model-based Analytics on hidden Data Sources" by Kajmakovic, Zupanc, Mayer, *et al.* [20], also established at the competence center Pro^{2}Future. It focuses on the usage of data in industrial environments from a safety point of view. It categorizes different sources of potentially valuable data within a production environment and discusses its further use for industrial fail-safe applications. These categories include for example self-diagnostic data, hardware & software data but also data on people's conditions and behaviours.

The second area that is related to this thesis is based on the prediction of temperature making use of low-cost IoT equipment sensors. The idea to use the processor's temperature data to make conclusions about the environment derives from the paper "Estimating outdoor temperature from CPU temperature for IoT applications in agriculture" written by Krintz, Wolski, Golubovic, *et al.* [21]. In their work the authors made use of the CPU temperature from low cost, single-board computers to make conclusions about the outdoor temperature in an agricultural setting. Similar to the

approach in this thesis a linear regression model was used. Their results showed that, considering a frequent recalibration of the system, errors of less than 1.5 $^{\circ}F$ (aprox. 0.84 $^{\circ}C$) could be achieved.

Another approach with focus on predicting human's body temperature exposed to hot environment conditions is presented in the paper "Prediction of human core body temperature using non-invasive measurement methods" [22]. The authors developed a multi linear regression model based on six parameters that included the skin temperatures at three different locations of the body, the measurement of the heart rate and two heat flow sensors. According to the authors the root mean squared deviation was in the range of 0.28 to 0.34 $^{\circ}C$.

The third area of related work can be summed up to the topic of change point detection in time series. During the early research phase for this thesis, it became clear that the possibility for a system to realize changes in its measured values opens up a whole new set of possibilities for manufacturing systems. These include topics such as predictive maintenance or safety related applications. Already in 1999, Guralnik and Srivastava [23] presented a paper that mentioned the promising field of data mining in time series data in connection with event detection and change point detection for streaming data but also for batch data.

The following authors shall be mentioned before hand, as their review papers, works and books were particularly helpfull to develop a change point detection algorithm:

- Aminikhanghahi and Cook on the topic of change point detection methods [24]
- Basseville, Nikiforov, *et al.* on the topic of detection of abrupt changes [25]
- Fu on the topic of time series data mining [26]
- Zeileis, Leisch, Hornik, *et al.* on the topic of structural change and Cumulated Sum [27]

While working on this thesis, parts of the results were published in the proceedings of the 9th international Conference on the Internet of Things in Bilbao in October 2019. The work was presented as a poster with the title: "Retrofit: Creating Awareness in Embedded Systems - A Usecase for PLCs" [28].

3 Time Series Data Analysis

The aim of this chapter is to give an introduction on the topic of data analysis in time series. After discussing the term time series an overview on the goals of data analysis is given. This includes representation and indexing, similarity, segmentation and data mining. Furthermore, the methods for online change point detection are discussed in detail.

3.1 Characteristics of Time Series

In general time series can be defined as sequences of data points spaced at strictly increasing times. Each point represents a set of values within a labeled set of dimensions that is paired with a time stamp. The time intervals between two successive points can either be uniform or changing [29].

Depending on the process, two adjacent points in time often show strong correlations. This means that one value at time t depends on the past values x_{t-1} , x_{t-2}, ... Therefore, many time series cannot be seen as independent and identically distributed (*iid*) sets of data points. For this reason the application of many conventional statistical methods is not suitable [30].

Unlike classical time-field data sets, time series track changes by appending the new information to a data set and not by updating the old one. This allows to draw conclusions about the history, detect abrupt changes in the present and make predictions about the future of the time series [31].

Furthermore, time series always have to be seen as a whole itself and not just as an individual numerical field. Time series have three main characteristics: large data size, high dimensionality and necessity to update continuously [26].

This creates an enormous amount of data that needs to be handled in an effective way when recorded or queried. As traditional databases are not designed for such a purpose, time series databases were developed. They allow a high throughput ingestion, compression and real-time querying of data. To ensure the storage of such a large scale of data, time series databases are made to compact the data automatically. This is realized by downsampling the data and applying data retention policies. Downsampling means that the raw data is kept at a high precision for a limited amount of time and later on stored in a lower-precision or summarized way. Depending on the retention policy, the data is kept for longer periods or discarded at a certain point of time [32].

3.2 Time Series Analysis

Since the topic of time series analysis has gained a lot of attention during the last decades, the goal of time series analysis has shifted. From a classical point of view with the objective of finding an ideal mathematical forecasting model, the objective has changed to a more general approach emphasizing the discovery of meaningful information. Moreover, with upcoming machine learning methods the objective has again broadened to almost automatically retrieve information out of time series data.

3.2.1 "Classical" Time Series Analysis

A classical definition for time series analysis is often stated as finding a mathematical model that describes a given sample of data and allows to make statements about the future [30].

As mentioned before, adjacent observations in time series are typically dependent from each other. Therefore, time series analysis is mainly concerned with analysing the sort of dependency [33].

The method of analysis depends very much on the nature of the time series. In general there exist two separate but not necessarily mutually exclusive approaches. The first one is called time domain approach, focusing on the

lagged relationships between the data points (e.g.: AutoRegressive-Moving Average [ARMA]). The second one is called frequency domain investigating the cycles of a time series (e.g. fourier analysis) [30].

However, there are also methods that combine the time and the frequency domain (e.g. wavelets) [34].

For a classical exploratory data analysis within the time domain, various statistical measures are used to define how a time series behaves. However, to be able to apply statistical analysis methods such as the ARMA-model it is necessary for a time series to fulfill the concept of weak stationarity. Weak stationarity is present if the mean, the variance and the covariances of the data are not dependent over time. Only if this is the case, it is possible to achieve a meaningful statistical analysis of the time series [30].

3.2.2 Time Series Analysis in the Sense of "Data Mining"

The definition of time series analysis in the previous chapter, neglects a big part of analyses often referred to as time series data mining. It includes several approaches to retrieve "hidden knowledge" of time series and can be categorized into the following tasks [26].

Time Series Representation and Indexing

To reduce the sheer amount of data, the task representation implicates reducing the dimension of the original data. Many different approaches exist that try to aggregate time series without loosing its core characteristics and preserving the salient points. One approach is to represent segments of the time series in a compressed way. This includes methods such as sampling, Piecewise Aggregate Approximation (PAA), segmented sum of variation and bit level approximation. Another approach is to approximate time series by using straight lines such as linear interpolation and linear regression. Furthermore, the dimension of a time series can be reduced preserving important points by discarding minor fluctuations and keeping major minima and maxima. Anomaly detection for example can be seen as an application of this approach. A last approach is to represent time series

in a symbolic form. One popular method is called Symbolic Aggregate Approximation (SAX) that represents segments of the time series by a symbol. Approaches that represent the time series in a transformation domain are the already mentioned Discrete Fourier Transformation (DFT) and Discrete Wavelet Transformation (DWT). An important task that is strongly connected to the representation of time series is indexing certain points or subsequences in an efficient way. It requires an approximation technique that produces an indexable representation. Hence, its efficiency depends strongly on the used approximation technique [26].

Similarity Measure and Motif Detection

In contrast to classical discrete data sets, the similarity for time series is not measured based on an exact match. Because time series have to be seen as a whole itself, the similarity measure is carried out in an approximate manner. Two different approaches can be differentiated, whole sequence matching and subsequence matching. Whole sequence matching focuses on the whole length of the data set. One approach is to evaluate the similarity by applying a distance function to the transformed representation of the time series. Common methods are the Euclidean distance and time warping. The aim of subsequence matching is to find a certain part within the whole time series that matches a known subset of data points. This again can be done working in both domains, time and frequency using methods such as SAX and DWT [26].

The goal of motif detection is to find frequent unknown patterns in a time series. Motifs are defined as recurring patterns, frequent trends, approximately repeated sequences or shapes within a time series. Common problems in subsequence matching and motif detection are ill-known patterns. These show approximate similarity, but vary in size and time and might have different phases or distortions. Moreover, motif detection can get complicated if the patterns are overlay with noise [35].

Segmentation and Change Point Detection

Segmentation can either be seen as a preprocessing step for other tasks such as representing and subsequence detection or as an analysis tool itself. The goal is to discretize a time series at certain cutting points creating segments without splitting meaningful patterns. Many methods focus on detecting special events and perceptually important points (PIP) that indicate changes. This results in a subarea of time series analysis called change point detection discussed in detail in chapter 4. Other approaches focus on finding the number of intervals, detecting cyclic periodicity or minimizing variances within segments [35].

3.2.3 Machine Learning Approach for Time Series Analyses

Recently, machine learning has gained attention as an alternative to statistical time series analysis methods. Especially in the field of forecasting, including one-step forecasting and multi-step forecasting, a variety of machine learning methods were applied. While one-step forecasting draws conclusion about the next data point multi-step forecasting performs a prediction for the next certain amount of data points [36].

Investigating a variety of machine learning methods it can be shown that traditional statistical methods still outperform upcoming machine learning algorithms. As a result machine learning methods lack behind in terms of performance and accuracy [37].

Another task that makes use of the developments in machine learning is anomaly detection in time series. With the help of Long Short Term Memory (LSTM) networks a model was trained detecting faults in time series such as an electrocardiogram and a multi-sensor engine dataset [38].

Furthermore, deep learning algorithms are used for time series classification in different domains such as human activity recognition and sleep stage identification [39].

It is important to mention that the presented applications are just a small selection of examples using machine learning for time series analysis. The

capability to learn automatically from given time series is definitely a promising field. However, looking at the level of performance and accuracy, machine learning will most probably not substitute the existing classical statistical and data mining approaches. Instead it will integrate itself as just another tool that can be used to retrieve knowledge out of time series data.

4 Change Point Detection

This chapter discusses the characteristics of change points and change point detection methods. After defining both terms the differences to the related but not identical topic of anomaly detection are lined out. Following this, an overview on the different problem statements for both online and offline change point detection is given. Furthermore, supervised as well as unsupervised methods are described and a list of change point detection methods is presented.

Two methods are explained in detail. Both were used to implement the online unsupervised change point detection algorithm for the I/O-modules of the PLC. The Shewhart control chart method is often used in quality engineering and the CUSUM method has its origin in the offline detection of change points.

When choosing an algorithm to detect changes, many tradeoffs have to be considered. Therefore, in the last subchapter performance measurements and evaluation indicators are discussed.

4.1 Characteristics of Change Point Detection

A change point can be defined as a point in time at which properties of time series data changes [40].

To be more specific, change points represent unexpected, structural changes in mean, variances or covariance representing a non stationary phenomena. Furthermore, change points are described as abrupt variations, with respect to the sampling period. Whereas a trend can be defined as an estimation of gradual departure of past values from a time series [41].

This however does not necessarily imply large magnitudes of changes, nor fast changes. On the contrary, early warning systems often have to detect small deviations to avoid catastrophic consequences [25].

In contrast to change points, anomalies are defined as points in time where the behaviour of the system is unusual and significantly different from past behaviour but does not imply a transition between two states [42]. Depending on the domain of application, anomalies are also often referred to as outliers, discordants and observations just to name a few [43].

As mentioned in the previous chapter, change point detection can be seen as a subarea of segmentation of time series. The aim is to find those points where a transition between two states occurs [35].

In the industrial sector, change point detection is used for robot control, predictive maintenance or fault detection [41], [44]. Besides that, it is also used in other fields such as medical condition monitoring, climate change, speech and image analysis and human activity analysis [24].

The following subchapters give an overview on the different categories of change point detection methods.

4.1.1 Offline vs. Online Change Point Detection

The detection can either take place in real time, or as a retrospective detection. The first one is also known as incremental, sequential or online detection, while the latter one is referred to as batch or offline detection [45], [23], [25].

Online Change Point Detection

Online change point detection algorithms aim to detect changes in data streams. In principle, new data points are tested whether they still belong to the current interval or if they are the very first point of a new interval, marking a change point [46].

Several different approaches exist that focus on online data processing. These can be statistical methods or machine learning based algorithms [25].

In general indicating change points in online data is more complicated than offline detection because of not having the whole data available at once. As storing the whole historical data stream for high frequency measurements is often not possible, so called, "reference window" approaches are commonly used. Depending on the approach, historical data is either lost completely or limited to a given size of window [46].

Offline Change Point Detection

Depending on the situation, the following problem statements can be investigated using an offline change point detection:

1. **Hypothesis Testing**
   Testing whether there is a change for a parameter θ within a batch of data. This does not require the estimation of the change time t_0. For this kind of problem the tradeoff between sensitivity and robustness of the algorithm has to be kept in mind. Change points shall be detected without reacting to noise effects [25].

2. **Single Change Point Detection**
   Finding the right change time t_0, assuming there is exactly one change point within the data set. Depending on the use case, the estimation algorithm can possibly use information about θ_0 and θ_1 [25].

3. **Multiple Change Point Detection with Given Amount of Intervals**
   Finding the right change times t_0, ... ,t_{N-1} for a series of change points given a predefined number of N intervals [25].

4. **Multiple Change Point Detection without Given Amount of Intervals**
   Splitting a given set of data into an uncertain amount of segments [25].

4.1.2 Supervised vs. Unsupervised Change Point Detection

To detect non stationary phenomena such as change points or anomalies both, supervised and unsupervised methods exist. While supervised methods use labeled data with flagged points, the unsupervised ones explore the statistical properties by statistical algorithms from unlabeled data. Supervised methods rely a lot on the predefined observations and are not suitable for an application with unpredictable events. Therefore, mainly unsupervised methods are used [47].

Figure 4.1 shows a categorization between supervised and unsupervised change point detection methods as presented by Aminikhanghahi and Cook in the review paper "A Survey of Methods for Time Series Change Point Detection" [24].

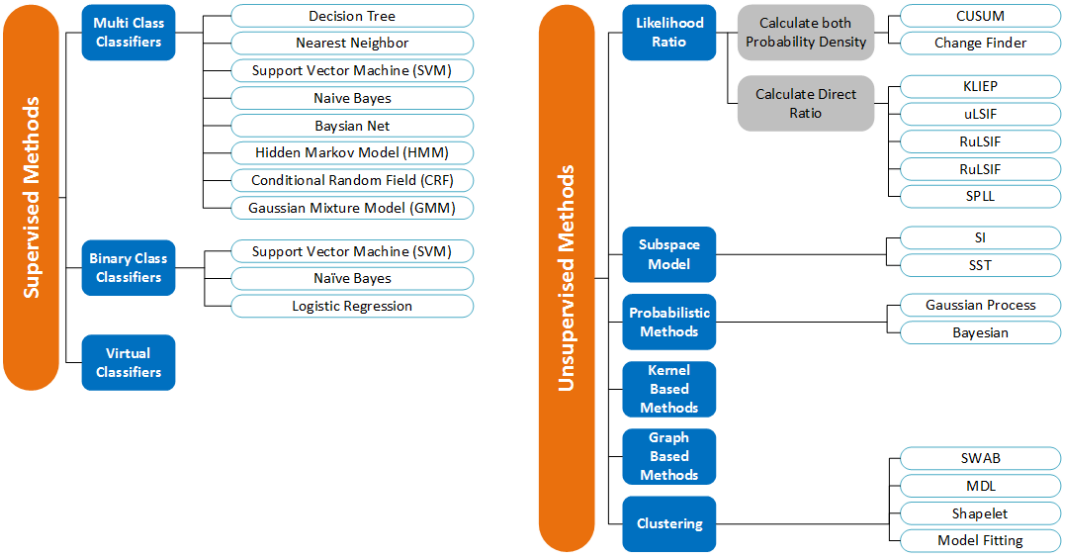


Figure 4.1: Change point detection methods, according to [24].

As shown in the figure there exists a variety of different approaches for change point detection. Because of the previously mentioned reason of unpredicted events this thesis focuses on unsupervised methods. After a screening period a log-likelihood ratio approach was chosen. It provides the base for the methods of control charts and CUSUM that were used

to develop the change point detection algorithm. The log-likelihood ratio approach itself and both methods are explained in detail in the following chapter. It is the prerequisite to understand how the implemented algorithm works and to understand its strengths and weaknesses.

4.2 Log-Likelihood Ratio Methods for Change Point Detection

This chapter presents the necessary theory about the concept of log-likelihood ratio in change point detection. As already stated it just represents one out of many other approaches. Log-likelihood ratio provides the framework for the statistical methods control charts and CUSUM. Both methods are well known, but do not have their roots in the area of online change point detection. While the control chart is actually an instrument for process control, CUSUM is mainly used for offline change point detection. The idea of combining both methods was influenced by the book *Detection of abrupt changes: theory and application* written by Basseville, Nikiforov, *et al.* [25].

4.2.1 Log-Likelihood Ratio

Two different sub approaches of likelihood ratio approaches can be differentiated. On the one hand, the ratio can be computed by calculating both probability densities, one for the distribution before the change point and one for a distribution after the change point. This requires not only knowledge about the parameter of the current probability density but also an estimation of the parameters of the upcoming one, after the change point. On the other hand the likelihood ratio can be computed directly without taking the intermediate step of calculating each probability density separately. While the first method depends on knowing the parameters of the distributions (mean μ and standard deviation σ), the second one does not necessarily imply knowing the characteristics of the distributions [40].

Representative algorithms for a separate calculation of probability density are: [24]

- Cumulated sum (CUSUM), accumulating deviations relative to a specified target of incoming measurements and
- Change Finder based on an auto regression model.

Approaches based on direct ratio calculation are: [24]

- Kullback-Leibler importance estimation procedure (KLIEP) estimating the density ratio based on Kullback-Leibler divergence.
- Unconstrained Least-Squares Importance Fitting (uLSIF) fitting the Pearson divergence as a dissimilarity measure to the true ratio.
- Relative uLSIF (RuLSIF) and
- Semi-Parametric Log-Likelihood Change Detector (SPLL) based on Kullback-Leibler statistics.

In general the log-likelihood ratio can be seen as a cost function. The purpose of such a function is to provide a measurement on whether a change point belongs to the current or to a new segment [48]. Both, the control chart and the CUSUM are based on log-likelihood ratio values. First the control chart is explained. After that a detailed insight into the method of CUSUM is given.

4.2.2 Control Chart

Control charts are a very common tool within statistical process control. The method dates back to the year 1924 when Walter A Shewhart first used it to improve the reliability of transmission systems. The aim of control charts is to reduce the variance of processes. Therefore, the tool helps to identify and distinguish between the two causes of variation namely ”special causes” and ”common causes” [49]. Two types of control chart schemes can be differentiated: While the one-sided scheme detects deviations of parameters only in one direction, the two-sided scheme detects changes in either direction [50].

For a given process that is observed for changes in both directions, the measurements are plotted and embedded in a chart with three horizontal lines as shown in Figure 4.2. These are the lower control limit, the center line and the upper control limit. If the data points are within the limits a

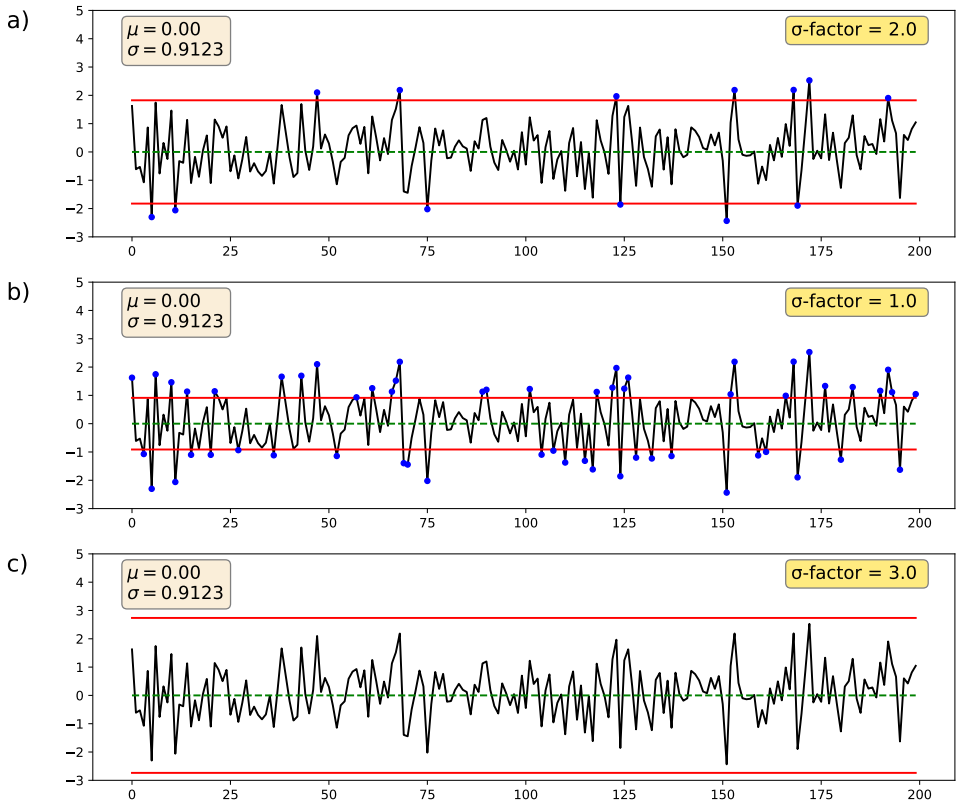


Figure 4.2: Two-sided control chart with different upper and lower control limits
a) control limit distance of 2σ
b) control limit distance of σ
c) control limit distance of 3σ

process is defined as within control . To be precise, control limits should not be confused with specification limits. While the control limits determine whether a process is in control or out of control, the specification limits define whether a product's quality is acceptable or if the product has to be discarded [51].

Depending on the distance of the upper and lower control limit to the center line, the control chart will be more or less sensible to variations within the measured values. On the one hand, if the limits are put too close the risk is that common variation cause the process to appear out of control (see figure 4.2 b)). On the other hand, if the tolerance is too big special causes might

not be detected (see figure 4.2 c)) [51].

Measured values are often expected to be independent and identical distributed (*iid*). Independent means that the value of a measurement is not dependent on the value of a preceding data point. Mathematically this can be expressed as

$$Pr(B|A) = Pr(B) \tag{4.1}$$

This means that the probability of an event B given an event A in the past is equal to the probability of the event B without the information about A. If this equality holds true, then A and B are said to be independent. Identical distributed, on the other hand states that the underlying distribution of the values are the same at any point of time. Hence, the likelihood that a certain value is measured is equally likely at different points in time [52].

A common way to define the tolerance window for a control chart is to calculate the distribution of a data sample for a given process assuming a normal distribution. Only if the underlying data set does not contain anomalies or structural changes the mean μ and the standard deviation σ will truly represent the characteristics of the process. The mean value is used as an estimator for the center line and the control limits are often set two or three standard deviations away from the center line. As a matter of fact the probability for independent and identical distributed processes to be in control depends only on the set distance of the control limits to the center line and can be stated as follows: [51]

$$Pr(\mu - 1\sigma \leq X \geq \mu + 1\sigma) \approx 0.6827 \tag{4.2}$$
$$Pr(\mu - 2\sigma \leq X \geq \mu + 2\sigma) \approx 0.9545 \tag{4.3}$$
$$Pr(\mu - 3\sigma \leq X \geq \mu + 3\sigma) \approx 0.9973 \tag{4.4}$$

Taking the reciprocal of the probability to be out of control shows on average how often values appear out of the set boundaries due to common causes. This performance measurement of control charts is called Average Run Length (ARL) and can only be used for uncorrelated or independent measurements. Although the tolerance window might be quite big, there is still some probability left that values appear out of control without an

underlying special cause. For the three tolerance window the average run length are: [51]

$$\frac{1}{1 - Pr(\mu - 1\sigma \leq X \geq \mu + 1\sigma)} \approx 3.15 \tag{4.5}$$

$$\frac{1}{1 - Pr(\mu - 2\sigma \leq X \geq \mu + 2\sigma)} \approx 21.98 \tag{4.6}$$

$$\frac{1}{1 - Pr(\mu - 3\sigma \leq X \geq \mu + 3\sigma)} \approx 370.37 \tag{4.7}$$

For a defined control limit that is three sigma away from the center line, an independent and identically distributed process appears on average once every 370 data points out of control. Because of this natural variation within processes, performance measurements of algorithms such as sensitivity and robustness have to be kept in mind. These are going to be discussed in the following chapter.

As mentioned at the beginning of chapter 4, time series are often not considered as *iid*, and a normal distribution representing the likelihoods of the expected values of a data stream can not be assumed. That also implies that performance indicators such as ARL can not be applied.

4.2.3 CUSUM

The cumulated sum algorithm better known under its acronym CUSUM is a likelihood ratio based method to detect structural changes in stochastic processes. Back in 1954 when first proposed by Page in "Continuous Inspection Schemes" it was already presented as a tool *"to detect changes in the quality of the output from a continuous production process"* [50].

However, many other use cases from various scientific areas such as finance or socioeconomics apply CUSUM to look for structural changes within a time series [27].

The idea behind the first CUSUM test introduced by Brown, Durbin, and Evans [53], also referred to as recursive CUSUM or BDE, is to accumulate

the recursive residuals to test for a structural change of the regression coefficients in a linear regression relationship. Another CUSUM test, proposed by Ploberger and Krämer [54] in 1992, is based on ordinary least squares (OLS) residuals. When comparing both methods, BDE and OLS show different properties but neither variant is uniformly superior to the other.

Stable processes show the characteristic of a stationary time series. As explained in chapter 3, this means a sequence of data points that is independent and identically distributed around a constant mean without changes in variance and covariance.

When a stable process is given, the sum of its residuals due to natural variance can also be interpreted as the integral of a white noise Gaussian process. This is also known as Wiener process, which represents a special case of a Brownian motion [55].

Considering the assumption of independent and identical distributed data points, this sum tends to fluctuate around its mean value. Only if the property of stationarity does not hold anymore the accumulated sum will move away from its mean.

As mentioned in the very beginning the CUSUM method is considered to be a log-likelihood ratio method. For a given set of measurements, that represent a stable process the distribution parameters can easily be calculated. Assuming a normal distribution $f_0(x)$, the distribution parameter vector θ_0 consists of two components, the mean μ_0 and the standard deviation σ_0. In the case of a normal distribution these are already enough to compute the likelihood of each measurement. When testing for a change in mean the hypothesis H_0 states that the measurements are part of the distribution $f_0(x)$ and H_1 states that the measurements are not part of the given distribution $f_0(x)$. As explained in 4.1.1, two cases can be differentiated when testing for an unknown change point. Either there is or there is no knowledge about the parameter vector θ_1 of the other distribution $f_1(x)$ that can be used to calculate the likelihood. Especially in the case of online change point detection not much is known about the second distribution.

In order to make the log-likelihood ratio method applicable for an online detection the problem is solved by defining another distribution $f_1(x)$ to test H_0 on. Instead of just performing the hypothesis testing whether H_0 holds

true or not the log-likelihood ratio is calculated. Its result can be interpreted as whether a data point is more likely to be part of distribution $f_0(x)$ or the admittedly "user dependent" picked distribution $f_1(x)$. Once a decision is made what to test the distribution of the stable process against with, the log-likelihood ratio can be calculated as follows:

$$s_i = ln(\frac{P_{\theta_1(x_i)}}{P_{\theta_1(x_i)}}). \tag{4.8}$$

These log likelihoods are then summed up using

$$S_k = \sum_{n=1}^{k} s_i \tag{4.9}$$

Basseville, Nikiforov, *et al.* [25] propose a set of different derivations of CUSUM tests and give a detailed explanation of their implementation. One of them is a good example to explain the use of offline change point detection in a batch of data, the other one shows the implementation of an online change point detection as a Repeated Sequential Probability Ratio Test.

Offline change point detection test

In search of applicable change point detection algorithms in various scripting languages such as Python and R, one will quickly notice that most packages mainly provide functions for offline change point detection. This is probably due to the fact that given a batch of data the change point in a parameter can be found quite easily.

The following example helps to understand how such an algorithm works: [25]

Figure 4.3 a) shows a process that can easily be divided into two segments that differ in mean but have a similar variance.

Figure 4.3 b) presents the cumulated sum S_k. While the value is at first continuously drifting, a specific point can be identified when the sum

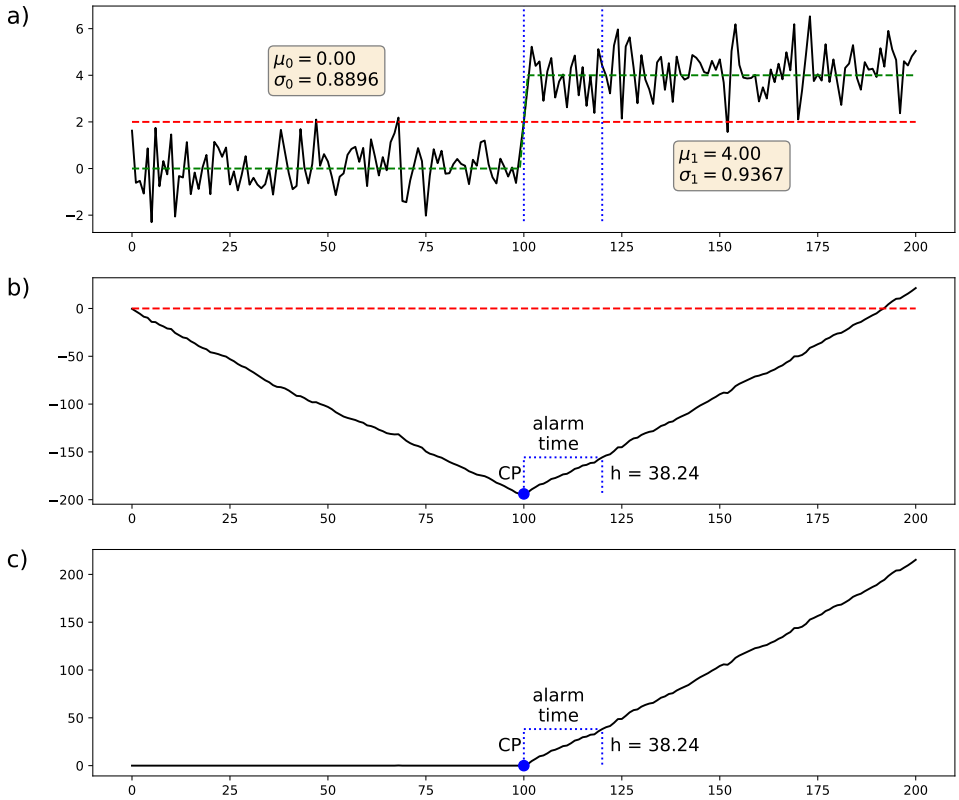


Figure 4.3: CUSUM: Offline change point detection, based on [25].
a) Process with an abrupt change in mean
b) Cumulated sum S_k
c) Goal function g_k with threshold h

is going up again. Once this change in the drift exceeds a certain value measured from the very minimum, the alarm for a change point is set. The time between the actual change point and the alarm can be interpreted as the alarm time that is necessary to differentiate between a natural variance and an actual change point. Hence, it can also be used as a parameter for the sensitivity of the algorithm.

Another way to show this is defining a decision function g_k as follows: [25]

$$g_k = S_k - m_k \geq h \tag{4.10}$$

where

$$m_k = \min_{1 \leq j \leq k} S_j \tag{4.11}$$

When plotting g_k (as shown in Figure 4.3 c)) it becomes clear that the CUSUM test is a simple comparison between S_k and an adaptive threshold $m_k + h$ [25].

Online change point detection test

As mentioned before, finding a change point in a data sequence is challenging because there is no information of the future data points. One way to still detect changes is to repeatedly test the following hypotheses: [25]

$H_0 : \theta = \theta_0$

$H_1 : \theta = \theta_1$

Again, log-likelihood ratios are cumulated and a basic decision rule d, also known as stopping rule, has to be defined by a simple thresholding scheme. Three cases can be differentiated: [25]

1. if $\epsilon \leq S_k \leq h$ the process continues.
2. if $S_k \leq \epsilon$ hypothesis H_0 is true
3. if $S_k \geq h$ hypothesis H_1 is true

In terms of the decision function this can also be expressed as: [25]

$$d = \begin{cases} 0 & \text{if } S_1^T \leq \epsilon \\ 1 & \text{if } S_1^T \geq h \end{cases} \tag{4.12}$$

where T is the time the lower threshold ϵ or the upper threshold h is exceeded, also called exit time defined as: [25]

$$T = T_{\epsilon,h} = min\{k : (S_1^k \geq h) \cup (S_1^k \leq \epsilon)\} \tag{4.13}$$

The thresholds have to be chosen depending on the use case, and ϵ is typically set to 0. A log-likelihood ratio of zero indicates that the likelihoods

are equal for one specific measurement. The decision function however depends on the cumulated sum of the log likelihoods. In other words, if the sum is between ϵ and h the process continues to sum up the log likelihood ratios (case 1). If the sum is smaller than ϵ it is more likely that the original distribution $f_0(x)$ is still valid (case 2). If the sum exceeds the threshold h there must have been a significant amount of points that are more likely to be part of the distribution $f_1(x)$ that it was tested against with (case 3).

The size of the window being $h - \epsilon$ defines how sensitive or how robust the algorithm is on changes. That can also be measured by observing the exit time. Again depending on the use case, short exit times might indicate that the boundaries are set too narrow.

4.3 Performance Measurements

The aim of defining performance measurements for classification methods is to identify indicators for an objective evaluation of algorithms. This includes either comparing different methods with each other or tuning the parameters for one specific algorithm.

Basseville, Nikiforov, *et al.* list five intuitive performance indicators for change detection algorithms:[25]

1. mean time between false alarms
2. probability of false detection
3. mean delay for detection
4. probability of nondetection
5. accuracy of the change time and magnitude estimates

To define such metrics the first step is to choose a classification model that maps the actual classes of the data points in a time series to its predicted classes. This can either be done using discrete class labels spanning a confusion matrix or using a continuous labelling approach calculating the probability of the class membership of data points [56].

However, a classification on its own is not enough as it neglects the fact that the difference in time between the detected change point and the actual

change point is also an important parameter of change point detection algorithms. To measure this timely performance regression metrics are used.

A detailed overview on discrete classification metrics is given that are based on the confusion matrix. After that, four common regression metrics are presented which focus on the timely difference between the occurrence and the detection of a change point.

4.3.1 Classification Metrics - Confusion Matrix

A confusion matrix, sometimes also referred to as contingency table, is an $n \times n$-matrix mapping the actual class against the predicted class. In case of a binary classification (as it is the case when testing between two distributions), the matrix looks as shown in Figure 4.4 [24].

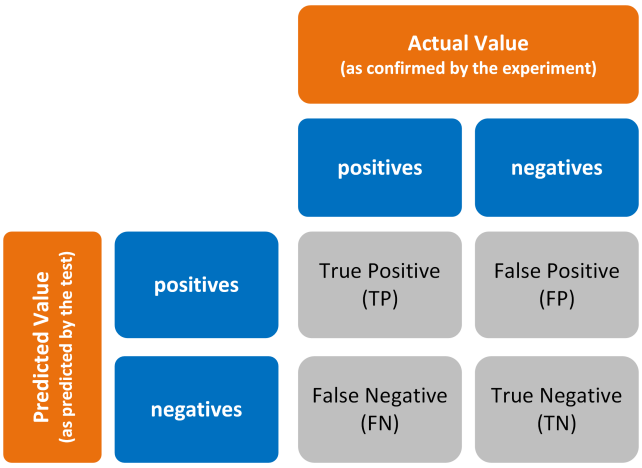


Figure 4.4: Confusion matrix, according to [24]

In the context of change point detection the four categories can be explained as follows: [24]

- True positives (TP) are defined as the number of correct positive predictions representing the amount of correctly detected change points.

- False negatives (FN) are defined as the number of incorrect negative predictions representing the amount of actual change points that were not detected.

- False positives (FP) are defined as the number of incorrect positive predictions representing the amount of wrongly detected change points.

- True negatives (TN) are the number of correct negative predictions representing the amount of points that were correctly classified as not being a change point.

Out of these four categories the following measures of performance can be calculated: [24]

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.14}$$

Also known as True Positive Rate the *Sensitivity* puts the amount of correctly detected change points in relation to all actual change points within a data set including the one that were not detected.

$$Specificity = \frac{TN}{FP + TN} \tag{4.15}$$

Also known as True Negative Rate the $Specificity$ puts all the points that did not show a change in relation to all points classified as such including those that were wrongly classified as change points.

$$PPV = \frac{TP}{TP + FP} \tag{4.16}$$

The Positive Predictive Value also known as *Precision* puts the correctly found change points in relation to all change points indicated, including the ones that were wrongly identified as such.

$$NPV = \frac{TN}{FN + TN} \tag{4.17}$$

The Negative Predictive Value sets all correctly classified points without any change in relation to the total of points without any change.

The following measurements based on the confusion matrix are explained in the context of change point detection as done in the review paper "A Survey of Methods for Time Series Change Point Detection" written by: [24]

$$Accuracy = \frac{TP + TN}{(TP + FP) + (FN + TN)} \tag{4.18}$$

$$ErrorRate = 1 - Accuracy = \frac{FP + FN}{(TP + FP) + (FN + TN)} \tag{4.19}$$

Accuracy and *ErrorRate* give a first insight on how the algorithm works but are not suitable in case of change point detection problems. The reason is that the ratio of changes to total data is small. However, both types of classification errors FP and FN are considered as equally important. As there is no information on the source of error or the distribution among the different classes both measurements are ineffective for evaluating the performance in a class-imbalanced data set.

$$\begin{aligned} \textit{G-mean} &= \sqrt{Sensitivity \times Specificity} \\ &= \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{FP + TN}} \end{aligned} \tag{4.20}$$

Due to the imbalanced class distribution in change point detection problems the *G-mean* is commonly used as an indicator. It is based on the indicators *Sensitivity* and $Specificity$.

$$\begin{aligned} \textit{F-measure} &= \frac{(1 + \beta)^2 \times Sensitivity \times Precision}{(\beta^2 \times Sensitivity) + Precision} \\ &= \frac{(1 + \beta)^2 \times \frac{TP}{TP+FN} \times \frac{TP}{TP+FP}}{(\beta^2 \times \frac{TP}{TP+FN}) + \frac{TP}{TP+FP}} \end{aligned} \tag{4.21}$$

The *F-measure* also known as f*-score* or $f1$ *score*, combines *Precision* and *Sensitivity* using a weighting factor β. It can be interpreted as the effectiveness of a change point detection algorithm.

Another performance measure of algorithms is the Receiver Operating Characteristics (ROC) curve as depicted in figure 4.5. The ROC curve is a technique for visualizing, organizing and selecting classifiers based on their performance. It is used to depict the tradeoff between true positives and false positive rates of classifiers. Since its first use in machine learning by Spackman in 1989, it has become a common tool to compare different algorithms, especially because simple classification accuracy is often a poor metric for measuring the performance [56].

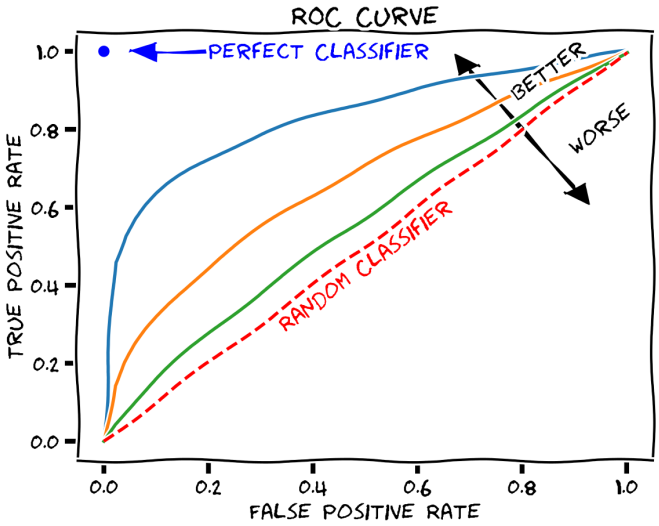


Figure 4.5: ROC curve [58]

The ROC graph is a two dimensional space in which the TP- and the FP-rate are plotted on the X-axis respectively on the Y-axis. The true positive and the false positive rate are defined as shown above. As can be easily seen the TP_{Rate} is similar to the previously explained measurement *Sensitivity*.

$$TP_{Rate} = \frac{TP}{TP + FN} \tag{4.22}$$

$$FP_{Rate} = \frac{FP}{FP + TN} \tag{4.23}$$

The ROC curve allows several interpretations as discussed in the paper "An

introduction to ROC analysis" written by Fawcett [56]. Some characteristic points of this graph shall be explained:

- (0,0): A classifier that never issues any positive classification and hence has a true positive rate as well as a false positive rate of 0.
- (1,1): A classifier that is unconditionally issuing positive classifications and therefore has a true positive rate as well as a false positive rate of 1.
- (0,1): The perfect classifier that detects all true positives without making any mistakes.

The diagonal line describes the performance of a random classifier. If for example 30% of the points are randomly issued as a positive classification the classifier is located at (0.3,0.3). Classifiers that perform on the right side of the random classifier are actually performing worse than a random selection of positives would. The goal is to achieve a classification that is located on the left side of the ROC random classifier diagonal.

In general it can be said that the farther up left the point of the classifier the better the algorithm. Hence, having a high TP_{Rate} but a low FP_{Rate}. Points on the very left side of an ROC make positive classifications only with strong evidence, so there are only a few false positive errors. This however comes with the tradeoff of having low true positive rates as well. Points further to the random classifier line make positive classifications already with weak evidence. As a result they often have high false positive rates [56].

This tradeoff in classifying algorithms is sometimes referred to as "robustness" [25].

In case of change point detection a classifier closer to the random classifier line might indicate too many points, while a classifier on the left hand side risks to neglect a change that has already taken place. In the end, it is again a question of the given use case. When safety or security is a topic, a more conservative approach might be the right choice. However, if the severity of not detecting a change point is low, but the costs for getting a system run after an alarm are high, the choice will be to go for an algorithm on the very left side on the ROC.

Another technique to evaluate an algorithm is looking at the Area Under the Curve (AUC). As a high TP_{Rate} and a low FP_{Rate} is desirable, the closer

the AUC is to 1 the better. One more measurement connected with the ROC is the Equal Error Rate (EER) which describes the point where the FP_{Rate} and the FN_{Rate} are equal. This point can also be constructed geometrically by drawing a diagonal from the upper left side to the lower right side and indicating the intersection of the ROC curve. The lower the EER value the better [24].

4.3.2 Regression Metrics

Although change point detection can be seen as a segmentation task and hence a classification problem, using classification metrics only is not enough as they neglect time performance. Therefore, regression metrics can be used that focus on the difference in time between the actual change point and its detection. The goal of any online change point detection algorithm is to detect a structural change as close to real time as possible while keeping the number of false alarms very small.

To measure the time performance, four common metrics are presented, where k is the number of change points within a subset, $T_{predicted}$ is the predicted time of a change point and T_{actual} is the time where the actual change point took place [59], [60]:

Mean Signed Difference:
$$MSD = \frac{\sum_{i=1}^{k}(T_{predicted} - T_{actual})}{k} \tag{4.24}$$

Mean Absolute Error:
$$MAE = \frac{\sum_{i=1}^{k}|T_{predicted} - T_{actual}|}{k} \tag{4.25}$$

Mean Squared Error:
$$MSE = \frac{\sum_{i=1}^{k}(T_{predicted} - T_{actual})^2}{k} \tag{4.26}$$

Root Mean Squared Error:
$$RMSE = \sqrt{\frac{\sum_{i=1}^{k}(T_{predicted} - T_{actual})^2}{k}} \tag{4.27}$$

4.3.3 Efficiency and Computational Complexity

Another way to compare algorithms is to analyse its computational costs and its scalability for a big amount of data to be processed.

Two main metrics can be differentiated: While "build time" is the time required to train a classifier on a given data set, the "classification speed" is the time necessary to classify a data point [61].

However, the metric of "Big O" is by far the most common way to compare the efficiency of algorithms. It describes the time and space complexity of an algorithm by indicating its maximum time or space dimension $f(n)$ as a function of the number of input variables n. These can be ordered by its tendency to grow with an increasing number of n. A scalable algorithm should therefore be at maximum of type $O(n\,log(n))$. Algorithms of higher order such as $O(n^2)$ or even higher lead to unacceptable execution times or might exceed the available memory space [62].

5 Experimental Setup

The aim of this chapter is to give an overview on the devices and software used to create a temperature aware programmable logic controller. At the beginning, the main architecture is presented. This is followed by a detailed explanation of the used devices. After that, a physical model used in problems of thermal transfer and thermal conduction is presented. The motivation behind this is to follow a holistic approach to gain a deeper understanding of the data produced by the experimental setup. Furthermore, the procedure of collecting the data is explained showing a data set of 16 days with characteristic changes due to external influences.

5.1 Architecture Overview

The experimental setup shown in Figure 5.1 consists of a Programmable Logic Controller *SIMATIC S7/1500* (PLC) with fail-safe Input/Output-modules (F-I/O-modules), a single board computer *Raspberry PI 3 Model B+* (RPi), and an *ATmega328 microcontroller Arduino UNO*.

Each of the PLC's I/O-modules has a processor that is capable, among other safety features, to measure its own temperature T_P. The temperature's values from all I/O-modules are transferred to the RPi via OPC UA using a local router. Furthermore, three external thermal sensors measuring the environmental temperature T_E are connected to the Arduino's input ports and their readings are transferred to the RPi via serial communication.

The RPi runs a linux operating system and the time series database application InfluxDB that receives and processes the time-stamped information from the PLC and the Arduino. The operating system of the single board computer can easily be accessed via HDMI, or from any workstation within

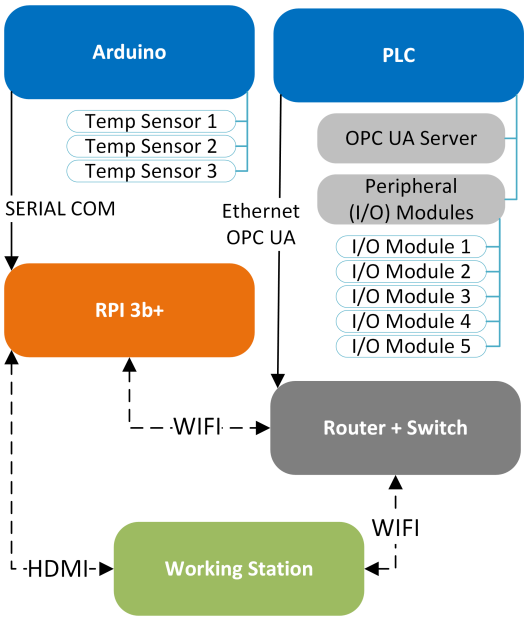


Figure 5.1: Experimental setup

the Network via its WiFi and a Virtual Network Computing desktop sharing system. This allows to setup the software infrastructure on the single board computer and script the handling of the data stream.

5.2 Hardware & Software of Subsystems

This chapter gives an overview on the hardware and the software of the mentioned subsystems and presents its main characteristics.

5.2.1 Programmable Logic Controller

Programmable logic controllers are currently the most widely used hardware platform in control technology. Such a controller is structured like a computer. Its function is typically stored as a cyclic processed program. In the simplest case, the programmable logic controller consists of a power

supply (PS) a central processing unit (CPU), some digital or analogue input and output modules and an internal bus system. Controls are available as modular and as compact systems for different requirement levels [63].

The requirements for industrial control computers are very demanding and can be summarized as follows: [64]

- survive in harsh environment (temperature, humidity, dirt etc.)
- capable of dealing with bit form digital Input/Output signals at the usual voltages encountered in industry (24 VDC to 240 VAC)
- easy understandable programming language, that can easily be changed
- monitor the plant operation and assisting in fault detection
- operate sufficiently fast for real-time control
- consider safety aspects

The programmable logic controller (PLC) is the core of the experimental setup. It consists of a Central processing Unit (CPU) *SIMATIC S7-1500 CPU 1516F-3 PN/PD* which is powerd by a *SIMATIC S7-1500/ET 200MP Load power supply module PM190W 120/230 VAC*.

The CPU can be seen as the heart of the PLC as it executes the user program and connects the controller with other automation components. The signal modules or I/O-modules that form the interface between the controller and an arbitrary process can be used in a centralized as well as in a decentralized configuration [65].

For a decentralized configuration, an interface module *ET 200SP, Interface-Modul IM 155-6PN HF* in combination with a bus adapter *ET 200SP, bus adapter BA 2xRJ45* is needed. They provide the backplane of the decentralized configuration with energy and allow a communication via PROFINET BUS between the CPU and the interface module. The backplane consists of five base units, at which the first base unit provides an additional supply voltage terminal that receives its energy from the interface module. The base units are serially connected to the interface module and completed with a server module. Each base unit provides a slot to connect an I/O-module. In the described experimental setup five fail-safe I/O-modules were connected to the base units.

In comparison to standard modules, the fail-safe signal modules differ in terms of their internal two channel structure. Two integrated processors monitor each other, automatically test the I/O-circuits and force the fail-safe signal module into safe state if a fault or an error has been detected. Fail-safe automation systems are usually used in plants which are subject to more stringent safety standards. They control the processes and force the plant into a safe state after a shutdown [64].

Unlike in any other automatization setup, in our experimental setup no sensors nor actuators were connected to the signal modules. The focus of this thesis lies on the extraction and interpretation of the internal data of the I/O-modules. The use of a specialized firmware allows access to internal data of the modules such as temperature of the processor, supply voltage and current. Moreover, the programmable logic controller is connected to a gateway *SIMATIC IOT2040* that, however, is not used in our setup. The gateway is able to collect, process and connect the data of the production environment and the office IT via cloud services independent of the manufacturer of the automatization components, its language and protocols [66].

The programming software necessary to create the control program follows the basic software architecture as defined in the international standard IEC 61131 for programmable logic controllers. It consists of five different programming languages, out of which two are textual, two are graphical and one is a higher level sequence language with textual and graphic elements: [67]

- Structured text (ST), textual
- Instruction list (IL), textual
- Ladder diagram (LD), graphical
- Function block diagram (FBD), graphical
- Sequential function chart (SFC), textual and graphical

The control program used in this setup was written using the SIEMENS Software solution SIMATIC STEP 7 TIA-Portal . The main task of the program is to check the temperature of the processor of the signal modules.

The totally integrated automation Portal (TIA Portal) offers a standard interface to integrate the components of any automation project such as

distributed I/O-modules. The data is then sent to the RPi that serves as a client using a local router via OPC UA.

OPC UA is a freely available, interoperability communication standard focused on the reliable exchange of data between industrial equipment and systems. One of its main advantages is its independence of the platform that ensures a seamless flow of information among devices from multiple vendors [68].

5.2.2 Microcontroller

Microcontrollers have all required building blocks of a microprocessor system integrated on one chip. Depending on the type this includes: [69]

- Central processing unit (CPU)
- Oscillatory circuits
- System control
- Interrupt Controls
- Input/Output ports
- Parallel and serial interfacing ports
- Analog/Digital converters
- Digital/Analog converters
- Counter
- Watchdog-Timer
- Read Only Memory (ROM)
- Random Access Memory (RAM)

In this setup, the microcontroller Arduino Uno is used to gather the information of three external temperature sensors *DS18B20* at a rate of about 1 Hz.

The temperature sensors in use can be categorized as resistive transducers that are based on a substantially linear resistance/temperature coefficient. For a rapid response and to avoid local temperature gradients, these temperature sensors must have a small thermal capacity [70].

To avoid a noisy signal, a program is executed on the Arduino that takes the mean value of the last ten measurements of the external temperature sensors. These are then send to the RPi using serial communication. To develop the small script for gathering and sending the data the Arduino IDE was used. It allows to write programs and upload them to the board.

5.2.3 Single Board Computer

Single Board Computers (SBCs) are computers of the size of a playing card. They consist of a single circuit board memory and a processor and provide several Input/Output interfaces for sensors or other devices [71].

In contrast to a microcontroller a SBC such as the Raspberry Pi is a fully functional computer that is designed to run an operating system such as Linux or Windows. It is much faster having a 1.4 GHz processor whereas an Arduino uses only up to 20 MHz. Moreover, a single board computer provides several connection possibilities such as Bluetooth and WIFI, while a microcontroller would have to make use of hardware attached on top to provide it. SBCs also provide several ports such as HDMI and USB that are not available in microcontrollers. In general it can be said that microcontroller are mostly used to perform simple and repetitive tasks like reading temperature, while single board computers perform multiple tasks [72].

5.3 Time Series Database

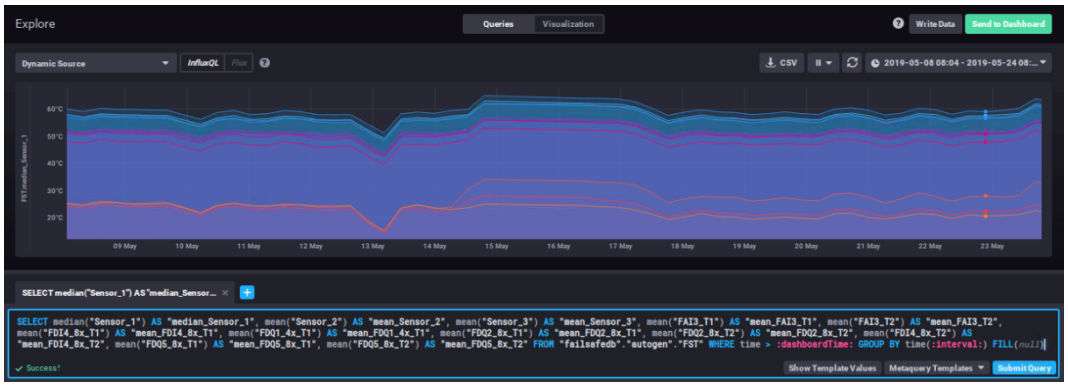


Figure 5.2: Chronograf visualization engine

As already mentioned in section 3 traditional databases are not designed to handle a high amount of time-stamped data. Therefore, there is a need for time series databases such as InfluxDB that provide a high availability

of the data and a high performance for reading and writing operations. In time series databases the data is co-located in chucks within the same time range on the same physical part of the database cluster. This allows to serve requests predictably and quickly during peak loads. Data is often pushed in a high frequency onto such databases, reaching frequencies of more than 1 Hz. Therefore, they perform data compression tasks and offer functionalities to handle large scales of data [73].

In this experimental setup the time series database InfluxDB developed by InfluxData was used. Together with other services such as Telegraf, Chronograf and Kapacitor it forms the TICK stack. According to the developers it is a "loosely coupled yet tightly integrated set of open source projects designed to handle massive amounts of time-stamped information to support your metrics analysis needs" [32].

The aim of these four services can be described as follows: [32]

- **Telegraf** is a server agent to collect data from a variety of sources and sends it to databases such as InfluxDB. In the case of the presented experimental setup it is not used though. Instead, a Python script is running on the RPi that is started right after booting the system. It gathers the data from the PLC and the Arduino and sends it onto InfluxDB.

- **InfluxDB** is a schemaless time series database that provides a SQL-like querying language and is mostly open-source. As all the other services of the TICK stack it is independent of the platform and offers application programming interfaces for several programming languages such as Python.

- **Chronograf** is the user interface and visualization engine of the database. It provides the necessary infrastructure to monitor the continuously updated data. It allows to easily build up dashboards with real-time visualizations. Figure 5.2 shows the temperature measurements of the I/O-modules tracked in the period between the 8^{th} and the 24^{th} of may.

- **Kapacitor**, the last tool of the stack group is a processing engine that allows to process alerts and compute statistical anomalies. It serves

as the interface between the original data and the algorithms that are applied onto the data. However, in this thesis Kapacitor was not used to process the data [32].

5.4 Physical Description

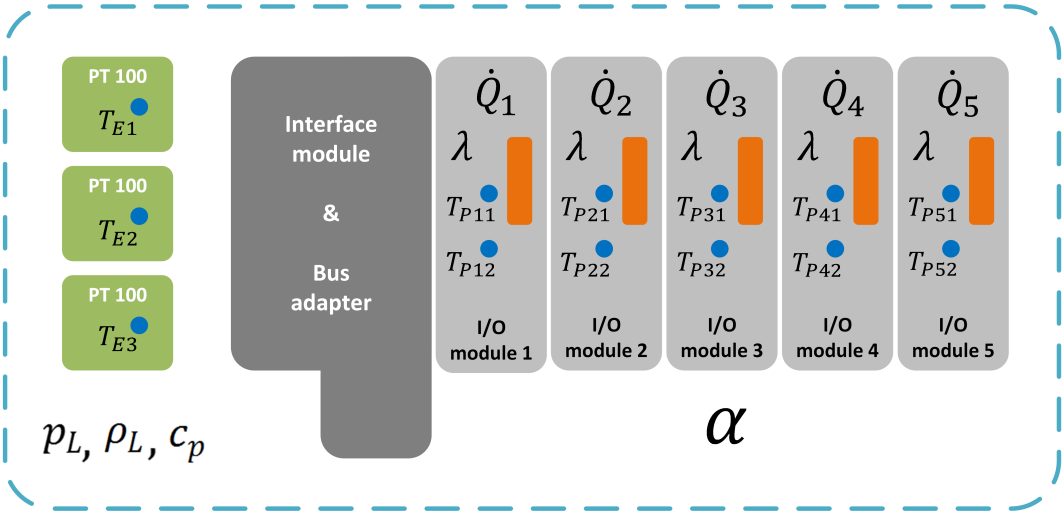


Figure 5.3: Physical model of experimental setup

After describing the hardware and software used in the experimental setup a short overview of the physical description of the model is given. Physical models are often based on differential equations and hence much more complicated than models that focus on the detection or the monitoring of a process only. Therefore, in this thesis the physical phenomena of heat transfer and thermal conduction was not simulated.

Following Occams' razor also known as law of parsimony, the simplest model that fulfills a task was chosen [74].

The whole process was treated as a black box and only the output, was used. Nevertheless, knowing the basic of the physical description and having

some domain knowledge on the topic of heat transfer helps validating and interpreting the measured data.

Figure 5.3 shows the interface module and the bus adapter of the setup with its five I/O-modules. Each module consists of two temperature sensors, represented as the blue dots that measure the Temperature T_{Pi1} and T_{Pi2}. They are used to monitor the temperature of the processor depicted in orange. The processor can be interpreted as a source of heat with the power $\dot{Q}_i$. The parameter λ represents the thermal conductivity under the assumption of *Fourie's law*. It describes the heat conduction within a material. The parameter α represents the heat transfer coefficient under the assumption of *Newton's law of cooling*. The three green devices represent the external temperature sensors measuring the environmental temperatures T_{Ei}. The physical quantities p_L, ρ_L and c_p describe the pressure, the density and the specific thermal capacity of the surrounding medium (air).

The following mathematical correlations describe the physical relations of the given case of heat conduction: [75]

Simplified heat conduction equation:
$$\frac{\partial T}{\partial t} = \frac{\lambda}{\rho c_p}\Delta T + \frac{\dot{q}_s}{\rho c_p} \tag{5.1}$$

with λ ...thermal conductivity
ρ ...density
c_p ...specific heat capacity
$\dot{q}_s$ ...local heat flux density

where $a = \frac{\lambda}{\rho c_p}$ ...thermal diffusivity

and $\Delta T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}$ ...Laplace Operator

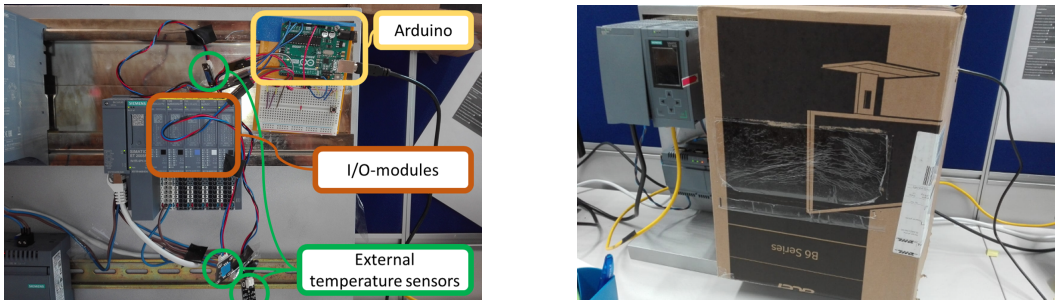
Thermal conduction (Fourie's law):
$$q_s = -\lambda \nabla T \tag{5.2}$$

where $\nabla T = \frac{\partial T}{\partial x} + \frac{\partial T}{\partial y} + \frac{\partial T}{\partial z}$ ...temperature gradient

Heat transfer (Newton's law of cooling): $q_s = -\alpha(T_E - T_H)$ (5.3)

with α ...heat transfer coefficient
T_E ...environmental temperature
T_H ...housing temperature

5.5 Data Collection



(a) Experimental setup (open) (b) Experimental setup (closed)

Figure 5.4: Real experimental setup

Figure 5.4 illustrates two pictures of the real experimental setup. The left one shows the setup in an open state with its five I/O-modules and three environmental temperature sensors that are connected to the Arduino. The right one shows the experimental setup when surrounded by a box. The idea of enclosing the whole setup with a box was to create the possibility to trigger a change point in temperature. Moreover the setup was put in an office in vicinity to a window of an approximately 20 m^2 room. Figure 5.5 shows the reaction of the processor's temperature when the box and the window got opened. This part of the time series was later on used to develop the change point detection algorithm. Furthermore, it is used to explain the

functionalities, parameters and shortcomings of the algorithm as explained in the following chapter.

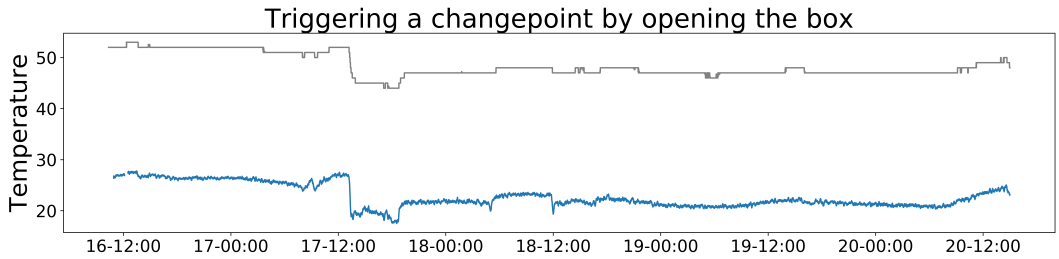


Figure 5.5: Triggering a change point

In the experimental setup, the data is sent by the Arduino and the PLC to the Raspberry Pi where a connection with the time series database InfluxDB is established. The data is marked with a timestamp and further pushed onto the database about every 2.2 seconds. This is considered a time frame that is more than sufficient to enhance temperature awareness to an automation device. Figure 5.6 shows how the time is distributed among the respective tasks.

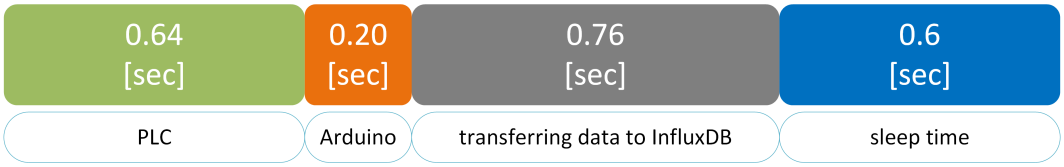


Figure 5.6: Time division pushing data onto time series database InfluxDB

Starting from May till mid August 2019 data was gathered with minor interruptions and stored onto the mentioned database. Looking at a period of 16 days as illustrated in figure 5.7 there is a clear difference between working days (not colored) and holidays (colored). The local minima are numbered from 1 to 17. Some of these very low minima indicate that a window was opened. As can be seen this happened mainly during the week. It is even possible to see, that apparently one of the colleagues tends to go to office on the weekend as well. Comparing the extrema of the external sensor with the time series of the temperature of one of the I/O-modules

this figure indicates the potential of these automation devices to gain some environmental awareness. The given set of data will later on be used to present the results of the change point detection algorithm.

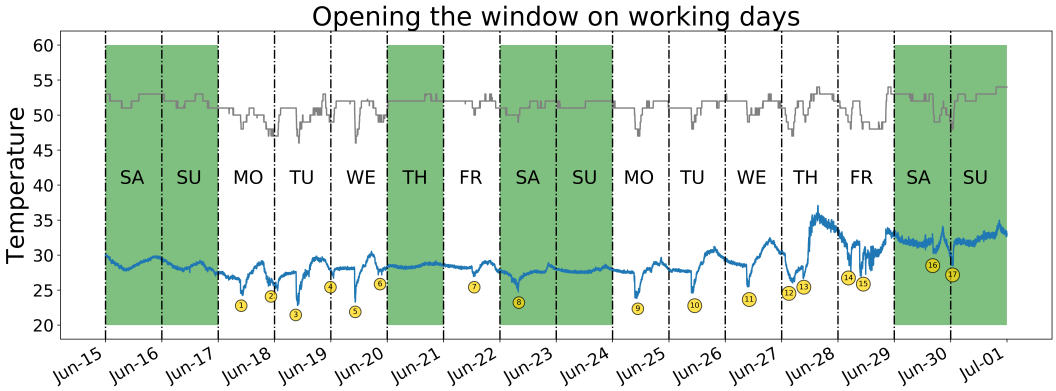


Figure 5.7: Sensibility of the system

6 Creating Awareness

This chapter gives an insight on the process of creating a temperature aware system making use of the internal temperature sensors within the I/O-modules of the PLC and the three external sensors. After a short overview of the requirements for such a system, the two features, change point detection and the temperature prediction are explained in detail. Furthermore, the idea of a similarity comparison among the I/O-modules is briefly sketched.

6.1 Requirements

The initial motivation of this thesis was to investigate the general usage of already available but yet unused data in an embedded system like the processor temperatures of the I/O-modules. Research on this topic resulted in the idea of creating a system that is capable of detecting changes in its own measurements and that is able to make predictions about the environment.

The requirements for such as system were only vaguely defined as the process of investigating the general usage of data was in focus. However, it was clear from the very beginning that if concrete requirements could be specified it would be very likely to meet them. This is particularly true when dealing with a physical quantity such as temperature, because of two main reasons: First of all, changes in temperature show a high time constant, meaning that because of the thermal inertia a change needs rather long than in changes of other physical quantities. Therefore, the requirement for the reaction time on a change point is rather low and more measurements can be taken into account before deciding whether a new data point indicates a

change point or not. The second reason is that the system endures relatively high temperature differences. Hence, detecting a temperature deviation of some degrees is already enough. This of course is a big advantage when trying to differentiate a statistical deviation inherent to the measurement chain from a systematic change because of environmental changes.

Another requirement elaborated during the work was the need to create a reliable feature with a minimum of false positive alarms, keeping in mind the tradeoff between sensitivity and robustness as explained in chapter 4.3. Only a method such as the change point detection that can be trusted on will add its share to a more intelligent manufacturing system.

6.2 Creating Self-Awarness: Online Change Point Detection

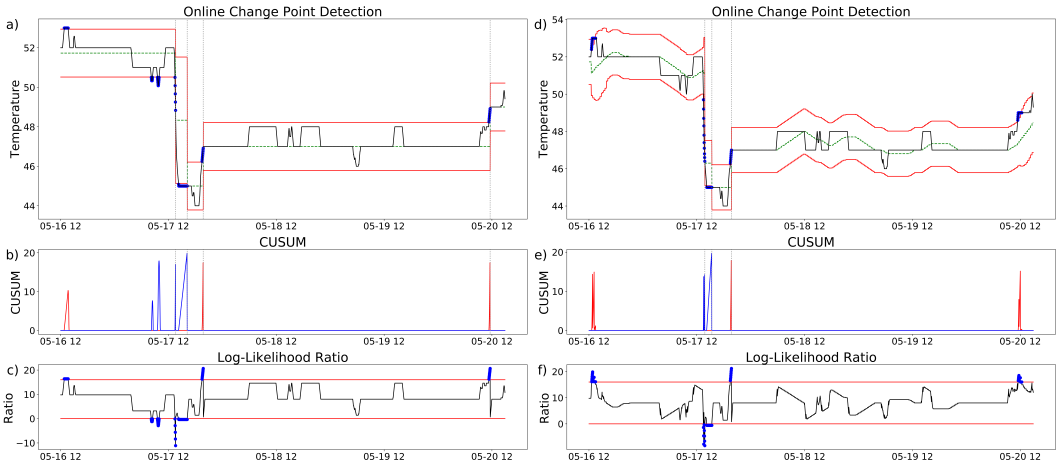


Figure 6.1: Online change point detection - comparison of two different algorithms

The developed online change point detection algorithm is a mixture of an adaptable control chart with elements of a CUSUM algorithm. On the one hand it uses the idea of upper and lower control limits similar to the control charts. On the other hand it accumulates the log-likelihood ratios of data points that are exceeding these limits following the idea of CUSUM. The

combination of both systems allows to create a robust method to constantly test streaming data for a change point. This includes data that does not fulfill the property of being independent and identical distributed (*iid*).

Both versions of the algorithm that can be chosen of are explained in detail, by introducing the global input parameters, followed by the three support functions. These functions are of statistical nature and are used within the main function. Finally the main algorithm is presented introducing the function **create_control_window** and the main script.

In the explanation of the algorithm three kind of inputs are differentiated. *Global parameters* have to be set beforehand and are used within functions but do not have to be passed explicitly. *Function parameters* must be passed explicitly. *Data* in form of a data point or a data array is passed to the function. It is the data that is actually manipulated. Furthermore, the sources for the so called "damping mechanisms" are going to be discussed. At the end a closer look is taken on the performance of the change point detection algorithm. This is done by answering the question how a small change in the global parameters influences the quality of the detection. Different settings of the global parameters are tested using the 16-days data set introduced in the previous chapter.

6.2.1 Explanation of the Algorithm

Two versions of the algorithm can be differentiated that rely on the same functions but follow a different approach. The first version is characterized by a discrete recalculation of the control window given by its mean μ_{seg_i} and the standard deviation σ_{seg_i}. This means that whenever a change point occurs, the control window is just recalculated once and valid for the whole segment. In contrast, the other version recalculates the control window in a preset frequency leading to a flexible tolerance corridor defined by μ_j or σ_j, that is nevertheless capable of detecting abrupt changes. Before starting to explain the different support functions and explaining the main algorithm in detail, the global parameters need to be set.

Global Parameters

Global parameters define the sensibility and the adaptability of a process. As discussed the goal is to create a trustful but robust change point detection algorithm. Therefore, several parameters can be set that are going to be referred to later on in the explanation of the subfunctions and the main algorithm.

- *win* defines the size of the rolling window to calculate the average temperature and the average log-likelihood ratio.
- *sigmafactor* defines the distance between the upper and the lower control limits *ucl* and *lcl*. It sets the amount of standard deviations between the control window's center line and the respective control limit. The higher this factor the lower the number of outliers in a normal distributed stationary process.
- *mowin* sets the size of the moving data window that is memorized. It can not be smaller than *win*.
- *continuously* can be set to 0 or 1 and activates the second version of the algorithm as will be explained further down.
- *degree* sets the frequency that the algorithm is recalculating the control window in case of a continuous recalculation of the control window when *continuously* is set to 1.
- *his* defines the number of the last most recent points that are used to define a new control window after a change point was detected, it should be at least of size *win*.
- *ucl*2 defines the value that has to be passed by the cumulated sum of upper deviations to indicate a change point.
- *lcl*2 defines the value that has to be passed by the cumulated sum of lower deviations to indicate a change point.

Support Functions

The support functions as explained in the introduction of this chapter are used within the main functions and work as follows:

- **parameter**: This function returns the mean μ and the standard deviation σ of a given set of data points.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.1}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{6.2}$$

- **log_likelihood_ratio**: Taking the parameter μ, σ as well as the value $sigmafactor$ as an input, this functions calculates the log-likelihood ratio for a given data point. This is done assuming two normal distributions with the same standard deviations but a mean that is $2 \cdot sigmafactor \cdot \sigma$ away from each other.
The likelihood ratio is calculated dividing the likelihood of a data point to be within the distribution $p(x|\mu_{up}, \sigma)$ by the likelihood of being in the distribution that assumes a lower mean temperature $p(x|\mu_{lo}, \sigma)$. The return value of the function is then computed taking the logarithm of that value.

$$s_i = ln\left(\frac{p(X = x_i|\mu_{up}, \sigma)}{p(X = x_i|\mu_{lo}, \sigma)}\right) \tag{6.3}$$

where:

$$\mu_{up} = \mu \tag{6.4}$$

$$\mu_{lo} = \mu - 2 \cdot sigmafactor \cdot \sigma \tag{6.5}$$

Hence, the following cases can be differentiated:

$$s_i = \begin{cases} < 0 & \text{if } p(X = x_i|\mu_{lo}, \sigma) > p(X = x_i|\mu_{up}, \sigma) \\ = 0 & \text{if } p(X = x_i|\mu_{lo}, \sigma) = p(X = x_i|\mu_{up}, \sigma) \\ > 0 & \text{if } p(X = x_i|\mu_{lo}, \sigma) < p(X = x_i|\mu_{up}, \sigma) \end{cases} \tag{6.6}$$

- **reverse_likelihood**: This function calculates the temperature x_i for a given log-likelihood ratio s_i. It is used to translate a certain control limit of a segment in form of a log-likelihood ratio back to the corresponding temperature within the segment.

$$x = \frac{(2 \cdot s_i \cdot \sigma^2) + (\mu_{up}^2 - \mu_{lo}^2)}{4 \cdot sigmafactor \cdot \sigma} \tag{6.7}$$

Main Algorithm

The main algorithm can be divided into two parts. One part is the function **create_control_window** that besides other values calculates the parameters for a new control window. The other part is the main script that calls the function **create_control_window** whenever a change point was detected. This is done by calculating the log-likelihood ratio of the incoming data points for the given control window. The differences of the rolled mean log-likelihood ratios of the data points exceeding the upper and the lower limits are then accumulated and compared to a second predefined limit. Only if this limit is exceeded a change point is indicated and the function **create_control_window** is called to calculate the parameter of the control window for the new segment.

First, a set of training data $\vec{x}_{train}$ has to be defined, representing the measurements of the processor's temperature under stationary conditions. This means that only a data set that does not show a change in mean at a stationary environmental temperature shall be used as training data. It is then passed as an input for the function **parameter** to calculate the mean μ_0 and the standard deviation σ_0 of the given temperature data points. Both values, μ_0 and σ_0 are necessary to define the initial control window. However, σ_0 is particular important as it is used as the minimum standard deviation for future control windows. In case a subsequence with a very small variance is taken to recalculate the control window after a change point, the standard deviation will be set again to σ_0.

In a next step the function **create_control_window** is called setting up the dataframe *oda* (**o**nline **d**etection **a**rray) making use of the three presented support functions. The dataframe contains all necessary columns to perform an online change point detection. The function takes the data $\vec{x}$ (or $\vec{x}_{train}$ in the very first case) as an input and requires the global parameters win and $sigmafactor$.

The dataframe *oda* results in a matrix with n rows and 14 columns, where n is the number of passed temperature values in $\vec{x}$. It contains the following values for each data point:

temperature: original temperature value
temperature_rm: rolled mean temperature value over the window of size *win*

log_likelihood_ratio: log-likelihood ratio calculated using the function **log_likelihood_ratio**
log_likelihood_ratio_rm: rolled mean log-likelihood ratio over the window of size *win*

lcl: lower control limit in form of a log-likelihood ratio that is set to 0. As explained this value indicates a data point that is equally likely to be a part of either the upper one or the lower distribution.

ucl: upper control limit in form of a log-likelihood ratio that depends on the global parameter $sigmafactor$. Common values are:

$$ucl = \begin{cases} 12 & \text{for } sigmafactor = 1 \\ 16 & \text{for } sigmafactor = 2 \\ 20 & \text{for } sigmafactor = 3 \end{cases} \tag{6.8}$$

templcl: lower control limit in terms of degree Celsius, calculated calling the function **reverse_likelihood** taking *lcl* as an input

tempucl: upper control limit in terms of degree Celsius, calculated calling the function **reverse_likelihood** taking *ucl* as an input

p_likeli_cusum: this value is set to 0 at the beginning as the data $\vec{x}_{train}$ is assumed to be stationary

n_likeli_cusum: this value is set to 0 at the beginning as the data $\vec{x}_{train}$ is assumed to be stationary

index_interval: Counting the number of elements in each segment. When setting up the dataframe *oda* no values are filled into that column.

recalcflag: Counting the number of segments. When setting up the dataframe *oda* no values are filled into that column.

par_mean: μ of the data set $\vec{x}$

par_sigma: σ of the data set $\vec{x}$

The dataframe *oda* can be seen as a moving window that is supposed to store only a certain amount of data points. This amount is set using another

global parameter called *mowin*. The very first training data $\vec{x}_{train}$ is not checked for its length. This is the reason why in a next step the length of the dataframe is read out and used as a criteria for different cases. Depending on the version of the algorithm it includes either two or all three of them. For version1 with a discrete calculation of control windows the cases *case1* and *case3* apply. For version2 with a continuous recalculation the cases *case1* and *case2* mainly apply with a single exception for *case3* as will be explained later on. To distinguish between those versions the global parameter *continuously* has to be set to 1 or 0.

case1: The length of *oda* is smaller than the set value *mowin*. In this case the new incoming data point is just added to the column *[temperature]* of *oda*. All other values in the row are then added as well step by step during the main script. This includes calculating the cumulated sum of rolled average log-likelihood ratios for consecutive data points exceeding the set upper or lower limits.

case2: This only applies for version2 of the algorithm if *continuously* is set to 1. The length of *oda* still has to be smaller than *mowin* but as soon as it reaches the length of $mowin - 1$ it enters this case. This results in taking the last *his* values of *oda* to recalculate the control window. The global parameter *degree* defines the modulo of the recalculation frequency. If *degree* is set to one, the control window is recalculated for every new data point, while if *degree* is set to the value of two the dataframe *oda* will first enter *case1* till it is again of length $mowing - 1$ before it enters again *case2*. For $degree = 3$ it would take three new data points, for $degree = 4$ four and so on and so forth. Therefore, values bigger than one lead to some sort of terraced history of control windows. After the recalculation, all other values in the row of the new data point are again added just as it was the case in *case1*.

case3: This mainly applies for version1 of the algorithm. To enter this case the length of *oda* must be at least of length *mowin*. In the case of the very first training data $\vec{x}_{train}$ it could also be bigger. The only difference to *case1* is that it drops the oldest data points to reach again the length of $mowing - 1$. That means that as in *case1* and *case2* all other values of the most current data point are added step by step as well. The algorithm will proceed entering this case if there is no sign of a change point.

To summarise the two versions they can be described as follows.

version1: After calculating the first control window a dataframe *oda* is generated and a new data point is added. If the length of *oda* is smaller than *mowin* the next data points are just added without dropping the older ones (*case1*). If the length of *oda* is equal to *mowin* the exceeding data points are dropped to reach again the length of $mowin - 1$ before adding a new data point (*case3*). In case of the very first training data more rows might be dropped. The algorithm continues to stay in *case3* till a change point is detected.

version2: If $\vec{x}_{train}$ is smaller than $mowin - 1$ the dataframe is filled up with new data points as explained in version1. Once it enters *case2* a recalculation takes place and repeats itself in a frequency that is defined by the global variable *degree*. The algorithm continues to alternate between the two cases till a change point is detected. For the special case that $\vec{x}_{train}$ was bigger than *mowin* the algorithm enters *case3* a single time but returns back to *case2* right afterwards.

Damping property of the algorithm

So far the algorithm was only explained for a stationary temperature process without any change point. As already briefly mentioned, a change point is in general not detected once the a log-likelihood ratio exceeds the upper or lower control limits *ucl* respectively *lcl*. Although this could technically be the case it is very unlikely as one single outlier would have to outperform all "damping meachanisms" of the algorithm. The effect of these mechanisms is to make the online change point detection robust against single or small groups of outliers. They can be summarized as follows:

The first damping property as illustrated in figure 6.2 comes from taking the rolled mean log-likelihood ratio calculated over a window of size *win* instead of the log-likelihood ratio itself. This leads to the fact that even if the log-likelihood ratio of a single data point was outside the control window, the rolled average that is taken of the last *win* data points can still be within the set limits. The rolled average log-likelihood ratio can be defined as:

$$\bar{s}_i = \frac{\sum_{i=(1-win)}^{1} s_i}{win} \tag{6.9}$$

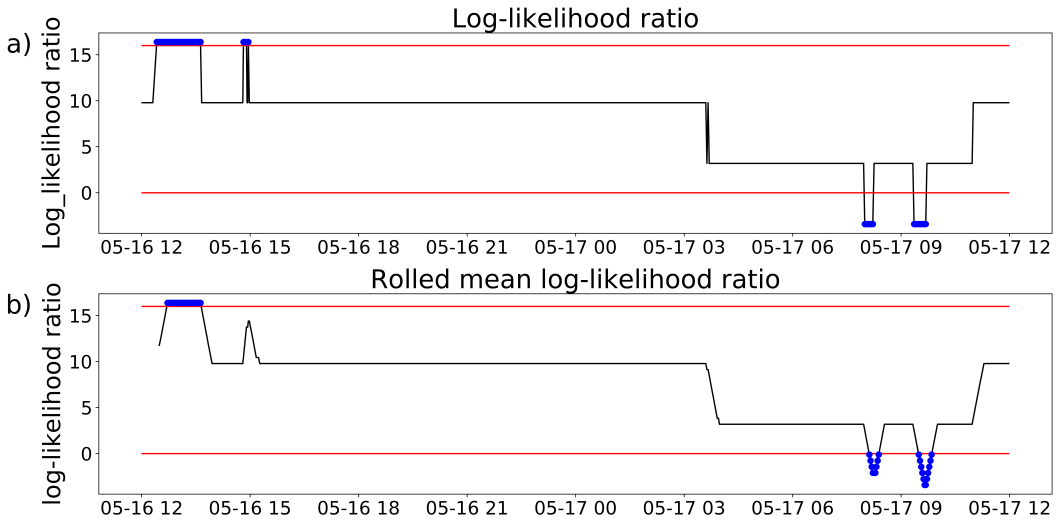


Figure 6.2: Damping mechanism due to rolled mean log-likelihood ratio

A second mechanism results from the fact that the cumulated sum of the differences between the rolled average log-likelihood ratio of the points and the set limits cannot get smaller than 0, which can be expressed as follows. Note that both cumulated sums are defined positive.

$$\overline{S}_{up,k} \geq 0 \tag{6.10}$$

$$\overline{S}_{lo,k} \geq 0 \tag{6.11}$$

with

$$\overline{S}_{up,k} = \sum_{n=(1-win)}^{1} (\overline{e}_{up,i}) \tag{6.12}$$

$$\overline{S}_{lo,k} = \sum_{n=(1-win)}^{1} (\overline{e}_{lo,i}) \tag{6.13}$$

where

$$\overline{e}_{up,i} = \overline{s}_i - ucl \tag{6.14}$$

$$\overline{e}_{lo,i} = lcl - \overline{s}_i \tag{6.15}$$

and

$$|\overline{e}_i| \leq \overline{S}_{i-1} \tag{6.16}$$

leading to

$$\overline{S}_k = \begin{cases} \overline{S}_{k-1} + \overline{e}_i & \text{for } |\overline{e}_i| < \overline{S}_{k-1} \\ 0 & \text{for } |\overline{e}_i| > \overline{S}_{k-1} \end{cases} \tag{6.17}$$

Equation 6.10 to 6.17 state that only differences ($\overline{e}_{up,i}$ respectively $\overline{e}_{lo,i}$) are taken into account that do not lead to a negative cumulated sum ($\overline{S}_{up,k}$ respectively $\overline{S}_{lo,k}$) when added to the sum. Because of that, $\overline{e}_i$ as defined in equation 6.14 and 6.15 does not necessarily have to be positive to be added to the cumulated sum. When negative, their absolute value must be smaller than the cumulated sum, otherwise the cumulated sum is set equal to 0. In other words the cumulated sum can grow steadily but decrease steadily or abrupt depending on the residuals. This results in a damping effect. Data points that are again within the set limits after the cumulated sum had already been growing lead to a reduction of that cumulated sum. Small sequences of data sets that are alternating around a control limit will therefore not necessarily lead to a change point.

The third reason leading to a robust algorithm is that a change point is only indicated, when the cumulated sum is bigger than the global variables $ucl2$ and $lcl2$ which represent the second upper respectively the second lower limits. Note again that both limits are defined as positive values.

$$\overline{S}_k \geq ucl2 \tag{6.18}$$

$$\overline{S}_k \geq lcl2 \tag{6.19}$$

Again the value $ucl2$ and $lcl2$ are of the unit log-likelihood ratio. As the other global parameters they have to be set beforehand. What differentiates them from the other parameters is that their suitable value for a reasonable change point detection depends on the frequency of the data. Using an absolute value for the second control limit leads to a more sensible algorithm the higher the frequency gets, a disadvantage that would need to be fixed in future.

Once one of the two second control limits are exceeded by the respective cumulated sum $\overline{S}_{up,k}$ or $\overline{S}_{lo,k}$, the sums are set back to 0. A flag is then set which triggers the algorithm to enter the case that a new control window has to be calculated. A predefined amount of data points set in the global variable *his* is passed to the function **create_control_window** to create the new upper and lower control limits. When the new dataframe *oda* is returned to the main script, the explained process starts again. First, the dataframe is filled up again to the size of *mowin* and then depending on the used version of the algorithm the control window is continuously, frequently or not at all recalculated till the next change point.

6.2.2 Performance Measurement

To apply the described performance measurements, the position of the change points separating different segments would have to be exactly defined to classify the data points into the categories of a confusion matrix. As this is not the case the algorithm shall be analyzed in a more qualitative way. This is done by applying both versions of the change point detection method to the same temperature curve with minor adaptations in the parameter settings. These preliminary results are then used to discuss the sensitivity of the algorithm and the effect on its ability to detect the structural change within the given data set.

First the change point detection algorithm with a fixed control window is discussed as presented in figure 6.3. Table 6.1 shows that the chosen parameters of "detection 1" and "detection 2" only differ in the value of $sigmafactor$. While the first result on the left side ("detection 1") indicates up to 14 changes the other one ("detection 2") only shows four change points. The evaluation of the reliability and the robustness of a change point detection algorithm depends a lot on the given use case. However, the segmentation on the right side of figure 6.3 looks quite promising as its segments are coherent and easy to follow. Analysing the granularity of the temperature resolution it can be said that the tolerance window between upper and lower control limit is hardly more than $2°C$, which is more than enough in case of creating a temperature aware automation device.

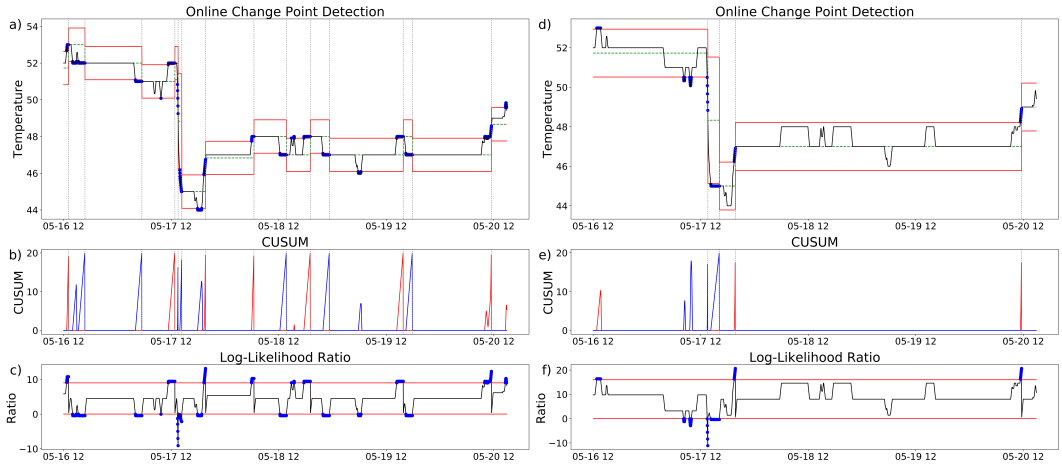


Figure 6.3: Discrete online change point detection - comparison of parameter settings

The necessary time for the recognition of a change point is strongly dependent on the set parameters too. A long window *win* leads in general to a slower adaptation when a change occurs as the influence on each point on the average value of the log likelihood ratio diminishes. Also, the set control limits for the control window (*lcl* and *ucl*) but also for the maximum cumulated sum (*lcl*2 and *ucl*2) have a strong influence on the reaction time. Again a tradeoff has to be found between slow but very likely change points or fast reaction time and the risk of false positive change point detections.

Figure 6.4 shows the application of the continuous change point detection algorithm on the very same temperature curve. This time the parameters of "detection 3" and "detection 4" as presented in table 6.2 differ only in the length of the *degree* window that states how often a recalculation of the control window is performed. "Detection 3" on the left side indicates that no change point got detected. Apparently the control window is adapting too fast to the changes in temperature. On the right side in "detection 4" the algorithm detects three change points. The results of figure 6.4 d)-f) are very similar to the one of figure 6.3 d)-f) except for the change point at the very end of the given interval. In this case the size of the control window looks also quite promising. However, when it comes to the topic of computing performance more resources are needed for the continuous recalculation of the control window every *degree*-times. It might be a problem if the time

Table 6.1: Discrete online change point detection - parameter settings

global parameter	**detection 1**	**detection 2**
win	12	12
sigmafactor	**1.5**	**2**
mowin	200	200
continously	false	false
degree	-	-
his	12	12
*ucl*2	20	20
*ucl*1	20	20

between two pushed data points is less than the amount of time needed to perform the calculations on a new data point. This is definitely not the case when dealing with temperature data that is pushed every 2.2 seconds onto the database.

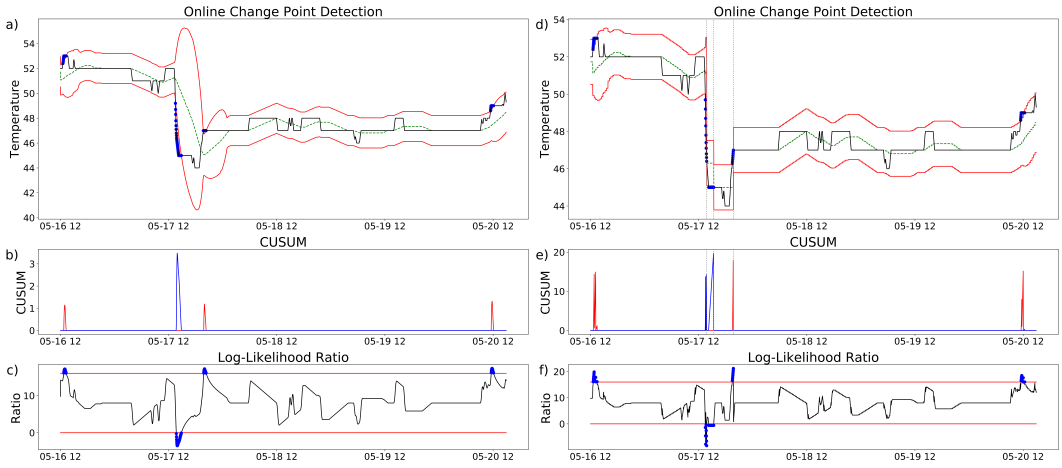


Figure 6.4: Continuous online change point detection - comparison of parameter settings

Summing up the performance comparison it can be said that both algorithms can be parameterized to show promising results. Both versions provide a useable resolution of changes in temperature and a fast and robust detection of the change points. There are however some shortcomings that shall be discussed in the following chapter.

Table 6.2: Continuous online change point detection - parameter settings

global parameter	**detection 3**	**detection 4**
win	12	12
sigmafactor	2	2
mowin	200	200
continously	true	true
degree	**1**	**10**
his	12	12
*ucl*2	20	20
*ucl*1	20	20

6.2.3 Shortcomings

After describing the algorithms in detail and comparing the performance of different settings of parameters, it is important to mention the shortcomings of the presented change point detection method. Knowing them helps to interpret the results and allows to know what to expect from the given method. In this context the following two shortcomings shall be discussed.

Supposedly Statistical Method

The developed algorithm is manly based on a log-likelihood ratio criteria. The likelihood ratio was defined as the likelihood of a data point to be within the distribution $f_0(x)$ divided by the likelihood of distribution $f_1(x)$. Using such a measurement to evaluate whether a point accounts to a predefined control window suggests that its results are of real statistical nature. However, this is not the case. The defined ratio is rather used as a measurement to define the distance of a point to a certain expected value in an interval.

The reason is that the given measurements are not independent from each other and not identically distributed. Their value can not be used to calculate real likelihoods. Figure 6.5 explains the main discrepancies to a real statistically sound hypothesis testing. First of all the data is not normally

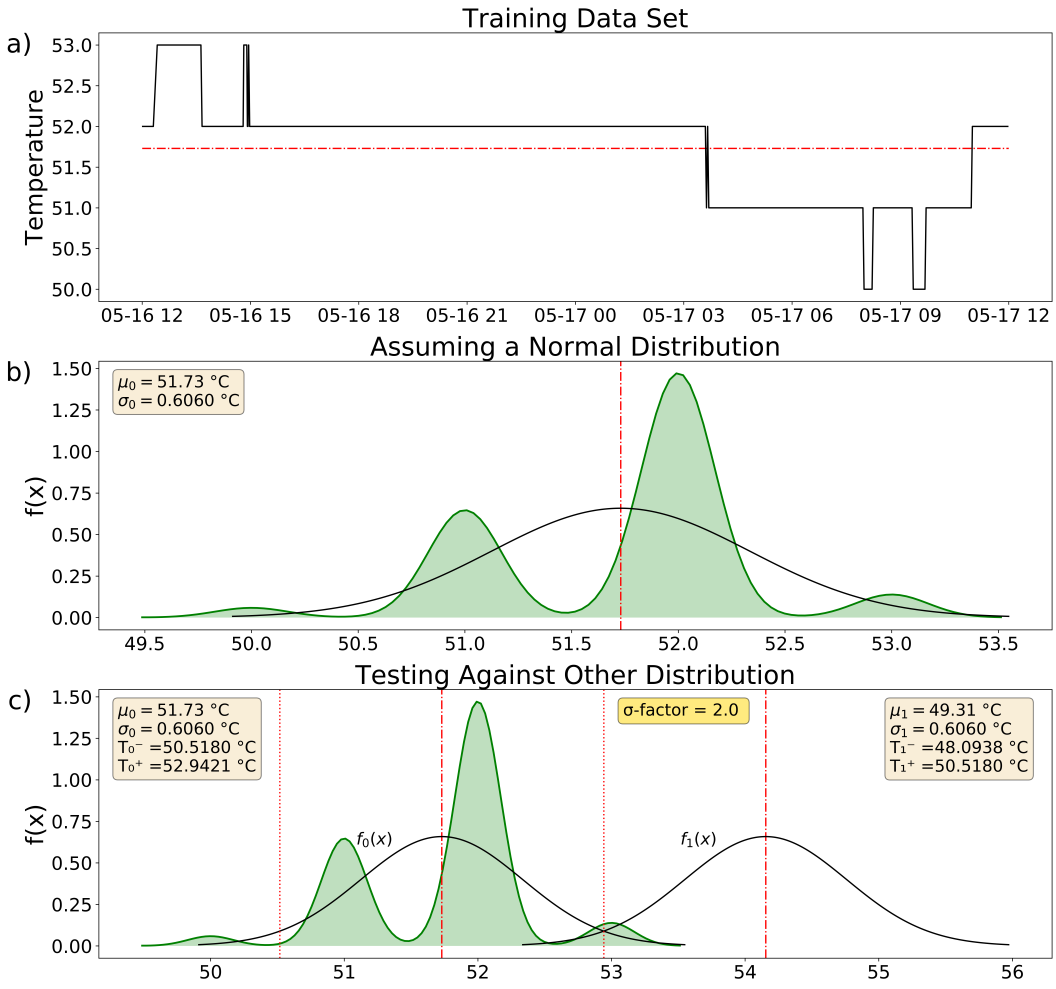


Figure 6.5: Log-likelihood ratio - a distance measurement

distributed, as shown in Figure 6.5 b). Hence, the likelihood of being within a certain interval does not represent the real likelihood that a data point belongs to a certain distribution.

Second as indicated in 6.5 c) the definition of the distribution $f_1(x)$ that a given distribution is tested against with is arbitrary chosen to be able to use the log-likelihood ratio criteria. Usual hypothesis tests would rather check whether a data point is within a given distribution (hypothesis H_0) or not (hypothesis H_1).

In other words, the used criteria to calculate the distance of a data point for a given distribution is only based on statistical methods but its intermediate results can not be interpreted as such. Knowing this has however no impact on the functionality and the reliability of the presented change point detection algorithm. As shown in the previous chapter, when setting the right parameters the results lead to a segmentation that is transparent and easy to follow.

Sample Rate Dependent Parameter Setting

Setting the values for the global parameter can only be done partly independent of the given time series. Especially the limits $ucl2$ and $lcl2$, but also the window sizes *win*, *his* and *mowin* have to be set in concordance with the given use case. The reason for that is that in order to achieve a certain sensibility and robustness of the change point detection, their value is dependent on the sampling rate of the incoming data. When increasing the frequency of the incoming data, both limits $ucl2$ and $lcl2$ also have to be increased. This applies also to the parameter *win*, *his* and *mowin* as their size is not given in time but in a number of data points. This leads to longer implementation periods and higher costs of setting up the system for a given process. As a result the parameter would have to be specified in the algorithm without dependency of the sampling rate.

6.2.4 Results of Online Change Point Detection

The algorithm is applied to the 16-days data set as introduced in chapter 5.5. The data stored on InfluxDB was retrieved with a time lag of 10 seconds between every data point. This means that the values actually represent a mean value of the original data uploaded every 2.2 seconds over a window of 10 seconds. Hence, a time series with 138.240 data points was analyzed. Due to computational power of the private computer used for this thesis the set was further reduced to one tenth of its size, resulting in a time series with an interval of 100 seconds. As explained in the shortcomings, this is important to know as the parameter settings and hence the sensitivity is frequency dependent.

Table 6.3 shows the chosen parameters for six different settings of the algorithm that shall be compared regarding its ability to detect the marked local minima. While settings 1 to 3 use the discrete version of the algorithm in setting 4 to 6 the continuous version is applied. Parameters changed are printed in bold numbers.

Table 6.3: Parameter settings for comparison

global parameter	**setting 1**	**setting 2**	**setting 3**
win	12	12	12
sigmafactor	2	2	**2.5**
mowin	200	200	200
continously	false	false	false
his	12	12	12
*ucl*2	20	**25**	20
*ucl*1	20	**25**	20
	setting 4	**setting 5**	**setting 6**
win	12	12	12
sigmafactor	2	2	2
mowin	200	200	200
continously	true	true	true
degree	10	10	**20**
his	12	12	12
*ucl*2	20	**12**	20
*ucl*1	20	**12**	20

Figure 6.6 and 6.7 show the results of the online change point detection for each setting. As explained in the previous chapter the red lines mark the current control window. In addition to that, grey dashed vertical lines show the point in time a change got detected.

To evaluate the performance of the chosen parameter settings the ability of the algorithm is tested to detect the local minima numbered from 1 to 17. In general a local minimum implies having at least two change points, one before and one after the respective minimum. However, this does not necessarily apply to all minima. In case that previous or consecutive points of a minimum lie within the control limits only one or even no change point gets detected. For deep minima it is also possible that more than two change points are encountered.

Before comparing the overall results, the change points for each setting are numbered by its appearance and related to the given minima as shown in

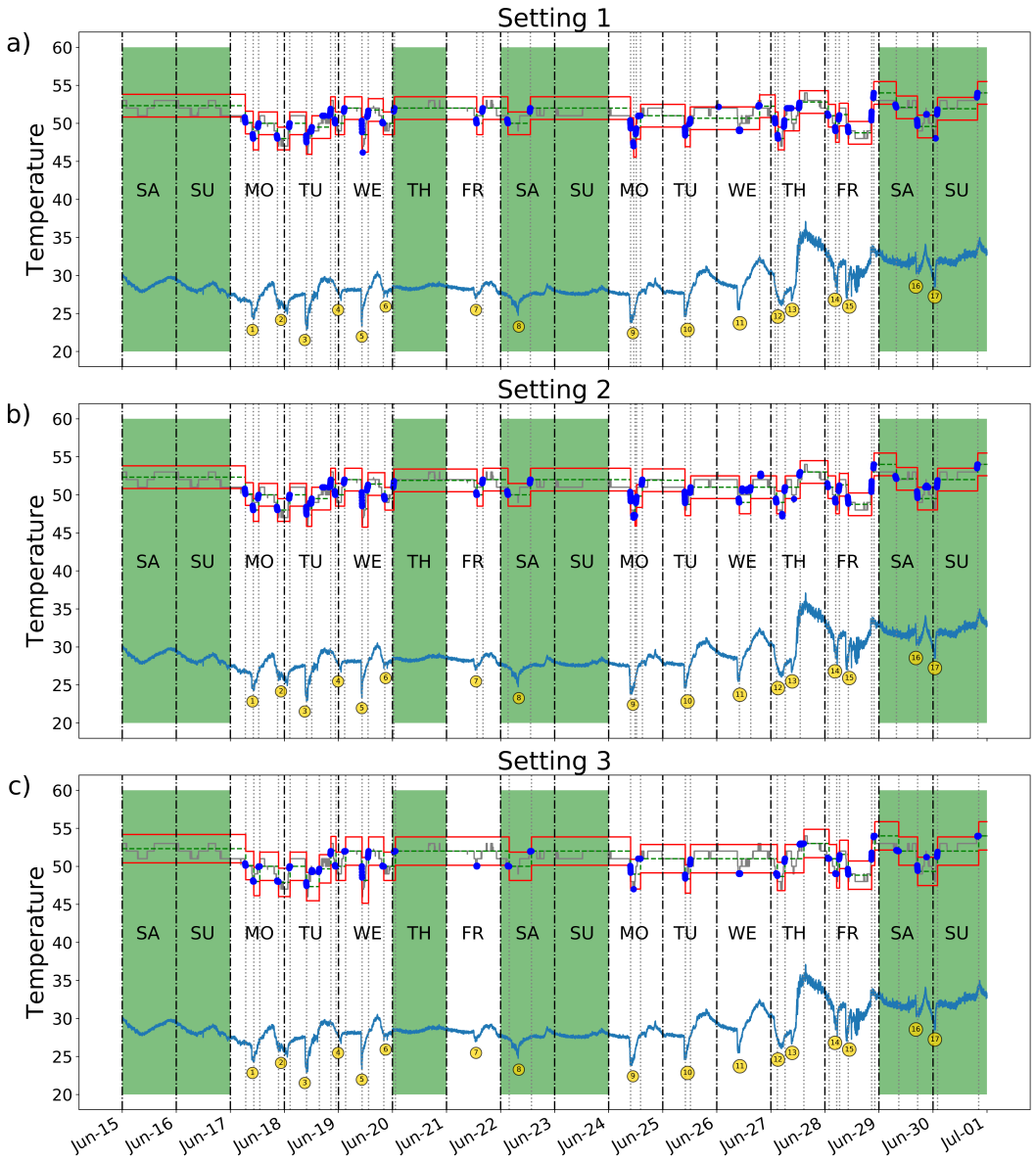


Figure 6.6: Results of discrete online change point detection

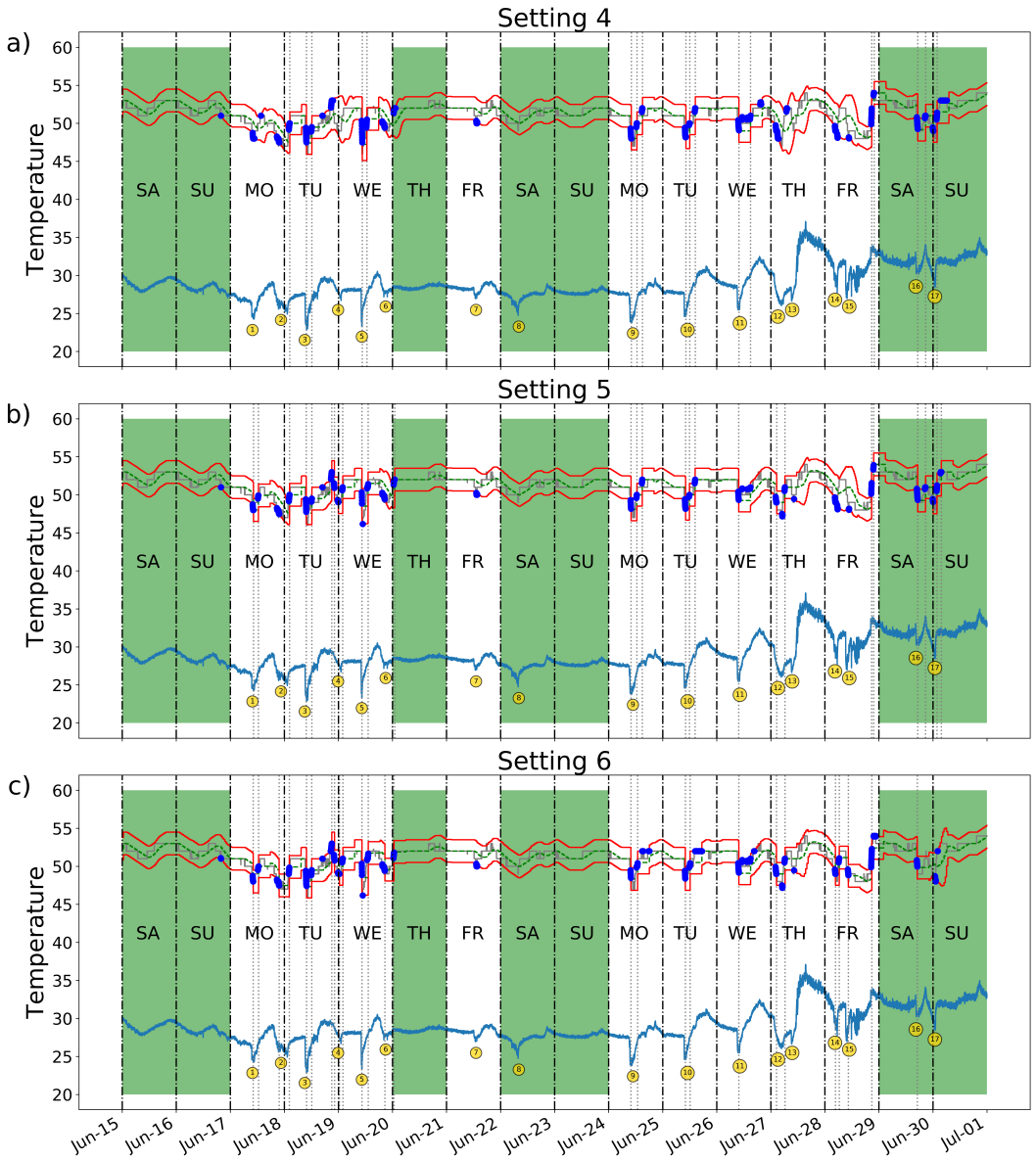


Figure 6.7: Results of continuous online change point detection

figure 6.8. This was done manually and therefore represents a subjective interpretation of the quality of the given change points. Nevertheless, it clearly emphasizes the potential of the developed algorithm. The following classification is used:

- green true positive (TP): The change point either marks the beginning or the end of a section with a local minimum.
- blue true positive (TP): The change point corresponds to a minimum but lays in between a starting or an ending change point.
- red false positve (FP): Change points without a transparent and comprehensive reason, thus representing false alarms.
- light orange Other (OT): Change points with a transparent and comprehensive reason that can however, either not be related to one of the predefined minima or that would not have detected a given minimum. This is for example the case for change points that indicated the end of a local minimum section without having a corresponding initial change point.

Figure 6.9 shows a comparison of the given settings summing up the previous results. For each setting the total number of change points (total), the number of false positives (FP) as well as the number of other change points (OT) are given. The quality of the minima detection is categorized in another three classes. These are defined as follows:

- Green boxes indicate correctly identified minima with exactly two change points in close proximity - one before the minimum and one after it.
- Blue boxes represent correctly identified minima with an amount of change points different to two. This includes minima with only one change point and deep minima that show more than one change points.
- Orange boxes indicate that the minimum was not found at all.

Interpreting the results lead to the following statements:

- Minima 3, 5 and 10 always got detected. Comparing them with each other shows that they are characterized by a very steep drop from its temperature base level to the minimum and a steep rise afterwards. Unfortunately change point number 11 with similar characteristics

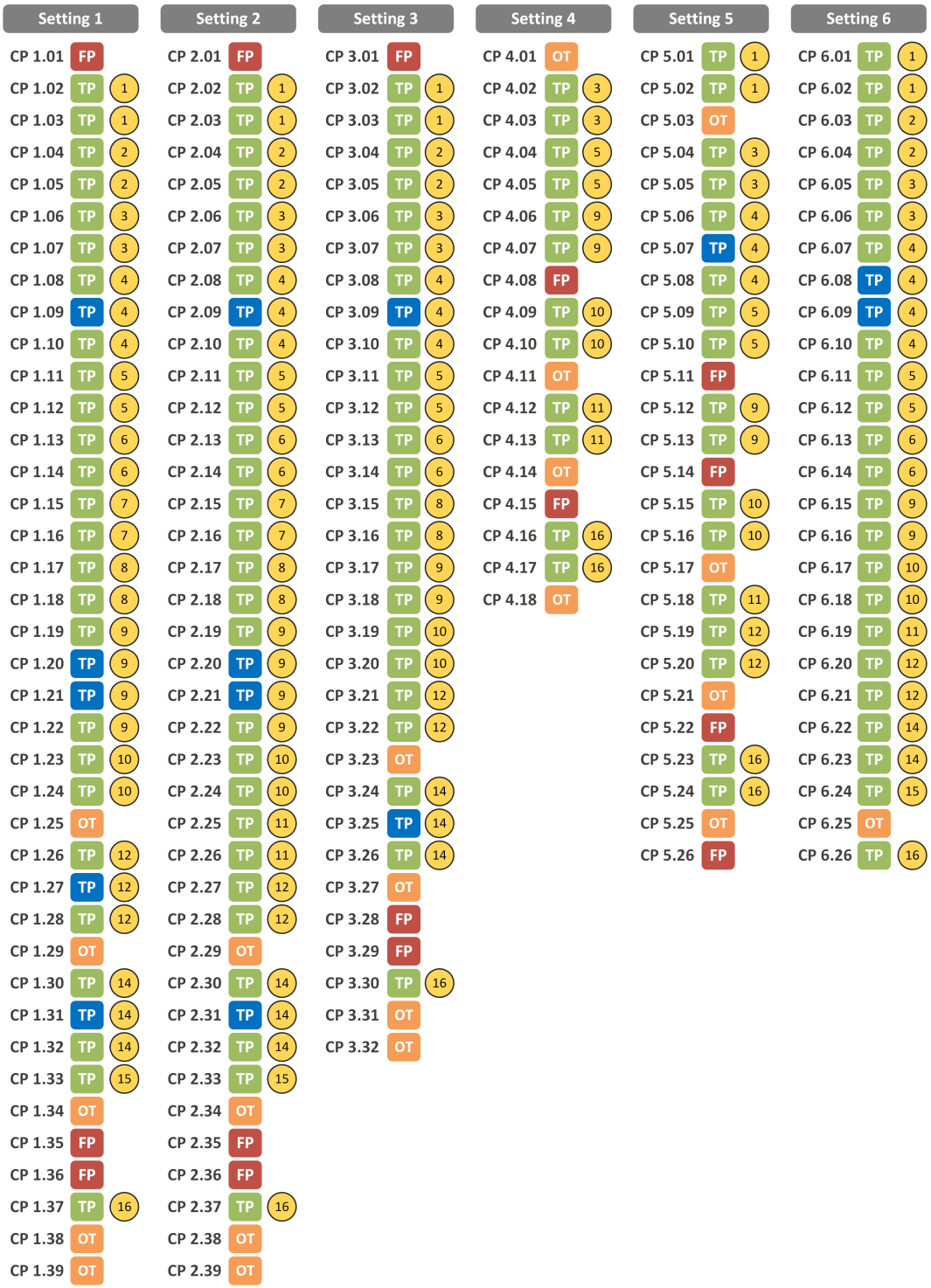


Figure 6.8: Change point - minima correlation map

	total	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	FP	OT
Setting 1	39	2	2	2	3	2	2	2	2	4	2	0	3	0	3	1	1	0	3	5
Setting 2	39	2	2	2	3	2	2	2	2	4	2	2	2	0	3	1	1	0	3	4
Setting 3	32	2	2	2	3	2	2	0	2	2	2	0	2	0	3	0	1	0	3	4
Setting 4	18	0	0	2	0	2	0	0	0	2	2	2	0	0	0	0	2	0	2	4
Setting 5	26	2	0	2	3	2	0	0	0	2	2	1	2	0	0	0	2	0	4	4
Setting 6	26	2	2	2	4	2	2	0	0	2	2	1	2	0	2	1	1	0	0	1

Figure 6.9: Comparison of Results

in the environmental sensor but a rather small extremum within the processor's temperature was not always found.

- Minima 4, 9 and 14 show a tendency of indicating more than two change points. This is due to the fact that the drop from the temperature base level is rather big.
- Minima 7 and 8 as well as the absolute number of found change points indicate a higher sensitivity of the discrete version of the algorithm when choosing similar values for the second upper and lower control limits. This can also be seen when comparing the results for minima 1, 2, 6 and 14 that always got detected by all discrete settings but not by all continuous settings.
- Minimum 13 and 17 was not detected at all although its minimum is clearly visible within the processor's temperature data as well as within the environmental temperature data. This is obviously due to its proximity to minimum number 12 respectively minimum number 16. Obviously the appearance of change points in pairs has an influence on the sensitivity of the algorithm to find consecutive changes.
- Minimum 15 and 16 show a tendency of only indicating one change point. While minimum 16 is clearly visible within the temperature data this is not the case for minimum 15.

Finally, summing it all up the results proof that extrema that exceed a certain level of change in its values and that occur in a certain distance to each other are very likely to be detected without indicating false alarms. This kind of robustness of an analytical method allows its application in any service that profits from finding structural changes within a time series. Therefore, the given online change point detection algorithm aligns itself as yet another

reliable tool in the list of methods to retrieve information out of a given set of data.

6.3 Creating Contextual-Awarness: Temperature Prediction

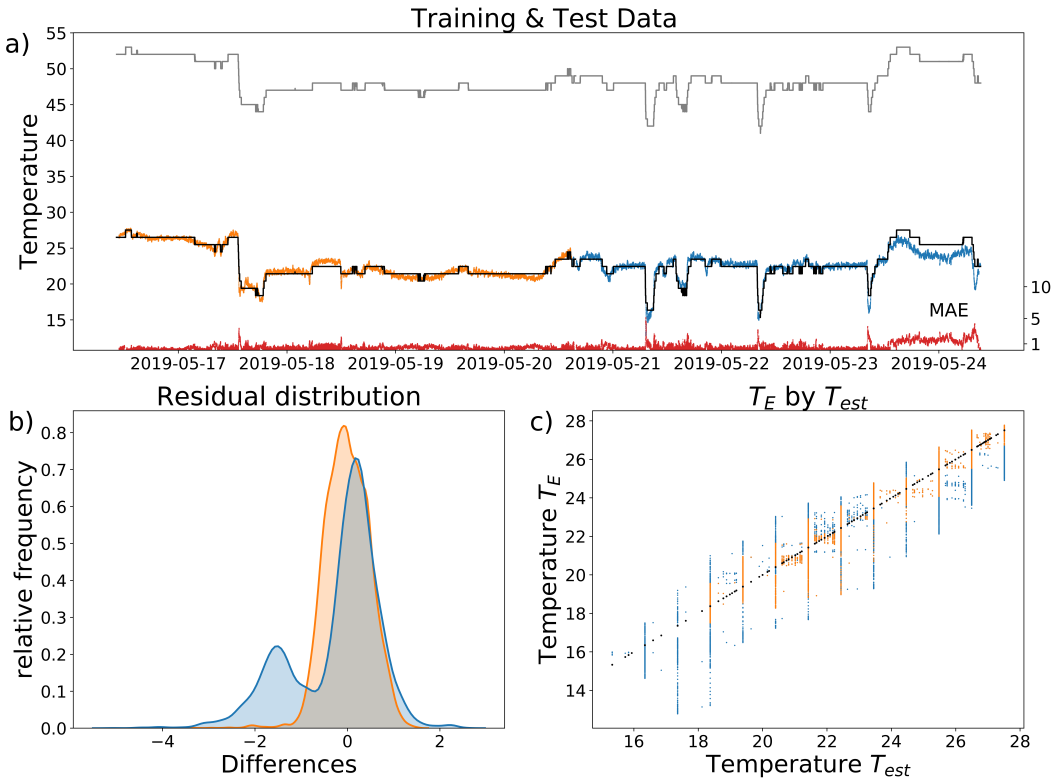


Figure 6.10: Temperature prediction right after training

The developed temperature prediction model is based on a simple linear regression. The processor's temperature T_p of one I/O-module was taken as the only explanatory variable x_t to estimate the environmental temperature T_{est} represented by $\hat{y}_t$. Hence, the linear regression model is defined as:

$$\hat{y}_t = \alpha + x_t\beta \quad (t = 1, ..., n) \tag{6.20}$$

Figure 6.10 a-c) shows the result of that linear regression. The grey line in figure 6.10 a) represents the temperature measurements of one processor, whereas the two-coloured graph in orange and blue shows the temperature measurements of one environmental sensor. A different color is taken to differentiate the training interval of the linear regression with the test interval. The black line T_{est} is the result of applying the first calculated linear regression to the temperature of the processor T_p. The calculated values of the regression coefficients are:

- $\alpha = -26.30$
- $\beta = 1.015$

Furthermore, the mean absolute error (MAE) is visualized and scaled according to the second y-axis on the right side. Figure 6.10 b) shows the distribution of the residuals between the result of the linear regression and the measured values of the environment. Again the orange distribution depicts the errors within the training interval, and the blue one the errors of the test interval. In figure 6.10 c) a scatter plot is shown that plots the environmental training values in orange and the environmental test values in blue. The regression line is represented by black data points. As the values of the explanatory variable T_p are recorded in integers the values appear mainly in grouped lines. Deviations from these integer values derive from the fact that the data stored on the time series database was uploaded in an interval of approximately 2.2 seconds but retrieved on a 10 second interval basis for further analyses. Hence, the values represent a mean value for the given interval.

In general the whole figure shows that the prediction of the temperature using an environmental sensor to train the model is quite promising. A fact that is underlined by the high coefficient of determination R^2 within the training set as well as within the first test data set ($R^2_{train} = 0.95$, $R^2_{test,1} = 0.77$). However, further analysis revealed two major shortcomings that are now going to be discussed.

6.3.1 Shortcomings

Unstable Linear Regression Coefficients

The first shortcoming is about the instability of the linear regression coefficients α and β. When conducting several linear regressions on different training intervals the values of both coefficients are unstable and strongly dependent from each other.

The reason is the high correlation within the values of the processors temperature, which is the only explanatory variable of the model. As already mentioned in chapter 3, adjacent points with strong correlations are a sign of a non independent time series. A value at time t is then strongly dependent on the past values x_{t-1} , x_{t-2}, ... This has an effect on the linear regression that is similar to a collinearity in a multiple linear regression. Looking at the general formulation of a linear regression in matrix format reveals the problem: [76]

$$\underline{Y} = \underline{\underline{X}}\,\underline{b} \tag{6.21}$$

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{bmatrix}}_{\underline{Y}} = \underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_t \end{bmatrix}}_{\underline{\underline{X}}} \underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{\underline{b}} \tag{6.22}$$

To solve for the regression coefficients the equation can be transformed to

$$\underline{b} = (\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \underline{\underline{X}}^T \underline{Y} \tag{6.23}$$

The predicted values for incoming values are hence

$$\underline{\hat{Y}} = \underline{\underline{X}}\,\underline{b} = \underbrace{\underline{\underline{X}}(\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \underline{\underline{X}}^T}_{\underline{\underline{H}}}\, \underline{Y} \tag{6.24}$$

where the matrix $\underline{\underline{H}}$ is referred to as *projection matrix* also known as *hat-matrix* as it takes $\underline{Y}$ as an input and calculates the predicted "hat-values" $\underline{\hat{Y}}$. In case the columns of $\underline{\underline{X}}$ are not linearly independent then $\underline{\underline{X}}^T\underline{\underline{X}}$ is singular and the least square coefficients $\underline{b}$, namely α and β are not uniquely defined [76].

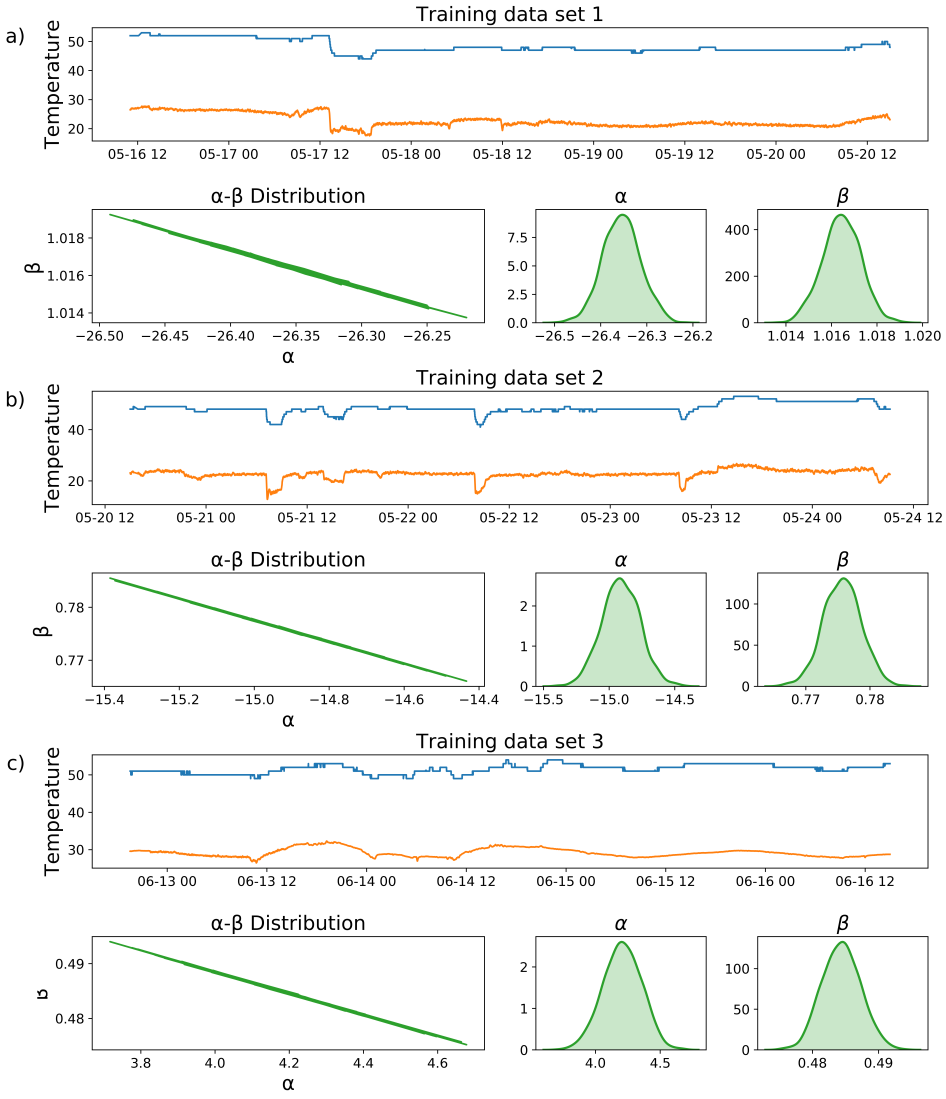


Figure 6.11: α-β Distribution for three different training data sets

Although this is not the case in the presented linear regression a multiple recalculation of the coefficients shows that the values for α and β are unstable and strongly dependent on each other. This indicates that $\underline{\underline{X}}^T \underline{\underline{X}}$ is close to being singular.

The figures 6.11 a) to c) illustrate the problem. For three different training intervals the coefficients α and β were calculated. Therefore 40% of the data points of an interval were randomly chosen and a linear regression was conducted. For each interval this procedure was repeated a thousand times. The values of α and β of each interval are then plotted against each other making up the solution space of all the linear regressions for the given interval. The distribution of each coefficient alone is then plotted right next to it.

Comparing the figures 6.11 a), b) and c) it can easily be seen that the distributions of α and β differ a lot between the different training data sets. In figure 6.11 a) β is almost one and hence α can be interpreted as the offset between the explanatory variable and the estimated variable. This however, is not the case for the other two distributions.

Increase of Mean Absolute Error (MAE)

The second shortcoming becomes clear when applying the prediction model to the consecutive incoming data. Using the same coefficient values α and β as before shows that there is an increase in the error of the model. As illustrated in figure 6.12 a) the predicted values depicted in black seem to drift away from the real environmental temperature values shown in blue. Hence, the mean absolute error in red increases.

When looking at the box plots of the error's distribution at a given processor temperature level as shown in figure 6.13, there is no sign for a processor temperature dependent error.

Thus, there is a bias that generally underestimates the value of the real temperature. This results in a linear regression with mainly positive errors between the prediction and the measured environmental temperature.

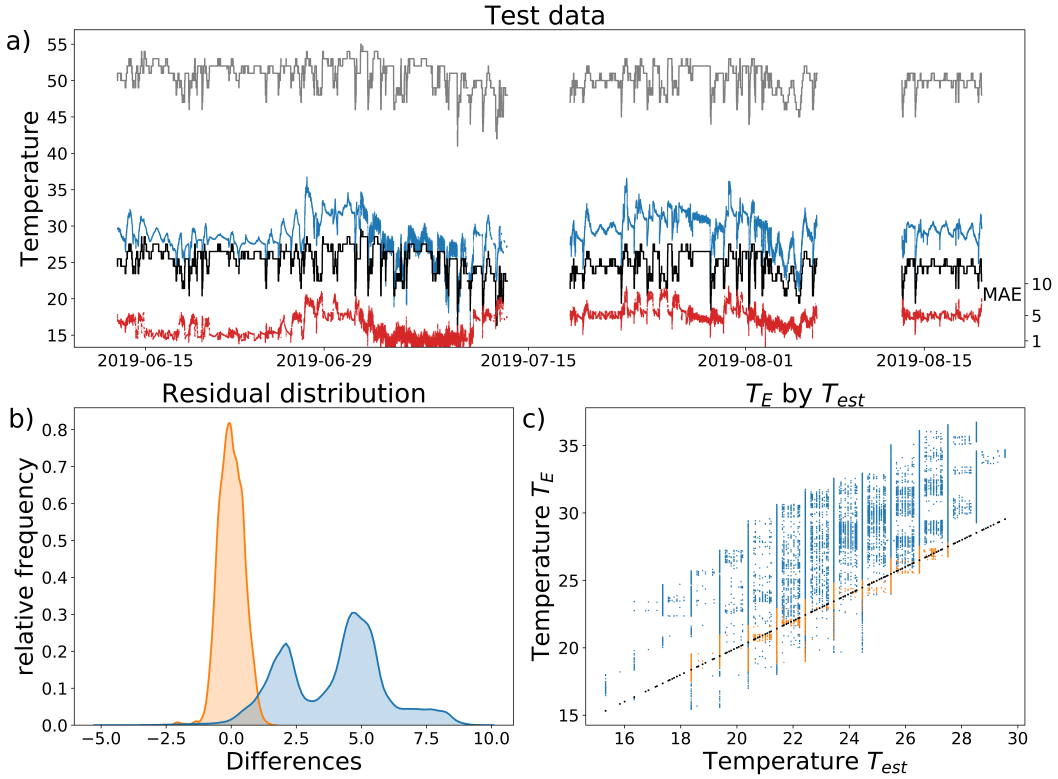


Figure 6.12: Temperature prediction over longer periods of time

While figure 6.13 a) shows pretty narrow distributions of errors for the training data, figure 6.13 b) indicates a systematic increase of the errors but no systematic deviation that is dependent on the temperature of the processor. This indicates that there is no correlation between an increase of the processor's temperature and an increase of the error. Hence, there is most probably a need for a frequent recalibration due to systematic measurement errors of the processor's temperature sensors.

These errors might include possible deviations due to linearity errors, hysteresis errors, zero point error and gain errors that have an increasing impact on the prediction over time [77].

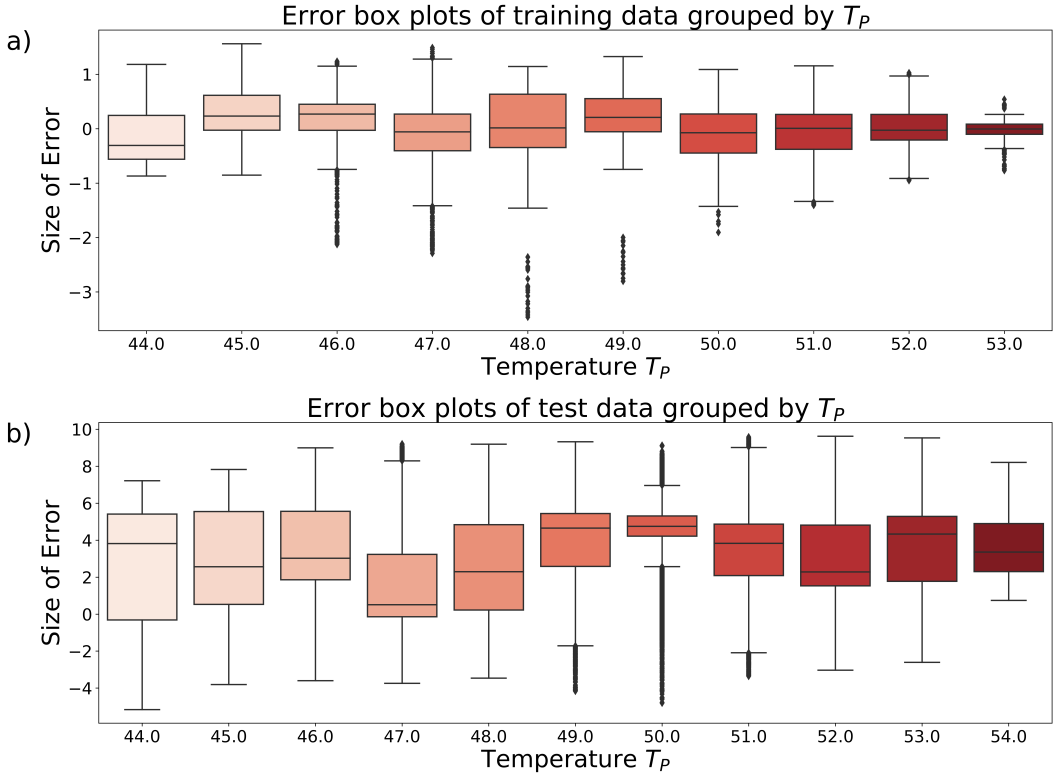


Figure 6.13: Error box plots

6.3.2 Results of Temperature Prediction

Considering the results and the shortcomings a prediction model using the processor's temperature as the only explanatory variably is still promising. To react on the identified shortcomings the following actions were taken.

Due to the non independent explanatory variable the linear regression as defined in equation (6.20) is changed to

$$y_t = \alpha + x_t \quad (t = 1, ..., n) \tag{6.25}$$

Hence, β is set to 1 and α is then representing the offset between the processor's and the predicted temperature. Conducting such a linear regression model on the initial training data leads to a regression coefficient

of $\alpha = -25.56$ and shows with $R^2_{train} = 0.95$ the same high coefficient for determination as the initial model.

To solve the second shortcoming of the increasing mean absolute errors over time, one solution would be to extend the size of the training data set to the whole given data set. Following this approach always minimizes the mean average error as it is inherent to resizing the training data. Especially as the initial training data set turns out to be not a good representation of the overall data. Doing so leads to a value of $\alpha = -22.07$.

The two calculations shall now be compared. Figure 6.14 illustrates the results for both cases based on the new model. Similar to figures 6.10 and 6.12 the grey line in figure 6.14 a) and b) shows the temperature measurement of one processor and the blue line depicts the temperature measurements of one environmental sensor. Both cases of the new linear regression models and their corresponding mean average errors (MAE) are illustrated as well. The first one, based on the initial training data, is depicted in black with its corresponding MAE in red. The other one, using the whole data set as a training interval, is depicted in green with its MAE in purple.

Comparing the two intervals shown in figure 6.14 a) and b) it can be seen that for the first data recorded between 17th and 24th of May the new α-value with $\alpha = -22.07$ shows worse results than the initial with $\alpha = -25.56$. On the long run however the MAE curve for the new α-value is almost always lower than the MAE curve of the first α value. This can also be seen looking at the box plots grouped by T_p for the α-value based on the whole data as illustrated in 6.14 c). It shows the same characteristics as in 6.13 b) but without a systematic error.

Hence, by using the new α-value, the mean absolute errors can be minimized but the results for certain segments of the gathered data will not get as close to the real value as a frequent recalibration of the model would. Depending on the use case either case has its advantages.

Summing up these results, the temperature of the surrounding can be predicted quite well. Due to the mentioned effects such as hysteresis errors, zero point error and gain errors there is, however, a need to constantly recalibrate the system making external temperature sensors necessary.

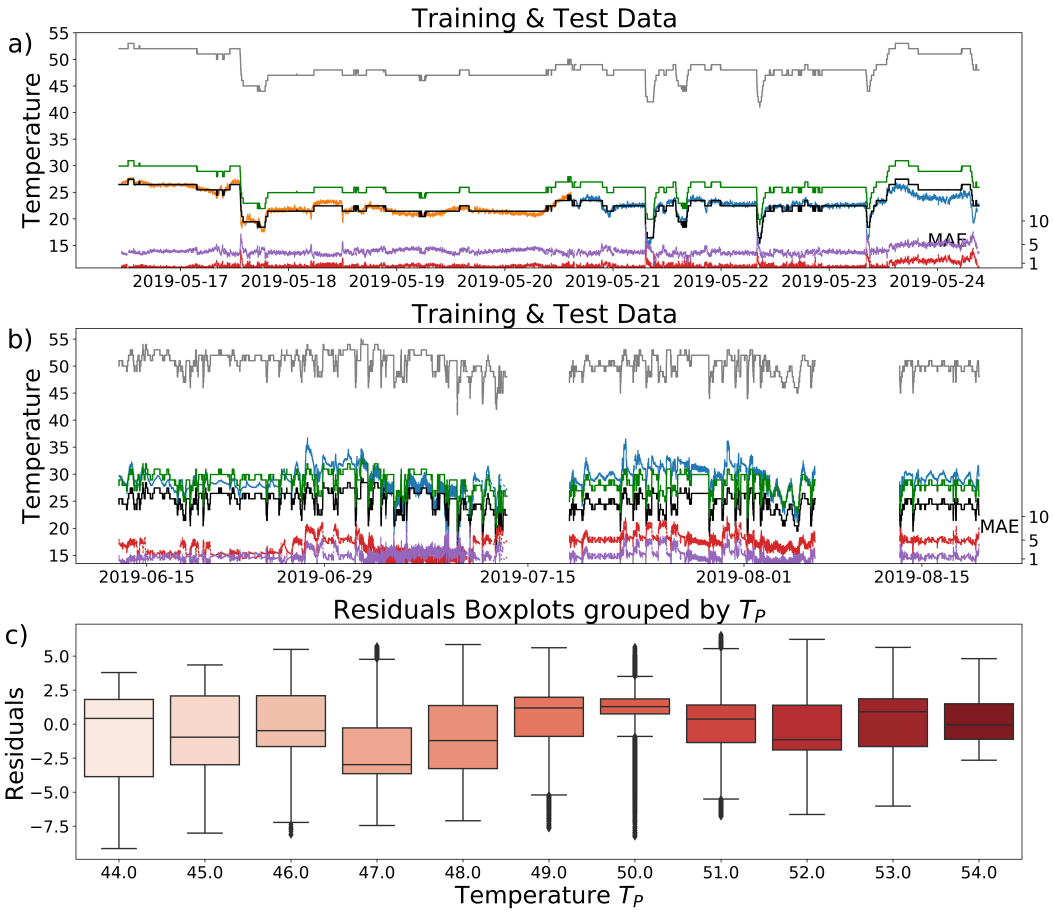


Figure 6.14: Model comparison

6.4 Creating Peer-to-Peer Awareness: Similarity Comparison

In this chapter the idea of a peer-to-peer similarity comparison between I/O-modules will be briefly discussed. No tests regarding an analytical method to compare different time series sections have been performed. This chapter shall only give a short overview on possibilities that might derive out of such an analytical tool. Figure 6.15 shows the temperature of five different I/O-modules. As all of them besides one provide information

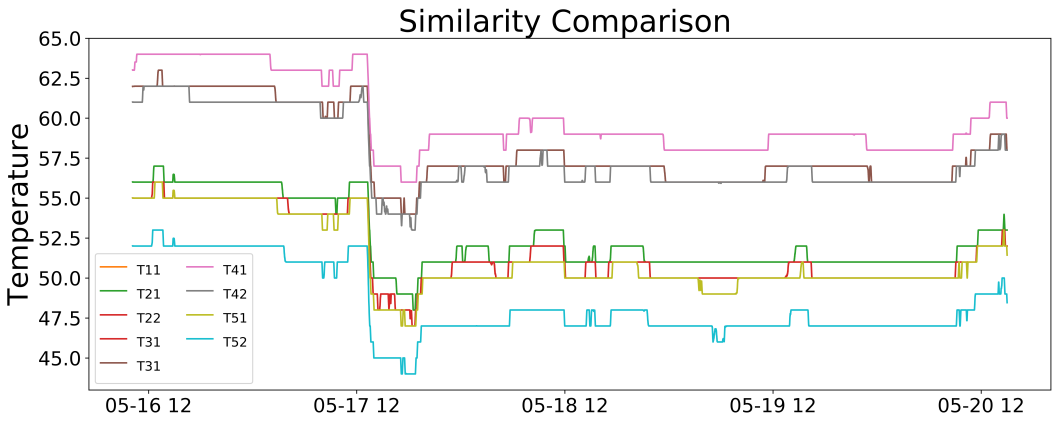


Figure 6.15: Similarity comparison

of two temperature sensors within the module a total number of nine temperature sequences are depicted. The temperature differ because of the simple reason that modules that are locked between other modules will show a lower convection than modules on the very end of a set. This leads to a difference in temperature of up to 10°C. Hence, having the temperature data of a set of devices available already allows to make conclusions on their relative position within the system.

However, the main idea behind a similarity comparison is matching the time series of several modules with each other. This might again help to develop predictive maintenance or safety related services. Comparing a set of data points can either be done geographically or temporally.

Geographically means that one or a set of modules can be compared to other modules that provide their data within a cyber physical production system. These can either be located

- on the very same PLC,
- on another PLC but in the same manufacturing area or
- anywhere else in the world

Temporally means that a set of data points can either be compared to other real time data or to data of the past. This might allow to detect problems that already occurred within the very same module or any other module that provides its historical data.

Having these two different dimensions in mind it is easy to come up with new services that could make use of an automated similarity check. This includes for example the detection of a malfunctioning I/O-module. Similarity Comparison allows to identify internal failures due to hardware errors that probably would not have been detected otherwise.

7 Conclusion

Originating from the initial aim to investigate the potential usage of available but still unused data of an automation device, the idea was born to develop suitable analytical methods to make a system not only aware of its own status but also aware of its environment and its peers. This derived from the fact that within the roadmap to smart manufacturing "phase 1" can be considered as completed for most organizations. This includes the capability of data retrieval and contextualization. In "phase 2" organizations have to posses the right analytical techniques to be able to further process that data.

As a result, it is necessary to gain the knowledge on how to handle time series data. This includes not only the topic of the right infrastructure like storage and latency in cloud and edge computing systems but also the basic knowledge about time series data and its characteristics. As explained in the theoretical part of the thesis, time series data can be characterized by its statistical properties like its underlying distribution, the independence of the data points and the stationarity of the data. As more and more devices are going to provide its data in form of time series it is necessary to know these properties to process the data with the right methods.

Furthermore, self-diagnostic capabilities have been propagated as core scientific fields of research to improve smart manufacturing for the future. In case of an automation device with available temperature data, the ability to predict the outside temperature and to detect change points within the processor's temperature have been defined as base analytical features.

With the given experimental setup it was possible to develop and test two methods that contribute to this need for new features. As could be demonstrated clearly both developed online change point detection methods work very robust and allow individual parametrization for different use cases.

Moreover, the conducted linear regression simplified to an offset shifting function was proven to be able to make conclusions about the environmental temperature. Although there is a need for frequent recalibration, this feature could be provided for several services. In addition to that, the idea for similarity comparison among I/O-modules offer even more possibilities.

In general it could be shown that following the “roadmap to smart manufacturing” or the “IoT layer scheme” it is possible to integrate former embedded systems into a cyber physical system. Hence, already a small set of low-cost IoT and some analytical techniques are sufficient to enhance new services to existing products and processes.

8 Outlook

As an outlook for further research an example of a complete tool chain combining the three presented analytical methods shall be briefly sketched. The idea is to have an end-to-end system, starting from detecting the temperature till an automatically sent alarm message in case of predefined alarm criteria.

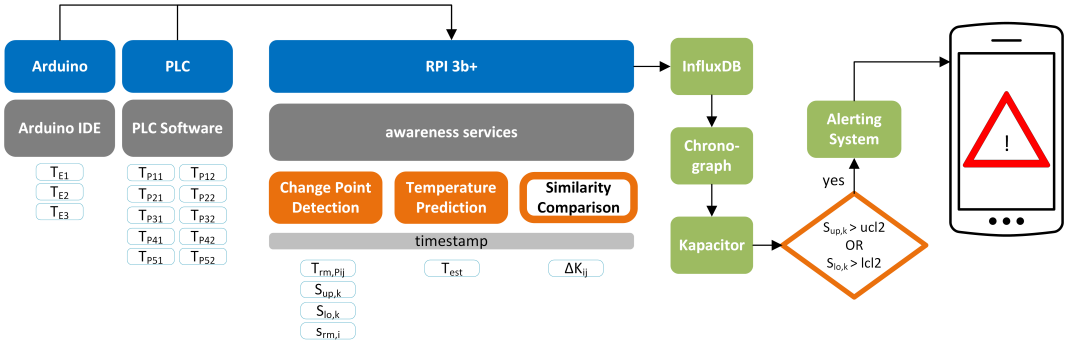


Figure 8.1: Tool chain

Figure 8.1 shows the whole process from gathering the data till setting an alarm on a mobile device. As indicated in the figure and already described in chapter 5 the temperature data of the environmental sensors and the temperature data of the I/O-modules of the PLC are gathered and pushed to a single board computer (Raspberry Pi 3b+). On this edge device the data is processed by performing the respective calculations for each service. The results together with the original data are then sent onto a time series database with a common time stamp. This includes the following variables that are depicted in the figure:

T_{Ei}	Temperature measured by the environmental sensors
T_{Pij}	Temperature measured by the I/O-modules of a given PLC
$T_{rm,Pij}$	Rolled mean temperature of a given I/O-module
$S_{up,k}$	Current sum of log-likelihood differences due to measurements outside of the upper control window
$S_{lo,k}$	Current sum of log-likelihood differences due to measurements outside of the lower control window
$s_{rm,i}$	Rolled mean log-likelihood ratio
T_{est}	Estimated temperature due to the given prediction model
ΔK_{ij}	Measured difference between given temperature curves

So far the calculations were only tested on pseudo-online test data sets that were gathered between May and August 2019. Future studies would have to investigate the full application of the presented services on live data. Therefore the interplay between edge devices and cloud solutions should be further examined. This includes dealing with topics like safety and security within such a systems. Finally the lessons learned of such a prototype would help to define requirements for implementing such systems in industrial applications.

Appendix

Bibliography

[1] P. O'Donovan, K. Leahy, K. Bruton, and D. T. O'Sullivan, "An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities," *Journal of Big Data*, vol. 2, no. 1, 2015.

[2] P. O'Donovan, K. Leahy, K. Bruton, and D. T. J. O'Sullivan, "Big data in manufacturing: A systematic mapping study," *Journal of Big Data*, vol. 2, no. 1, Dec. 2015. DOI: 10.1186/s40537-015-0028-x. [Online]. Available: http://www.journalofbigdata.com/content/2/1/20 (visited on 06/24/2019).

[3] B. B. Gupta, Y. Rahulamathavan, S. Yamaguchi, T. Brooks, and Z. Yan, "IEEE Access Special Section Editorial: Recent Advances in Computational Intelligence Paradigms for Security and Privacy for Fog and Mobile Edge Computing," *IEEE Access*, vol. 7, pp. 134 063–134 070, 2019. DOI: 10.1109/ACCESS.2019.2940302. [Online]. Available: https://ieeexplore.ieee.org/document/8856278/ (visited on 11/20/2019).

[4] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *Journal of Manufacturing Systems*, vol. 48, pp. 157–169, 2018.

[5] F. Tao and Q. Qi, "New it driven service-oriented smart manufacturing: Framework and characteristics," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–11, 2017.

[6] J. Lee, B. Bagheri, and H. A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing letters*, vol. 3, pp. 18–23, 2015.

[7] N. Jazdi, "Cyber physical systems in the context of industry 4.0," in *2014 IEEE international conference on automation, quality and testing, robotics*, IEEE, 2014, pp. 1–4.

[8] E. A. Lee, "Cyber-physical systems-are computing foundations adequate," in *Position paper for NSF workshop on cyber-physical systems: research motivation, techniques and roadmap*, Citeseer, vol. 2, 2006, pp. 1–9.

[9] R. F. Babiceanu and R. Seker, "Big data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook," *Computers in Industry*, vol. 81, pp. 128–137, 2016.

[10] J. Lee, E. Lapira, B. Bagheri, and H.-A. Kao, "Recent advances and trends in predictive manufacturing systems in big data environment," *Manufacturing letters*, vol. 1, no. 1, pp. 38–41, 2013.

[11] G. Shao, S.-J. Shin, and S. Jain, "Data analytics using simulation for smart manufacturing," in *Proceedings of the Winter Simulation Conference 2014*, IEEE, 2014, pp. 2192–2203.

[12] O. Niggemann and V. Lohweg, "On the diagnosis of cyber-physical production systems," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[13] L. Monostori, "Cyber-physical production systems: Roots, expectations and r&d challenges," *Procedia Cirp*, vol. 17, pp. 9–13, 2014.

[14] A. Jantsch and K. Tammemäe, "A framework of awareness for artificial subjects," in *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, IEEE, 2014, pp. 1–3.

[15] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM transactions on autonomous and adaptive systems (TAAS)*, vol. 4, no. 2, 2009.

[16] M. Weinberger, D. Bilgeri, and E. Fleisch, "IoT business models in an industrial context IoT," 2016.

[17] (2018). Gartner identifies top 10 strategic IoT technologies and trends, [Online]. Available: `https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends` (visited on 07/19/2019).

[18] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A survey on sensor-based threats to internet-of-things (IoT) devices and applications," *arXiv:1802.02041 [cs]*, Feb. 6, 2018. arXiv: 1802.02041. [Online]. Available: http://arxiv.org/abs/1802.02041 (visited on 07/19/2019).

[19] S. Thalmann, J. Mangler, T. Schreck, C. Huemer, M. Streit, F. Pauker, G. Weichhart, S. Schulte, C. Kittl, C. Pollak, M. Vukovic, G. Kappel, M. Gashi, S. Rinderle-Ma, J. Suschnigg, N. Jekic, and S. Lindstaedt, "Data analytics for industrial process improvement a vision paper," in *2018 IEEE 20th Conference on Business Informatics (CBI)*, vol. 02, Jul. 2018, pp. 92–96. DOI: 10.1109/CBI.2018.10051.

[20] A. Kajmakovic, R. Zupanc, S. Mayer, N. Kajtazovic, M. Hoeffernig, and H. Vogl, "Improving the Safety of Industrial Environments through Model-based Analytics on hidden Data Sources," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, 2018.

[21] C. Krintz, R. Wolski, N. Golubovic, and F. Bakir, "Estimating outdoor temperature from CPU temperature for IoT applications in agriculture," in *Proceedings of the 8th International Conference on the Internet of Things - IOT '18*, Santa Barbara, California: ACM Press, 2018, pp. 1–8. DOI: 10.1145/3277593.3277607. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3277593.3277607 (visited on 01/08/2019).

[22] R. Niedermann, E. Wyss, S. Annaheim, A. Psikuta, S. Davey, and R. M. Rossi, "Prediction of human core body temperature using non-invasive measurement methods," *International Journal of Biometeorology*, vol. 58, no. 1, pp. 7–15, Jan. 2014. DOI: 10.1007/s00484-013-0687-2. [Online]. Available: http://link.springer.com/10.1007/s00484-013-0687-2 (visited on 01/08/2019).

[23] V. Guralnik and J. Srivastava, "Event detection from time series data," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99*, San Diego, California, United States: ACM Press, 1999, pp. 33–42. DOI: 10.1145/312129.312190. [Online]. Available: http://portal.acm.org/citation.cfm?doid=312129.312190 (visited on 07/25/2019).

[24] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and information systems*, vol. 51, no. 2, pp. 339–367, May 2017. DOI: 10.1007/s10115-016-0987-z. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5464762/ (visited on 04/15/2019).

[25] M. Basseville, I. V. Nikiforov, *et al.*, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.

[26] T. C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[27] A. Zeileis, F. Leisch, K. Hornik, C. Kleiber, B. Hansen, E. C. Merkle, and M. A. Zeileis, *Package 'strucchange'*, 2015.

[28] R. Bjetak, K. Diwold, and A. Kajmaković, "Retrofit: Creating awareness in embedded systems - a usecase for plcs," in *Proceedings of the 9th International Conference on the Internet of Things*, ser. IoT 2019, Bilbao, Spain: ACM, 2019, 33:1–33:4. DOI: 10.1145/3365871.3365907. [Online]. Available: http://doi.acm.org/10.1145/3365871.3365907.

[29] V. Kashyap, "Taxonomy: Biomedical health informatics," *Encyclopedia of Database Systems*, pp. 2908–2911, 2009.

[30] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2017.

[31] A. Kulkarni. (Nov. 19, 2018). What the heck is time-series data (and why do i need a time-series database)? Timescale Blog, [Online]. Available: https://blog.timescale.com/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/ (visited on 07/19/2019).

[32] (2019). Influxdb 1.x: Open source time series platform — influxdata, [Online]. Available: https://www.influxdata.com/time-series-platform/ (visited on 07/22/2019).

[33] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*, 4th ed, ser. Wiley series in probability and statistics. Hoboken, N.J: John Wiley, 2008, 746 pp.

[34] L. Cohen, *Time-frequency analysis*. Prentice hall, 1995, vol. 778.

[35] S. Torkamani and V. Lohweg, "Survey on time series motif discovery: Time series motif discovery," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 2, Mar. 2017. DOI: 10.1002/widm.1199. [Online]. Available: http://doi.wiley.com/10.1002/widm.1199 (visited on 07/22/2019).

[36] M.-A. Aufaure and E. Zimányi, Eds., *Business intelligence: second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, tutorial lectures*, Lecture notes in business information processing 138, Heidelberg ; New York: Springer, 2013, 233 pp.

[37] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PloS one*, vol. 13, no. 3, 2018.

[38] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, Presses universitaires de Louvain, 2015.

[39] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[40] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statistical Analysis and Data Mining*, vol. 5, no. 2, pp. 114–127, Apr. 2012. DOI: 10.1002/sam.10124. [Online]. Available: http://doi.wiley.com/10.1002/sam.10124 (visited on 07/25/2019).

[41] S. Sharma, D. A. Swayne, and C. Obimbo, "Trend analysis and change point techniques: A survey," *Energy, Ecology and Environment*, vol. 1, no. 3, pp. 123–130, Jun. 2016. DOI: 10.1007/s40974-016-0011-1. [Online]. Available: http://link.springer.com/10.1007/s40974-016-0011-1 (visited on 07/25/2019).

[42] S. Ahmad and S. Purdy, "Real-time anomaly detection for streaming analytics," *arXiv:1607.02480 [cs]*, Jul. 8, 2016. arXiv: 1607.02480. [Online]. Available: http://arxiv.org/abs/1607.02480 (visited on 07/25/2019).

[43] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 1, 2009. DOI: 10.1145/1541880.1541882. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1541880.1541882 (visited on 07/25/2019).

[44] M. Basseville, "Statistical methods for change detection," *Control systems, robotics and automation*, vol. 16, 2009.

[45] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, Jul. 2013. DOI: 10.1016/j.neunet.2013.01.012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0893608013000270 (visited on 07/25/2019).

[46] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, VLDB Endowment, 2004, pp. 180–191.

[47] H. Wang, H. Guo, X. Wu, and J. Li, "Temporal and spatial anomaly detection using phase spectrum of quaternion fourier transform," in *2015 IEEE International Conference on Information and Automation*, Lijiang, China: IEEE, Aug. 2015, pp. 657–662. DOI: 10.1109/ICInfA.2015.7279368. [Online]. Available: http://ieeexplore.ieee.org/document/7279368/ (visited on 07/25/2019).

[48] M. Neuhaus and H. Bunke, "Automatic learning of cost functions for graph edit distance," *Information Sciences*, vol. 177, no. 1, pp. 239–247, Jan. 2007. DOI: 10.1016/j.ins.2006.02.013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025506000569 (visited on 08/09/2019).

[49] M. Best and D. Neuhauser, "Walter a shewhart, 1924, and the hawthorne factory," *BMJ Quality & Safety*, vol. 15, no. 2, pp. 142–143, 2006.

[50] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1, pp. 100–115, 1954. DOI: 10.2307/2333009. [Online]. Available: https://www.jstor.org/stable/2333009 (visited on 08/05/2019).

[51] C. W. Kang and P. H. Kvam, *Basic Statistical Tools for Improving Quality: Kang/Basic*. Hoboken, NJ, USA: John Wiley & Sons, Inc., Apr. 8, 2011. DOI: 10.1002/9781118491751. [Online]. Available: http://doi.wiley.com/10.1002/9781118491751 (visited on 07/30/2019).

[52] L. B. Castañeda, V. Arunachalam, and S. Dharmaraja, *Introduction to Probability and Stochastic Processes with Applications: Castañeda/Introduction*. Hoboken, NJ, USA: John Wiley & Sons, Inc., Jun. 11, 2012. DOI: 10.1002/9781118344972. [Online]. Available: http://doi.wiley.com/10.1002/9781118344972 (visited on 08/01/2019).

[53] R. L. Brown, J. Durbin, and J. M. Evans, "Techniques for testing the constancy of regression relationships over time," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 37, no. 2, pp. 149–163, 1975.

[54] W. Ploberger and W. Krämer, "The cusum test with ols residuals," *Econometrica*, vol. 60, no. 2, pp. 271–285, 1992. DOI: 10.2307/2951597. [Online]. Available: https://www.jstor.org/stable/2951597 (visited on 08/05/2019).

[55] R. Jarrow and P. Protter, "A short history of stochastic integration and mathematical finance: The early years, 1880–1970," in *Institute of Mathematical Statistics Lecture Notes - Monograph Series*, Beachwood, Ohio, USA: Institute of Mathematical Statistics, 2004, pp. 75–91. DOI: 10.1214/lnms/1196285381. [Online]. Available: http://projecteuclid.org/euclid.lnms/1196285381 (visited on 08/05/2019).

[56] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006. DOI: 10.1016/j.patrec.2005.10.010. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016786550500303X (visited on 08/08/2019).

[57] K. A. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," in *Proceedings of the sixth international workshop on Machine learning*, Elsevier, 1989, pp. 160–163.

[58] M. Thoma, *ROC Curve*, Jun. 2018. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/3/36/Roc-draft-xkcd-style.svg (visited on 11/30/2019).

[59] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, 2005. DOI: 10.3354/cr030079. [Online]. Available: http://www.int-res.com/abstracts/cr/v30/n1/p79-82/ (visited on 08/09/2019).

[60] L. J. Wells, F. M. Megahed, C. B. Niziolek, J. A. Camelio, and W. H. Woodall, "Statistical process monitoring approach for high-density point clouds," *Journal of Intelligent Manufacturing*, vol. 24, no. 6, pp. 1267–1279, Dec. 2013. DOI: 10.1007/s10845-012-0665-2. [Online]. Available: http://link.springer.com/10.1007/s10845-012-0665-2 (visited on 08/09/2019).

[61] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, Oct. 10, 2006. DOI: 10.1145/1163593.1163596. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1163593.1163596 (visited on 08/09/2019).

[62] A. Mohr, "Quantum computing in complexity theory and theory of computation," *Carbondale, IL*, 2014.

[63] G. Wellenreuther and D. Zastrow, "Automatisierungsgeräte," in *Handbuch Elektrotechnik*, W. Plaßmann and D. Schulz, Eds., Wiesbaden: Springer Fachmedien Wiesbaden, 2016, pp. 775–778. DOI: 10.1007/978-3-658-07049-6_57. [Online]. Available: http://link.springer.com/10.1007/978-3-658-07049-6_57 (visited on 05/06/2019).

[64] E. Parr, "Programmable controllers," in *Electrical Engineer's Reference Book*, Elsevier, 2003, pp. 1–52. DOI: 10.1016/B978-075064637-6/50016-2. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780750646376500162 (visited on 05/06/2019).

[65] (2019). SIMATIC s7-1500, siemens.com Global Website, [Online]. Available: https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500.html (visited on 04/29/2019).

[66] (2019). SIMATIC IOT2000, siemens.com Global Website, [Online]. Available: https://new.siemens.com/global/en/company/sustainability/education/sce/iot2000.html (visited on 05/01/2019).

[67] G. Wellenreuther and D. Zastrow, "Grundzüge der SPS-Norm IEC 61131-3," in *Handbuch Elektrotechnik*, W. Plaßmann and D. Schulz, Eds., Wiesbaden: Springer Fachmedien Wiesbaden, 2016, pp. 779–785. DOI: 10.1007/978-3-658-07049-6_58. [Online]. Available: http://link.springer.com/10.1007/978-3-658-07049-6_58 (visited on 05/06/2019).

[68] (2019). What is OPC? OPC Foundation, [Online]. Available: https://opcfoundation.org/about/what-is-opc/ (visited on 08/13/2019).

[69] H. Gierens, "Mikrocomputertechnik," in *Handbuch Elektrotechnik*, W. Plaßmann and D. Schulz, Eds., Wiesbaden: Springer Fachmedien Wiesbaden, 2016, pp. 661–728. DOI: 10.1007/978-3-658-07049-6_52. [Online]. Available: http://link.springer.com/10.1007/978-3-658-07049-6_52 (visited on 08/12/2019).

[70] M. Cunningham and G. Bibby, "Electrical measurement," in *Electrical Engineer's Reference Book*, Elsevier, 2003, pp. 1–43. DOI: 10.1016/B978-075064637-6/50011-3. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780750646376500113 (visited on 05/06/2019).

[71] U. Isikdag, *Enhanced Building Information Models*, ser. SpringerBriefs in Computer Science. Cham: Springer International Publishing, 2015. DOI: 10.1007/978-3-319-21825-0. [Online]. Available: http://link.springer.com/10.1007/978-3-319-21825-0 (visited on 08/12/2019).

[72] (2019). Raspberry pi 3 vs arduino - learn the 6 amazing differences, [Online]. Available: https://www.educba.com/raspberry-pi-3-vs-arduino/ (visited on 08/12/2019).

[73] S. N. Z. Naqvi, S. Yfantidou, and E. Zimányi, "Time series databases and influxdb," *Studienarbeit, Université Libre de Bruxelles*, 2017.

[74] W. Hamilton, *Discussions on philosophy and literature, education and university reform*. Harper, 1855.

[75] H. Steiner, *Hoehere stroemungslehre und waermeuebertragung*, Script, WS 2016/2017.

[76] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed, ser. Springer series in statistics. New York, NY: Springer, 2009.

[77] M. Paulweber and K. Lebert, *Mess- und Prüfstandstechnik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014. DOI: 10.1007/978-3-658-04453-4. [Online]. Available: http://link.springer.com/10.1007/978-3-658-04453-4 (visited on 09/20/2019).